METHODOLOGY FOR SIMULATION AND ANALYSIS OF PREEMPTIVE REBOOKING FOR CANCELLED AIRLINE PASENGERS

by

Sanja Avramovic
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Information Technology

Committee:

______________________________	Dr. Lance Sherry, Dissertation Director
______________________________	Dr. George Donohue, Committee Member
______________________________	Dr. Brian L. Mark., Committee Member
______________________________	Dr. John F. Shortle, Committee Member
______________________________	Dr. Stephen Nash, Senior Associate Dean
______________________________	Dr. Kenneth S. Ball, Dean, Volgenau School of Engineering

Date: ______________________________ Summer Semester 2015
George Mason University
Fairfax, VA

Methodology for Simulation and Analysis of Preemptive Rebooking for Cancelled Airline Pasengers

A Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

by

Sanja Avramovic

Director: Dr. Lance Sherry, Professor
Department of System Engineering and Operations Research

Summer Semester 2015
George Mason University
Fairfax, VA

DEDICATION

To my daughter Tara, my biggest inspiration, my husband Ivan, who is always my support, and my parents Radmila and Slobodan, who taught me the importance of imagination and discovery.

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor Dr. Lance Sherry, for his guidance through my research. With his expertise, he helped me put my work in perspective and perfect it. I owe almost all I know about air transportation management to Dr. Lance Sherry.

I am grateful to my committee members, Dr. George Donohue, for his difficult questions which inspired my work, Dr. Brian Mark for insights in helping shape my dissertation and Dr. John Shortle for helping me with analysis. Thank my committee for provided feedback that help improve my work.

Also, I would like to thank Paula Lewis for her unending support.

Thanks for Bozidar Avramovic for his help in reviewing my work.

I am grateful for the support of my friends and family, especially my parents Slobodan and Radmila Stanisic, my sister Maja Blagojevic, and husband Ivan Avramovic and daughter Tara Avramovic who supported me at every stage of my life. Their encouragement helped me to reach whatever I could imagine.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Air Navigation Service Providers	ANASP
Airfares Not Refund	ANR
Airline On-Time Performance	AOTP
Airline Operations Center	AOC
Airline Operations Control Centers	AOCC
Airline Origin and Destination Survey	DB1B
Airline Service Quality Performance	ASQP
American Airlines	AA
Antonio B. Won Pat International Airport (Guam International Airport)	GUM
Baltimore Washington International Airport	BWI
Bureau of Transportation Statistics	BTS
Chicago Midway International Airport	MDW
Chicago O'Hare International Airport	ORD
Collaborative Decision Making	CDM
Computer Reservation Systems	CRS
Corporates Travel Expense Savings	CTES
Dallas Love Field	DAL
Dallas/Fort Worth International Airport	DFW
Denver International Airport	DEN
Department of Transportation	DOT
Detroit Metropolitan Wayne County Airport	DTW
Enhanced Traffic Management System	ETMS
George Bush Intercontinental Airport	IAH
Hartsfield–Jackson Atlanta International Airport	ATL
International Civil Aviation Organization	ICAO
Irregular airline flight operations	IROPS
LaGuardia Airport	LGA
Logan International Airport	BOS
Los Angeles International Airport	LAX
McCarran International Airport	LAS
Narita International Airport	NRT
National Airspace	NAS
Newark Liberty International Airport	EWR
Official Airline Guide	OAG
Origin Destination	O-D
Origin Hub Destination	O-H-D
Orlando International Airport	MCO
Overnight Passengers	SPAX
Passenger delay costs	PDM
Passenger disruption costs	DPM

Passenger Trip Delay Calculator	PTDC
Passenger Trip Delay	PTD
Passengers	PAX
Philadelphia International Airport	PHL
Phoenix Sky Harbor International Airport	PHX
Port Columbus International Airport	CMH
Remaining Passengers	RPAX
San Francisco Airport	SFO
Schedule B-43 Aircraft Inventory	B-43
Southwest Operational Disruption Accommodation	SODA
Total passenger trip delay	TPTD
United Airlines	UA
William P. Hobby Airport	HOU

ABSTRACT

METHODOLOGY FOR SIMULATION AND ANALYSIS OF PREEMPTIVE REBOOKING FOR CANCELLED AIRLINE PASENGERS

Sanja Avramovic, Ph.D.

George Mason University, 2015

Dissertation Director: Dr. Lance Sherry

Although, on average, only 2.1% of airline flights are cancelled each year, some of these cancellations occur in batches due to events that impact network operations such as snow storms, equipment outages, and labor issues. Batch cancellations impact a large number of passengers at the same time in the same location and have a negative effect on airline revenue (due to refund obligations), corporate profits (due to unbudgeted costs of employee travel), and overall traveler experience. Since some of these batch flight cancellations can be now be accurately predicted in advance, along with changes in the National Airspace System (NAS) that allow airlines to plan cancellations in advance (i.e. Collaborative Decision Making programs) and the ubiquity of inexpensive and reliable broadband communication services for communication between airlines and passengers, it is now possible for airlines to migrate from a concept-of-operations of rebooking passengers after the cancellation event, to re-booking passengers before the large scale cancellation event (i.e. pre-emptive rebooking).

This dissertation describes a method for Monte Carlo analysis of the feasibility and benefits of pre-emptive rebooking of airline passengers. Five case-studies of one-day cancellation events for major U.S. network carriers hub showed that: (i) pre-emptive rebooking is feasible accommodating more than 70% of the passengers seeking to rebook pre-emptively, (ii) rebooking passengers on the same day before the departure time of the cancelled flight, accommodates a majority of the passengers (i.e. previous day rebooking is not required), (iii) airlines can recoup up to an average of $297K per one-day event at each hub in airfare refund obligations, and (iv) corporations sponsoring business travel can save collectively up to an average of $49K on unplanned travel expenses for one-day event at each hub. The implications of these results for airlines, corporations and travelers are discussed.

1. INTRODUCTION

During the course of a year, there are a number of reduced flight capacity events in which, due to increased coordination amongst National Airspace (NAS) operational stakeholders, flights are cancelled by airlines *in advance* of the scheduled departure time. For example, this may occur in preparation for a major snow storm impacting a hub airport, in anticipation of a labor action, or in anticipation of staffing or equipment shortages. In 2012, there were 273 events for which airlines cancelled 5 or more flights departing from a given airport.

The traditional concept-of-operations for an airline is to *wait-and-see*. It is in the airline's best interest to wait until two hours before the scheduled departure to announce the flight cancellation. Passengers are then rebooked on alternate flights by on-site customer service agents, or increasingly, self-service kiosks and on-line websites. The outcome of this concept-of-operations is reduced options to rebook passengers, resulting in overnight passengers, and adverse impact on passengers' ability to arrive on time for time-critical scheduled events (e.g. business meetings, life-cycle events, sporting events, performances, etc.).

This thesis describes an alternative approach to mitigating the effects of IROPS based on exploiting the short term forecasts of cancellations and flexibility in airline passenger travel plans to rebook passengers in a preemptive manner, in advance of the anticipated IROPS. This concept-of-operations is made feasible by four changes: (1) improved forecast accuracy of airspace bottlenecks by Air Navigation Service Providers

(ANSPs), (2) dissemination of airspace information and coordination of slots by Collaborative Decision Making (CDM), (3) ubiquitous availability of wireless broadband for immediate communication with passengers, and (4) internet reservation systems for rebooking.

This thesis examines the feasibility and benefits of preemptive rebooking on the *same day before the ticketed scheduled departure time*, and on the *previous day*. Extensive case-studies of preemptive rebooking for U.S. domestic operations, undertaken in this thesis using a Monte Carlo assignment of passengers willing to be rebooked on an earlier flight, showed that:

- Preemptive rebooking in advance of forecast large-scale cancellation events is feasible.
- About two thirds of the passengers seeking preemptive rebooking can be accommodated before their original scheduled flight time. The remaining one third of passengers seeking preemptive rebooking cannot be re-accommodated due to insufficient seats.
- Preemptive rebooking on the previous day was not required as it did not significantly change the percentage of passengers re-accommodated.
- Airlines are able to recoup up to just over a half in airfares that would otherwise need to be refunded.
- Corporations sponsoring business travel for their employees also benefit by savings of a similar magnitude, up to 53% of unbudgeted overnight costs, which would need to be paid if Preemptive Rebooking was not used.

- Passenger Trip Delay (PTD) and the number of overnight and passenger which cannot be rebooked decrease by a half (up to 50%), and the decrease in the number of passengers that cannot be rebooked at all is 55%.

The results of the case-studies show the way in which increased travel flexibility for airline passengers, enabled by rapid, inexpensive wireless/broadband capabilities for communication, yields win-win results both for airline passengers and airlines.

A degree to which flight cancellations may be subject to preemptive rescheduling is evident in the statistics from U.S. domestic operations in 2012, where there were over 3000 events in which an airline cancelled more than 5 flights (tactical cancellations as opposed to mechanical failure) on a given day, departing a given airport or arriving at a given airport.

The preemptive rebooking concept of operations creates a win-win scenario for all stakeholders. Airlines are able to recoup revenues that would otherwise be lost to refunds. Airlines are also able to distribute the rebooking task in advance of the IROPS, taking the pressure off after the IROPS when resources are the scarcest. Airlines increase their flight operations flexibility for mitigating IROPS and maintain a greater degree of customer good will.

By taking advantage of preemptive rebooking, Corporates can reduce their loss in cost for overnight stays, transportation, and similar travel expenses that would be required by their travelling employees.

Passengers who are offered preemptive rebooking would maintain control of their destiny, by making decisions on the value of alternate travel plan options given the cancellation of their flight. The advance warning provides them the chance to avoid last minute changes in plans that can affect business relationships and attendance at fixed-time events.

Airports and ANSP also benefit by avoiding passenger overnight stays at airports as well as excessive demand for flights in capacity constrained conditions.

This thesis is organized as follows: Chapter 1 is the introduction. Chapter 2 provides a literature review. Chapter 3 describes the proposed method and the associated models. Chapter 4 describes the method of analysis and detailed results of one case. Chapter 5 includes a discussion of the results, future work and conclusions.

Airline Transportation Network

Airlines schedule flights to operate in a time-space network. In this network airports are represented as nodes, and aircraft movements are arcs. The arcs may appear in the same airport or between airports, and wraparound between the first and last time node.

Well-designed itineraries (and their corresponding networks) are made so as to maximize revenue, by meeting passengers travel demands with appropriate seat capacity, and by minimizing costs through using the most cost-effective aircraft, maximizing aircraft utilization, and minimizing the impact of disruptions.

Hub-and-spoke networks are centralized networks, designed like a chariot wheel, with a hub as a center, connecting all spokes. Passengers travel from their origin to their destinations, via the hub. The benefits of this kind of networks are that only *n*-1 routes are necessary to connect all *n-1* spokes, which increases efficiency and lowers operating costs. This means that an airline can operate many origin-destination markets with fewer departures and fewer aircrafts. A centralized model decreases flexibility, increases scheduling complexity, and introduces a single point of failure (bottleneck).

An example of a hub-and-spoke network, featuring United Airlines, is shown in figure 1. The hubs are in Chicago (ORD), Newark (EWR), Boston (BOS), and San Francisco (SFO).

Another major airline transportation network type is *point-to-point*. In this network a central location does not exist, and $\frac{n\cdot(n-1)}{2}$ routes are required to connect every node in the network. Often point-to-point networks operate focusing on non-stop flights with no connecting passengers, which improves aircraft and crew productivity.

Southwest Airlines is an example of an airline which uses a point-to-point network, but it uses "airport of interests" to connect passengers. Those airports are Chicago (MDW) Las Vegas (LAS), Baltimore Washington (BWI), Dallas (DAL), Phoenix (PHX), Denver (DEN), Houston (HOU), Atlanta (ATL), Los Angeles (LAX) and Orlando (MCO).

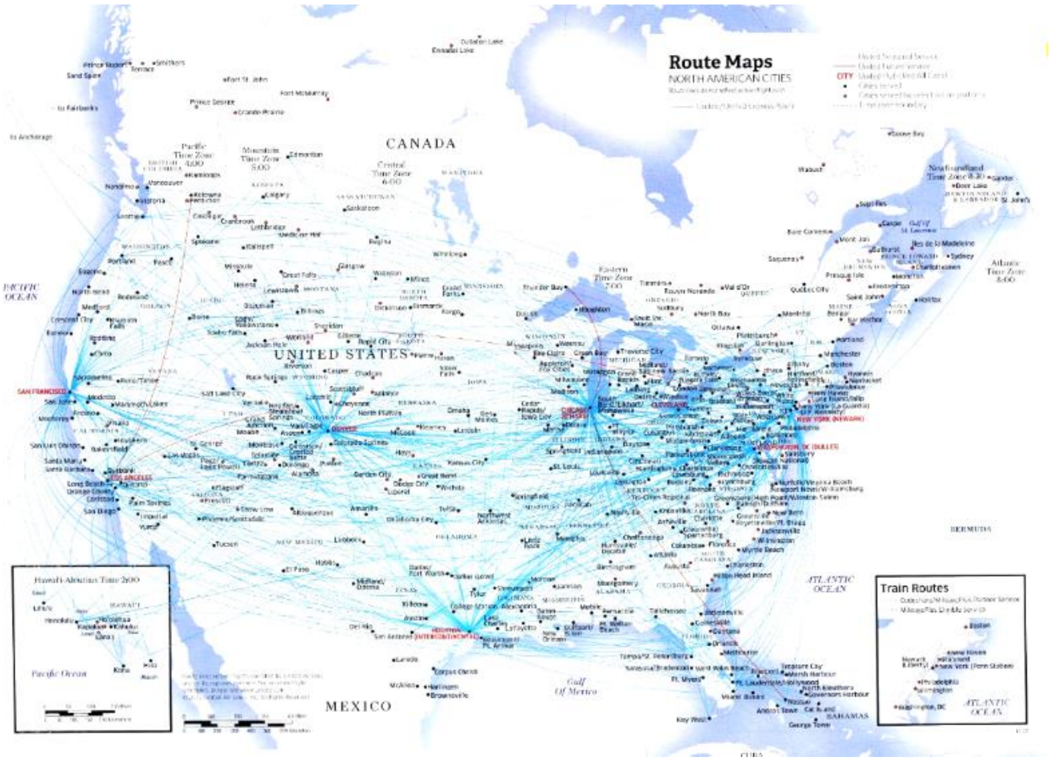


Figure 1 Hub and Spoke network: United Airlines, route map

Irregular Operations

Even if cancellations are not very frequent events, their impact can be big. Daily Number of Cancelled Flights during 2012, for the UA network is shown in the figure 2. There were a total of 82 days which had more than 10 cancelled flights. A total of 7 days had more than 100 cancellations. The largest number of cancelled flights was 720 (on October 29th), while the biggest event (a 4-day cancellation event) involved 1860 cancellations total.

This dissertation shall only analyze one day events.

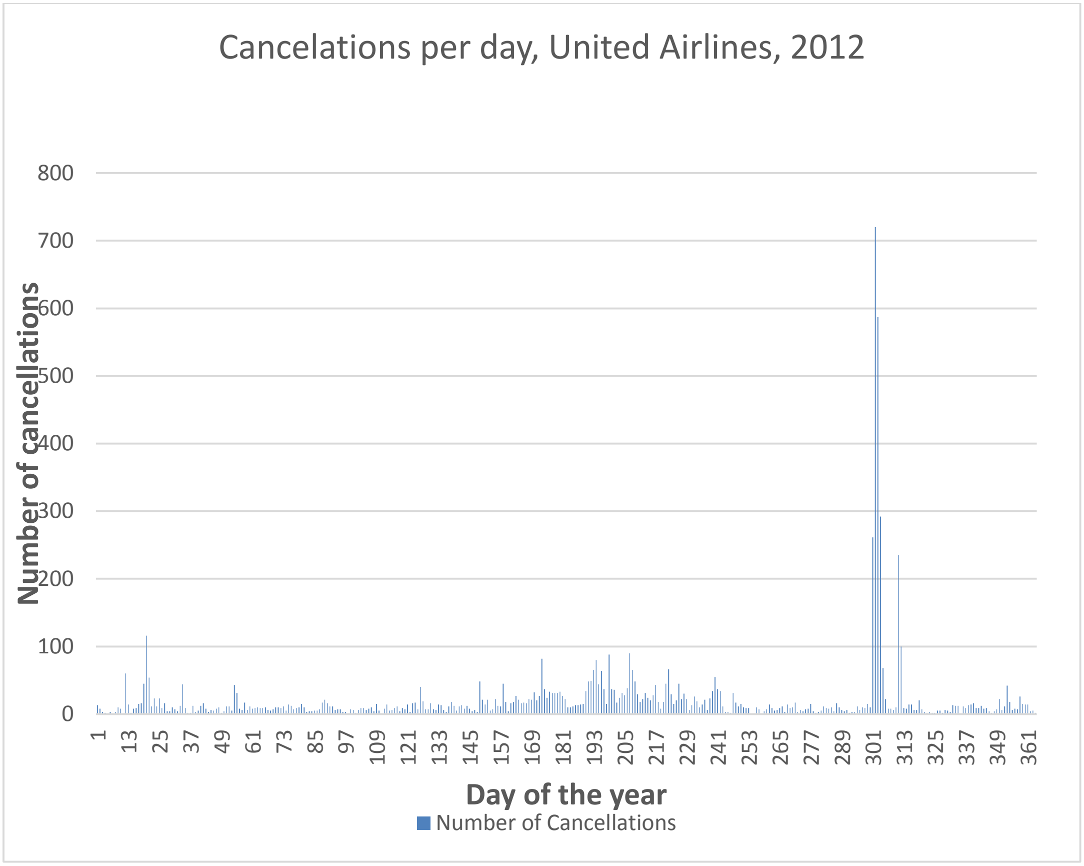


Figure 2 United Airlines, 2012, Cancellations per day

Figures 3-6 show the number of cancelled flights daily for the hubs ORD, DEN, SFO and EWR.

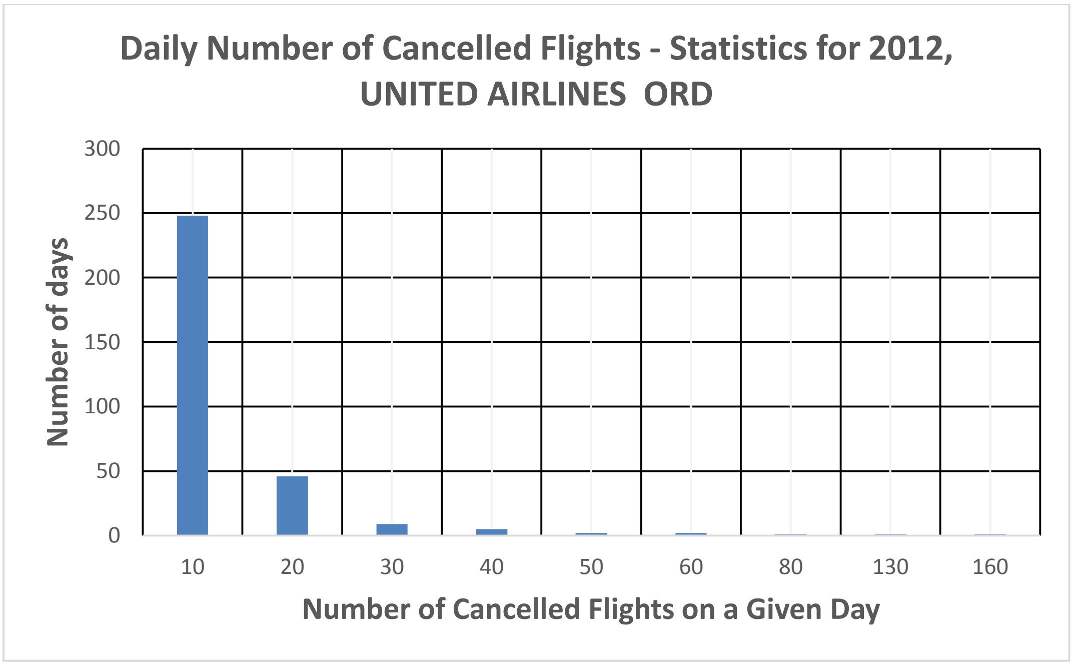


Figure 3 ORD: Number of cancelled flights

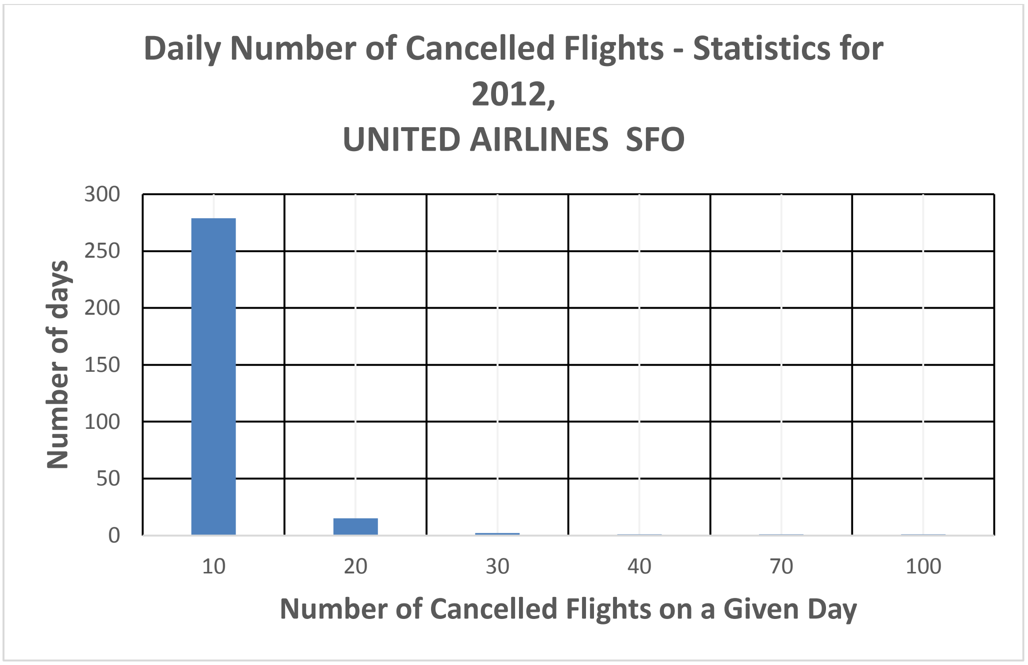


Figure 4 SFO: number of cancelled flights

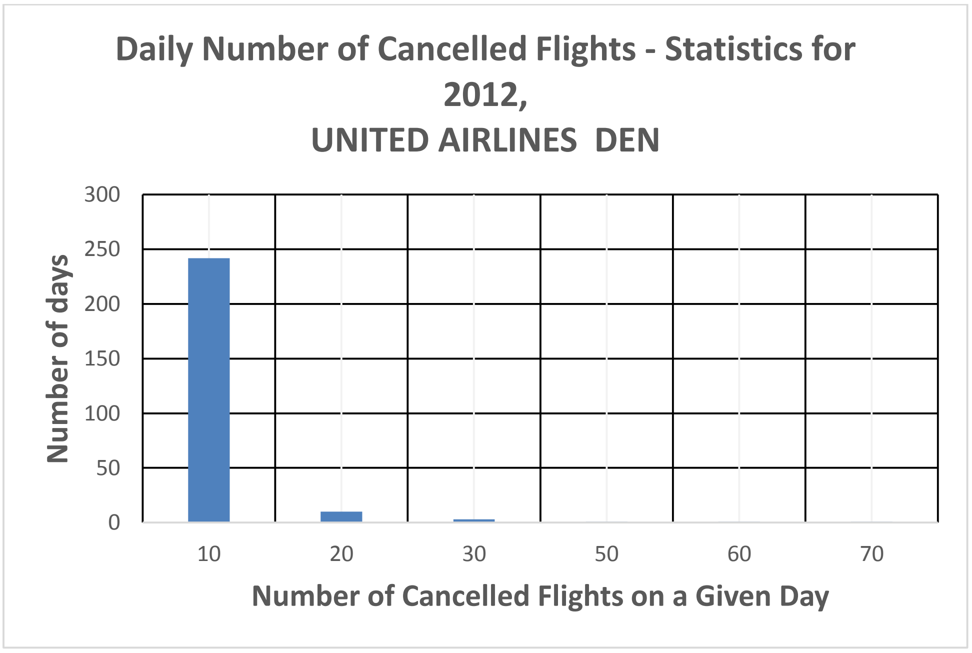


Figure 5 Den: Number of cancelled flights

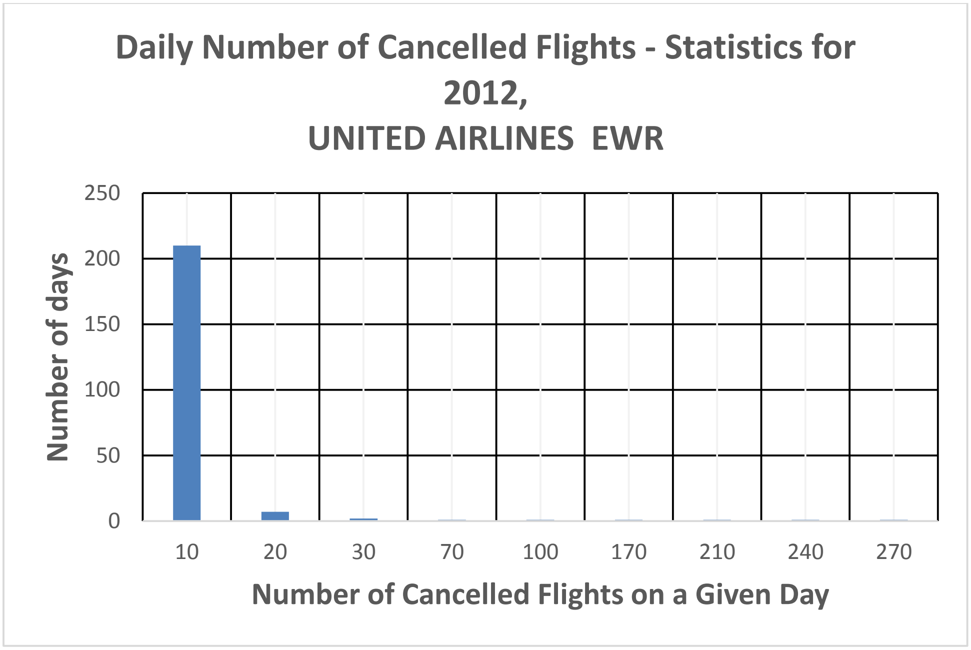


Figure 6 EWR: Number of cancelled flights

The largest number of cancellations per day was through EWR – 265 cancellations, then ORD – 153 cancellations, SFO – 99 cancellations, and DEN – 63 cancellations.

The statistics for Southwest Airlines during the same year look like this (Figure 7):

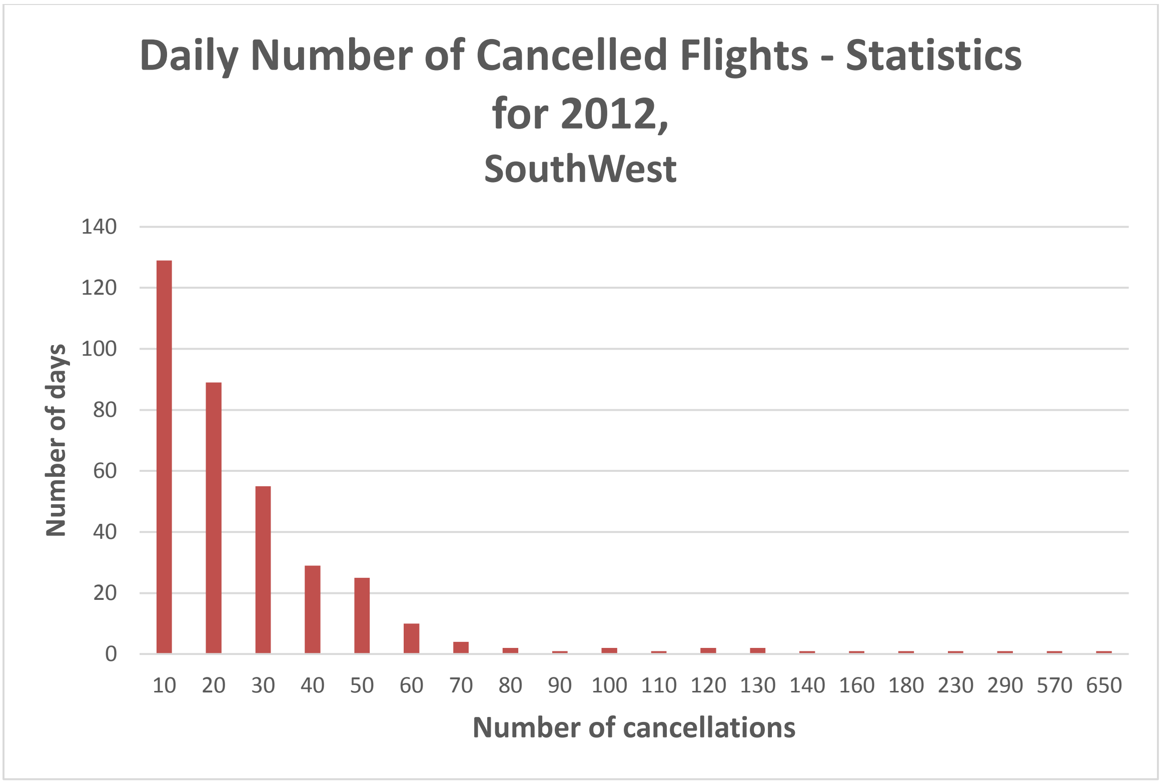


Figure 7 Southwest: Number of cancelled flights

On Southwest, the majority of days had no more than 10 cancelled flights (129 days total). 140 days had more than 20 cancelled flights per day, out of which 14 days had more than 100 cancelled flights each. The largest number of cancelled flights on a single day was 650.

Chicago Midway (MDW) had the largest number of cancellations of all Southwest hubs (Figure 8):

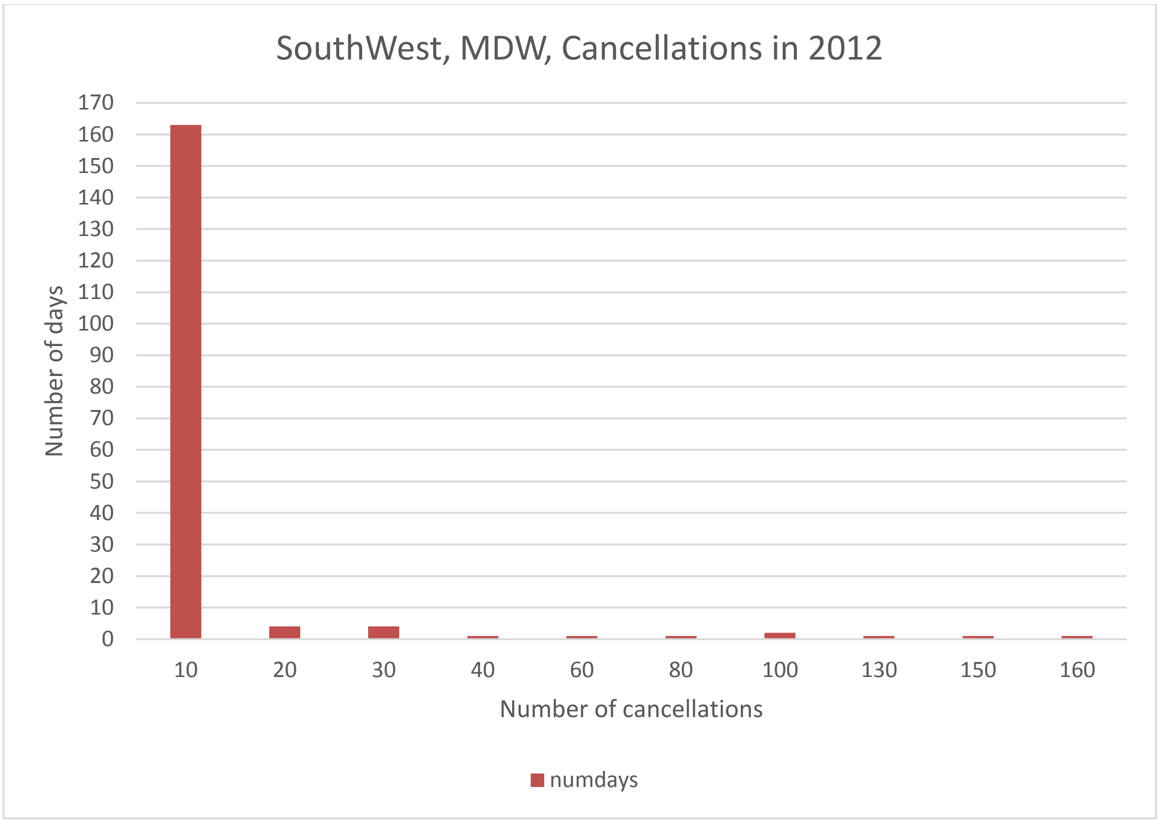


Figure 8 MDW: Number of cancelled flights

There were 163 days during 2012 on which there were no more than 10 cancelled Southwest flights at the MDW hub. Just 16 days had more than 10 cancelled flights, but the number of cancelled flights had a high upper bound of 160 per day. Although these events are not frequent, the impact of the cancellations is large.

For example, on January 20th, Southwest cancelled 96 flights (inbound and outbound MDW) impacting 11,000 passengers. On the same day, 7140 passengers were rebooked, and 3560 passengers stayed overnight as a result of next-day rebooking. 300 passengers couldn't be rebooked at all. If it is assumed that all of the passengers that could not be rebooked on the same day would ask for the airfare refund, the estimated cost to the airline would be $1.5M for this event.

Cancellations may arise due to different circumstance, for example a major snow storm impacting a hub airport, or in anticipation of a labor action, or staffing shortfall or equipment outage. In February 2015. Southwest Airlines grounded 128 aircrafts after they discovered that the airplanes had not received required inspections. In the situation that the imminence of a big disruption is known, an airline has time to find better rebooking options for passengers. The idea of this research is to analyze the use of preemptive rebooking to add more options for the passengers' recovery process. This means that instead of the wait-and-see approach, announcing flight cancellations at the last minute, passengers would be inexpensively contacted in advance of the cancellation to coordinate rebooking, in a way that leverages the willingness of the passengers to accept a flight earlier than the scheduled (now cancelled) flight. Advances in communication technology, in particular broadband wireless access, would allow the possibility for airlines to instantaneously, inexpensively contact all ticketed passengers on a cancelled flight to provide options for rebooking.

For the airline, preemptive rebooking increases revenue by avoiding reimbursement costs for passengers that no longer want to travel, and can improve customer relations. For the passenger, the preemptive recovery capability provides additional options to meet their attendance obligations by arriving in advance.

Concept of Operations

Two important changes have enabled the ability to improve passenger mobility, in the event that irregular operations which lead to large scale cancellations can be foreseen. Those two things are increasingly pro-active planning on the part of airlines, and the development of fast and efficient technologies for communicating to passengers. The former is the result of increased coordination amongst National Airspace (NAS) operational stakeholders, and the second comes in the form of inexpensive and reliable broadband communication services. In this chapter, a new concept of operations will be described: Proactive Preemptive Rebooking Concept.

Traditional concept of operations: Reactive

The traditional concept of operations for rebooking passengers for irregular operation is reactive. This means that the airline waits until a few hours before departure time to cancel the flight. Possible reasons for this include:

- Cancelling as late as possible leaves options open for achieving an on-time flight through aircraft *swapping*, in the case that the cancellation is the result of a maintenance issue.
- Cancelling as late as possible leaves options open for achieving an on-time flight through *slot allocation,* in the case of a ANSP coordinated delay program.
- There was previously no way to quickly and inexpensively contact passengers and re-accommodate them to suit their needs, thus there was no significant advantage to announcing cancellations earlier.

Due to the fact that there is no contact with passengers before they arrive at the airport, passengers can be re-booked on alternate flights by on-site customer service agents, self-service kiosks or on-line websites.

Often, not all passengers can be re-accommodated on the same day, so remaining passengers will be rebooked on flights the following day (overnight passengers). This increases travel costs for all passengers and adds unbudgeted travel costs to corporations whose employees are on business travel. If passengers choose to cancel their tickets, it would result in a loss of revenue for the airline, due to the need to refund airfares. The sequence of events from a passenger view point for the traditional and proposed mode of operations are described below, and summarized in the Event Sequence diagram in Figure 10.

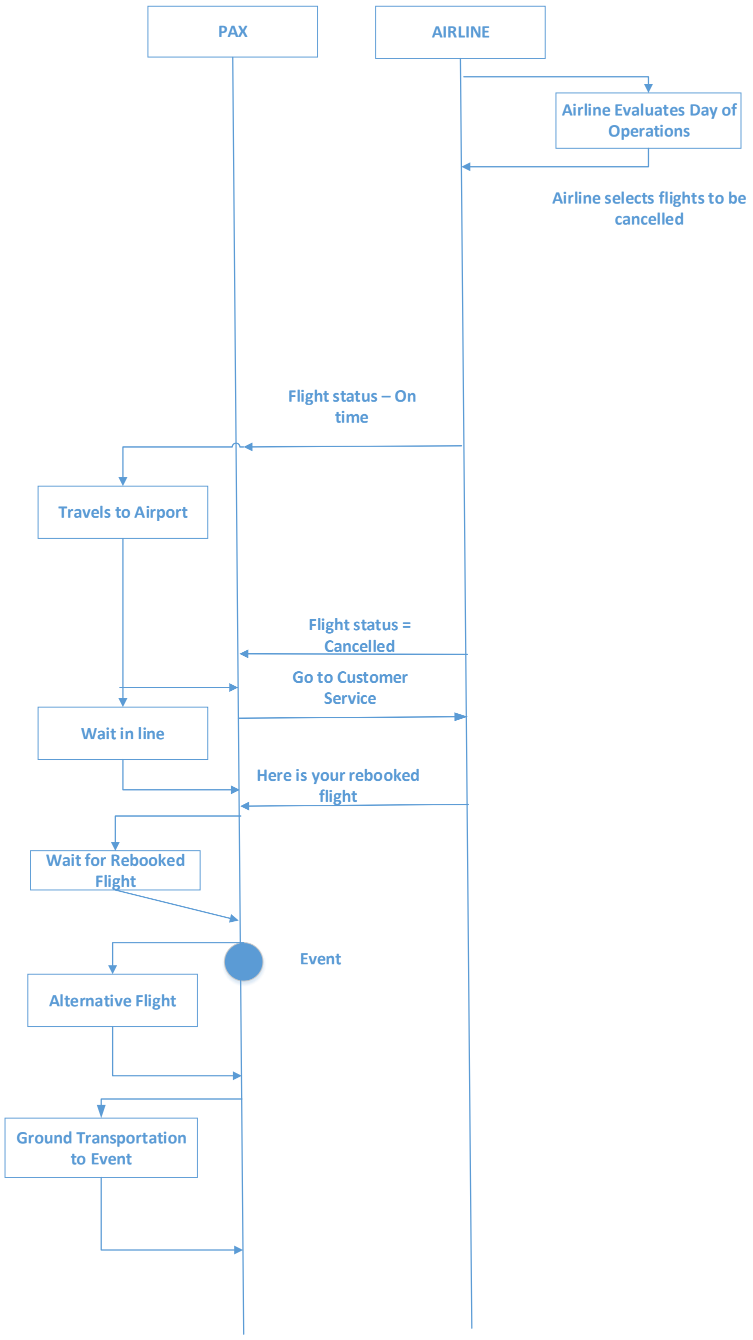

PAX
AIRLINE
Airline Evaluates Day of Operations
Airline selects flights to be cancelled
Flight status – On time
Travels to Airport
Flight status = Cancelled
Go to Customer Service
Wait in line
Here is your rebooked flight
Wait for Rebooked Flight
Event
Alternative Flight
Ground Transportation to Event

Figure 9 - Sequence of events diagram from Post Irregular Operations rebooking: traditional (reactive) Concept of Operations

Post Irregular Operations Rebooking sequence of Events

As shown in Figure 10, the airline starts the day with an assessment of expected operations. On days where flights must be cancelled due to severe weather events or ANSP delay programs, the flights that are likely to be affected (e.g. cancelled or delayed) are identified. Due to uncertainty in the airspace system, airlines keeps their options open by listing the flight status to the ANSP and passengers as scheduled on-time.

As the scheduled departure time of a flight approaches, the airline decides to cancel it. This information is provided to the ANSP as well as to the passenger. The information to passengers is posted in the sense that is made available (i.e. through airport flight status monitors, airline websites, etc.), but not sent directly to the individual passenger.

At this time, most airlines reservation systems unilaterally automatically rebook the passenger based on next available flight. Preference is given to passenger considered high value passengers by Frequent Flyer status or other indications. The initial rebooking is made to expedite airline operations.

At this time the passengers can use customer service agents at the airport, reservation kiosks at the airport, or airline websites to accept or renegotiate the rebooking.

Unless a destination has “shuttle” operations with high frequency, it is unlikely that a passenger on an itinerary with a cancelled flight will be rebooked without a significant amount of time before the next flight. Further, passengers on an itinerary with a cancelled flight late in the day will generally be required to overnight. In some cases the passenger will not be able to arrive at the destination in time for a planned event.

Although many of large scale cancellation events can be forecast (i.e. snow storms, labor issues, and equipment outages), uncertainty in Air Traffic Control (ATC) and in competing airline response to these events obliged airline's to *reactively* rebook passengers *after* the event. However, the advent of Collaborative Decision Making to enable equitable advance planning by the airlines (Wambsganns, 2001), along with the ubiquity of broadband wireless communication between airline and passengers, now enables airlines to *proactively* cancel flights in advance and rebook passengers *before* the cancellation event.

New concept of operations: Proactive Preemptive Rebooking Concept

With increased coordination amongst National Airspace (NAS) operational stakeholders, there are reduced airport and airspace capacity events (e.g. snow storms, labor shortages, infrastructure closures, equipment failures) in which flights can be cancelled by the airlines well in advance of the scheduled departure time. The development of communications, which have become ubiquitous, instantaneous, and inexpensive, has enabled a new concept of operations: proactive, preemptive rebooking.

Collaborative decision-making by ATC and airlines, has changed the way the fight operations are managed from reactive to pro-active. The proactive mode allows airlines to make decisions in advance of events. This means that capacity shortfalls at the runway are now identified at the start of the day. Sometimes the impacted flight operations can be identified in advance (from 48 to 72 hours), in the case of:

- a planned shutdown, for example runway closure, or
- predictable severe weather, for example a heavy snow storm.

As a result, some flight cancellations are known well in advance.

The other significant factor that was mentioned is that modern passengers can be contacted via social networks, text messages or emails, in order to check their willingness to re-schedule flights. In this way, it is possible to better manage large scale cancellation events. The Internet coupled with interactive airline reservation systems and broadband/wireless communication now enables airlines to contact and coordinate changes in reservations at speed and low cost considered unimaginable a decade ago.

These factors have all conspired to provide an opportunity to change the usual operations paradigm and facilitate more choice in addressing irregular operations from a passenger standpoint. The combination of airline passengers travel flexibility with rapid, inexpensive wireless/broadband capabilities to communicate with passengers, and pro-active Flow Management can yield a win-win for airlines, passengers, and corporations that rely on business travel. Airlines can increase revenue by avoiding reimbursement for passengers that no longer want to travel (Airfare Not Refund, ANF) and improve customer relationship (based on trust). Corporates can reduce unexpected costs on corporate travel (Corporate Travel Expanse Savings, CTES). Passengers, have a greater chance to meet their attendance obligations by arriving in advance (lower PTD, decrease in number of overnight passengers and passengers that cannot be rebooked).

A sequence of event diagram depicting rebooking under Post Irregular Operations circumstances, showing the proactive preemptive rebooking concept, is shown in Figure 11.

PROACTIVE (PRE-EMPTIVE)

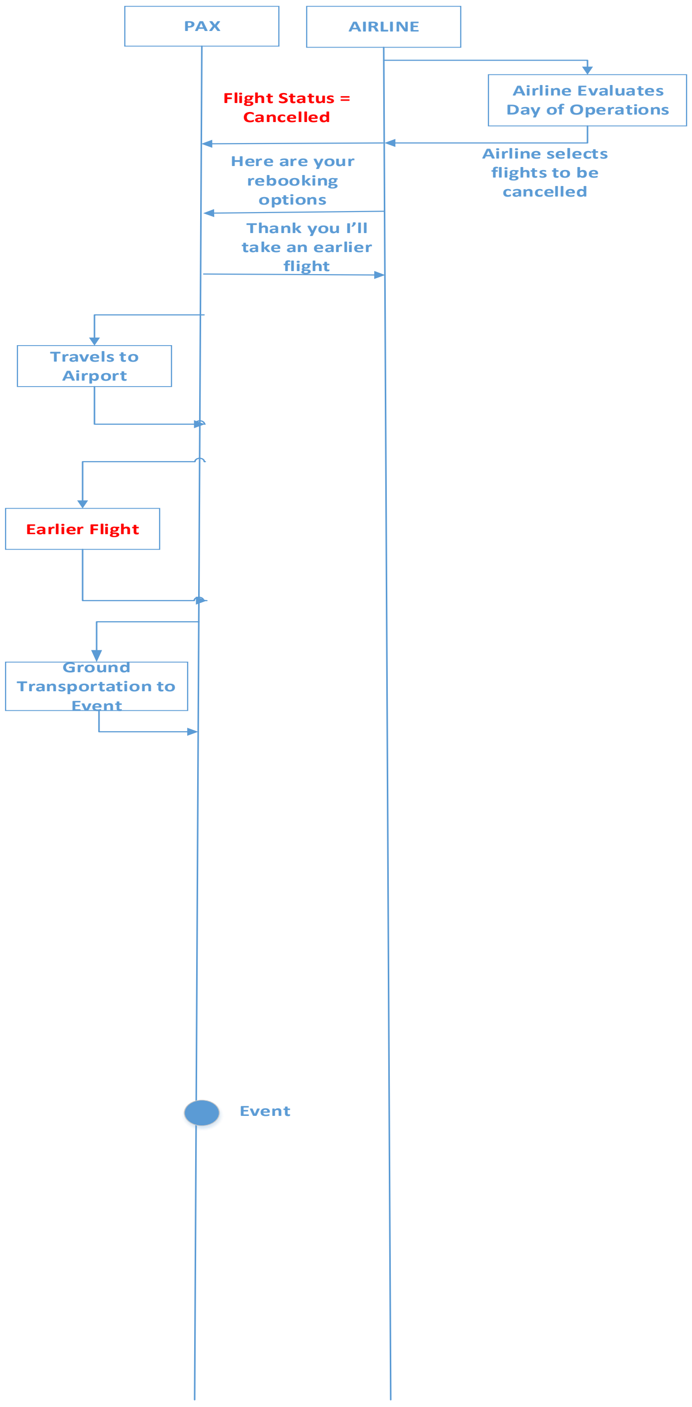

PAX
AIRLINE
Airline Evaluates Day of Operations
Flight Status = Cancelled
Airline selects flights to be cancelled
Here are your rebooking options
Thank you I'll take an earlier flight
Travels to Airport
Earlier Flight
Ground Transportation to Event
Event

Figure 10 Sequence of events diagram from Post Irregular Operations rebooking: the preemptive rebooking scenario

In this scenario, the airline starts the day with an assessment of expected operations. On days where flights must be cancelled due to severe weather events or ANSP delay programs, the flights likely to be affected (e.g. cancelled or delayed) are identified. Due to the level of collaboration and planning available from the ANSP, the airline immediately lists the flight status to the ANSP and passengers as cancelled.

At this time, the airline proactively contacts the passenger and offers preemptive rebooking options. The passenger selects the option that works best, *under the circumstances*, for the passenger. The passenger adjusts their personal schedule and is able to take advantage of the proactive rebooking and arrive on-time for their event at the destination.

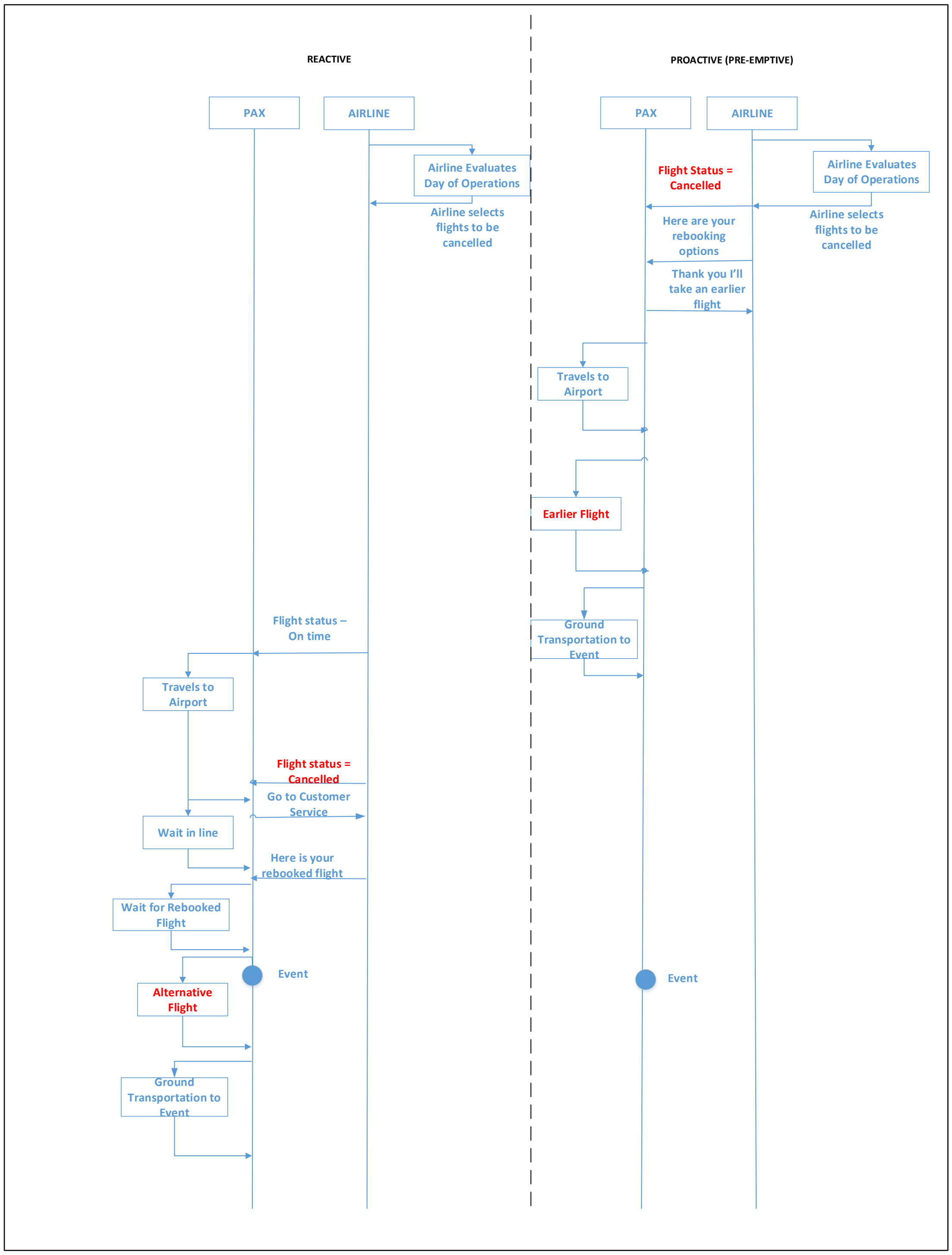

REACTIVE
PROACTIVE (PRE-EMPTIVE)
PAX
AIRLINE
Airline Evaluates Day of Operations
Airline selects flights to be cancelled
Flight status – On time
Travels to Airport
Flight status = Cancelled
Go to Customer Service
Wait in line
Here is your rebooked flight
Wait for Rebooked Flight
Event
Alternative Flight
Ground Transportation to Event
PAX
AIRLINE
Flight Status = Cancelled
Airline Evaluates Day of Operations
Here are your rebooking options
Airline selects flights to be cancelled
Thank you I'll take an earlier flight
Travels to Airport
Earlier Flight
Ground Transportation to Event
Event

Figure 11 Sequence of events diagram from Post Irregular Operations rebooking: Comparison between traditional and preemptive rebooking

Research Question: Preemptive Rebooking

There are five questions which this research intends to answer:

1. Does preemptive rebooking solve the problem of providing a suitable number of rebooking options? This is defined in terms of the fraction of passengers who are accommodated on preemptive flights. If passengers accept preemptive rebooking at a rate of 70%, then to be considered a satisfactory alternative, there must be at least enough preemptive seats available to rebook them on preemptive itineraries.
2. Is preemptive rebooking financially beneficial for airlines? If the assumptions is that for every preemptively rebooked passenger the airline saves $377 (DOT, 2012), the question is how much the airline will save in recouped revenue. It is assumed that in addition to recouped revenue, airlines employing preemptive rebooking will benefit through building a stronger trust relationship between the airlines and passengers.
3. Is preemptive rebooking financially beneficial for Corporates? Can Corporates reduce their loss in cost for overnight stays, transportation, and similar travel expenses that would be required by their travelling employees in case of cancellations?
4. Is preemptive rebooking financially beneficial for passengers? The costs to the passengers are modelled in terms of lost costs due to trip delays, overnight stays, and itineraries which cannot be rebooked. The passenger costs can be reduced through improving PTD, improving timeliness for private or business events, and lowering number of overnight passenger and passenger who cannot be rebooked. So if there is a net reduction in costs due to reduced delays, overnight stays, and the number of unbooked passengers, then preemptive rebooking is considered financially beneficial to passengers.
5. Is there an advantage to rebooking preemptively further than a day in advance (Same day early, Previous day)? Define the threshold for preemptive rebooking in sense the necessary time frame to rebooked passengers in advance.

Research Hypothesis

To answer the five research questions, the null hypotheses below have been formulated per question. Each must be false for the corresponding questions to be answered in the affirmative.

H_0: Preemptive rebooking does not accommodate more than 10% of passengers on cancelled flights (Preemptive passengers / total cancel passengers < 10%).

H_0: Preemptive rebooking does not result in additional revenue to airlines.

H_0: Preemptive rebooking does not result in additional saving to Corporates.

H_0: Preemptive rebooking does not result in reducing PTD and the number of overnight and passenger which cannot be rebooked.

H_0: The number of additional passengers accommodated by adding an additional day of preemptive rebooking is greater than 25%.

Validation Plan

This thesis examines the feasibility of preemptive rebooking and the benefits to airlines and passengers. Historical data for flight statuses and passenger booking is available from the US Department of Transportation's Bureau of Transportation Statistics. This data includes airline on-time performance, flight delays, and cancellations from the Airline Service Quality Performance (ASQP) System; aircraft inventory and seat capacity from the Schedule B-43 Aircraft Inventory database; monthly aggregate airline load factors from the Form 41 Schedules T-100 database; a 10% ticketed sample showing passenger demand and route information from the Airline Origin and Destination Survey (DB1B); historical and future flight schedules from the Official Airline Guide (OAG); and on-time arrival data for non-stop domestic flights from the Airline On-Time Performance (AOTP) database.

The combined data is a base for computing average PTD for a day's flight schedule, to serve as a reference. Any model that is developed will produce its PTD estimate, which can be then compared to the historical reference value in order to verify the accuracy of the model.

.

2. BACKGROUND AND LITERATURE REVIEW

Airlines provide rapid, safe and affordable transportation of passengers between their origins and destinations. This service is critical to the national economy. The *reliability* of this service can be expressed by the difference between passengers' ticketed and actual arrival time.

A single aircraft flying from a departing airport to its next arrival destination is called a *leg*. Every leg flown by an aircraft is identified by a *flight number*, which is an identifying number consisting of one to four digits, with an optional alpha suffix code. One or more consecutive legs by an aircraft using the same *flight designator* (airline code, flight number, and operational suffix) is called a *flight*.

Every flight has a number of passengers, and each passenger has an itinerary. An *itinerary* is the combination of all portions of the passenger's trip, from origin to destination, which may include one or more flight legs. The portion of a flight from the point when a passenger boards to the point when he disembarks is called a *flight segment.* Flight segments separated by gaps may still be part of the same itinerary, and a single flight may have passengers with many different itineraries. For example, a flight traveling from ORD airport (Chicago) to SFO (San Francisco). In addition to the passengers traveling from Chicago to San Francisco, the flight may contain passengers with itineraries that take them to other cities, via San Francisco.

During the course of the day, any airline will encounter problems, including delays and cancellations of flights. A delay or cancellation may cause a passenger to miss a subsequent flight in their itinerary. This is called missed connections. A passenger who has missed a flight due to a misconnection can have their reservations changed to put them on the next available flight in the direction of their destination, a practice which is known as *rebooking*.

The term *flight status* is used to denote the state of the flight with respect to its scheduled time. It could be on time, delayed or canceled. A flight which arrives within fifteen minutes of its scheduled time is still considered on time, but longer than that is considered a delay.

Every flight's status could be *on time*, *delayed* or *canceled*. These flight statuses can apply to any flight, but within a flight cycle, subsequent flights can be affected by the status of the previous one. It means that if a flight is delayed, it makes it more likely that the next flight will also be delayed. A flight that is canceled surely means that the next one must also be canceled. If a flight is on time, the next one may also be on time, but it could still be delayed or canceled.

A flight with a particular tail number leaves an origin location (spoke) and reaches a destination (hub), and after some time, the next flight with the same tail number leaves the destination (hub) and returns to the origin of the round-trip (spoke). This refers to one flight cycle. During a day it is possible to have one or more flight cycles.

In this dissertation the reliability of service will be measured by *passenger trip delay* (PTD), which is the difference between the ticketed arrival time and the actual passenger arrival time. In general, less reliable service means more flight disruptions, which tends to lead to greater delays reflected in the PTD.

Disruptions

During a day different events, caused by airline resource shortages or airport and airspace capacity shortages may lead to irregular operations. Airline Operations Control Centers (AOCCs) are responsible for safety and managing of operations, communications between Air Navigation Providers and the other airlines, and disruption recovery.

Controllers in AOC respond to these disruptions in different ways, e.g. reassigning resources, or adjusting schedules. The schedule recovery process recovers aircrafts, crews and passengers from disruptions. Although there is similarity between flight and passenger itinerary disruptions, there are also important differences, as shown in the next table:

Table 1Types of disruptions

Flight Disruption	**Itineraries Disruption**
Delayed	Delayed flights
Cancelled	Cancelled flights
Diverted	Missed Connections
	Over-booking
	Diversions

Passengers itineraries can be disrupted by delayed or cancelled flights, but also by diversions, over-booking and missed connections.

Recovery Process

Previous research describes methods for addressing the general class of irregular operations including: robust scheduling (Clarke, 1998; Bratu, Barnhart, 2006; AhmadBeygi at al, 2010; Chiraphadhanakul, Barnhart 2013), and aircraft assignment (Eggenberg Salani, Bierlaire, 2010; Jafari, Zegordi, 2011). The effects of passenger rebooking has been analyzed by Yan et.al. (submitted 2015). Pre-emptive passenger rebooking for a possible missed connection for a forecast single delayed flight was discussed in McCarty (2012).

When the disruptions occur, scheduled operations may be adjusted in a several ways (Belobaba et al 2009). This includes delaying departures or postponing departure times (either to wait until an aircraft is ready, or to accommodate delayed passengers), canceling flights, rerouting planes, *swapping* (reassigning) aircraft between flights, and rebooking disrupted passengers. Any adjustments must be done within the framework of existing rules and regulations. Adjustments should also take into account estimated gate-to-gate ready times for flight legs.

Cancellations and delays may occur for a number of reasons, including mechanical problems, absent crew, disruptions to other flights or earlier legs of a flight, or airport capacity issues (especially due to weather), and even because of missed flight inspection. Southwest grounded 128 airplanes in February 2015, when discovered that they had not finished the required inspection by the deadline.

Flight cancellations are often used to bound the propagating effect of the delays to the rest of the system. When canceling a flight, the number of aircraft at each airport should stay roughly the same. The reason is that it is important to have enough aircraft to service future flights. For this reason, cancellations are often done in pairs (two legs or two planes). To avoid overnight delays, the tendency is to cancel a morning leg rather than an evening leg. Cancellation decisions may also take into account aircraft crew. Since missing crew may prevent an aircraft from taking off, an aircraft is less likely to be canceled if the crew has connections on multiple different aircraft than if the crew stays together for their flights.

In the event of a disruption, the plan for recovery from the disruption may include several objectives, often based on minimizing costs such as spare aircraft, reserve crews, recovery time, or passenger time and loss of goodwill. Recovery plans are often done in phases, where recovery of planes is top priority, followed by crew. Passenger rebooking is handled only after flights and crews are accounted for. The sequential process is intended to speed up recovery decisions, and decision support tools tend to reflect the process by focusing on single phases of the process.

During the aircraft recovery phase decisions will be made involving flight leg cancellations, delays and aircraft rerouting. The goal of the aircraft recovery phase is to ensure that aircraft are present to service future flights.

In the second phase, the crew recovery phase, crews will be assigned to uncovered flight legs, either by reassignment or by calling in reserve crews. Crew can be flown as passengers in order to ensure that they are available where they need to be, or crew can be reassigned from another flight to fill in for the missing crew. A third option, using reserve crews, is a more expensive option, because both the reserve crew and the original crew need to be paid.

Finally, disrupted passengers will be rebooked. Because of the fact that a disrupted aircraft may have passengers with a number of different itineraries, a number of passengers are likely to have misconnections. According to a study by Bratu and Barnhart (2005), passengers who missed connections due to delays were *very likely* to be given the best (minimum delay) alternative to their original itinerary, whereas those that missed connections due to cancellation were about *half as likely* to be given the best alternative. This is due to the number of people needing to be rebooked in the event of the cancellation.

Schedule recovery

Schedule recovery should minimize cost, including the estimated cost of: reserve crews, spare aircrafts, passengers' recovery, loss of good will and time. This process is sequential: first aircrafts, then crews and in the end passengers. Dividing the task speeds up recovery, but the big problem remains making decisions in uncertainty.

Airline schedules are optimized under an assumption of normal weather operating conditions, with slack time largely stripped from the schedule to maximize resource productivity. This leads to more difficult handling during disruptions (Lan et al., 2006).

Slack time

Slack time is defined as "additional time allocated beyond the expected time required for each aircraft connection, passenger connection, or flight leg" (Chiraphadhanakul, Barnhart, 2013). This means that lack of slack time leads to increased difficulty in recovery from disruption. However, maximizing resource utilization can result in plans that minimize ground time as much as possible. Chiraphadhanakul and

Barnhart proposed a flight schedule adjustment model that strategically re-allocates existing schedule slack to minimize some proxy of expected delays or disruptions, and introduced the notion of *effective slack.*

Passenger Rebooking

In most cases, recovery models deal primarily with aircraft, crew, and flight schedule recovery, in that order of priority. Recovery models developed by Bratu and Barnhart (2006) plan decisions to adjust the schedules of aircraft, crew, and passengers in response to delays and disruptions, in order to optimize passenger and operating costs. One of the models focuses on passenger delay costs (PDM), while the other focuses on passenger disruption costs (DPM). The models reduce delays and disruptions noticeably in a simulator, over a span of 3 days of operation under varying levels of disruption. The DPM is fast enough to be used during operations, whereas the PDM is too slow to be used in real-time during the single day window which is needed to make decisions.

Constraint specific recovery network for solving airline recovery problems

Eggenberg et al. model a problem as a “Recovery network”. In the model, the aircraft, crew and passengers are units. Each unit is associated with a specific recovery network (a set of nodes and arcs). Each feasible recovery method corresponds to a path through the nodes and arcs in the network. Each unit has a resource limit and the resource limit for passengers is the delay of their itinerary. The recovery network is based on a two dimensional coordinate system. The network generation algorithm is a dynamic programming algorithm.

The advantage of this approach is that the units and constraints are separated. Using a column generation algorithm is was possible to compute exact delays and recovery costs.

Solving the Airlines Recovery Problem Considering Aircraft Rerouting and Passengers

Le, Zhan,Wu (2008) proposed the model for aircrafts and passengers, with idea that each passenger must be delivered to his or her destination and to minimize total delay cost. The paper presents an objective function to minimize delay costs, and provides a genetic algorithm approach to finding an approximate solution.

Preemptive Rerouting of Airline Passengers under Uncertain Delays

Based on an assumption that delays are known in advance, McCarty and Cohn introduced preemptive rerouting of airline passengers, which occur before the length of the delay is realized, to minimize the sum of the delay costs for all passengers on the delayed flight. The idea is to rebook passengers proactively; as soon as it is known that a flight will be delayed instead waiting for missed connections. This model is a 2-stage stochastic model. In the first stage passengers are preemptively assigned to new itineraries. The second stage modifies itineraries for passengers who still miss connections. The second stage is modeled as a minimum cost model.

Suppose that a passenger has an itinerary consisting of two flights:

Flight A from EWR to DTW, departing at 12:00pm and arriving at 2:10pm.

Flight B from DTW to LAX, departing at 3:40pm and arriving at 6:25pm.

A mechanical delay occurs on Flight A, from 45 minutes up to 3 hours. Other itineraries that have available capacity:

Itinerary 1:

Flight C from EWR to CMH, departing at 12:30pm and arriving at 2:30pm

Flight D from CMH to LAX, departing at 5:30pm and arriving at 8:00pm

Itinerary 2:

Flight A from EWR to DTW, departing at 12:00pm and arriving at 2:10pm

Flight E from DTW to LAX, departing at 7:10pm and arriving at 9:55pm

If the delay on the first flight is 45 minutes, then the passenger will make the second flight and reach the destination on time. If the delay is three hours, it means that the second flight will be misconnected, and the new itinerary uses Flight E, arriving at 9:55pm.

If the passenger proactively switches to Flight C when the delay on her first flight is discovered, the passenger will arrive at the destination at 8:00pm, 1 hour and 35 minutes later than planned, rather than the 3 hours and 30 minutes that would result from misconnection.

McCarty and Cohn's results showed that the model reduced the length of passenger delays, but the model takes into account just one delay, which is often not a realistic situation. In many cases there is a combination of delays, and sometimes just one delay can product other, concurrent delays. The model does not scale well to large model.

Passenger Trip Delay

Passenger Trip Delay (PTD) was defined as the difference between scheduled arrival time and actual arrival time. This definition is derived from the definition of *Passenger trip time* which is the period of time between the scheduled gate departure time to the scheduled gate arrival time. Passenger trip time is not flight time. It includes flight delays due to cancellations, missed connections and diversions (Sherry, Wang 2007).

Trip delays are a function of the number of passengers on itineraries in the time-space network of a flight. An itinerary is the sequence of flights taken by a given passenger in traveling from their Origin to Destination. An itinerary can be a direct itinerary, which consists of exactly one flight, or a connecting itinerary. One flight may have passengers with both direct and connecting itineraries on board.

PTD on direct and connected itineraries experience trip delays due several types of situations, shown in the table below.

Table 2 Direct vs Connecting Itineraries, trip delays

Passengers on *Direct Itineraries* (Origin-Destination) :	Passengers on *Connecting Itineraries* (Origin-Hub-Destination):
Delayed flight (Origin to Destination) Rebooking due to cancelled flights (Origin to Destination) Diverted flights (Origin to Destination) Denied Boarding (Origin to Destination)	Delayed flights (Hub to Destination) Rebooking due to Cancelled flight (Origin to Hub) Rebooking due to Cancelled flight (Hub to Destination) Rebooking due to Missed Connection (Hub) Diverted flights (Hub to Destination) Denied Boarding

When a flight is disrupted the associated itineraries will be disrupted as well. This relationship between flight and itinerary disruptions is shown in the next table.

Table 3 Relationship between flight and itinerery disruptions

Itinerary status	Direct Itinerary	Connecting Itinerary	PTD function
On time	Origin – Destination flight on time	Origin-Hub flight on time/delayed/diverted with connection time, AND Hub-Destination flight on time	N/A
Delayed	Origin-Destination flight delayed	Origin-Hub flight on time/delayed/diverted within connection time AND Hub-Destination flight on time	Magnitude of flight delay
Passenger rebooked due to Missed Connection	N/A	Origin-Hub flight delayed/diverted and NOT within connection time, AND rebooked Hub-Destination flight on time/delayed/diverted	Availability of seats on later flights and frequency of flights from Hub-Destination
Passenger rebooked due		Origin-Destination flight cancelled AND rebooked	Availability of seats on later

to cancelled flights	Rebooked Origin-Destination flight on time/delayed/diverted	Origin-Destination flight OR Origin-Hub-Destination flight on time/delayed/diverted	flights and frequency of flights from Origin-Destination, Origin-Hub-Destination and Origin-Hub1-Destination
		Origin-Hub flight on time/delayed/diverted BUT within connection time AND Hub-Destination flight cancelled and rebooked	
Diverted	Origin-Destination flight diverted	Hub-Destination flight diverted	Magnitude of delay due to diversion

Passenger itinerary data is not publicly available, for privacy and security reasons. However, a fractional sampling of itinerary data is available in public databases for analysis purposes. In practice, an airline wanting to continue experiments for its own purposes would have complete itinerary information on its own passengers.

This dissertation focuses primarily on passengers disrupted due to cancelled flights.

Metric for Passenger On-Time Performance

Sherry and Wang (2007) described a method for computing PTD for all hubs and airlines, but for direct flights only. The statistics were generated from the Airline On-Time Performance (AOTP) Database. Flight delays are computed based on the Computer Reservation Systems (CRS) scheduled gate arrival time.

The flight delays are computed as follows:

- 15-OTP On-Time Flights Percentage: For each route, the number of on-time flights (with less than 15 minutes) is summed and divided by the total number of scheduled flights on that route.
- Average Magnitude of Flight Delays: For each route, the flight delays for delayed flights (with flight delay more than 15 minutes) are summed and divided by the total number of delayed flights on that route, yielding the Average Magnitude of Flight Delays for the route.
- Average Worst-Case Magnitude of Flight Delays: For each route, the top 5% of flights by flight delay are computed, and the delays are summed and divided by the total number of flights in the top 5%.

The Total PTD (TPTD) is computed through the combination of 2 algorithms:

6. TPTD due to Delayed Flights: compute delay time for each flight based on the AOTP data, multiplied by the average number of passengers on the flight from the T-100 data-base, in order to derive the passenger delay time for the flight. The TPTD for delayed flights is sum of the passenger delay times for all flights during the specified period.
7. TPTD due to Cancelled Flights: computed based on the assumption that a passenger displaced by a cancelled flight will be rebooked on a subsequent flight operated by the same carrier with the same origin/destination pair. The ability to rebook is determined by the load-factor and aircraft size of the subsequent flights.

The computed delay per passenger comes from the difference between their new arrival time and the time in their original schedule. Passengers who cannot be rebooked incur a fixed delay time.

Modeling Passenger Travel and Delays

Barnhart, Fearing, Vaze (2010) developed the model to analyze the causes and costs of historical U.S. passenger travel disruptions. This was a Low jet model for passenger itineraries. They developed a discrete choice model for estimating historical passenger travel, and extend a previously-developed greedy heuristic for re-accommodation and for estimating the resulting passenger delays.

In the process of passenger allocation, the first step was to join passenger and flight data from multiple sources into a large Oracle database. In the second step the data was processed in a way that would establish the inputs for the allocation (potential itineraries and seats for the passengers). The last step was to develop a discrete choice model for passenger itinerary allocation, and to train and validate the results using a small set of proprietary booking data.

The analysis is done based on the previous work of Bratu and Barnhart (2005). They analyzed a multi-day, multi-carrier rebooking process for passenger delays from 2007. In a simplified regression-based approach they estimated passenger delays directly. This linear regression model was developed to 1) identify critical characteristics of airline networks, schedules, and passenger itineraries that affect passenger delays; and 2) estimate passenger delays directly given public data, thus bypassing the process of passenger allocation and re-accommodation, shown in the following Figure:

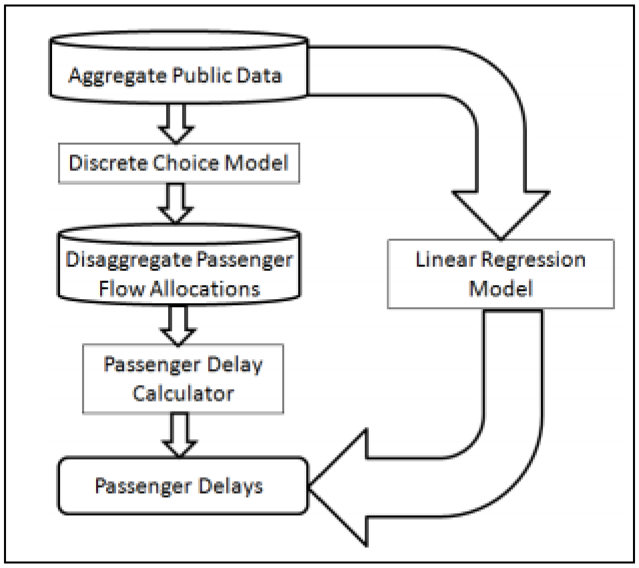


Figure 12 Linear regression model developt by Bratu and Barnhart, which estimate passenger delays (Barnhart, Fearing, Vaze, 2010)

They key findings were:

1. The ratio of average passenger delay to average flight delay is maximum for regional carriers and minimum for low-cost carriers, owing primarily to the cancellation rates and the connecting passenger percentages.
2. Passengers scheduled to transfer in one of 6 airports: Newark (EWR), Chicago O'Hare (ORD), New York La Guardia (LGA), Washington Dulles (IAD), New York Kennedy (JFK) or Philadelphia (PHL), were exposed to the longest average connecting passenger delays.
3. Domestic passenger connections are highly concentrated at the top three transfer airports: Atlanta (ATL), Chicago O'Hare (ORD), and Dallas / Fort Worth (DFW), representing approximately 43.2% of planned passenger connections.
4. Average evening passenger delay is 86.8% greater than the average morning passenger delay.

5. The average passenger delay for the three months of summer and the three months of winter was 56% higher than for the remaining six months. June was the worst month.
6. Delay to the non-stop disrupted passengers depends on the ease of rebooking and is lower for origin-destination pairs with higher daily frequency.
7. The relative benefits of flight frequency in terms of the ease of rebooking depends significantly on load factors
8. Monday and Saturday have the lowest ratio of average passenger delay to average flight delay, and these are the only two days when the ratio is lower than the overall average value for the week.
9. Southwest Airlines has the lowest average passenger delay, nearly 55% lower than its competitors, even though its average flight delay is only 36.3% lower than other airlines

Passenger Itinerary Generation Algorithm

Sherry, Samant and Calderon-Meza (2010) developed an algorithm for estimating PTD and a model for passenger itineraries. In this work, they analyzed the causes and costs of PTD for a particular year. The algorithm was used for all hubs and airlines, including direct and connecting flights.

The Passenger Itinerary Generations Algorithm was described in six steps, as follows:

Step 1: Identify all the possible itineraries. The DB1B is used to generate all the O-D and O-H-D itineraries flown during a month. These itineraries identify the sequence of airports flown by passengers on an airline (e.g. Delta Airlines: DCA-ATL, or Delta Airlines: DCA-ATL-DFW). Itineraries with more than 2 segments account for only 2.5% of the itineraries in 2007, so they are ignored. Similarly, itineraries with a very small number of passengers (less than 0.5 passengers per day) are also ignored.

Step 2: Identify the passenger itineraries. The AOTP is used to generate the individual passenger itineraries for each O-D and O-H-D itinerary in the DB1B. The output is a list of O-D and O-H-D itineraries for each day.

Step 3: Assign Seat Capacity and Total Passengers to each Flight. The T100 database is used to assign values for the number of available seats and the total number passengers on each flight.

Step 4: Estimate Total Passengers for each Day on each Itinerary. Since DB1B represents 10% of the passenger totals, then each DB1B generic itinerary has its total passengers per quarter multiplied by 10, in order to generate an estimate of the total quarterly passengers on the itinerary. The total quarterly passengers is divided by 90 days to estimate the total passengers per day on each itinerary.

Step 5: Estimate the Percentage of Passengers on each Segment of each Itinerary. Sort the itineraries that have an X-Y airport pair by airline and flight number and date. The X-Y could be in an O-D itinerary, an O-H itinerary, an O-H segment or an HD segment on an O-H-D itinerary. Divide the passenger count on individual itineraries by the sum the total passengers that share the X-Y segment. This yields the percentage of passengers on each X-Y segment of the itinerary.

Step 6: Estimate Passengers on each Flight in each Flight Itinerary. The T-100 average passengers per flight (Step 3) and the Percentage of Passengers on each segment of each Itinerary (Step 5) are used to estimate the number of passengers on each flight in each itinerary. On direct itineraries 10 passengers are added to account for a biased underestimation of passengers on DB1B direct itineraries. On connecting itineraries, passengers are allocated evenly between the different itineraries up to the actual load factor from T100. Passenger counts are rounded up as follows: flights with less than 0.3 passengers are assigned 0 passengers, while flights with 0.3 to 0.9 passengers are rounded up.

Passenger Trip Delay Model

The algorithm for estimating Passenger Trip Delays uses the Passenger Itineraries described above and the Airline OnTime Performance (AOTP) data-base. The algorithm is shown in the following figure (Sherry, Samant, Calderon-Meza, 2010):

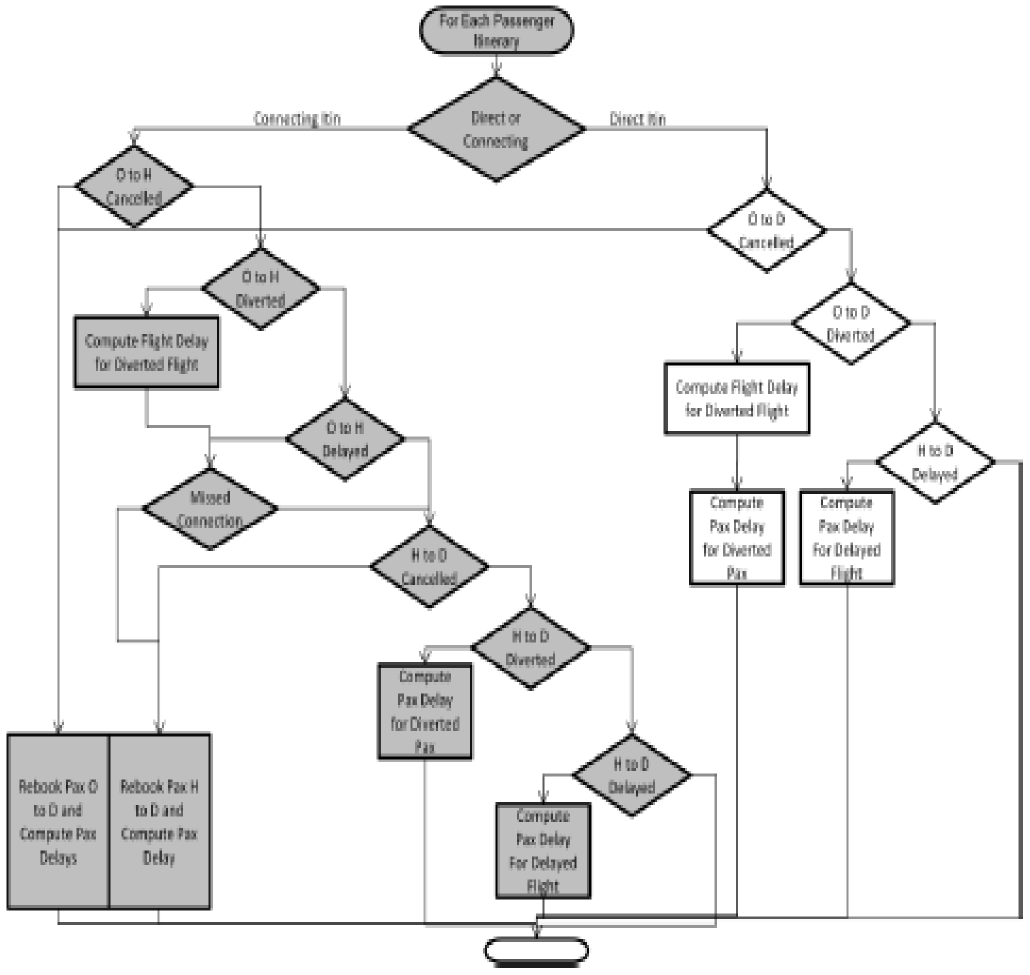

For Each Passenger Itinerary
Direct or Connecting
Connecting Itin
Direct Itin
O to H Cancelled
O to D Cancelled
O to H Diverted
O to D Diverted
Compute Flight Delay for Diverted Flight
Compute Flight Delay for Diverted Flight
O to H Delayed
H to D Delayed
Missed Connection
Compute Pax Delay for Diverted Pax
Compute Pax Delay For Delayed Flight
H to D Cancelled
H to D Diverted
Compute Pax Delay for Diverted Pax
H to D Delayed
Rebook Pax O to D and Compute Pax Delays
Rebook Pax H to D and Compute Pax Delay
Compute Pax Delay For Delayed Flight

Figure 3: PIDA Algorithm (Sherry, Samant, Calderon-Meza, 2010)

The shaded (left) part shows the processing of connecting itineraries, and right part of the figure shows the processing of direct itineraries.

<u>The Origin-Hub flight is cancelled</u>:

Passengers are rebooked on an itinerary by the same airline from the origin to the destination. The rebooked passenger itinerary may be direct, connect through the original hub, or connect through an alternate hub. The Passenger Trip Delay for these passengers is based on the frequency of service and availability of seats. Passengers that cannot be rebooked on the day of the flight are assigned a Passenger Trip Delay of 900 minutes (15 hours), which is an estimate of the delay accrued due to rebooking a flight the following day.

<u>The Origin-Hub flight is not cancelled:</u>

Check to see if the flight was diverted. An estimate of the delay due to diversion is computed based on roundtrip flying time and turnaround time to the closest airports with appropriate length runways. If the diverted flight returns to the Hub airport and misses the connecting flight, the passengers are rebooked from the Hub to the Destination on the same airline. The rebooked passenger itinerary is direct to the destination. The Passenger Trip Delay for these passengers is based on the frequency of service and availability of seats. Passengers that cannot be rebooked on the day of the flight are assigned a Passenger Trip Delay of 900 minutes (15 hours).

<u>Arrival time on the Origin-Hub flight:</u>

If the O-H flight is delayed and arrives less than 30 minutes from the departure of the H-D flight, the passengers are considered to have missed their connection. Passengers who miss connections are rebooked from the Hub to the Destination.

<u>Hub-Destination flight is cancelled.</u>

The passengers are rebooked from the Hub to the Destination. The Passenger Trip Delay for these passengers is based on the frequency of service and availability of seats.

Passengers that cannot be rebooked on the day of the flight are assigned a Passenger Trip Delay of 900 minutes (15 hours).

Diversion on the H-D segment.

If the flight is diverted, the estimate of the delay due to the diversion is computed based on roundtrip flying time and turnaround time to the closest airports with appropriate runways.

Finally, the algorithm checks the H-D flight for delays. If the flight is more than 15 minutes late, the non-zero PTD is computed.

Direct Itinerary,

Check for cancelled flights, diverted flights and delayed flights, rebooking and/or assigning passenger delays as described above (Sherry, Samant, Calderon-Meza, 2010).

Modeling Passenger Trip Reliability

Sherry (2011) introduced a probabilistic model for the operation of hub-and-spoke networks, describing passenger trip reliability metrics. The passenger trip reliability is a complex phenomenon, and it is determined not just by flight on-time performance, but by factors that have nothing to do with flight performance: load factors, frequency of service, and itinerary design.

This model shows that flight delays are one of four factors that affect passenger trip reliability. An important part are structure of the airline space-time network, fleet mix, and airline revenue and competitive strategies.

Factors which have a significant impact on passenger trip reliability metrics include:

1. Changes in aircraft size, and revenue management improvements that increase load factor (the largest impact on passenger trip reliability metrics).
2. Banking structure and frequency of service.
3. Shifting itineraries (lowest impact).

A method for quantifying travel productivity for corporate travel managers

Corporate Travel Managers do not plan for travel disruptions, and indirect charges due to disruptions are not budgeted and not provided in available data sets. Disruptions will thus impact the magnitude of lost billable revenue, which is not known. Sherry (2014) described a method for providing Corporate Travel Departments travel disruption statistics and their impact on revenue and profits.

The paper describes the implications for Corporate Travel Management (CTM), including the possibility to improve productivity strategies, corporate travel and indirect budgets, contracts with travel providers, and travel insurance.

Handling flight delays and cancellations

Airlines try to be as reliable as possible, but disruptions are an unavoidable phenomena. Sometimes weather, air traffic control problems and mechanical problems can result in cancellations and large delays. In many of these situations, passengers don't need to be rebooked, but in some cases, it is necessary. In the next section, the way that Southwest Airlines and United Airlines handle passenger disruption accommodation shall be described.

Southwest Airlines

Southwest Airlines (Southwest, 2015) will not begin the boarding process if they know that a flight will be delayed at the gate for two or more hours. If the duration of the delay is known the boarding will start 30 minutes in advance of a revised estimated departure time.

If an itinerary is disrupted when the departure is from the passengers' city of origin and Southwest is unable to transport passengers to their destinations as scheduled, passengers will be accommodated on the next flight(s) to their destinations with no charge (if seats are available).

If the passengers miss the last possible flight or connection of the day to their destinations, and if it is the case of circumstances *within the airline's control* (for example due to swapping aircraft or mechanical problems), Customer Service will arrange for overnight lodging at no additional cost. In the case when it is *not within the airlines control*, Southwest offers to assist passengers for overnight lodging, by providing a discounted rate at a hotel or motel at or near the airport. Southwest refunds tickets on its own flights, but it does not pay for tickets from another airline, nor does it match the difference between their fares and higher fares on other airlines.

The Southwest disruption policy is called Southwest Operational Disruption Accommodation (SODA). Under SODA, if a flight does not operate as scheduled, the company would refund the unused portion of the fare, without additional charge. Another option that is provided is to arrange transportation for the passenger on another airline's flight, assuming availability of seating.

In the case of events that may affect departures (inclement conditions, etc.) Southwest offers the opportunity to change travel dates and/or flight times at no additional charge. Even the original flight(s) may operate as scheduled, changing travel dates and/or flight times is voluntary and not required.

In those cases The Southwest Network Operations Control Team anticipates and announces, via a Travel Advisory on its site southwest.com, that the flight schedule will be impacted. Affected passengers can be re-accommodated at no charge provided that:

1. Travel is completed within 14 days of the originally scheduled itinerary and no changes are made to the originally reserved city-pairs: for example, if San Francisco International Airport is affected, passengers can change it to the San Jose or Oakland airports.
2. If the passengers chooses to travel beyond 14 days, they are responsible for any fare difference for the new reservation.
3. If changes are made outside the parameters, the changes are subject to the original fare restrictions and may result in a higher fare.

Customers who have booked a Business Select Fare may change to another flight with Business Select or may switch to an Anytime Fare when Business Select is unavailable.

SODA is a rule under which passengers have to be rebooked in the same buckets: if passengers are rebooking to a flight within 14 days of the event, Southwest will overbook the bucket, even if those seats aren't available and confirm the passengers. If the airplane is sold out, they may standby at no additional charge.

United Airlines

Notification of delays and cancellations (the status of flights) is available from several sources (United, 2015):

1. On flight information screens located within airport facilities.
2. United representatives provide information in airports and onboard the aircraft in a reasonably timely manner as to known delays, cancellations and diversions.
3. Passengers may subscribe to flight status notification subscription service, which allow themselves to be notified of disruptions in advance via mobile device. Passengers provide reliable contact information, including possibly mobile device information, during reservations.

In situations where rebooking is required, United has automated systems which will attempt to confirm passengers on the next United flight that has available seats. Using the original confirmation number, passengers can find out if and how they have been rebooked in several ways, including visiting airport kiosks, using the flight status notification subscription service, contacting United Customer Contact Center, speaking with a United airport representative, or managing reservations from the on-line site. If the next available flight is not within the next several hours (it could potentially be several days in the future), and the trip is severely delayed, passengers' options include:

1. Standby for flights. The passenger is rebooked on a later flight, but an earlier flight with no available seats may exist. Passengers may request to stand by on the earlier flight for no charge. If seats become available, the standby passengers will be inform approximately 15 minutes before departure. If seats are not available, passengers will be placed on a standby list for the next flight, or they may request additional assistance.

2. Using another airport. Flights into or out of nearby airports may be available. Typically, ground transportation and re-route checked bags are services which are not provided.
3. Using another airline. If United delayed or canceled a flight due to reasons which are within airlines control and the delay is more than two hours, United will try to rebook passengers on another airline with which they have a rebooking agreement. When the delay or cancellation is due to reasons which are not within airlines control, United typically does not rebook customers on other airlines.
   At the United hub airports, Chicago (ORD), Denver (DEN), Guam (GUM), Houston (IAH), Los Angeles (LAX), New York/Newark (EWR), San Francisco (SFO), Tokyo (NRT) and Washington, D.C. (IAD), it is less likely you will be rebooked on another airline because other airlines offer fewer flights compared to United.
4. Rescheduling trip. If a flight is delayed two or more hours, and passengers want to postpone or cancel, United may waive any change penalties that may apply to your ticket.

In the case of disruptions within United’s control, United my offer food and beverages. For delays that exceed 4 hours and occur between 10 p.m. and 6 a.m., customers in a connecting city may be offered overnight lodging, depending on the amount of time involved and the location of the hotel. Hotel accommodations may not be provided if transportation is offered to or from a nearby airport.

In the case of disruptions that ware not within United’s control, and depending on the duration of the delay, United may offer food and beverages. If passengers have to stay overnight, United offers distressed passenger rate vouchers for a discounted rate at nearby hotels. However, accommodations will not be refunded.

Data Source

The data used for computing passenger trip delay, including flight schedules, aircraft inventory, on-time performance of airlines and passenger demand, is derived from publicly available data sources provided by the Bureau of Transportation Statistics (BTS).

The lists of datasets described in this section are shown in the following table:

Table 4 List of data sets

Airline Service Quality Performance (ASQP)	Airline on time performance, flight delays, cancellations (BTS, updated monthly)
Schedule B-43 Aircraft Inventory (B-43)	Aircraft inventory: seats, serial number (BTS)
Enhanced Traffic Management System (ETMS)	Traffic surges, gaps on national, local scale Used by FAA (not available in the public domain)
Form 41 Schedules T-100	Monthly aggregate load factors (4 data sets: domestic & international market & segment)
Airline Origin &Destination Survey (DB1B)	10% ticketed sample : pax demand and route info (3 data sets: coupon, market & ticket)
Official Airline Guide (OAG)	Historical and future flight schedules
Airline On-time Performance (AOTP)	On time arrival data (nonstop domestic flights)

1) Airline Service Quality Performance (ASQP) (BTS, 2012a)

The Airline Service Quality Performance System (ASQP) provides information about airline on-time performance, flight delays, and cancellations. It is based on data filed by airlines each month with the Department of Transportation’s Bureau of Transportation Statistics (Office of Airline Information), as described in 14 CFR Part 234 of DOT's regulations. The ASQP system includes data for U.S. air carriers (15 as of April 2011) that have at least one percent of total domestic scheduled-service passenger revenues, and one carrier that currently reports data voluntarily. ASQP data are updated on a monthly basis, approximately 25 days after the end of the month.

2) Schedule B-43 Aircraft Inventory (2012f)

The Schedule B-43 Aircraft Inventory, published by BTS (Bureau of Transportation Statistics) provides an annual list of aircraft in inventory for most airlines including seat capacity, serial number, tail number, aircraft manufacturer, model, total capacity of the aircraft, acquisition date etc.

3) Enhanced Traffic Management System (ETMS) (2012e)

ETMS is the system used by the FAA’s Traffic Management Personnel to predict, on national and local scales, traffic surges, gaps, and volume based on current and anticipated airborne aircraft. ETMS database provides schedule information for all flights with its International Civil Aviation Organization (ICAO) code tracked by the air traffic control. This dataset is not available in the public domain.

4) Form 41 Schedules T-100 (BTS, 2012b):

The Form 41 Schedules T-100 database includes the following four datasets.

a) T-100 Data Bank 28DM *Domestic Market Data*: This file contains data reported by U.S. carriers operating between airports located within the boundaries of the United States and its territories, and contains information on passengers; freight and/or mail enplaned at the origin airport and deplaned at the destination airport.
b) T-100 Data Bank 28DS *Domestic Segment Data*: This file contains data reported by U.S. carriers operating nonstop between airports located within the boundaries of the United States and its territories and contains information by aircraft type and service class for departures performed, available capacity and seats, passengers transported, freight and mail transported, scheduled departures, and aircraft hours ramp-to-ramp and airborne.
c) T-100 Data Bank *28IM International Market Data*: International Market Data are subject to access restrictions. The data fields contain information on passengers; freight and/or mail enplaned at the origin airport and deplaned at the destination airport.
d) T-100 Data Bank 28IS *International Segment Data*: International Market Data are subject to access restrictions. These data fields contain information by aircraft type and service class for departures performed, available capacity and seats, passengers transported, freight and mail transported, scheduled departures, and aircraft hours ramp-to-ramp and airborne.

5) Airline Origin and Destination Survey (DB1B) (BTS, 2012c)

The Airline Origin and Destination Survey (DB1B) is a 10% sample of airline tickets from reporting carriers collected by the Office of Airline Information of the Bureau of Transportation Statistics. Data includes origin, destination and other itinerary details of passengers transported. This database is used to determine air traffic patterns, air carrier market shares and passenger flows. It consists of three data sets.

a) *DB1B Coupon*: provides coupon-specific information for each domestic itinerary of the Origin and Destination Survey, such as the operating carrier, origin and destination airports, and number of passengers, fare class, coupon type, trip break indicator, and distance.
b) *DB1B Market*: provides directional market characteristics of each domestic itinerary of the Origin and Destination Survey, such as the reporting carrier, origin and destination airport, prorated market fare, number of market coupons, market miles flown, and carrier change indicators.
c) *DB1B Ticket*: contains summary characteristics of each domestic itinerary on the Origin and Destination Survey, including the reporting carrier, itinerary fare, number of passengers, originating airport, roundtrip indicator, and miles flown.

6) Official Airline Guide (OAG) (2012g)

The Official Airline Guide (OAG) database stores future and historical flight details and schedules for more than 1,000 airlines and over 4,000 airports.

7) Airline On-Time Performance (AOTP) (BTS, 2012d)

The Airline On-Time Performance database contains on-time arrival data for non-stop domestic flights by major air carriers, and provides such additional items as departure and arrival delays, origin and destination airports, flight numbers, scheduled and actual departure and arrival times, cancelled or diverted flights, taxi-out and taxi-in times, air time and non-stop distance.

3. METHOD AND MODEL

Method of Analysis

To analyze the impact of preemptive rebooking strategies, the PTD Calculator (Sherry et al, 2011) is embedded in a Monte Carlo Simulation (Figure 13). The PTD Calculator, Monte Carlo Simulation, and the Design of Experiment are described in this section.

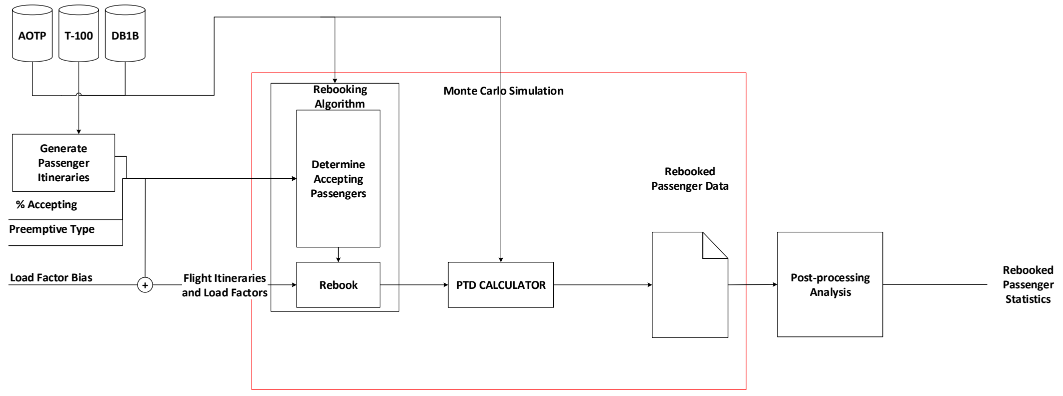


Figure 13 PTD Calculator, embedded in a Monte Carlo Simulation

Passenger Itinerary Delays Algorithm

The "Passenger Itinerary Delays Algorithm (PIDA)" is used to generate statistics for trip delays for each ticketed passenger itinerary (Sherry et. al., 2011). The PIDA takes as an input each individual passenger itinerary. This includes both direct itineraries and connecting itineraries. The scheduled itineraries are compared against the actual performance of the flights associated the itinerary. If a flight is on-time, no passenger trip delay is accrued. If the flight is delayed more than 15 minutes, PIDA calculates the delay. In the case of cancellations or missed connections, passengers are rebooked on flights operated by the same airline or their subsidiaries, using departure times after the scheduled departure of the cancelled or miss connected passenger.

Passengers are rebooked according to the following heuristics. First, passengers are rebooked on flights operated by the same airline or their subsidiaries with departure times after the scheduled departure of the cancelled or miss connected passenger. If the passenger cannot be rebooked on those flights (due to unavailability of seats), the passenger is then rebooked on flights operated by other airlines that depart after the scheduled departure time of the cancelled or miss connected passenger.

Passengers who are not accommodated on the same day are considered “overnight” passengers. These passengers are rebooked on the following day, first on the same airline and its subsidiaries, and then on other airlines. Passengers that cannot be accommodated in this way are considered “passengers not rebooked.”

The ability to be rebooked is dependent on the availability flights from the passenger’s location (i.e. origin airport or hub airport) to the destination, as well as the availability of seats on those flights. In this way, the Seat Size of the aircraft along with the Load Factor (the % seats occupied) determines the ability for passengers to get rebooked.

Model of preemptive rebooking

In the model of preemptive rebooking, passengers are divided into two big groups:

1) *Passengers on cancelled flights*, and
2) *Passengers on not cancelled flights*

Passengers on the cancelled flights are further grouped as:

1) *Passengers that CAN be rebooked*, and
2) *Passengers that CANNOT be rebooked*

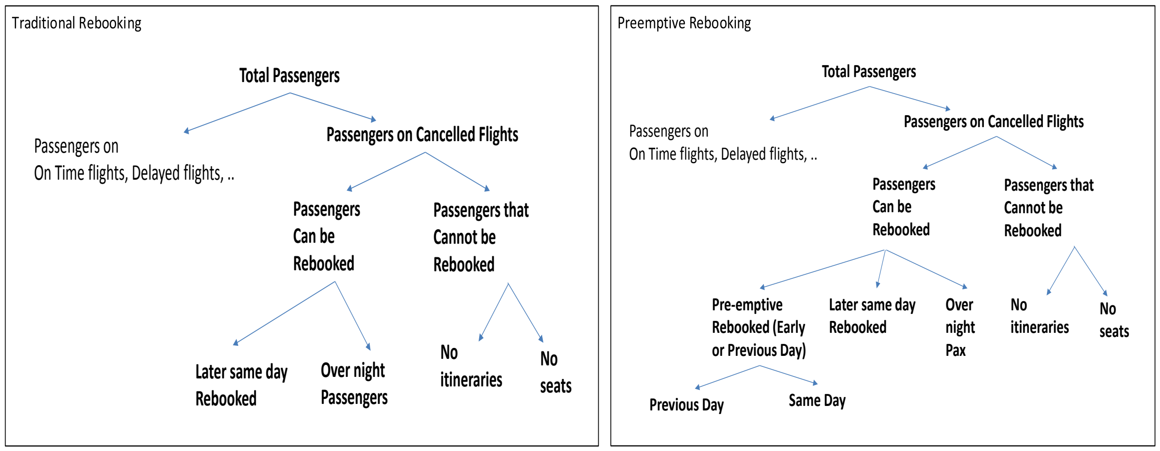
If a passengers cannot be rebooked, it is either because there are no available alternative itineraries, or because the itineraries that exist lack available seats.

Passengers that can be rebooked, in traditional rebooking model, can be rebooked on the *same day later* (after the scheduled departure time), or if there is not available itinerary or seats, on the first available flight on the next day. Those that are rebooked on the following day are considered *overnight passengers*.

In the preemptive rebooking model there are four options for Passengers that CAN be rebooked. They can be rebooked:

1. *Preemptively Same Day.* The passenger is rebooked on a flight earlier in the day, before the scheduled departure time. If there are no such available seats, then the passenger is rebooked later, following the traditional rebooking model.
2. *Preemptively Previous + Same Day.* Like the preemptively same day rebooking option, the passenger is rebooked on a flight earlier that the scheduled departure time. If no alternate flights are available on the day of the original flight, then then alternate flights are sought on the previous day, before resorting to the traditional rebooking model.
3. *Same Day Later.* Describes traditional rebooking, taking place after the scheduled departure time of the original flight.
4. *Next Day*. If there is no available itinerary or seats, the passenger is rebooked on the first available flight on the next day. Such a passenger is considered an *Overnight Passenger*.

The difference between traditional and preemptive rebooking passenger groups is shown in the figure 14.


Traditional Rebooking
Total Passengers
Passengers on On Time flights, Delayed flights, ..
Passengers on Cancelled Flights
Passengers Can be Rebooked
Passengers that Cannot be Rebooked
Later same day Rebooked
Over night Passengers
No itineraries
No seats
Preemptive Rebooking
Total Passengers
Passengers on On Time flights, Delayed flights, ..
Passengers on Cancelled Flights
Passengers Can be Rebooked
Passengers that Cannot be Rebooked
Pre-emptive Rebooked (Early or Previous Day)
Later same day Rebooked
Over night Pax
No itineraries
No seats
Previous Day
Same Day
Figure 14 Passengers' groups in Traditional Rebooking vs Preemptive Rebooking

The methodology for preemptive rebooking is shown in Figure 13.

This section will explain how the rebooking algorithm takes the available itinerary information and uses it to make rebooking decisions.

The rebooking algorithm gathers all disrupted itineraries which require rebooking, and for each one, prepares a list of suitable alternative itineraries. Each itinerary includes the number of passengers using the itinerary, and each alternative has a certain number of available seats.

The algorithm selects disrupted itineraries one by one, and takes all of the passengers that need to be rebooked. The passengers in need of rebooking can be split into groups, depending on whether they are willing to accept preemptive rebooking or not. For as long as passengers remain in need of rebooking, and for as long as alternate flights remain, remaining passengers are assigned to available seats.

First, passengers who have accepted preemptive rebooking are checked on preemptive alternative flights on the same day. Failing in that, rebooking options are checked on preemptive flights on the previous day. After that, any remaining passengers who have accepted preemptive rebooking are merged with the group of other remaining passengers, and rebooking options are checked for same-day flights later in the day. Finally, next day flights are checked. Having completed all of that, any remaining passengers are considered not rebooked, and the algorithm repeats using the next disrupted itinerary.

For example, if a flight is cancelled at 11:36 am, traditional rebooking would rebook passengers on the first available flight after original (cancelled) scheduled departure time (figure 15). With preemptive rebooking they will be rebooked on the first available flight before original (cancelled) scheduled departure time of 11:36, for instance at 10am. If there are no itineraries or seats available between 11:36 am and 6 am, passengers will be rebooked on the previous day (starting from 11pm to 6 am). If there are no itineraries or available seats on the previous day, they will be rebooked first available flight after 11:36 am, the original scheduled departure time.

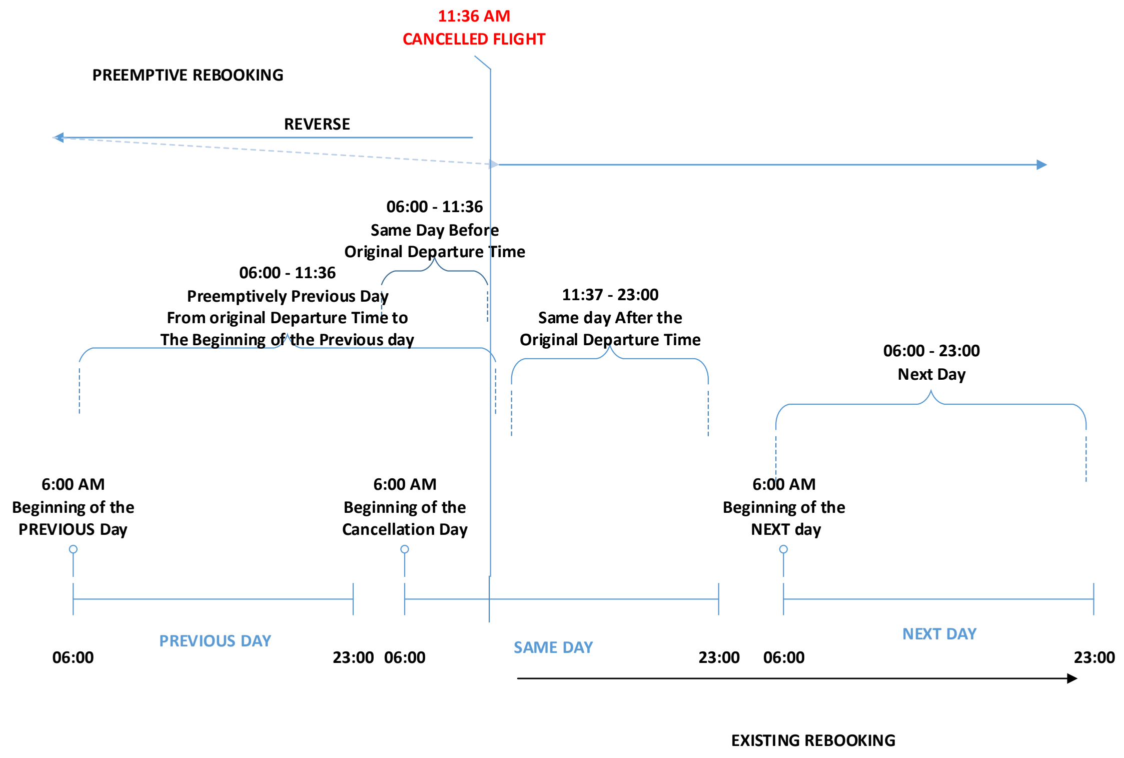
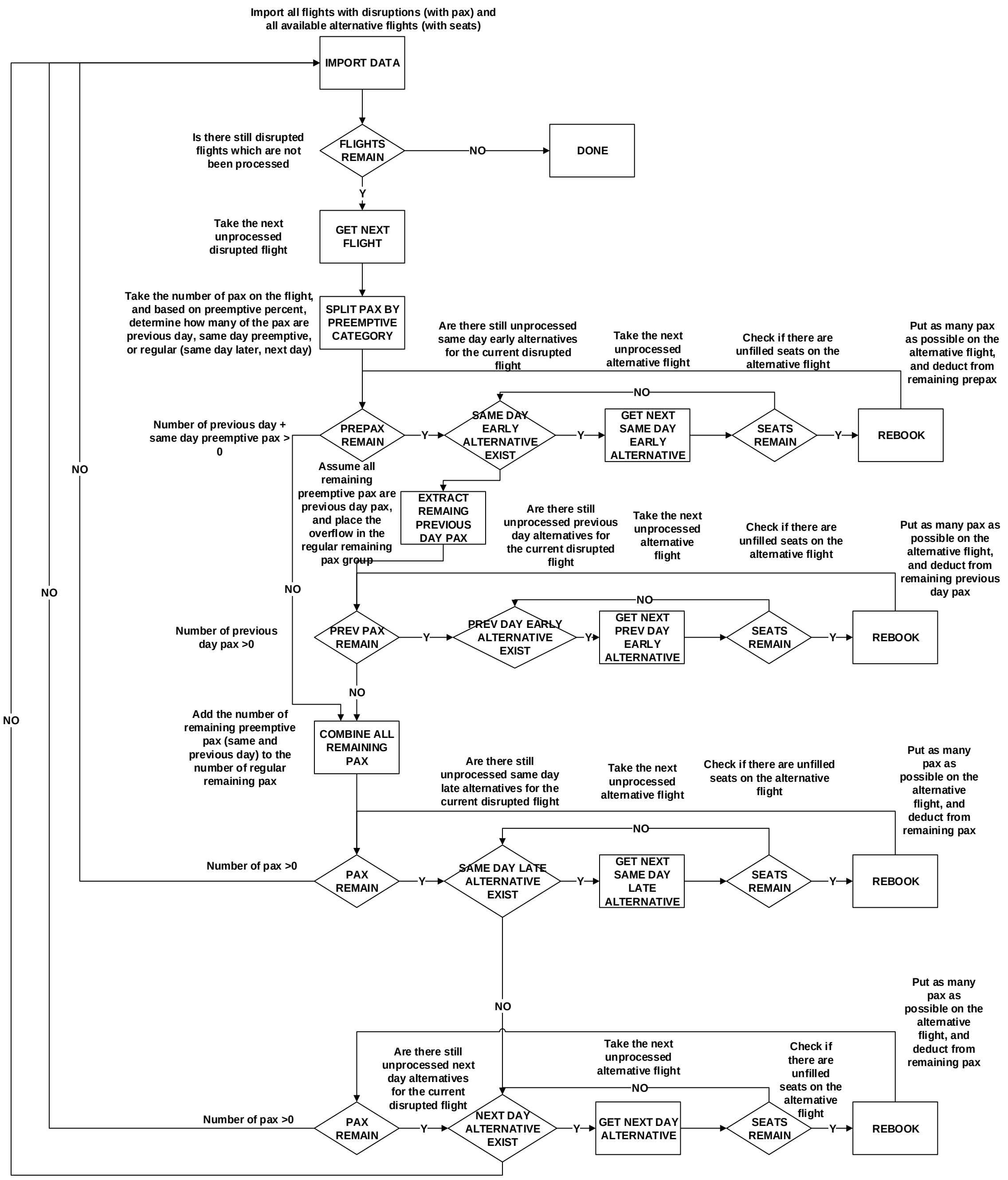


Figure 15 Preemptive rebooking

Preemptive rebooking algorithm

The methodology for preemptive rebooking is shown in Figure 16.

REBOOKING METHODOLOGY



Import all flights with disruptions (with pax) and all available alternative flights (with seats)
IMPORT DATA
Is there still disrupted flights which are not been processed
FLIGHTS REMAIN
NO
DONE
Y
Take the next unprocessed disrupted flight
GET NEXT FLIGHT
Take the number of pax on the flight, and based on preemptive percent, determine how many of the pax are previous day, same day preemptive, or regular (same day later, next day)
SPLIT PAX BY PREEMPTIVE CATEGORY
Are there still unprocessed same day early alternatives for the current disrupted flight
Take the next unprocessed alternative flight
Check if there are unfilled seats on the alternative flight
Put as many pax as possible on the alternative flight, and deduct from remaining prepax
Number of previous day + same day preemptive pax > 0
PREPAX REMAIN
Y
SAME DAY EARLY ALTERNATIVE EXIST
Y
GET NEXT SAME DAY EARLY ALTERNATIVE
SEATS REMAIN
Y
REBOOK
NO
Assume all remaining preemptive pax are previous day pax, and place the overflow in the regular remaining pax group
EXTRACT REMAING PREVIOUS DAY PAX
Are there still unprocessed previous day alternatives for the current disrupted flight
Take the next unprocessed alternative flight
Check if there are unfilled seats on the alternative flight
Put as many pax as possible on the alternative flight, and deduct from remaining previous day pax
NO
Number of previous day pax >0
PREV PAX REMAIN
Y
PREV DAY EARLY ALTERNATIVE EXIST
Y
GET NEXT PREV DAY EARLY ALTERNATIVE
SEATS REMAIN
Y
REBOOK
NO
Add the number of remaining preemptive pax (same and previous day) to the number of regular remaining pax
COMBINE ALL REMAINING PAX
Are there still unprocessed same day late alternatives for the current disrupted flight
Take the next unprocessed alternative flight
Check if there are unfilled seats on the alternative flight
Put as many pax as possible on the alternative flight, and deduct from remaining pax
Number of pax >0
PAX REMAIN
Y
SAME DAY LATE ALTERNATIVE EXIST
Y
GET NEXT SAME DAY LATE ALTERNATIVE
SEATS REMAIN
Y
REBOOK
NO
Put as many pax as possible on the alternative flight, and deduct from remaining pax
Are there still unprocessed next day alternatives for the current disrupted flight
Take the next unprocessed alternative flight
Check if there are unfilled seats on the alternative flight
Number of pax >0
PAX REMAIN
Y
NEXT DAY ALTERNATIVE EXIST
Y
GET NEXT DAY ALTERNATIVE
SEATS REMAIN
Y
REBOOK

Figure 16: Preemptive Rebooking Methodology

This section will show how the rebooking algorithm takes the available itinerary information and uses it to make rebooking decisions. Passengers who cannot be rebooked on the same day, will be rebooked on the next day. Remaining passengers who cannot be rebooked on the next day will be considered as passengers not rebooked. Rebooking depends on:

- The availability flights from the passengers' location (i.e. origin airport or hub airport) to the destination.
- The availability of seats on those flights.

Due to the fact that seat availability is a factor, the Seat Size of the aircraft and the Load Factor (the % of seats occupied) are determining factors in the ability for passengers to be rebooked.

The algorithm selects disrupted itineraries one by one, and takes all of the passengers that need to be rebooked. The passengers needing to be rebooked can be split into groups depending on whether they are willing to accept preemptive rebooking or not. For as long as passengers remain in need of rebooking, and for as long as alternate flights remain, remaining passengers are placed in empty seats.

First the passengers who accept preemptive rebooking are checked on preemptive alternative flights on the same day. Failing in that, rebooking options are checked on preemptive flights on the previous day. After that, any remaining passengers who have accepted preemptive rebooking are merged with the other remaining passengers, and rebooking options are checked for same-day flights later in the day. Finally, next day flights are checked. Having completed all of that, any remaining passengers are considered unbooked, and the algorithm repeats with the next disrupted itinerary.

Code Optimization and Method Contribution

The code used by the Monte Carlo simulator and PTD calculator (Figure 16 above pp 57) was derived from existing code for the PTD calculator without preemptive rebooking.

Previously existing code consists of five steps written in Microsoft SQL which needed to be run consecutively. The code has limited flexibility in terms of parameters such as the dates, number of days, and companies used in the simulation. Code written for this research simplified the existing code into a single step, supported by several stored procedures with configurable parameters. Several capabilities were added to the code, as described below.

Code Contribution

Many of the sections of code which involved creating, populating, and updating tables were moved into stored procedures. This made it easier to modify parameters and call repeated sections of code without redundancy. The remaining code was greatly simplified by observing that repeated sections of code could be combined to reduce redundancy. This especially impacted the DB1B to T100 conversion, as well as the rebooking algorithm itself. The difference is shown in the figure 17.

The code contribution is summarized in the next table:

Table 5 Preemptive Rebooking: code contribution

Preemptive rebooking		
Load factor bias		
Passengers groups:	Passengers who can be rebooked	Previous Day,
		Same Day early,
		Same Day After,
		Next Day (Overnight: Stranded passengers)
	Passengers who cannot be rebooked:	No Itineraries
Trip Time (Scheduled, Alternative, Actual)		
Automated Statistics:	Airfares Refund/Not Refund, / Corporate Travel Expense Saving,	
	Corporate Travel Expense/ Corporate Travel Expense Saving,	
	Passenger statistics	
	Run statistics	
Efficiency of preemptive scheduling		

Debugging

A significant error in the original code was located and corrected. The cursor which processed all alternative options was a static cursor, meaning that all updates to the underlying tables would not be reflected in the data retrieved by the cursor. As a result, changes made by the rebooking algorithm would be ignored as the rebooking algorithm continued to operate. This meant that, for example, a flight that was filled by the rebooking algorithm could be refilled again later. The error was fixed by maintaining the needed data in local variables whenever possible (for example, the number of remaining passengers on a flight is now stored in a local variable after a new flight is read), and dynamic cursors are used when the underlying data can change.

The new code uses a Monte Carlo simulation which did not exist in the original code. In order to implement the simulator, the relevant parts of the code (e.g. step three of the original code) were placed in a loop that iterates the desired number of times.

The ability to process preemptive rebooking was added to the code. This involved adding a loop which would re-run the simulation for each choice of preemptive percent. The original code would consider alternate flights in order. The new code does the same, but early flights are added to the group of flights to be considered. That means that alternatives are considered in the order beginning with flights departing prior to, but on the same day as, the original scheduled departure of the flight, followed by flights departing on the previous day, followed by flights departing after, but on the same day, followed by flights departing on the next day. Additionally, in each group, alternate flights on different airlines were considered after all alternates on the same airline were exhausted. A number representing priority category was introduced into the code, and used to establish the sorting order.

To make it easier to keep track of the number of passengers which could not be rebooked, two virtual priority categories were added, accounting for passengers which could not be rebooked, either due to the lack of available alternative flight options (no itineraries), or the lack of available seats. The new code is more rigorous about counting flights without alternative itineraries, due to the fact that those flights are now included in the alternative table under a priority level which is unused by the rebooking algorithm.

Due to the fact that some passengers rebook preemptively and some do not, a binomial distribution calculator was added to the code, and used to randomly determine the number of passengers which would accept preemptive rebooking. Whenever a new flight is processed, the distribution calculator is be used together with the preemptive acceptance rate, to determine the number of passengers for previous day preemptive rebooking and same day preemptive rebooking. The preemptive alternatives would only be processed for the relevant preemptive passengers, whereas the non-preemptive alternative options would be processed for all remaining passengers (see the model in the previous section).

The ability to adjust a load factor bias was also added to the code. This feature was implemented by adding a parameter which would adjust the number of available seats at the time that the list of alternative flight options was being created.

Another added capability was the ability to compute trip time, including actual, scheduled, and alternative trip time. The trip time capability is supported by adding per passenger multipliers to the alternative options table. The trip time in each category can be determined by multiplying the number of passengers on each alternative option by the multiplier for that option.

The new code works together with a new *statistics script*, which automates the computation of results such as revenue recouped, indirect costs, and the efficiency of preemptive scheduling. To enable a detailed statistics computations, the code now stores a table of results which includes information broken down by Monte Carlo run, alternative priority, date, and type of disruption. The statistics script uses the information from different runs to compute means, medians, and standard deviations of the desired values.

Code sequence diagram

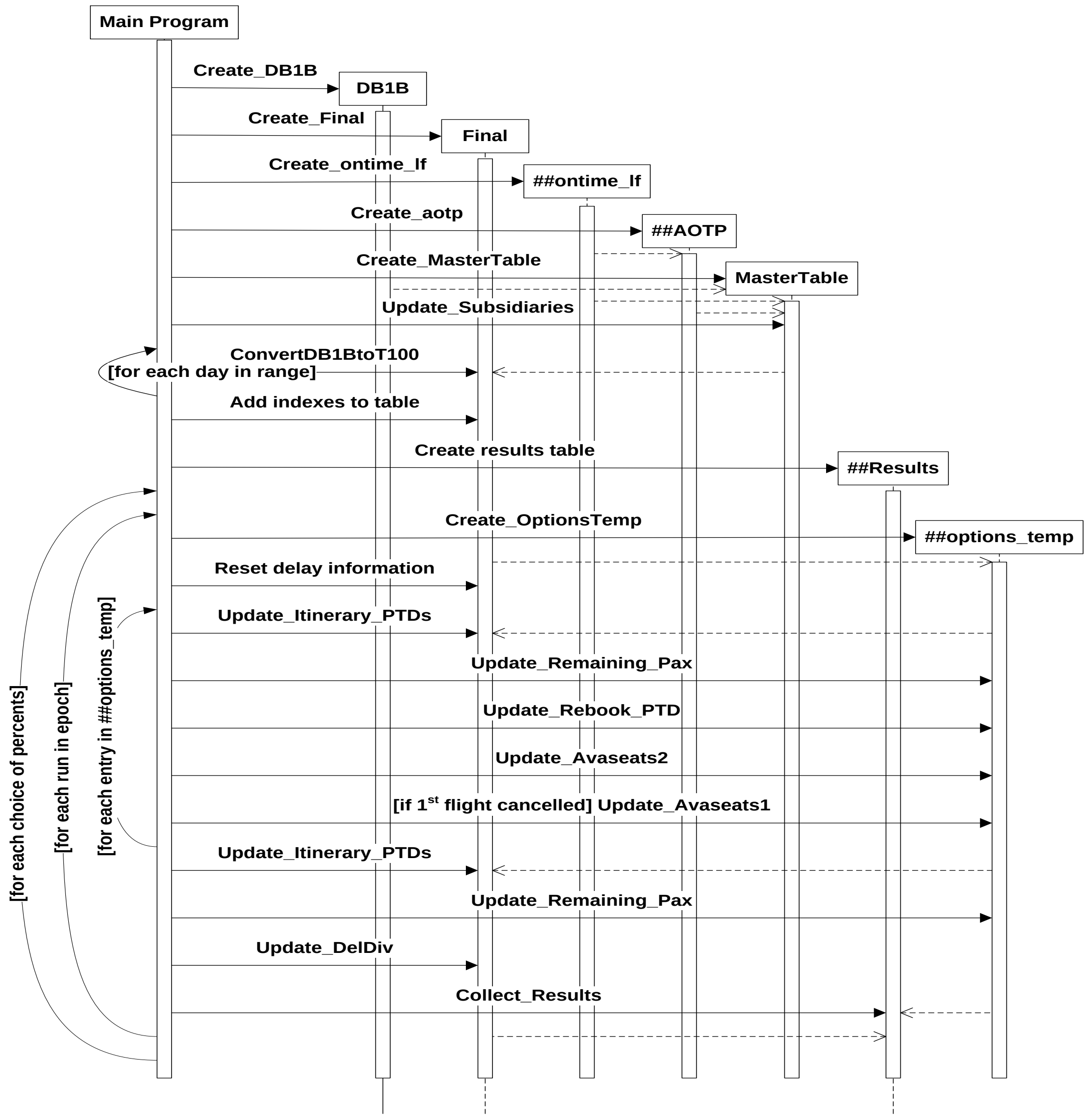


Figure 17 Code Sequence Diagram

Code statistics:

- 26 stored procedures (Table 6):

Table 6 Stored Procedures

Stored procedures	Lin es	Stored procedures	Line s	Stored procedures	Lines

Binomial_rand	233	CreateFinal	56	PopulateDelays	158
Collect_results	428	CreateMasterTable	163	PopulateDiverted	156
Comput_rem_pax	41	CreateOntime_lf	93	UpdateAvaseats1	30
Compute_total_pax	75	CreateOptionsBase	132	UpdateAvaseats2	29
Convert DB1BtoT100	140	CreateOriginDest	54	UpdateDelDiv	112
Create AOTP	95	PopulateCancellations	172	UpdateItinerarPtds	64
CreateDB1B	96	PopulateCancellations 2	160	Update RebookedPTD	45
Update Remaining Pax	34	UpdatedSubsidiery	51		

- Main program file: 425 lines
- Statistical calculation program: 129 lines

Stranded/Remaining Passengers Calculation

Stranded passengers are the passengers who are rebooked on a subsequent day, meaning that they must stay overnight to meet their flight schedule. Remaining passengers are passengers who do not have an alternative option for their itinerary and must remain unbooked.

The number of stranded passengers is the number of passengers who have been rebooked on a next-day alternative itinerary. In the code, this number can be determined by the number of passengers in the alternative priority categories 7 and 8 in the results table. The number of remaining passengers is the number of passengers who have not been rebooked. This is stored in the results table as the number of passengers in alternative priority categories -1 (no itineraries) and 0 (no seats), which are determined from the number of remaining passengers on flights after the rebooking algorithm has completed.

Formulas and Definitions

i: Itinerary, assuming that there are *n* itineraries, numbered 1 through *n*.

Sch: Scheduled, referring to the departure and arrival times scheduled for the flight.

Act: Actual, referring to the actual departure and arrival times of a flight.

Reb: Rebooked, referring to rebooked flights. A rebooked flight will also have scheduled and actual departure and arrival times, so for example, the scheduled departure time of a rebooked flight would be written as $SchDeparture_{Reb}$.

PTD: Passenger Trip Delay, a measure of the delay between each passenger's scheduled arrival time and their actual arrival time (assuming that a window of up to 15 minutes is not considered a delay). If a passenger arrives before the ticketed time (either due to preemptive rebooking, or simply a fast flight), the passenger's contribution to the PTD is zero.

$$PTD = \sum_{i=1}^{n} \max(0, ActArrival(i) - SchArrival(i) - 15min)$$

Reduction in airline lost revenue, AIRFARE NOT REFUND (ANF)

Airfare not refund: *ANR* is the difference in airfare refunded by the airline without preemptive rebooking and with preemptive rebooking. The assumption is that for passengers not rebooked, airlines have to refund an average airfare of \$377 per passenger.

Airfare refund is an amount which airline pays to passengers not rebooked. The baseline is the amount of refund expected without preemptive rebooking. The difference between baseline and the amount with preemptive rebooking is the airline's savings (ANR).

$AR_{base} = (PAX_{RebNDBase} + PAX_{RemBase}) \bullet \$377$

$AR = (PAX_{RebND} + PAX_{Rem}) \bullet \$377$

$ANR = AR_{base} - AR$

$\%ANR = 100\% \bullet ANR/AR_{base}$

Where:

$PAX_{RebNDBase}$: Number of passengers rebooked Next Day, with no preemptive rebooking.

PAX_{RebND}: Number of passengers rebooked Next Day, using the chosen level of preemptive rebooking.

$PAX_{RemBase}$: Number of unbooked passengers remaining, with no preemptive rebooking.

PAX_{Rem}: Number of unbooked passengers remaining, using the chosen level of preemptive rebooking.

AR_{base}: Airfare refund for the baseline case without preemptive rebooking

CORPORATES TRAVEL EXPENSE SAVINIGNS (CTES)

CTES is the difference in additional travel expenses accrued by corporate travelers without preemptive rebooking and with each of the preemptive rebooking percentages that are required to overnight due to rebooking the next day. The assumption is that corporates have to pay additional costs for overnight business travelers, which is \$250 on average, \$160 for hotel accommodation and \$90 for food and transportation expenses (Sherry, 2014). It is assumed that 50% of the passengers are not at their home town airport and would require overnight hotel accommodation (Li, Baik, Trani, 2010). Further 50% of these passengers are estimated to travel on corporate expense accounts (DoT, 2015).

The baseline is the cost which corporates have to pay in the case without preemptive rebooking. The difference between the baseline and amount with preemptive rebooking is the corporates' savings.

$CTE_{Base} = \$250 \bullet (PAX_{RebNDBase}+PAX_{RemBase}+PAX_{RebPDBase}) \bullet 0.5 \bullet 0.5$

$CTE = \$250 \bullet (PAX_{RebND}+PAX_{Rem}+ PAX_{RebPD}) \bullet 0.5 \bullet 0.5$

$CTES = CTE_{Base} - CTE$

$\%CTES = 100\% \bullet CTES/CTE_{Base}$

$PAX_{RebPDBase}$: Number of passengers rebooked Previous Day, with no preemptive rebooking.

PAX_{RebPD}: Number of passengers rebooked Previous Day, using the chosen level of preemptive rebooking.

PASSENGERS

Percentage of preemptively rebooked passengers is calculating like this:

% preemptively rebooked pax = 100% · Pax_{reb}/Pax_{can}

Pax_{reb}: Number of pax rebooked preemptively.

Pax_{can}: Number of pax on the cancelled flights.

Percent of passengers accommodated

Let us suppose that 10% of passengers *accept* preemptive rebooking. Then how many of those can be *accommodated* on an early flight? Number of passengers accommodated is defined as number of passengers willing to be rebooked earlier (preemptively rebooked), who are rebooked on an earlier flight. Not all passengers who are willing to be preemptively rebooked can be rebooked on earlier flights, because itineraries or seats might not be available. The algorithm first checks the available itineraries and available seats.Seats are number of passengers who can be rebooked earlier. Then the algorithm fills the seats until there are empty seats.

Overnight passengers, also called or *stranded passengers,* are passengers who are rebooked on the day after a flight's scheduled departure time.

Passengers that cannot be rebooked are remaining passengers who cannot be rebooked, as there are no options for their rebooking.

TTT : Total Trip Time, a measure of the combined time that each passenger is scheduled to travel, from initial departure to final arrival.

$$TTT_{Sch} = \sum_{i=1}^{n} [SchArrival(i) - SchDeparture(i)]$$

$$TTT_{Reb} = \sum_{i=1}^{n} [ActArrival_{Reb}(i) - SchDeparture_{Reb}(i)]$$

ΔTTT : Change in Total Trip Time, from the original schedule to the final actual flight time including rebooking.

$$\Delta TTT = TTT_{Reb} - TTT_{Sch}$$

As an example, a passenger may have a 2 hour flight scheduled at noon, so their scheduled trip time would be 2 hours. Suppose that their flight is cancelled, and they are preemptively rebooked on a 2 hour flight that leaves at 8 in the morning, but the flight is delayed for an hour. Their rebooked trip time would now be 3 hours, due to the delay, and they would arrive at 11am, which is still earlier than the original scheduled arrival. That means that the trip time has gone up while the PTD is zero.

Monte Carlo Simulation

To achieve the objectives of the analysis, various parameters that are inputs to the PTDC are modified over multiple runs in the Design of Experiment (see below). Parameters that are modified include:

- Percentage of passengers that choose to accept a preemptive rebooking option. The actual passengers that preemptively rebook is taken by randomly selecting passengers at a uniform rate from all passengers on all itineraries, using the preemptive percentage as the probability for each passenger.
- The time in advance that the preemptive rebooking option is made available to passengers. Passengers may choose Same day before or Previous day + Same day preemptive rebooking, and the algorithm will accepts these two options.

- Load Factor Bias. Levels of Load Factor which differ from the baseline Load Factor may be used.

The Monte Carlo Simulation is executed a fixed number *n* times for each replication. The epoch size *n* is set as a configurable parameter in the code. The results are stored and then used to generate statistics by a processing algorithm.

In order to determine the minimum but a sufficient number of runs *n* ,a confidence interval approach was used. For a confidence interval of 95%, it means that if the suggested number of runs *n* is used, there is less than 5% chance that a bad estimate will occur through pure chance. A confidence interval is calculated by finding the area contained in the tail ends of a normal distribution.

The value of *n* for a given confidence interval is given as $n = \left(\frac{z \cdot \sigma}{E}\right)^2$, where *z* is the critical value for the chosen confidence interval (for 95% confidence, the parameter *z* = 1.96), σ is the standard deviation of the data, and *E* is the interval of tolerance for the data. As an example, if the data values are 5 digits, and two significant figures are desired, then a tolerance of $E = 0.5 \times 10^{(5-2)} = 500$ is needed.

For rebooked passengers (the calculations is done for number of passengers rebooked preemptively –same day, preemptively same + previous day, same day later and next day), the data values and the range of σ is shown in table 7.

Table 7 Example of results for different percentages of passenger willing to be rebooked earlier, from 20% to 70%

Same Day Preemptively				**Same + Previous Day Preemptively**			
10% Pax Rebooked	μ	σ	μ/σ	**10% Pax Rebooked**	μ	σ	μ/σ

Same day Early	380.90	23.30	16.30	Same day Early	380.90	23.30	16.30
Same Day Late	2674.50	17.40	153.50	Same Day Late	2674.50	17.40	153.50
Next Day	2139.60	13.20	161.50	Next Day	2139.60	13.20	161.50
30% pax Rebooked	μ	σ	μ/σ	**30% pax Rebooked**	μ	σ	μ/σ
Same day Early	1144.08	42.30	27.00	Same day Early	1144.08	42.30	27.00
Same Day Late	2197.64	29.20	75.20	Same Day Late	2197.64	29.23	75.20
Next Day	1853.28	26.21	70.70	Next Day	1853.28	26.21	70.70
50% pax Rebooked	μ	σ	μ/σ	**50% pax Rebooked**	μ	σ	μ/σ
Same day Early	1910.80	34.10	56.00	Same day Early	1910.80	34.10	56.00
Same Day Late	1695.80	29.40	57.70	Same Day Late	1695.80	29.40	57.70
Next Day	1588.30	26.30	60.40	Next Day	1588.30	26.30	60.40
70% pax Rebooked	μ	σ	μ/σ	**70% pax Rebooked**	μ	σ	μ/σ
Same day Early	2678.60	47.56	56.30	Same day Early	2678.60	47.56	56.30
Same Day Late	1181.20	40.00	29.50	Same Day Late	1181.20	40.00	29.50
Next Day	1335.10	19.20	69.70	Next Day	1335.10	19.20	69.70

In the case σ=116, for 2 significant digit accuracy, E = 50, resulting in a need for at least n>20.7 runs, which confirms that 25 runs are sufficient to achieve the desired confidence. Every additional significant digit of accuracy of the results will divide the value of E by 10, meaning that it would require a 100 times increase in the number of runs

Design of Experiment

The Design of Experiment is summarized in Table 1 and 2. Two preemptive rebooking options are considered: Same day earlier (ticketed scheduled departure time), and Previous day + Same day earlier. The percentage of passengers rebooking is incremented beginning from none (i.e. 0%), 10%, 30%, 50%, 70%. The 0% is the baseline where no preemptive rebooking is used. The template for collecting results as a function of this independent parameter is shown in the following table.

Table 8 Design of Experiment

COONTROL PARAMETERS		RESULTS			
Preemptive Rebooking Time	% Passengers Accepting Rebooking Option	% Passengers accommodated	ANR	CTES	Total PTD
Same Day Before	0				
	10				
	30				
	50				
	70				
Previous day plus Same Day Before	0				
	10				
	30				
	50				

	70				

4. RESULTS

Multiple case-studies were run for United Airlines and Southwest Airlines scheduled flights in 2012 (table 9). The results are discussed in the next section.

Table 9 Case studies

United Airlines (UA)
Case study 1 – UA, ORD one day (Jan 12 2012)
Case study 2 – UA, ORD 2012
Case study 3 – UA, Hubs 2012
Southwest Airlines (SW)
Case study 4 – SW, MDW, 2012
Case study 5 – SW, Hubs, 2012

For all case studies, the H_0 hypotheses are rejected (Table 10). The p-value represents the probability that a null hypothesis cannot be rejected given the test results. As shown in the table, the probability represented by the p-value is small everywhere, with a value less than 0.1%, meaning that there is high statistical significance to the claim that the null hypothesis were all rejected.

Table 10 p-values for case studies:

Case Studies	H_0 1: PR does not accommodate more than 10% pax on cnx flights	H_0 2: Preemptive rebooking does not result in additional revenue to airlines	H_0 3: Preemptive rebooking does not result in additional saving to Corporations	H_0 4: PR does not result in reducing PTD and number of overnight and passenger which cannot be rebooked	H_0 5: Number of additional pax accommodated by adding an additional day of preemptive rebooking is greater than 10%
UA, Jan 12	$p<0.001$	$p<0.001$	$p<0.001$	$p<0.001$	$p<0.001$
UA, ORD	$p<0.001$	$p<0.001$	$p<0.001$	$p<0.001$	$p<0.001$
UA	$p<0.001$	$p<0.001$	$p<0.001$	$p<0.001$	$p<0.001$
SW,MDW	$p<0.001$	$p<0.001$	$p<0.001$	$p<0.001$	$p<0.001$
SW	$p<0.001$	$p<0.001$	$p<0.001$	$p<0.001$	$p<0.001$

United Airlines

In the first three case studies the observed airline was United Airlines. During 2012 the airline had a total of 531,245 flights, out of which, 7599 were cancelled (1.43%). The cancellations per day are shown in the next histogram (Figure 18).

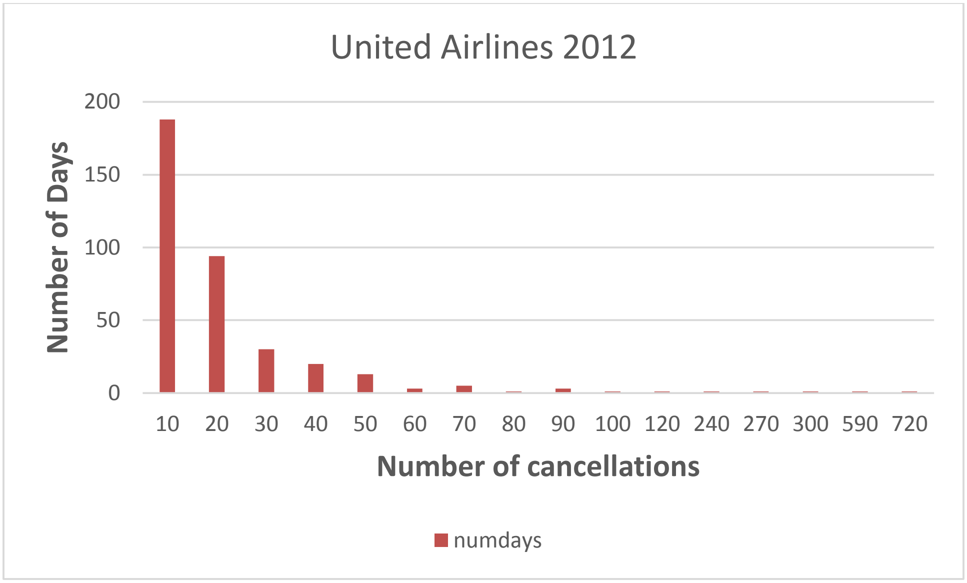


Figure 18 Number of cancellations per day, UA, 2012

The biggest number of days had less than cancellations (188 days), and more than 100 cancellations per day had 7 days. The biggest number of cancelled flight in one day was 720. Mostly all cancellation events were one-day events. In this thesis just those one day events, when number of cancelled flights is bigger then 20, will be analyzing. First case study is a one day event, January 12th 2012.

Case Study 1

A case-study was conducted for the scheduled domestic flights for a United Airlines operating from ORD hub for an event on January 12, 2012. Snow accumulation of 4.7" started around Noon and lasted through the evening ((NOAA, 2015). There were a total of 56 cancelled flights to or from the hub airport (Figure 19) impacting an estimated 5250 passengers (Table 11).

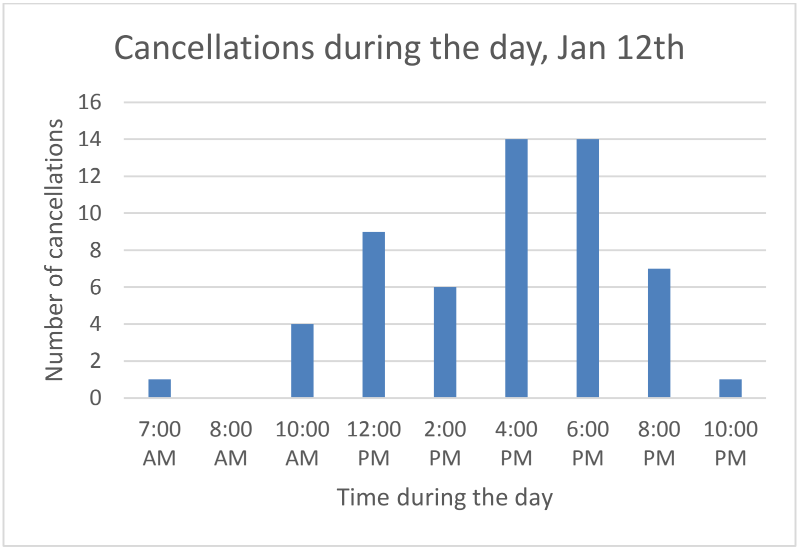


Figure 19 UA, ORD, January 12th 2012, Cancellations during the day

Table 11 Flight Statistic for the event January 12th, 2012

%Flights Cancelled	Cancelled Flights	Passengers	Passengers on the Cancelled Flights
16.4%	56	29,052	5,250

The average load factor was 80% with a minimum of 32% and maximum of 96% (Table 12).

Table 12 Load Factor statistics

LF percent	**Number of cancelled flights (with Origin/Destination in the hub)**	**Number of flights**	**Number of flights with Origin/Destination in the hub**

0-50%	**0**	**24**	**3**
50%-60%	**3**	**92**	**22**
60%-70%	**24**	**163**	**75**
70%-80%	**19**	**446**	**97**
80%-90%	**10**	**617**	**133**

Existing rebooking paradigm

For the existing re-booking paradigm, assuming all passengers are to be rebooked, 55% of the cancelled passengers are accommodated with seats on the same day after the departure time of the cancelled flight. Forty-three percent must be rebooked on flights the following day. Approximately 1% of the cancelled passengers cannot be rebooked due to the absence of available seats to their desired destination (Figure 20). The upper bound for refunded airfares is \$833K. The upper bound for unplanned (i.e. unbudgeted) Corporate Travel Expenses is \$292.8K.

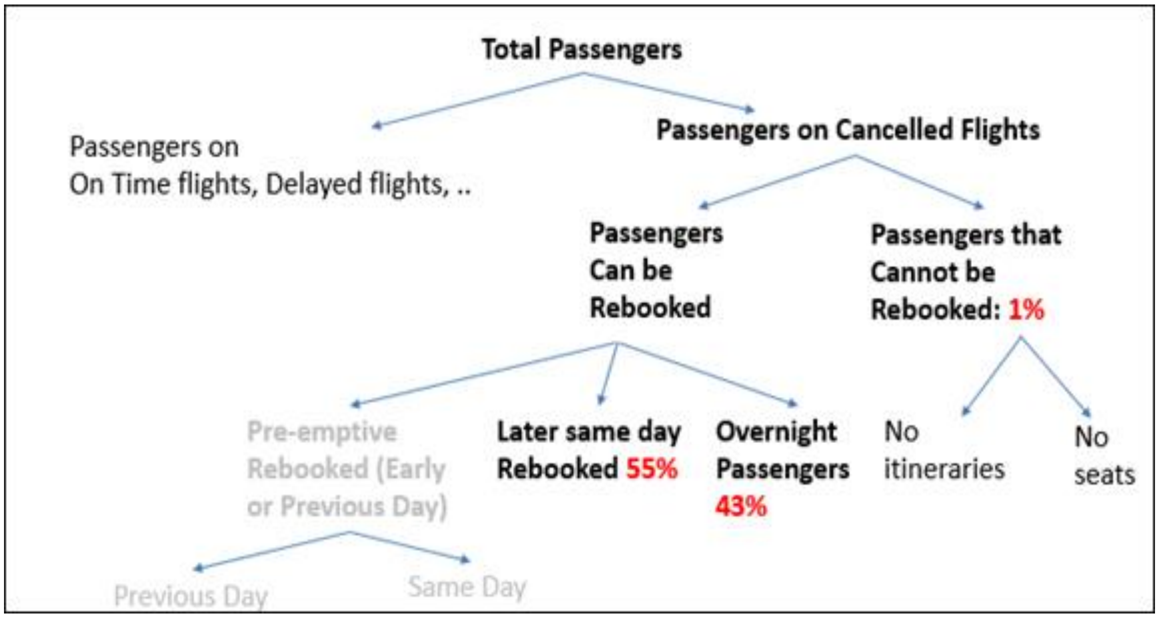

Total Passengers
Passengers on On Time flights, Delayed flights, ..
Passengers on Cancelled Flights
Passengers Can be Rebooked
Passengers that Cannot be Rebooked: 1%
Pre-emptive Rebooked (Early or Previous Day)
Later same day Rebooked 55%
Overnight Passengers 43%
No itineraries
No seats
Previous Day
Same Day

Figure 20 Existing re-booking paradigm

Case study 1: ***H_0: Preemptive rebooking does not accommodate more than 10% passengers on cancelled flights (Preemptive passengers / total cancel passengers < 10%)***

Based on results it is **shown that Preemptive Rebooking accommodated 73% of passengers,** hence Null hypothesis, H_0: Preemptive rebooking does not accommodate more than 10% passengers on cancelled flights (Preemptive passengers / total cancel passengers < 10%) is rejected.

Monte Carlo simulation of randomly selected passengers seeking preemptive rebooking for cancelled flights were accommodated 72% of the time (σ=6%). The relationship between the percent of Preemptively Rebooked Passengers and the percent of Passengers seeking Preemptive Rebooking is as follows (R^2=0.996):

% Pax Preemptively Rebooked = 0.72 • % Pax Seeking Rebooking

For example, for this case study, when 10% of the passengers pursued preemptive Same Day Early Rebooking, 7.2% were accommodated. When 70% of the passengers pursued preemptive Same Day Early Rebooking 51% were accommodated (Figure 21). The yellow line shows full accommodation projection, the projected value if all passengers willing to be preemptively rebooked are accommodated.

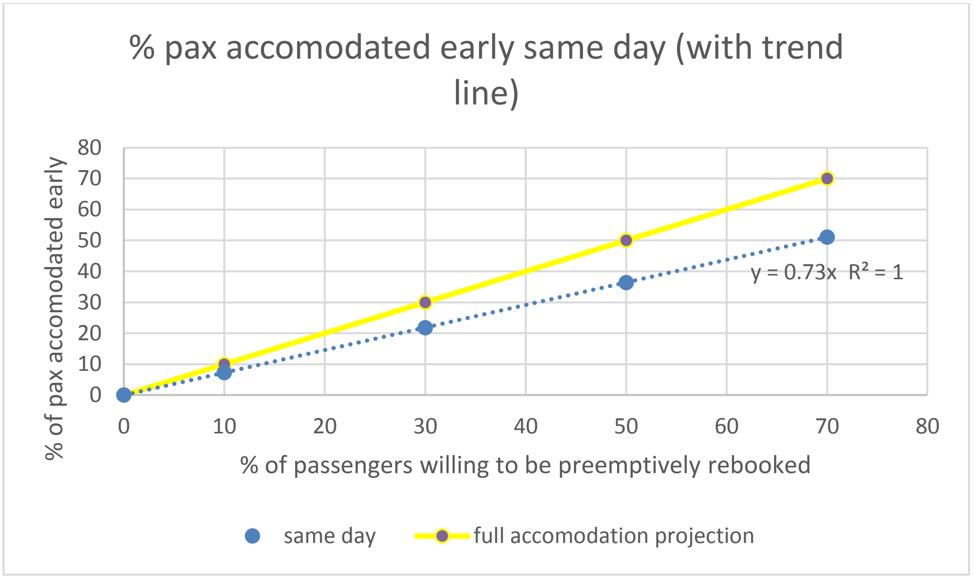


Figure 21 % of passengers accommodated on the Same Day Early

For the passengers that sought preemptive rebooking but could not be accommodated, it was due to the absence of seats on flights, not the absence of itineraries.

Case study 1: H0: The number of additional pax accommodated by adding <u>an additional day</u> of preemptive rebooking is greater than 25% of the original number of accommodated

Based on results it is shown that **<u>a majority of the passengers (>93%) can be accommodated earlier on the same day,</u>** hypothesis H0: The number of additional pax accommodated by adding an additional day of preemptive rebooking is greater than 25% of the original number of accommodated is rejected.

Randomly selected passengers seeking preemptive rebooking for cancelled flights were accommodated 79% of the time (σ=6%). The relationship between the percent of

Preemptively Rebooked Passengers and the percent of Passengers seeking Preemptive Rebooking is as follows (R2=0.996):

*% Pax Preemptively Rebooked = 0.79 * % Pax Seeking Rebooking*

For example, for this case study, when 10% of the passengers pursued preemptive Same Day Early Rebooking, 7.9% were accommodated. When 70% of the passengers pursued preemptive Same Day Early Rebooking 55.4% were accommodated.

For the passengers that sought preemptive rebooking but could not be accommodated, it was due to the absence of seats on flights, not the absence of itineraries.

These results were a 4% increase over the Preemptive Rebooking for the Same Day only. The benefits of Previous Day rebooking are marginal (Figure 22). The yellow line shows full accommodation projection, the projected value if all passengers willing to be preemptively rebooked are accommodated.

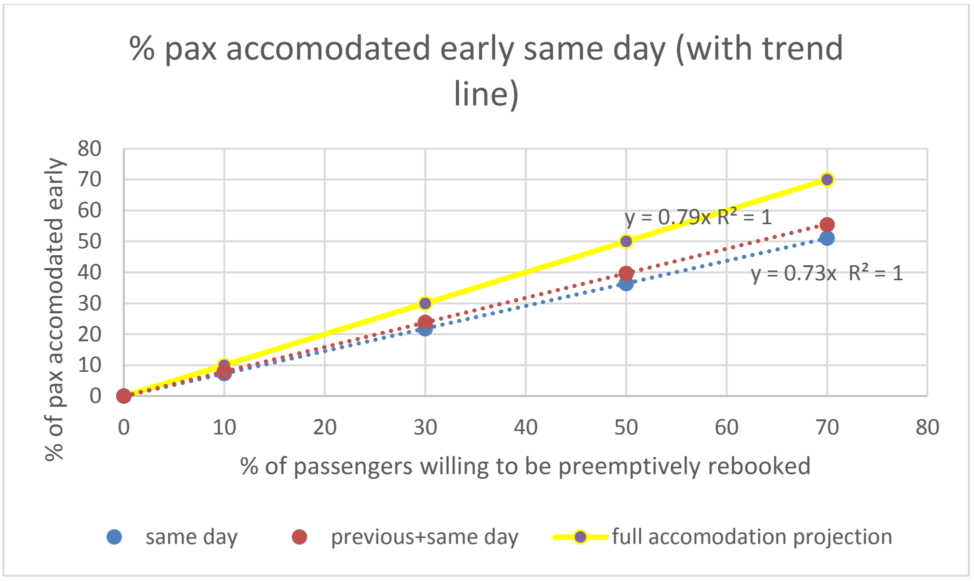


Figure 22 % of passengers accommodated early same day and same + previous day

Case Study 1: H0: Preemptive rebooking does not result in additional revenue to airlines: Rejected

Based on the Monte Carlo results it is shown that Preemptive rebooking results in additional revenue to airlines up to $359K for same day and $444K for same + previous day. Therefore, the hypothesis H0: Preemptive rebooking does not result in additional revenue to airlines is rejected.

The upper bound (i.e. case when all passengers who stay overnight and the passengers that cannot be rebooked ask for refunds) of ANR when 7.3% of the passengers are pre-emptively rebooked is $56K, and the upper bound of ANR when 51% of the passengers are pre-emptively rebooked is $359K for the same day, it is$67K for 10% and $444K for 70% for same+previous day (figure 23, shows percent of ANR with decrease of percentage of passengers willing to accept preemptive rebooking).

The total savings in ANR (upper bound) for same day preemptive rebooking and same + previous day preemptive rebooking is shown in the following table:

Table 13 Savings in $ for Airfares Not Refund

Airfares Not Refund		
% of Pax accepted preemptive rebooking	Same Day Early [$]	Same+Previous Day [$]
0	0	0
10	56K	67K
30	164K	201K
50	264K	326K
70	360K	444K

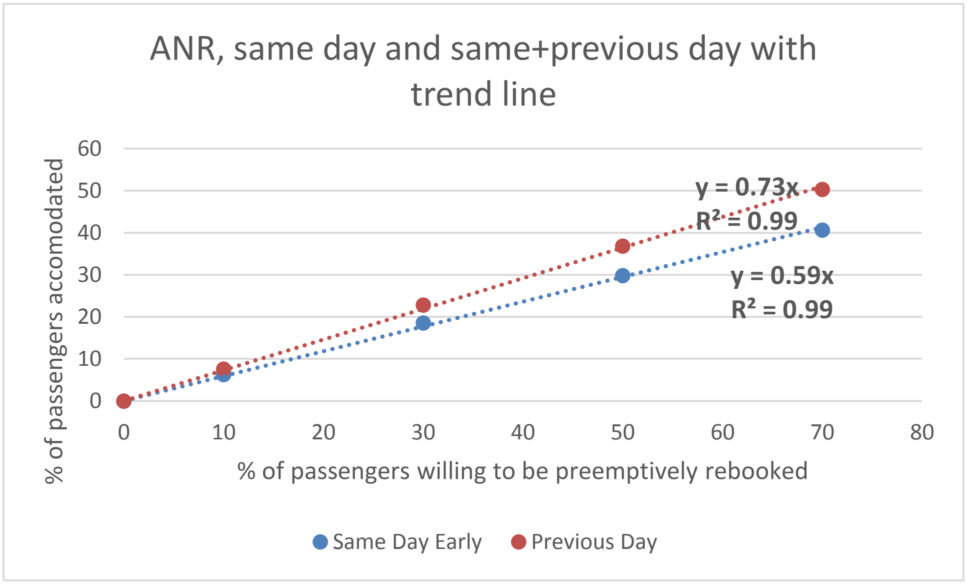


Figure 23 Airfares Not Refund, same and same + previous day

Case Study 1 H0: Preemptive rebooking does not result in additional saving to Corporates: Rejected

Based on the simulation results it is shown that Preemptive rebooking results in additional revenue to corporates up to K for same day and K for same + previous day. Hence, the hypothesis H0: Preemptive rebooking does not result in additional revenue to corporates is rejected.

Pre-emptive rebooking also has an effect of reducing the number of overnight passengers that could accrue unbudgeted Corporate Travel Expenses (Figure 24). The upper bound for Corporate Travel Expense Savings (CNES) is represented by the following relationship ($R^2 = 0.9987$):

*CNES ($) = 1.2K * % Pax Preemptively Rebooked*

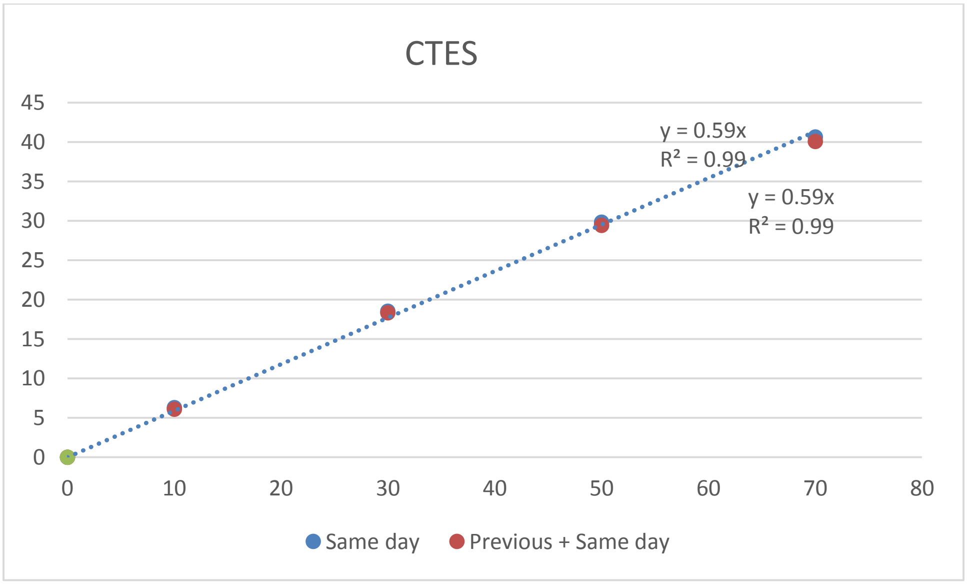


Figure 24 CTES

For example, the upper bound of ANR when 7.2% of the passengers are pre-emptively rebooked is $9K. The upper bound of ANR when 51% of the passengers are pre-emptively rebooked is $44K (Table 14).

Table 14 CTES in $, same day and same+previous day

CTES		
% of pax willing to be preemtively rebooked	Same day	Previous + Same day
0	0	0
10	9K	9K
30	27K	27K
50	44K	43K
70	59K	59K

Case Study 1: H_0: Preemptive rebooking does not result in reducing PTD and number of overnight and passenger which cannot be rebooked: Rejected

Based on the simulation results it is shown that Preemptive rebooking results in reducing PTD, the number of overnight stays, and passenger which cannot be rebooked. Hence, the hypothesis H_0: Preemptive rebooking does not result in reducing PTD and number of overnight and passenger which cannot be rebooked is rejected.

Preemptive rebooking accepted by random passengers from the cancelled flights has the effect of freeing-up seats on flights after the cancelled flight. This allows passengers that would otherwise be rebooked the next day to be rebooked on the same day. The percentage in reduction in the number of passengers rebooked on the next day as a function of the % Passengers Rebooked Preemptively is represented by the equation below ($R^2 = 0.9987$).

% Reduction in Pax Preemptively Rebooked on the Next Day =

0.35 • % Pax Preemptively Rebooked + 0.12

For example, for this case study, the preemptive rebooking of 7.2% of the passengers reduced the percentage of passengers rebooked overnight by 14%. The preemptive rebooking of 51% of the passengers, reduced the percentage of passengers rebooked overnight by 29.9% (figure 25).

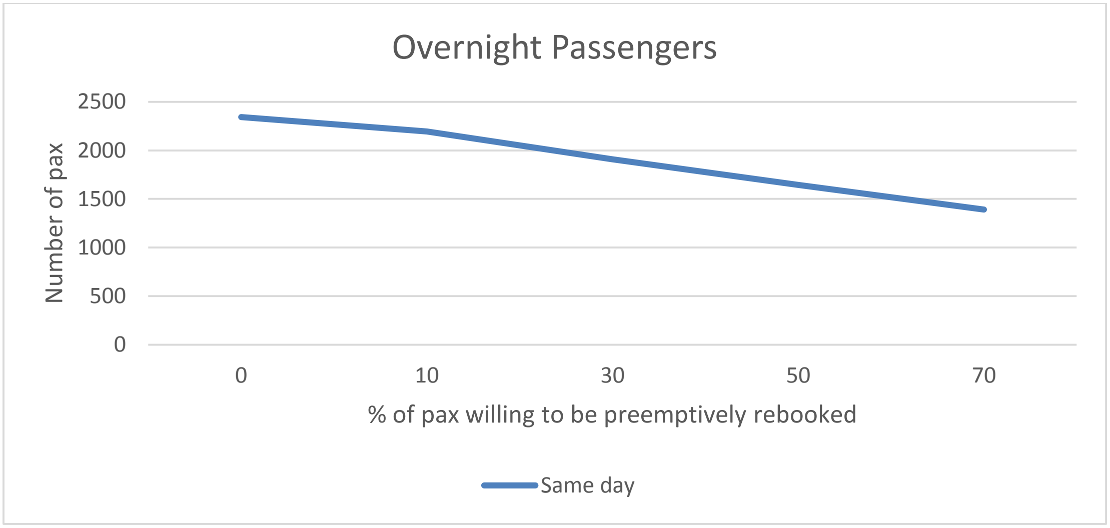


Figure 25 Decrease in Overnight Pax

Average PTD decreased from 499 (base line) to 274min (70%), and total PTD from 1818 days (base line) to 1000 days (70%), and number of passengers who cannot be rebooked is decreased by 42.5% . The decrease in PTD is shown in the figure 26.

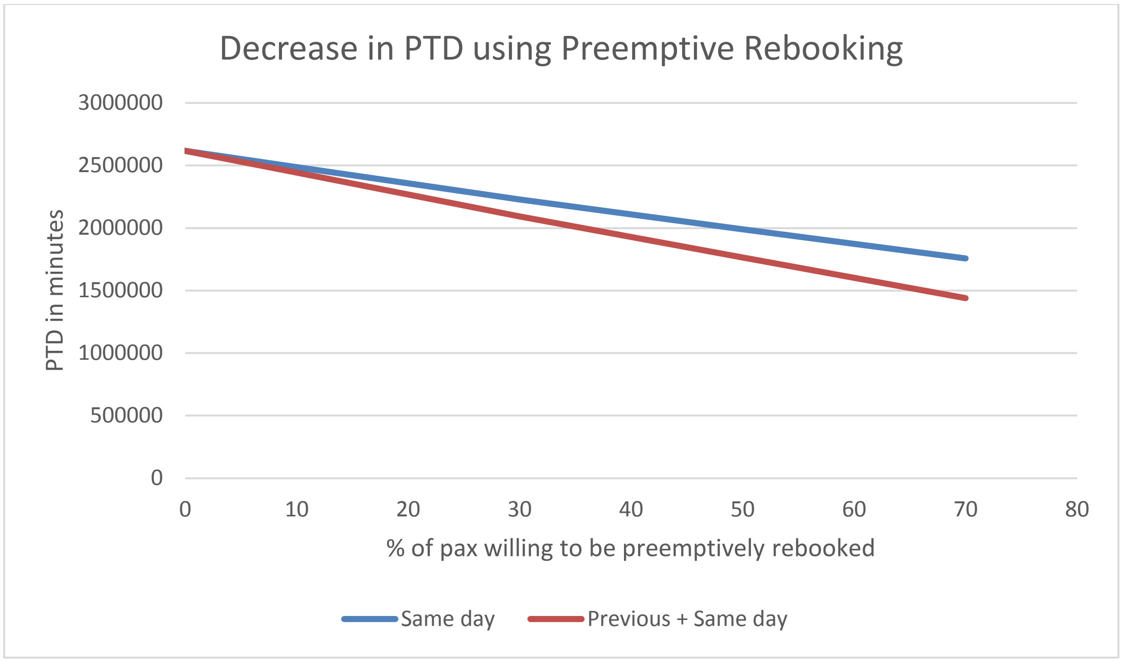


Figure 26 Decrease in Total PTD

Total trip time was not changed significantly (Figure 27).

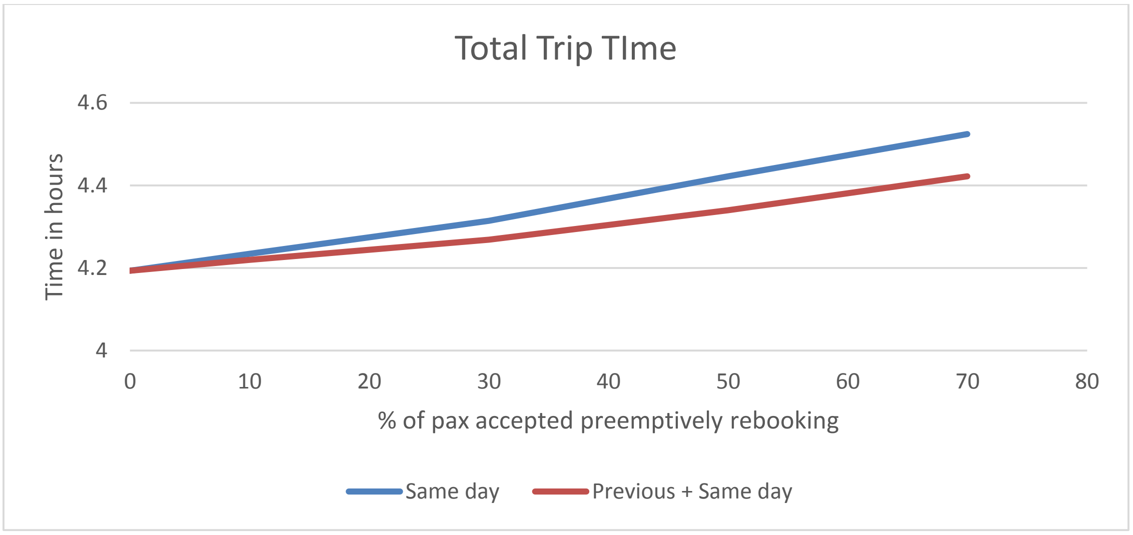


Figure 27 Total Trip Time

Case Study 1: UA, ORD, Jan 12, Summary of results

Case study 1 for United Airlines, hub ORD, January 12 shows that Pre-emptive rebooking in advance of forecast large-scale cancellation events is feasible. At least 72% of the passengers seeking pre-emptive rebooking can be accommodated before their original scheduled flight time. The remaining 28% of the passengers cannot be re-accommodated due to insufficient seats. Pre-emptive rebooking on the previous day was not required. Rebooking on the previous day did not significantly change the percentage of passengers re-accommodated. Pre-emptive rebooking also creates a win-win for all the stakeholders. Airlines are able to recoup up to $7.7K • %Passengers Rebook Preemptively that otherwise might be refunded. Corporations sponsoring business travel for their employees also benefit by saving up to $1.2K • %Passengers Rebook Preemptively through unbudgeted overnight costs.

Case Study 2

A case-study was conducted for the scheduled domestic flights for a United Airlines operating from ORD hub for all one day cancellation events in 2012. There were a total of 20 days with more than 20 cancelled flights in that year, out of which 13 days were 1-day cancellation events (Table 15).

Table 15 UA, ORD, Number of cancelled flights per day

Month	Day	Number of Cancelled Flights	Arrival Flights Cancelled	Departure Flights Cancelled
1	12	56	31	26
1	20	77	33	44
1	23	38	23	15
2	23	28	11	17
5	6	26	8	18
5	29	30	15	15
6	22	28	12	16
6	29	22	10	12
7	13	34	16	18
7	15	25	12	13
7	18	37	14	23

11	7	37	31	24
12	20	37	20	17

In 2012 at the mid-west hub the airline experienced 4 days (2.5%) with between 20 and 30 flight cancellations. There were 5 days (1.4% of the year) with 30 and 40 cancelled flight, 4 days (1.1% of the year) with between 40 and 60 cancelled flights, and 2 days (0.54% of the year) with more than 100 cancelled flights.

For the day of the one day cancellation events, the average load factor on *all* flights was 83% with a minimum of 25% and maximum of 100%. The average load factor on the cancelled flights was 80% with a minimum of 32% and maximum of 97%.

Figure 28 shows number of days with one day events, with more than 20 cancellations per day.

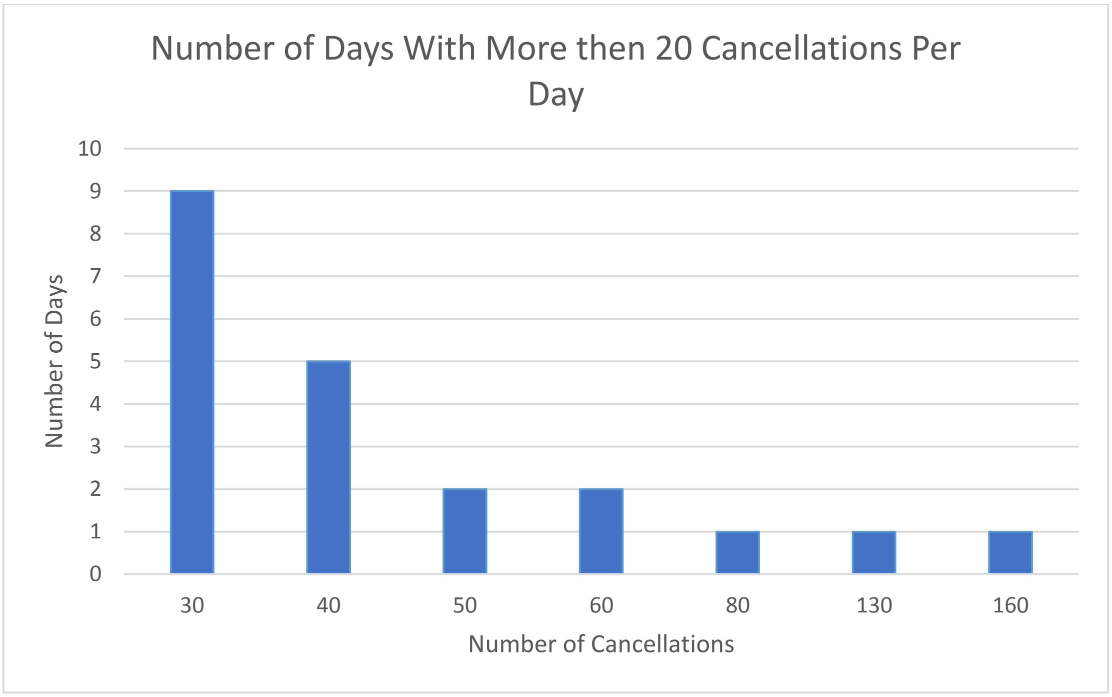


Figure 28 Number of days with more than 20 cancelled flights per day

The load factor statistics are shown in Table 16.

Table 16: Load Factor Statistics, 2012

LF percent	% of cancelled flights (with Origin/Destination in the hub)	% of flights	% of flights with Origin/Destination in the hub
0-50%	0.6	0.61	1.56
50%-60%	1.16	1.83	2.2

60%-70%	6.5	4.9	8.78
70%-80%	24.3	21.21	28.89
80%-90%	44.98	49.01	42.65
>90%	22.46	22.44	15.93

Baseline Rebooking (After Cancelled Flight)

For the existing re-booking paradigm, no passengers will be booked on preemptive flights, so the percent of passengers rebooked early will automatically be zero. The result is a slope of zero in the equation calculating the percent of passengers rebooked early. For similar reasons, the ANR and CTES slopes will also be zero.

Case study 2: *H_0: Preemptive rebooking does not accommodate more than 10% passengers on cancelled flights (Preemptive passengers / total cancel passengers < 10%)*

Based on the results that a projected 70% for upper bound of the passengers willing to be preemptively rebooked, will be accommodated it is shown that a hypothesis stetting that Preemptive rebooking does not accommodate more than 10% passengers on cancelled flights (Preemptive passengers / total cancel passengers < 10%) is rejected.

The Monte Carlo simulation shows that randomly selected passengers seeking pre-emptive rebooking for cancelled flights are accommodated at a rate of 70% of the number of passengers accepting early rebooking (the slope is 70%). If no passengers accept early rebooking, no passengers are accommodated preemptively, while if 70% of the passengers accept early rebooking, then 70%•70% = 49% of the total cancelled passenger are accommodated. If all passengers were to accept preemptive rebooking, a projected 70% would be accommodated.

The equation below represents the %Pax Accomodated, where the %Pax Accomodated represents the y-value, the %Pax Pre-emptively Rebooked represents the x-value, and 0.70 is the slope (all equations are in table 17). Subsequent equations follow a similar convention.

%Pax Accomodated = 0.70 • % Pax Pre-emptively Rebooked

Table 17 Preemptively rebooked passengers (m, R2) same and same+previous day

Preemptively rebooked passengers				
Same day Early			Same +Previous day	
	m	R^2	m	R^2
12-Jan	0.73	1	0.79	1
20-Jan	0.69	1	0.78	1
23-Jan	0.84	1	0.86	1
23-Feb	0.67	1	0.73	1
6-May	0.65	1	0.73	1
29-May	0.76	1	0.80	1
22-Jun	0.76	1	0.81	1
29-Jun	0.74	1	0.81	1
13-Jul	0.67	1	0.75	1
15-Jul	0.67	1	0.73	1
18-Jul	0.65	0.99	0.72	1
7-Nov	0.63	0.99	0.68	1
μ	0.70		0.77	
σ	0.066		0.05	
μ/σ	11.37		16	

Case study 2: H_0: The number of additional passengers accommodated by adding an additional day of preemptive rebooking is greater than 25% of the original number of accommodated: Rejected

Based on the Monte Carlo simulation results showing that a projected 70% for upper bound of passengers are accommodated on the same day which is 7% less than on the same + previous day, it is shown that the hypothesis H_0 :The number of additional passengers accommodated by adding an additional day of preemptive rebooking is greater than 25% of the original number of accommodated is rejected.

The next figure shows the percentage of passengers accommodated on the same day for the hub. All days with cancellations bigger than 20 shows a similar trend (more than 64% of passengers can be accommodated on the same day before the scheduled departure time).

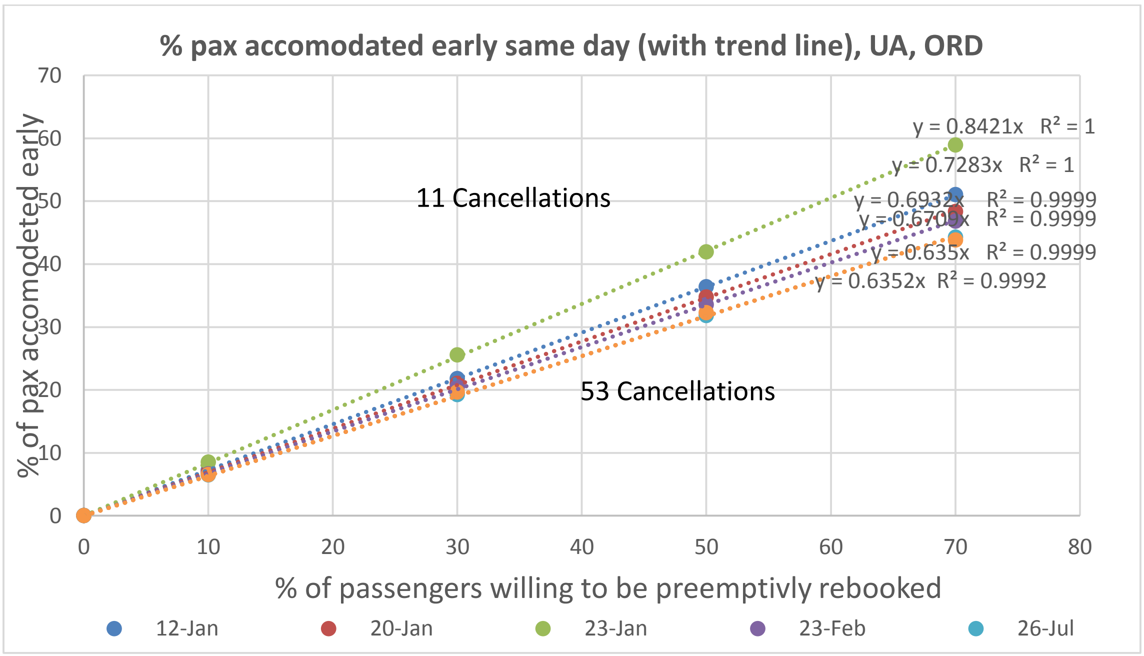


Figure 29 Percentage of passengers accommodated same day early

Case Study 2: H0: Preemptive rebooking does not result in additional revenue to airlines: Rejected

Based on the Monte Carlso simulation results that a decrease in airfares refunds is for 64% of passengers, it is shown that the hypothesis H_0: Preemptive rebooking does not result in additional revenue to airlines is rejected

Preemptive rebooking also has the effect of reducing the number of overnight passengers that could be eligible for airline refunded tickets. The relationship between Airfares Not Refunded (ANR) and preemptive rebooking is represented by the following equation:

%Decrease in Airfares Refunded = 0.64 • % Pax Pre-emptively Rebooked

For the same day preemptively rebooked the slope for ANR is 64%, giving an ANR of 45% when 70% of the passengers accept early rebooking (table 18).

Table 18 Airfares Not Refund (m and R2) same and same+ previous day

Airfares Not Refund					
Same day Early			Same +Previous day		
	m	R^2		m	R^2
12-Jan	0.58	0.9991	12-Jan	0.72	0.9993
20-Jan	0.46	0.9959	20-Jan	0.66	0.9994
23-Jan	0.65	0.9938	23-Jan	0.65	0.9942
23-Feb	0.52	0.9997	23-Feb	0.60	0.9998
6-May	0.62	0.9894	6-May	0.82	0.9975
29-May	0.82	0.9982	29-May	0.88	0.9985
22-Jun	0.92	0.9953	22-Jun	0.94	0.9963
29-Jun	0.39	0.996	29-Jun	0.64	0.9971
13-Jul	0.63	0.9997	13-Jul	0.73	0.9999
15-Jul	0.73	0.9977	15-Jul	0.76	0.997
18-Jul	0.60	0.9989	18-Jul	0.70	0.9994

7-Nov	0.61	0.9993	7-Nov	0.66	0.9999
μ	0.63		μ	0.74	
σ	0.14		σ	0.10	
μ/σ	4.60		μ/σ	7.4	

Case Study 2 H0: Preemptive rebooking does not result in additional saving to Corporates: Rejected

Based on the results that an increase in Corporate Travel Expense savings for 64% of passengers, it is shown that the hypothesis H_0 : Preemptive rebooking does not result in additional savings to corporates is rejected. Preemptive rebooking also has the effect of reducing the number of overnight passengers that could accrue unbudgeted Corporate Travel Expenses. The relationship between Corporate Travel Expense Savings (CNES) and preemptive rebooking is represented by the following equation:

%Increase in CNES =0.64 • % Pax Pre-emptively Rebooked

For the same day preemptively rebooked the slope CTES is 64% as well (table 19).

Table 19 CTES (m and R2) same and same + previous day

CTES					
Same day Early			Same +Previous day		
	m	R^2		m	R^2
12-Jan	0.58	0.9991	12-Jan	0.57	0.999
20-Jan	0.46	0.9959	20-Jan	0.42	0.9956
23-Jan	0.72	0.9938	23-Jan	0.63	0.9906
23-Feb	0.52	0.9997	23-Feb	0.50	0.9996
6-May	0.62	0.9894	6-May	0.55	0.9795
29-May	0.82	0.9982	29-May	0.81	0.997

22-Jun	0.92	0.9954	22-Jun	0.79	0.9778
29-Jun	0.39	0.996	29-Jun	0.23	0.974
13-Jul	0.63	0.9997	13-Jul	0.60	0.9997
15-Jul	0.73	0.9977	15-Jul	0.44	0.9898
18-Jul	0.60	0.9989	18-Jul	0.50	0.9951
7-Nov	0.61	0.9993	7-Nov	0.65	0.9917
μ	0.64		μ	0.56	
σ	0.14		σ	0.15	
μ/σ	4.6		μ/σ	3.7	

Case Study 2: H_0: Preemptive rebooking does not result in reducing PTD and number of overnight and passenger which cannot be rebooked: Rejected

Based on the results that number of overnight passengers is decreased up to 65% (same day) to 75% (same + previous day) and PTD is decreased 53% (same day) to 64% (same + previous day)it is shown that hypothesis H_0 : Preemptive rebooking does not result in reducing PTD and number of overnight and passenger which cannot be rebooked is rejected.

As noted above, preemptive rebooking has the effect of reducing the number of overnight passengers and decreasing PTD. The relationship between stranded passengers (next day passengers and passengers that cannot be rebooked) and preemptive rebooking is represented by the following equation:

%Decrease in SPAX =0.65 • % Pax Pre-emptively Rebooked

And the relationship between PTD and preemptive rebooking is represented by following equation:

% Decrese in PTD =0.53 • % Pax Pre-emptively Rebooked

The tables 20 and 21 show all results for stranded passengers and PTD.

Table 20 Slope and R2 for Overnight pax and pax that cannot be rebooked

Next day pax & Pax that cannot be rebooked					
Same day Early			Same +Previous day		
	m	R^2		m	R^2
12-Jan	0.59	1.00	12-Jan	0.73	1.00
20-Jan	0.48	0.99	20-Jan	0.67	1.00
23-Jan	0.69	0.99	23-Jan	0.69	0.99
23-Feb	0.52	1.00	23-Feb	0.60	1.00
6-May	0.65	0.99	6-May	0.84	1.00
29-May	0.83	1.00	29-May	0.89	1.00
22-Jun	0.97	0.99	22-Jun	0.97	0.99
29-Jun	0.40	0.99	29-Jun	0.66	1.00
13-Jul	0.64	1.00	13-Jul	0.74	1.00
15-Jul	0.65	1.00	15-Jul	0.73	1.00
18-Jul	0.61	1.00	18-Jul	0.70	1.00
7-Nov	0.62	1.00	7-Nov	0.66	1.00
μ	0.65		μ	0.75	
σ	0.15		σ	0.11	
μ/σ	4.43		μ/σ	7.09	

Table 21 Slope and R2: PTD

PTD					
Same day Early				Same +Previous day	
	m	R^2		m	R^2
12-Jan	0.48	1.00	12-Jan	0.65	1.00
20-Jan	0.47	1.00	20-Jan	0.64	1.00
23-Jan	0.54	0.97	23-Jan	0.55	0.98
23-Feb	0.39	1.00	23-Feb	0.49	1.00
6-May	0.55	0.98	6-May	0.74	0.99
29-May	0.74	1.00	29-May	0.79	1.00
22-Jun	0.65	0.99	22-Jun	0.68	0.99
29-Jun	0.44	1.00	29-Jun	0.61	1.00
13-Jul	0.47	1.00	13-Jul	0.58	1.00
15-Jul	0.46	0.99	15-Jul	0.51	0.99
18-Jul	0.52	1.00	18-Jul	0.61	1.00

7-Nov	0.61	1.00	7-Nov	0.66	1.00
μ	0.53		μ	0.64	
σ	0.10		σ	0.09	
μ/σ	5.26		μ/σ	7.06	

Case Study 2: UA, ORD, Summary of results

This case study rejects all H_o hypotheses. The analysis is shown in Table 22.

Table 22 Analysis, Case Study 2

Analysis	Same Day Preemptive			Same Day+Previous Day Preemptivly		
	m_μ	m_σ	m_μ/m_σ	m_μ	m_σ	m_μ/m_σ
% Pax Rebooked	0.70	0.07	11	0.77	0.05	16
%AF Not Refund	0.63	0.14	4.60	0.74	0.10	7.4
%CTES	0.64	0.14	4.57	0.56	0.15	3.7

Pre-emptive rebooking in advance of forecast large-scale cancellation events is feasible. At least 72% of the passengers seeking pre-emptive rebooking can be accommodated before their original scheduled flight time. The remaining 28% of the passengers seeking pre-emptive rebooking cannot be re-accommodated due to insufficient seats.

Pre-emptive rebooking on the previous day was not required as it did not significantly change the percentage of passengers re-accommodated.

The signal to noise ratio in Table shows that signal is 11 (same day) and 16 (same + previous day) times stronger than noise (for passengers accommodated case), and 4.6 (same day) and 7.4 (same + previous day) for ANR (table 10). The worst is for CTES, 4.6 for the same day and 3.7 for the previous + same day (table 11) In the case of CTES the best is if the passengers are rebooked on the same day because on other days (previous or next) they face extra costs due to lodging, etc.

Pre-emptive rebooking also creates a win-win for all the stakeholders. Airlines are able to recoup up to 0.74% • %Passengers Rebook Preemptively that otherwise might be refunded. Corporations sponsoring business travel for their employees also benefit by saving up to 0.64 • %Passengers Rebook Preemptively through unbudgeted overnight costs.

Case Study 3

Case study 3 is done for United Airlines, year 2012 and hubs: ERW and BOS, figure 30 shows the number of cancellations per day for UA.

The biggest cancellation events are shown in the table 23. A high percentage of cancellations was on November 7th (61%). Boston had the smallest number of flights per day. These hubs show similar results like ORD in the two previous studies.

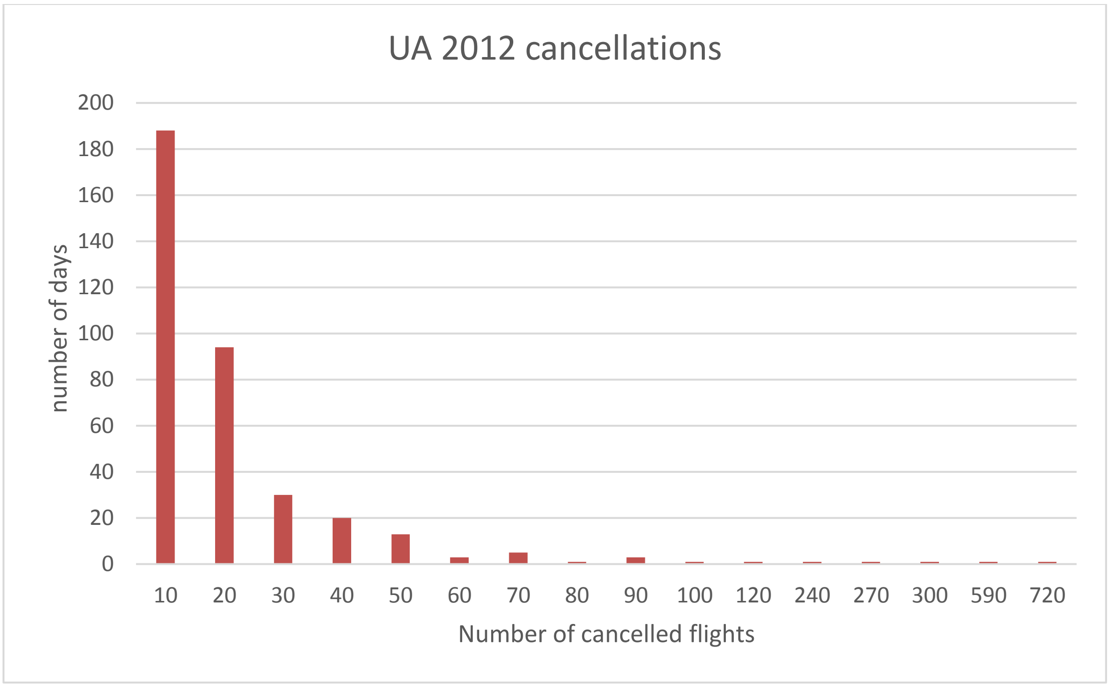


Figure 30 Number of cancelled flights per day

Table 23 Days with the biggest number of cancelled flights

>60		
month	day	cancellation
1	20	101
6	22	68
7	18	78
7	26	78
11	7	208
11	8	85

Baseline Rebooking (After Cancelled Flight)

Similar to the Case Study 2, for the existing re-booking paradigm, no passengers will be booked on preemptive flights, so the percent of passengers rebooked early will automatically be zero. The result is a slope of zero in the equation calculating the percent of passengers rebooked early. For similar reasons, the ANR and CTES slopes will also be zero.

Table 24 Preemptively rebooked passengers

Preemptively rebooked passengers									
Same day Early			Same +Previous day						
				Previous day		Same Day		Prev+Same	
	m	R		m	R	m	R	m	R
Den Feb 3rd	0.44	1.00		0.27	1.00	0.44	1.00	0.71	1.00
Den Jun7th	0.68	1.00		0.27	1.00	0.68	1.00	0.71	1.00
EWR July 18th	0.82	1.00		0.27	1.00	0.82	1.00	0.71	1.00
EWR Aug 14th	0.81	1.00		0.27	1.00	0.82	1.00	0.71	1.00
μ	0.69			0.27		0.69		0.71	
σ	0.18			0.00		0.18		0.00	
μ/σ	3.82					3.83			

Case study 3: *H_0: Preemptive rebooking does not accommodate more than 10% passengers on cancelled flights (Preemptive passengers / total cancel passengers < 10%)*

Based on the results that a projected 69% as the upper bound, of the passengers willing to be preemptively rebooked (Table 24), will be accommodated it is shown that the hypothesis that Preemptive rebooking does not accommodate more than 10%

passengers on cancelled flights (Preemptive passengers / total cancel passengers < 10%) is rejected.

Case study 3: H_0: The number of additional pax accommodated by adding an additional day of preemptive rebooking is greater than 25% of the original number of accommodated

Based on the results that a projected 69%, as the upper bound for passengers accommodated on the same day, which is the same as in the same + previous day (table 24), it is shown that the hypothesis H_0 :The number of additional passengers accommodated by adding an additional day of preemptive rebooking is greater than 25% of the original number of accommodated is rejected.

Case Study 3: H0: Preemptive rebooking does not result in additional revenue to airlines: Rejected

Based on the results that airlines can save in ANF up to $1M for the same day and $1.2M for previous day preemptively rebooked passengers for not refunded tickets (upper bound), it is shown that the hypothesis H_0 Preemptive rebooking does not result in additional revenue to airlines is rejected.

The full table of estimated savings is shown in the table 25.

Table 25 Savings in not refunded airfares

ANR			
	% of pax accepted preemptively rebooking	Same	Prev
Den Feb 3rd	0	$0	$0
	10	$21K	$43K
	30	$60K	$125K

	50	$98K	$197K
	70	$136K	$254K
Den Jun7th	0	$0	$0
	10	$38K	$45K
	30	$109K	$126K
	50	$167K	$201K
	70	$215K	$266K
EWR July 18th	0	$0	$0
	10	$64K	$64K
	30	$196K	$192K
	50	$310K	$311K
	70	$399K	$407K
EWR Aug 14th	0	$0	$0
	10	$47K	$46K
	30	$134K	$136K
	50	$203K	$221K
	70	$253K	$293K

The relationship between Airfares Not Refunded (ANR) and preemptive rebooking is represented by the following equation:

%Decrease in Airfares Refunded = 0.78 • % Pax Pre-emptively Rebooked

(Table 26).

Table 26 % of savings in ANF, slope and R2

Airfares Not Refund

Same day Early				Same +Previous day	
	m	R		m	R
Den Feb 3rd	0.31	1.00		0.59	0.99
Den Jun7th	0.68	0.99		0.83	1.00
EWR July 18th	1.07	0.99		1.08	1.00
EWR Aug 14th	1.06	0.98		1.18	1.00
μ	0.78		μ	0.92	
σ	0.37		σ	0.27	
μ/σ	2.14		μ/σ	3.46	

Case Study 3 H0: Preemptive rebooking does not result in additional saving to Corporates: Rejected

Based on the results that an increase in Corporate Travel Expense savings is for 64% it is shown that hypothesis H_0 : Preemptive rebooking does not result in additional savings to corporates is Rejected.

Preemptive rebooking also has the effect of reducing the number of overnight passengers that could accrue unbudgeted Corporate Travel Expenses. The relationship between Corporate Travel Expense Savings (CNES) and preemptive rebooking is represented by the following equation:

%Increase in CNES = 0.78 • % Pax Pre-emptively Rebooked

Table 27 UA, Hubs, CTES

CTES					
	Same day Early			Same +Previous day	
	M	R		M	R
Den Feb 3rd	0.31	1.00		-0.03	0.52
Den Jun7th	0.68	0.99		0.62	0.98
EWR July 18th	1.07	0.99		1.05	0.99
EWR Aug 14th	1.06	0.98		1.02	0.97
μ	0.78		μ	0.66	
Σ	0.37		σ	0.50	
μ/σ	2.14		μ/σ	1.32	

The slope for the same + previous day is lower (0.66, shown in the Table 27). The reason is that if more passengers are rebooked on the previous day, some of them have extra (unplanned) expenses for one more night (for hotel, transportation etc.) For corporates standpoint same day early is the best choice.

Case Study 3: H_0: Preemptive rebooking does not result in reducing PTD and number of overnight and passenger which cannot be rebooked

Based on the results that number of overnight passengers is decreased by 54% and PTD is decreased up 53% (upper bound for 70% passengers willing to accept preemptively rebooking), it is shown that hypothesis H_0 : Preemptive rebooking does not and result in reducing PTD and number of overnight and passenger which cannot be rebooked is rejected.

The results for PTD are shown in the table 28.

Table 28, Case Study 3, PTD results

PTD				% Improvement	
	%	Same Day	Same + Prev Day	% improvement same	% improvement same + prev
Den Feb 3	0	1663	1663	0	0
	10	1614	1551	2.95	6.73
	30	1530	1341	8.00	19.36
	50	1451	1153	12.75	30.67
	70	1378	990	17.14	40.47
Den Jun 7	0	1015	1015	0	0
	10	945	928	6.90	8.57
	30	808	768	20.39	24.33
	50	695	617	31.53	39.21
	70	603	489	40.59	51.82
EWR 18 Jul	0	1530	1530	0	0
	10	1413	1414	7.65	7.58
	30	1147	1154	25.03	24.58
	50	925	923	39.54	39.67
	70	753	739	50.78	51.70
EWR 14 Aug	0	918	918	0	0
	10	814	815	11.33	11.22
	30	629	624	31.48	32.03
	50	494	456	46.19	50.33
	70	396	299	56.86	67.43

As noted above, preemptive rebooking has the effect of reducing the number of overnight passengers and decreasing PTD. The relationship between stranded passengers (next day passengers and passengers that cannot be rebooked) and preemptive rebooking is represented by the following equation:

%Decrease in SPAX =0.78 • % Pax Pre-emptively Rebooked

And the relationship between PTD and preemptive rebooking is represented by following equation:

% Decrese in PTD =0.53 • % Pax Pre-emptively Rebooked

Table 29 Overnight passengers improvement

SPAX IMPROVEMENT			
		% Improvement	
	% of pax willing to be preemptively rebooked	improvement same	improvement same + prev
Den Feb 3	0	0.00	0.00
	10	3.32	6.76
	30	9.40	19.58
	50	15.31	30.70
	70	21.16	39.63
Den Jun 7	0	0.00	0.00
	10	7.97	9.62
	30	23.10	26.72
	50	35.36	42.52
	70	45.40	56.35
EWR 18 Jul	0	0.00	0.00
	10	11.62	11.62
	30	35.36	34.74
	50	56.05	56.13
	70	72.01	73.49
EWR 14 Aug	0	0.00	0.00
	10	13.05	12.73

	30	36.90	37.40
	50	55.99	60.90
	70	69.48	80.73

SPAX - slope		
	same	prev
Den Feb 3	0.30	0.60
Den Jun 7	0.68	0.83
EWR 18 Jul	1.07	1.08
EWR 14 Aug	1.06	1.18
μ	0.78	0.92
σ	0.37	0.26
μ/σ	2.12	3.54

Case Study 3: UA, Summary of results

Case Study 3 results are summarized in the table 30. Compared to Case Study 2, the percent passengers rebooked statistics is nearly the same as the same day preemptive, and within 0.2 for all other statistics.

Table 30 Case Study 3, Analysis

Analysis	**Same Day Preemptive**			**Same Day+Previous Day Preemptivly**		
	m_μ	**m_σ**	**m_μ / m_σ**	**m_μ**	**m_σ**	**m_μ / m_σ**
Slope for % Pax Rebooked	**0.69**	**0.18**	**3.82**	**0.70**	**0**	
Slope for ANR	**0.78**	**0.36**	**2.13**	**0.92**	**0.27**	**3.46**

Slope for CTES	**0.78**	**0.36**	**2.13**	**0.66**	**0.50**	**1.32**

,

Case Study 4

A Case Study 4 was conducted for the scheduled domestic flights for Southwest Airlines operating from Chicago Midway International Airport (MDW) for all one day cancellation events in 2012.

As is shown in Figure 31, Southwest didn't have a big number of cancellations during 2012. Out of 1,140,535 flights, 9580 flights were cancelled (0.84%) For MDW that statistics was similar: out of 158,515, 1368 were cancelled flights (0.86%).

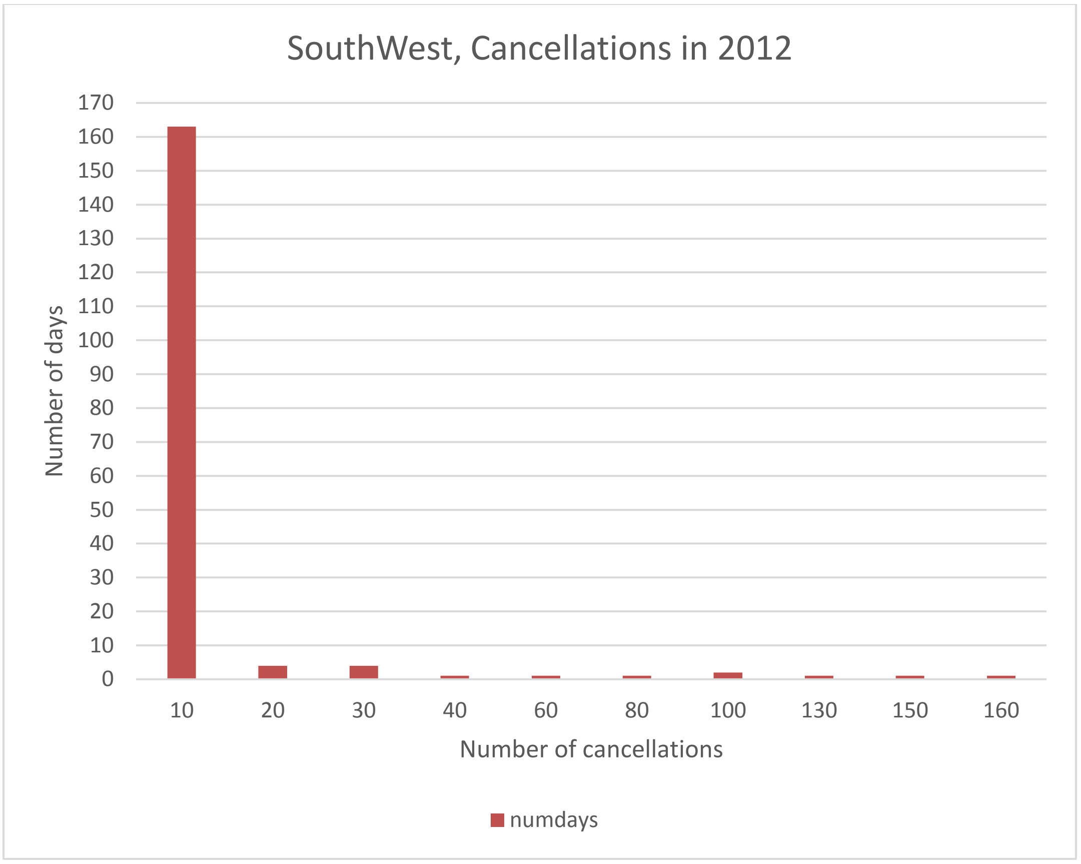


Figure 31 Cancellations per day, Southwest

Top cancellations events are shown in the table 31, the biggest event was on December 20th, when 160 flights were cancelled:

Table 31 SW, MDW, Top cancellation events

month	day	Number of cancelled flights

2	24	30
7	18	30
10	31	30
12	26	30
11	21	40
6	28	60
10	30	80
1	20	100
10	29	100
1	12	130
2	23	150
12	20	160

Baseline Rebooking (After Cancelled Flight)

The same as for Case Studies 2 and 3, for the existing re-booking paradigm, no passengers will be booked on preemptive flights, so the percent of passengers rebooked early will automatically be zero. The result is a slope of zero in the equation calculating the percent of passengers rebooked early. For similar reasons, the ANR and CTES slopes will also be zero.

Case study 4: *H_0: Preemptive rebooking does not accommodate more than 10% passengers on cancelled flights (Preemptive passengers / total cancel passengers < 10%)*

Based on the results that a projected 0.56%, as the upper bound of the passengers willing to be preemptively rebooked, will be accommodated it is shown that the hypothesis, Preemptive rebooking does not accommodate more than 10% passengers on cancelled flights (Preemptive passengers / total cancel passengers < 10%) is rejected.

During 2012 MDW had one rare event which had extremely big impact on the results. That was for November 21st. Because of the fact that this was an unusual event, it will be observe separately. The results for all days are given in the table 32.

Table 32 Preemptively rebooked passengers (m, R2)

Preemptively rebooked passengers					
	Same day Early			Prev+Same	
	m	R		m	R
20-Jan	0.65	1.00		0.68	1.00
23-Feb	0.80	1.00		0.80	1.00
7-Jun	0.69	1.00		0.69	1.00
18-Jul	0.37	0.99		0.69	1.00
21-Nov	0.37	0.99		1.29	1.00
20-Dec	0.75	1.00		1.64	1.00
26-Dec	0.75	1.00		0.57	1.00
μ	0.63			0.91	
σ	0.18			0.40	
μ/σ	3.5			2.27	

Case study 4: H0: The number of additional pax accommodated by adding an additional day of preemptive rebooking is greater than 25% of the original number of accommodated

For Case Study 4 this hypothesis is not rejected. November 21st started with a big number of cancelled flight early in the morning (5am). The cancellations during the day are at the noise level, as in the next figure.

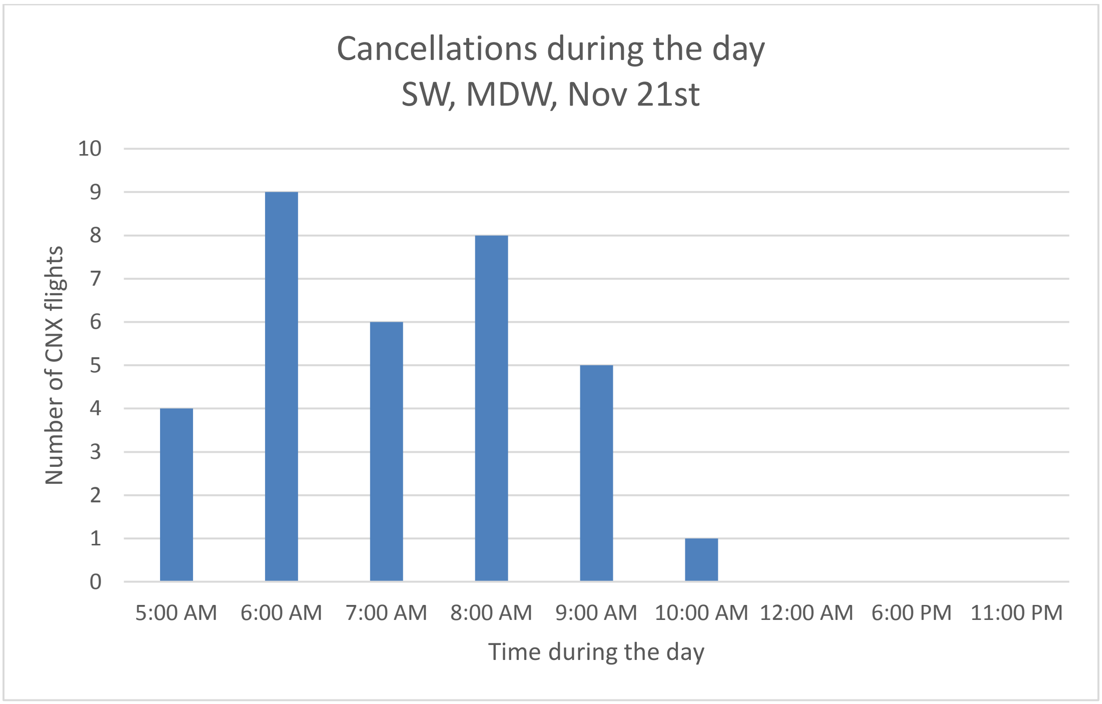


Figure 32 Cancellations during February 24th

In total, 33 cancelled flights affected 4664 passengers. The last cancelled fligh was at 11.30. Because of this situation, just a maximum of 37% of cancelled passengers willing to accept preemptive rebooking are able to be accommodated on the same day early (Figure 33) . On the other hand, previous day and same day later show good results.

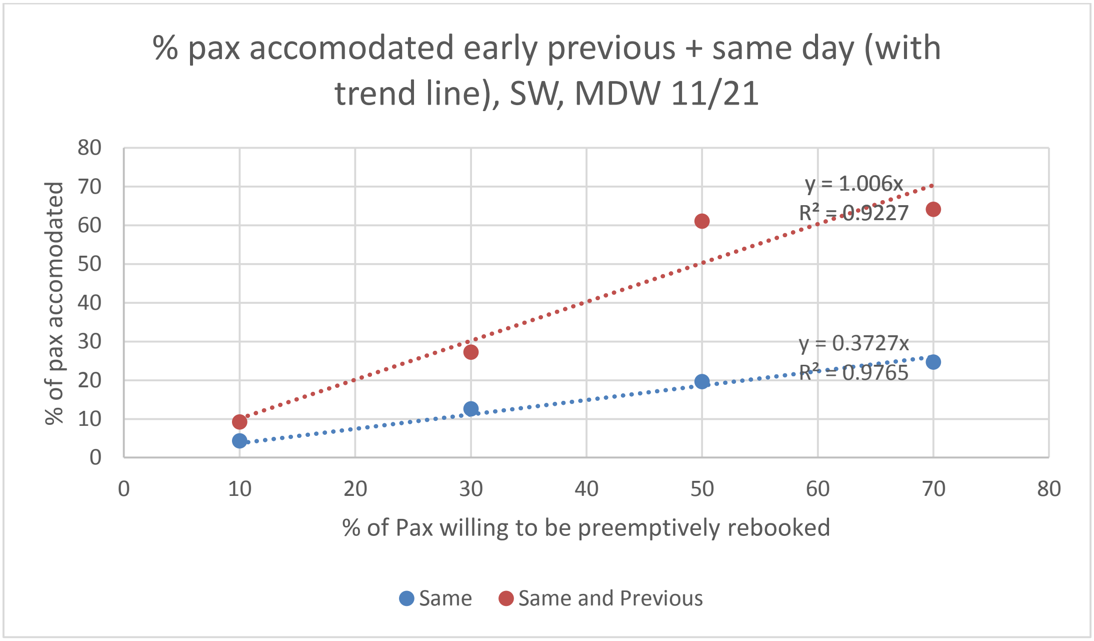


Figure 33 February 24, Same day early vs Previous day rebooked pax

Case Study 4: H0: Preemptive rebooking does not result in additional revenue to airlines

Based on the results that a projected 69% for upper bound for passengers accommodated on the same day which is the same like on the same + previous day (table 33), it is shown that hypothesis H_0 :The number of additional passengers accommodated by adding an additional day of preemptive rebooking is greater than 25% of the original number of accommodated is rejected.

Table 33 SW, MDW, Airfares Not Refund

Airfares Not Refund	
Same day Early	Same +Previous day

	m	R		m	R
20-Jan	0.74	1.00		0.78	1.00
23-Feb	0.79	1.00		0.80	1.00
7-Jun	0.89	1.00		0.89	1.00
18-Jul	0.89	1.00		0.89	1.00
21-Nov	0.29	0.60		1.32	0.89
20-Dec	0.82	1.00		0.95	1.00
26-Dec	0.53	1.00		0.59	1.00
μ	0.71		μ	0.70	
σ	0.22		σ	0.22	
μ/σ	3.21		μ/σ	3.98	

Case Study 4 H0: Preemptive rebooking does not result in additional saving to Corporates

This hypothesis can be rejected just for the previous + same day preemptive rebooking (0.69%). For the same day early it cannot be rejected, because of the situation that passengers cannot be rebooked for the same day early. CTES depends of extra cost for overnight stays. The CTES for November 21st is shown in the figure 34.

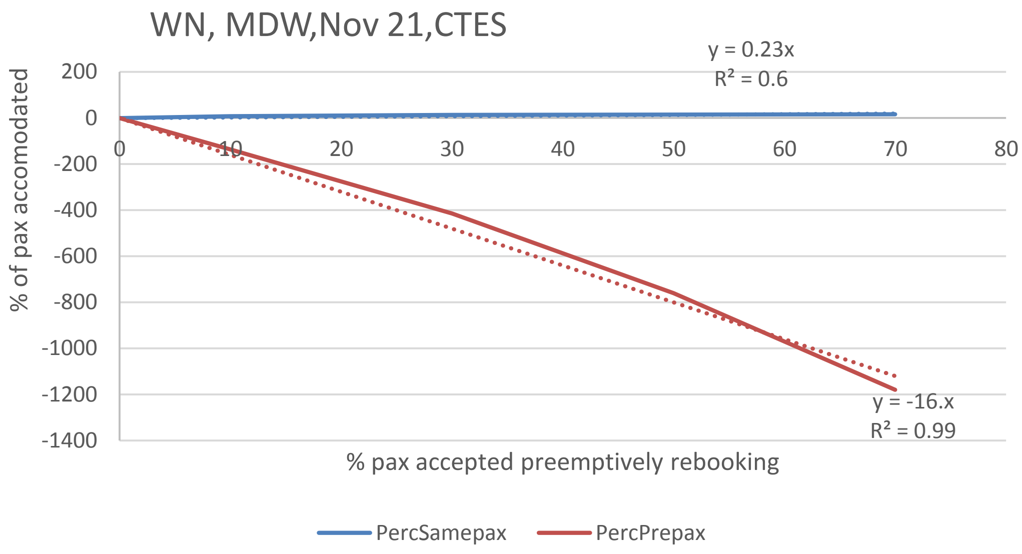


Figure 34 CTES, February 24

The result for CTES for 2012 for MDW is shown in the Table 34.

Table 34 SW, MDW, CTES

CTES					
Same day Early				Same +Previous day	
	M	R		M	R
20-Jan	0.70	1.00		0.74	1.00
23-Feb	0.79	1.00		0.79	1.00
7-Jun	0.88	1.00		0.89	1.00
18-Jul	0.88	1.00		0.89	1.00
21-Nov	-16.00	0.99		0.28	0.59
20-Dec	0.78	1.00		0.82	1.00
26-Dec	0.67	1.00		0.53	1.00
µ	-1.61		µ	0.70	
σ	6.35		σ	0.22	
µ/σ	-0.25		µ/σ	3.16	

Case Study 4: H_0: Preemptive rebooking does not result in reducing PTD and number of overnight and passenger which cannot be rebooked

Based on the results that number of overnight passengers is decreased by 54% and PTD is decreased by up to 53% (upper bound for 70% passengers willing to accept preemptively rebooking), it is shown that hypothesis H_0 : Preemptive rebooking does not and result in reducing PTD and number of overnight and passenger which cannot be rebooked is rejected.

As noted earlier, preemptive rebooking has the effect of reducing the number of overnight passengers and decreasing PTD. The relationship between stranded passengers (next day passengers and passengers that cannot be rebooked) and preemptive rebooking is represented by the following equation:

%Decrease in SPAX =0.76 • % Pax Pre-emptively Rebooked

Case Study 4: SW, MDW, Summary of results

This case study shows that if heavy cancellations occur early in the day, the number which can be rebooked same-day preemptive is minimal. In this case the biggest difference in the results show in the % of passengers accommodated same day early and CTES. The results are summarized in Table 35.

Table 35 Summary of results, Case Study 4

Analysis	Same Day Preemptive			Same Day+Previous Day Preemptivly		
	m_μ	m_σ	m_μ / m_σ	m_μ	m_σ	m_μ / m_σ

Slope for % Pax Rebooked	**0.63**	**0.18**	**3.48**	**0.91**	**0.40**	**2.28**
Slope for ANR	**0.71**	**0.22**	**3.21**	**0.89**	**0.22**	**3.98**
Slope for CTES	**-1.61**	**6.35**	**-0.25**	**0.70**	**0.22**	**3.16**

Relation of Preemptive rebooking acceptance rate to pax rebooked, ANR and CTES without Nov 21st (Table 36) :

Table 36 Analysis for the Case Study without the event

Analysis	**Same Day Preemptive**			**Same Day+Previous Day Preemptivly**		
	m_μ	m_σ	m_μ / m_σ	m_μ	m_σ	m_μ / m_σ
Slope for % Pax Rebooked	**0.67**	**0.15**	**4.35**	**0.85**	**0.4**	**2.13**
Slope for ANR	**0.77**	**0.13**	**5.83**	**0.93**	**0.13**	**6.48**
Slope for CTES	**0.78**	**0.09**	**8.8**	**0.77**	**0.13**	**5.87**

The strong impact of November 21st event shows that a big factor is the type of event (e.g. number of cancellations, when they start and how they are grouped).

Case Study 5

A case-study was conducted for the scheduled domestic flights for Southwest for all other hubs (except MDW, which is analyzed in the Case Study 4) for all one day cancellation events in 2012. In the study 10 airports are observed:

Airports:

- MDW
- LAS
- BWI
- DAL
- PHX
- DEN
- HOU
- ATL
- LAX
- MCO

For these 10 airports the biggest cancellation events are shown in the Table 37.

Table 37 Top cancelled event, SW

Month	Day	Cancellations:
1	12	126
1	20	116
2	3	165

2	23	150
4	3	71
6	28	67
7	18	90
8	26	62
12	10	63
12	17	270
12	20	201
12	26	86

Because of the big number of airports in the study just the events with more than 100 cancelled flights are observed.

Baseline Rebooking (After Cancelled Flight)

As in the previous studies, for the existing re-booking paradigm, no passengers will be booked on preemptive flights, so the percent of passengers rebooked early will automatically be zero. The result is a slope of zero in the equation calculating the percent of passengers rebooked early. For similar reasons, the ANR and CTES slopes will also be zero.

Case study 5: ***H_0: Preemptive rebooking does not accommodate more than 10% passengers on cancelled flights (Preemptive passengers / total cancel passengers < 10%)***

Based on the results that a projected 62% • % Pax Pre-emptively Rebooked for upper bound of the passengers willing to be preemptively rebooked, will be accommodated it is shown that the hypothesis Preemptive rebooking does not accommodate more than 10% passengers on cancelled flights (Preemptive passengers / total cancel passengers < 10%) is rejected. The relationship between percentage passengers accommodated and preemptive rebooking is represented by the following equation:

%Pax Accomodated = 0.62 • % Pax Pre-emptively Rebooked

Case study 5: H_0: The number of additional pax accommodated by adding <u>an additional day</u> of preemptive rebooking is greater than 25% of the original number of accommodated

Based on the results from the table below the upper bound for passengers accommodated on the same day is not greater than 25% , it is shown that hypothesis H_0 :The number of additional passengers accommodated by adding <u>an additional day</u> of preemptive rebooking is greater than 25% of the original number of accommodated is rejected (Table 38). Same day preemptively can accommodate 0.62% • % Pax Pre-emptively Rebooked and with Same+Previous day option for 0.17 is accommodate on the previous day (with the same, 0.62 on the same day).

Table 38 % pax accommodated Same vs Same + Previous Day

Preemptively rebooked passengers								
	Same day Early		Same +Previous day					
			Previous day		Same Day		Prev+Same	
	m	R	m	R	m	R	m	R

12-Jan	0.80	1.00		0.04	0.97	0.80	1.00	0.85	1.00
3-Feb	0.39	1.00		0.46	1.00	0.39	1.00	0.85	1.00
23-Feb	0.80	1.00		0.01	0.95	0.80	1.00	0.81	1.00
18-Jul	0.71	1.00		0.01	1.00	0.71	1.00	0.72	1.00
17-Dec	0.42	1.00		0.20	1.00	0.42	1.00	0.62	1.00
20-Dec	0.60	1.00		0.28	1.00	0.60	1.00	0.88	1.00
μ	0.62			0.17		0.62		0.79	
σ	0.18			0.18		0.18		0.10	
μ/σ	3.38			0.92		3.37		8.01	

Case Study 5: H0: Preemptive rebooking does not result in additional revenue to airlines:

Based on the results that airlines can increase their savings (ANF) shown in the table 39 it is shown that hypothesis H_0 Preemptive rebooking does not result in additional revenue to airlines is rejected.

Table 39 ANR, SW, 10 hubs (slope and R2)

Airfares Not Refund					
Same day Early				Same +Previous day	
	m	R		m	R
12-Jan	0.99	0.99		0.94	0.99
3-Feb	0.17	1.00		0.76	1.00
23-Feb	0.80	1.00		0.79	1.00
18-Jul	0.87	1.00		0.86	1.00
17-Dec	0.59	1.00		0.59	1.00

20-Dec	0.72	1.00		0.96	1.00
μ	0.69		μ	0.79	
σ	0.29		σ	0.13	
μ/σ	2.40		μ/σ	6.11	

The relationship between Airfares Not Refunded (ANR) and preemptive rebooking is represented by the following equation:

%Decrease in Airfares Refunded = 0.69 • % Pax Pre-emptively Rebooked

Case Study 5 H0: Preemptive rebooking does not result in additional saving to Corporates:

Based on the results that increase is Corporate Travel Expense savings for 74% for the same day early, it is shown that hypothesis H_0 : Preemptive rebooking does not result in additional savings to corporates is rejected.

As it is said, preemptive rebooking has the effect of reducing the number of overnight passengers that could accrue unbudgeted Corporate Travel Expenses. The relationship between Corporate Travel Expense Savings (CNES) and preemptive rebooking is represented by the following equation:

%Increase in CNES = 0.74 • % Pax Pre-emptively Rebooked

Table 40 CTES, SW, All hubs

CTES					
Same day Early				Same +Previous day	
	m	R		m	R
12-Jan	0.94	0.99		0.84	0.98
3-Feb	0.17	1.00		3.37	1.00

23-Feb	0.79	1.00		0.79	1.00
18-Jul	0.86	1.00		0.85	1.00
17-Dec	0.99	1.00		0.37	1.00
20-Dec	0.72	1.00		1.40	1.00
μ	0.74		μ	1.27	
σ	0.30		σ	1.08	
μ/σ	2.50		μ/σ	1.18	

Case Study 5: H_0: Preemptive rebooking does not result in reducing PTD and number of overnight and passenger which cannot be rebooked:

Based on the results that show PTD is decreased up56 % • % Pax Pre-emptively Rebooked (upper bound for 70% passengers willing to accept preemptively rebooking), it is shown that hypothesis H_0 : Preemptive rebooking does not and result in reducing PTD and number of overnight and passenger which cannot be rebooked is rejected. Preemptive rebooking has the effect of reducing the number of overnight passengers and decreasing PTD (Table 41).

The relationship between stranded passengers (next day passengers and passengers that cannot be rebooked) and preemptive rebooking is represented by the following equation:

%Decrease in SPAX =0.64 • % Pax Pre-emptively Rebooked

The relationship between PTD and preemptive rebooking is represented by the following equation:

%Decrease in SPAX =0.56 • % Pax Pre-emptively Rebooked

Table 41 Overnight and Remaining passengers and PTD statistics

Overnight&Remaining Passengers					
Same day Early				Same +Previous day	
	m	R		m	R
12-Jan	0.94	0.99		0.76	0.99
3-Feb	0.17	1.00		0.80	1.00
23-Feb	0.79	1.00		0.80	1.00
18-Jul	0.86	1.00		0.87	1.00
17-Dec	0.39	1.00		0.59	1.00
20-Dec	0.72	1.00		0.96	1.00
μ	0.64		μ	0.76	
σ	0.30		σ	0.10	
μ/σ	2.15		μ/σ	7.47	
PTD					
Same day Early				Same +Previous day	
	m	R		m	R
12-Jan	0.79	1.00		0.85	1.00
3-Feb	0.17	1.00		0.75	1.00
23-Feb	0.73	1.00		0.74	1.00
18-Jul	0.72	1.00		0.74	1.00
17-Dec	0.37	1.00		0.56	1.00
20-Dec	0.57	1.00		0.89	1.00
μ	0.56		μ	0.73	
Σ	0.24		σ	0.10	
μ/σ	2.30		μ/σ	6.99	

Case Study 5:SW, Summary of results

Results for the case study are summarized in the Table 42. Compering with the summary of the case study 3 (UA, all hubs) statistics for the same day preemptive are within 0.07 for percentage of passengers rebooked, 0.09 for ANR and 0.04 for CTES.

Signal to noise ratio is similar to the statistics for the same day preemptively case study 3.

Table 42 SW, Sumary of Results

Analysis	**Same Day Preemptive**			**Same Day+Previous Day Preemptively**		
	m_μ	**m_σ**	**m_μ/ m_σ**	**m_μ**	**m_σ**	**m_μ/ m_σ**
Slope for % Pax Rebooked	**0.62**	**0.18**	**3.38**	**0.79**	**0.10**	**8.01**
Slope for ANR	**0.69**	**0.29**	**2.40**	**0.79**	**0.13**	**6.11**
Slope for CTES	**0.74**	**0.30**	**2.50**	**1.27**	**1.08**	**1.18**

5. CONCLUSION AND FUTURE WORK

Although the large scale airline flight cancellations events are infrequent, they have significant impact on airline revenue, corporate travel expenses and passengers travel costs and inconvenience. In many cases these big events are reliably forecast enough in advance, so that it is possible to facilitate a pro-active response from the airlines. In this dissertation a new rebooking methodology, preemptive rebooking, was proposed as an option to accommodate airline passengers in case of flight cancellations. A Monte Carlo simulator was develop by enhancements of an existing code and additions of significant new modules for realistic modeling of airline rebooking, and then used to perform comprehensive analysis, using actual airline data, to validate the expected benefits of the proposed approach.

Significant contributions are described in chapter III, a PTD Calculator embedded in a Monte Carlo Simulation, which analyzes the impact of preemptive rebooking strategies. In the model passengers are divided into two big groups: passengers on the cancelled flights, and passengers on not cancelled flights. This model analyzes the first group, which is then divided into: passengers that can be rebooked (traditional rebooking model: rebooked on *same day later* or on the first available flight on the next day; and preemptive model: same day before the scheduled departure time, previous + same day before, same day later and next day). The rebooking algorithm takes the available itinerary information and uses it to make rebooking decisions. It selects disrupted itineraries one by one, over all of the passengers that need to be rebooked.

Those passengers are split into groups depending on whether they are willing to accept preemptive rebooking or not. For as long as passengers remain in need of rebooking, and for as long as alternate flights remain, remaining passengers are placed in empty seats.

Rebooking alternatives are considered in order beginning with flights departing prior to, but on the same day as, the original scheduled departure of the flight, followed by flights departing on the previous day, followed by flights departing after, but on the same day, followed by flights departing on the next day. Additionally, in each group, alternate flights on different airlines were considered after all alternates on the same airline were exhausted. The capability to handle the order is new to the algorithm.

Due to the fact that some passengers rebook preemptively and some do not, a binomial distribution calculator was added to the code, and used to randomly determine the specific passengers which would accept preemptive rebooking for a given percentage of passengers willing to be rebooked. Passengers are rebooked First Come Fist Serve and randomization is done in Monte Carlo simulation.

In Chapter IV detailed case studies with realistic industrial data showed that preemptive rebooking in advance of forecast large-scale cancellation events is feasible and that decrease in PTD and number of overnight and passenger which cannot be rebooked is up to 50%, and a decrease in Remaining passengers is up to 55 %. Previous day rebooking is not required as a majority of the passengers (>90%) can be accommodated earlier on the same day.

The number of rebooked passenger on previous day, and same and next day, depends on several factors:

1) How many cancellations were on the previous/same day;
2) When the cancellations occur (morning, midday, afternoon);
3) The size of an event (1day or more days event);
4) Number of flights previous day/same day/next day;
5) Number of available itineraries and the type of itineraries (number of “2-day” itineraries;
6) Load factors;
7) Frequency;

There is also an opportunity for airlines to generate additional revenue by offering a for-fee option that would move passengers to the front of the preemptive rebooking queue in the event of a large scale event (Raiteri, 2015).

In a future work it would be analyzed complexities that maybe applied in developing the model, for example rare extreme multi day events, how different load factors can influent results, and also considering the case that only partial information on cancelled flights is available (e.g. preemptive rebooking with possibility of unexpected cancellations) and increase fidelity of airfares refund model and corporate travel expanse model.

The general premise that the ability to mass communicate with individual clients has become feasible is one that can be implied in more industries than in aviation. Is worth investigating whether concept similar to preemptive rebooking can be used with significant gain in other areas.

APPENDIX I

1) Stored procedure Binomial Random, which selects binomially distributed random values:

```
USE [PaxDelay]
GO
/****** Object:  StoredProcedure [dbo].[binomial_rand] ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

-- =============================================
-- Author:              <Sanja,Avramovic>
-- Create date: <12/6/14>
-- Description:  <This function selects binomially distributed random value>
/*
 *  Binomial sampler function, SQL version derived from R library
 *  Copyright (C) 1998 Ross Ihaka
 *  Copyright (C) 2000-2014 The R Core Team
 *  Copyright (C) 2007 The R Foundation
 *  Copyright (C) 2014 Sanja Avramovic
 *
 *  This program is free software; you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation; either version 2 of the License, or
 *  (at your option) any later version.
 *
 *  This program is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with this program; if not, a copy is available at
 *  http://www.r-project.org/Licenses/
 *
 *  SYNOPSIS
 *
 *      binomial_rand @nin int, @pp float
 *  returns int
 *
 *  DESCRIPTION
 *
 *      Random variates from the binomial distribution.
 *
 *  REFERENCE
 *
 *      Kachitvichyanukul, V. and Schmeiser, B. W. (1988).
 *      Binomial random variate generation.
 *      Communications of the ACM 31, 216-222.
 *      (Algorithm BTPEC).
 */
-- =============================================
Create PROCEDURE [dbo].[binomial_rand]
(
         @nin int,
        @pp  float
```

```
)
as
begin

        set nocount on;

        declare @c    float;
        declare @fm   float;
        declare @npq  float;
        declare @p1   float;
        declare @p2   float;
        declare @p3   float;
        declare @p4   float;
        declare @qn   float;

        declare @xl   float;
        declare @xll  float;
        declare @xlr  float;
        declare @xm   float;
        declare @xr   float;

        declare @psave float;
        declare @nsave int;
        declare @m int;

        set @psave = -1;
        set @nsave = -1;

        declare @f    float;
        declare @f1   float;
        declare @f2   float;
        declare @u    float;
        declare @v    float;
        declare @w    float;
        declare @w2   float;
        declare @x    float;
        declare @x1   float;
        declare @x2   float;
        declare @z    float;
        declare @z2   float;

        declare @p    float;
        declare @q    float;
        declare @np   float;
        declare @g    float;
        declare @r    float;
        declare @al   float;
        declare @alv  float;
        declare @amaxp float;
        declare @ffm  float;
        declare @ynorm float;

        declare @i  int;
        declare @ix int;
        declare @k  int;
        declare @n  int;

    --if (!R_FINITE(nin)) ML_ERR_return_NAN;
    set @r = @nin;
    --if (r != nin) ML_ERR_return_NAN;
    --if (!R_FINITE(pp) ||
        /* n=0, p=0, p=1 are not errors <TSL>*/
        if (@r < 0 OR @pp < 0. OR @pp > 1.)      return -1;

    if (@r = 0 OR @pp = 0.) return 0;
    if (@pp = 1.) return @nin;

    --if (r >= INT_MAX)/* evade integer overflow, and r == INT_MAX gave only even values */
        --return qbinom(unif_rand(), r, pp, /*lower_tail*/ 0, /*log_p*/ 0);
    /* else */
    set @n = @nin;

        set @p = @pp;
        if (1-@pp < @pp) set @p = 1-@pp;
    set @q = 1. - @p;
    set @np = @n * @p;
    set @r = @p / @q;
    set @g = @r * (@n + 1);

    /* Setup, perform only when parameters change [using static (globals): */

    /* FIXING: Want this thread safe
       -- use as little (thread globals) as possible
    */
    if (@pp != @psave OR @n != @nsave) begin
                set @psave = @pp;
                set @nsave = @n;
                if (@np < 30.0) begin
            /* inverse cdf logic for mean less than 30 */
                        set @qn = power(@q, @n);
                        goto L_np_small;
                end -- if (@np < 30.0)
                else begin
                        set @ffm = @np + @p;
                        set @m = cast(@ffm as int);
                        set @fm = @m;
                        set @npq = @np * @q;
                        set @p1 = cast ((2.195 * sqrt(@npq) - 4.6 * @q) as int) + 0.5;
                        set @xm = @fm + 0.5;
                        set @xl = @xm - @p1;
                        set @xr = @xm + @p1;
                        set @c = 0.134 + 20.5 / (15.3 + @fm);
                        set @al = (@ffm - @xl) / (@ffm - @xl * @p);
                        set @xll = @al * (1.0 + 0.5 * @al);
                        set @al = (@xr - @ffm) / (@xr * @q);
                        set @xlr = @al * (1.0 + 0.5 * @al);
                        set @p2 = @p1 * (1.0 + @c + @c);
                        set @p3 = @p2 + @c / @xll;
                        set @p4 = @p3 + @c / @xlr;
                end -- if (@np < 30.0) else
    end -- if (@pp != @psave || @n != @nsave)
        else if (@n = @nsave) begin
                if (@np < 30.0) goto L_np_small;
    end -- if (@n = @nsave)

    /*-------------------------- np = n*p >= 30 : ------------------- */
    while 1=1 begin
                set @u = rand() * @p4;
                set @v = rand();
      /* triangular region */
                if (@u <= @p1) begin
                        set @ix = cast((@xm - @p1 * @v + @u) as int);
                        goto finis;
                end -- if (@u <= @p1)
      /* parallelogram region */
                if (@u <= @p2)  begin
                        set @x = @xl + (@u - @p1) / @c;
                        set @v = @v * @c + 1.0 - abs(@xm - @x) / @p1;
                        if (@v > 1.0 OR @v <= 0.) continue;
                        set @ix = cast(@x as int);
```

```
                end -- if (@u <= @p2)
                else begin
                        if (@u > @p3) begin     /* right tail */
                                set @ix = cast((@xr - log(@v) / @xlr) as int);
                                if (@ix > @n) continue;
                                set @v = @v * (@u - @p3) * @xlr;
                        end -- if (@u > @p3)
                        else begin /* left tail */
                                set @ix = cast((@xl + log(@v) / @xll) as int);
                                if (@ix < 0) continue;
                                set @v = @v * (@u - @p2) * @xll;
                        end -- if (@u > @p3) else
                end -- if (@u <= @p2) else
      /* determine appropriate way to perform accept/reject test */
                set @k = abs(@ix - @m);
                if (@k <= 20 or @k >= @npq / 2 - 1) begin
          /* explicit evaluation */
                        set @f = 1.0;
                        if (@m < @ix) begin
                                set @i = @m + 1;
                                while @i <= @ix begin
                                        set @f = @f * (@g / @i - @r);
                                        set @i = @i + 1;
                                end -- while @i <= @ix
                        end -- if (@m < @ix)
                        else if (@m != @ix) begin
                                set @i = @ix + 1;
                                while @i <= @m begin
                                        set @f = @f / (@g / @i - @r);
                                        set @i = @i + 1;
                                end -- while @i <= @m
                        end -- if (@m != @ix)
                        if (@v <= @f) goto finis;
                end -- if (@k <= 20 || @k >= @npq / 2 - 1)
                else begin
          /* squeezing using upper and lower bounds on log(f(x)) */
                        set  @amaxp  =  (@k  /  @npq)  *  ((@k  *  (@k  /  3.  +  0.625)  +
0.1666666666666) / @npq + 0.5);
                        set @ynorm = -@k * @k / (2.0 * @npq);
                        set @alv = log(@v);
                        if (@alv < @ynorm - @amaxp) goto finis;
                        if (@alv <= @ynorm + @amaxp) begin
              /* stirling's formula to machine accuracy */
              /* for the final acceptance/rejection test */
                                set @x1 = @ix + 1;
                                set @f1 = @fm + 1.0;
                                set @z = @n + 1 - @fm;
                                set @w = @n - @ix + 1.0;
                                set @z2 = @z * @z;
                                set @x2 = @x1 * @x1;
                                set @f2 = @f1 * @f1;
                                set @w2 = @w * @w;
                                if (@alv <= @xm * log(@f1 / @x1) + (@n - @m + 0.5) * log(@z
/ @w) + (@ix - @m) * log(@w * @p / (@x1 * @q)) + (13860.0 - (462.0 - (132.0 - (99.0 - 140.0
/ @f2) / @f2) / @f2) / @f2) / @f1 / 166320.0 + (13860.0 - (462.0 - (132.0 - (99.0 - 140.0
/ @z2) / @z2) / @z2) / @z2) / @z / 166320.0 + (13860.0 - (462.0 - (132.0 - (99.0 - 140.0 /
@x2) / @x2) / @x2) / @x2) / @x1 / 166320.0 + (13860.0 - (462.0 - (132.0 - (99.0 - 140.0 /
@w2) / @w2) / @w2) / @w2) / @w / 166320.)
                                        goto finis;
                        end -- if (@alv <= @ynorm + @amaxp)
                end -- if (@k <= 20 || @k >= @npq / 2 - 1) else
        end -- while 1=1

  L_np_small:
    /*---------------------- np = n*p < 30 : ------------------------ */
```

```
                while 1=1 begin
                        set @ix = 0;
                        set @f = @qn;
                        set @u = rand();
                        while 2=2 begin
                                if (@u < @f) goto finis;
                                if (@ix > 110) break;
                                set @u = @u - @f;
                                set @ix = @ix + 1;
                                set @f = @f * (@g / @ix - @r);
                        end -- while 2=2
                end -- while 1=1

          finis:
                if (@psave > 0.5)
                        set @ix = @n - @ix;
                return @ix;

        end
```
8)

2) Sored procedure Compute Total Pax, this procedure computes and returns the total number of passengers on a chosen origin/destination’s flight:

```
USE [PaxDelay]
GO
/****** Object:  StoredProcedure [dbo].[compute_total_pax]     Script Date: 3/15/2015 2:39:19 PM
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

-- =============================================
-- Author:              <Sanja Avramovic>
-- Create date: <10/22/2014>
-- Description: computes and returns the total
-- number of pax on a chosen origin/dest/flight
-- on a chosen day.
-- Based on Guermo's and Ashwin's code
-- =============================================
CREATE PROCEDURE [dbo].[compute_total_pax]
(
         @origin  varchar(3),
         @dest    varchar(3),
         @airline  varchar(3),
         @fl_num smallint,


         @start_date smalldatetime
)

AS
BEGIN
        set nocount on

        -- Declare the return variable here
        DECLARE @result int
```

```sql
        -- count pax for one day, so the end date is one day ahead
        DECLARE @end_date smalldatetime
        set @end_date =  dateadd(day,1,@start_date)

        -- sum all pax with a matching origin, dest, airline, and flight
        -- count the totals for all possibilities (origin/hub, hub/dest, and origin/dest)
        set @result =
        (select sum(a.total+b.total+c.total)
                                from
                                        (
                                                select isnull(sum(max_pax),0) as total
                                                from MasterTable
                                                where
                                                        origin = @origin
                                                and dest = @dest
                                                and airline = @airline
                                                and fl_1_num = @fl_num
                                                and hub is null
                                                and fl_1_sch_dep >= @start_date
                                                and fl_1_sch_dep < @end_date
                                        )a, -- Contains all the direct flights
                                        (
                                                select isnull(sum(max_pax),0) as total
                                                from MasterTable
                                                where
                                                        hub = @origin
                                                and dest = @dest
                                                and airline = @airline
                                                and (fl_1_num = @fl_num or fl_2_num = @fl_num)
                                                and fl_1_sch_dep >= @start_date
                                                and fl_1_sch_dep < @end_date
                                        )b,     -- Contains all the flights having b.hub = a.origin
and b.dest = a.dest
                                        (
                                                select isnull(sum(max_pax),0) as total
                                                from MasterTable
                                                where
                                                        origin = @origin
                                                and hub = @dest
                                                and airline = @airline
                                                and (fl_1_num = @fl_num or fl_2_num = @fl_num)
                                                and fl_1_sch_dep >= @start_date
                                                and fl_1_sch_dep < @end_date
                                        )c
        )
        -- Return the result of the function
        RETURN  @result

END
```

3) Stored Procedure Create AOTP matches pairs of flights to get 2-leg itineraries

```sql
USE [PaxDelay]
GO

/****** Object:  StoredProcedure [dbo].[Create_aotp]    Script Date: 3/15/2015 2:42:10 PM ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
```

```
-- ============================================
-- Author:              <Sanja Avramovic>
-- Create date: <10/15/2014>
-- Description:  create table ##aotp
-- Matches pairs of flights to get 2-leg itineraries.
-- Based on Guermo's and Ashwin's code
-- ============================================
CREATE PROCEDURE  [dbo].[Create_aotp]

AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;

        -- create the global temporary table
        create table ##aotp(
                origin varchar(3) not null,
                hub varchar(3),
                dest varchar(3) not null,
                airline varchar(10) not null,
                Flight_1_number int not null,
                Flight_1_Sch_Dep smalldatetime,Flight_1_Sch_Arr smalldatetime,
                Flight_1_Act_Dep smalldatetime,Flight_1_Act_Arr smalldatetime,
                Flight_1_Status bit not null,Flight_1_Div bit not null,
                Flight_1_Avg_Seat int,Flight_1_LF float,

                Flight_2_number int ,
                Flight_2_Sch_Dep smalldatetime,Flight_2_Sch_Arr smalldatetime,
                Flight_2_Act_Dep smalldatetime,Flight_2_Act_Arr smalldatetime,
                Flight_2_Status bit,Flight_2_Div bit,
                Flight_2_Avg_Seat int,Flight_2_LF float,

                --Total_flights int
        );

        -- populate the table from ##ontime_lf, using pairs of flights
        insert into ##aotp(
                origin,hub,dest,airline,

                Flight_1_number,
                Flight_1_Sch_Dep,Flight_1_Sch_Arr,
                Flight_1_Act_Dep,Flight_1_Act_Arr,
                Flight_1_Status,Flight_1_Div,
                Flight_1_Avg_Seat,Flight_1_LF,

                Flight_2_number,
                Flight_2_Sch_Dep,Flight_2_Sch_Arr,
                Flight_2_Act_Dep,Flight_2_Act_Arr,
                Flight_2_Status,Flight_2_Div,
                Flight_2_Avg_Seat,Flight_2_LF
        )

        select  distinct m.origin, m.dest, k.dest ,m.airline,

                m.flight_num,
                m.sch_dep,m.sch_arr,
                m.act_dep,m.act_arr,
                m.cancelled,m.div_delay,
                m.avgseat,m.lf,

                k.flight_num,
                k.sch_dep,k.sch_arr,
                k.act_dep,k.act_arr,
                k.cancelled,k.div_delay,
                k.avgseat,k.lf
                --count (*) -- Remove total_flights if we dont need it for connecting flights

        from ##ontime_lf m, ##ontime_lf k
        where -- Here I am setting the conditions
                k.airline = m.airline
                and k.origin = m.dest -- Origin of the second flight = Destination of the first
flight
                and k.dest  <> m.origin -- Destination of the second flight cannot be origin of the
first flight.
                and k.sch_dep > = m.sch_arr_1
                and k.sch_Dep < = m.sch_arr_2

        group by m.origin,m.dest,k.dest,m.airline,

        m.flight_num,m.sch_dep,m.sch_arr,m.act_dep,m.act_arr,m.cancelled,m.div_delay,m.avgseat,m.l
f,

        k.flight_num,k.sch_dep,k.sch_Arr,k.act_dep,k.act_arr,k.cancelled,k.div_delay,k.avgseat,k.l
f
        order by m.origin,m.dest,k.dest,m.airline,m.sch_dep,m.sch_arr,k.sch_dep,k.sch_arr;

        -- create indexes for the table
        create index IDX_ORIGIN on ##aotp(origin);
        create index IDX_HUB on ##aotp(hub);
        create index IDX_DEST on ##aotp(DEST);
        create index IDX_AIRLINES on ##aotp(airline);
        create index IDX_FLNUM_1 on ##aotp(flight_1_number)
        create index IDX_FL_1_SCH_DEP on ##aotp(flight_1_sch_dep);
        create index IDX_FL_1_SCH_ARR on ##aotp(flight_1_sch_arr);
        create index IDX_FLNUM_2 on ##aotp(flight_2_number)
        create index IDX_FL_2_SCH_DEP on ##aotp(flight_2_sch_dep);
        create index IDX_FL_2_SCH_ARR on ##aotp(flight_2_sch_arr);

END

GO
```

4) Stored Procedure create the table DB1B, which contains passeners' statistics for origin/hub/destination tuples

```
USE [PaxDelay]
GO

/****** Object:  StoredProcedure [dbo].[Create_DB1B]    Script Date: 3/15/2015 7:11:39 PM ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:               <Sanja Avramovic>
-- Create date: <10/15/2014>
-- Description:  create the table DB1B,
-- which contains pax statistics for
-- origin/hub/destination tuples.
-- Based on Guermo's and Ashwin's code
-- =============================================
create PROCEDURE [dbo].[Create_DB1B]
                @start_date smalldatetime

AS
BEGIN
```

```
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;

        -- extract the year and quarter of the year from the date
        declare @year as smallint
        declare @quarter as tinyint

        set @year = year(@start_date)
        set @quarter = (month(@start_date) - 1) / 3 + 1

        ----------------------CREATING #DB1B---------------------------------------------------

        -- drop the table if it already exists
        IF OBJECT_ID('dbo.db1b', 'U') IS NOT NULL
    DROP TABLE dbo.db1b;

        -- create the table
        create table db1b(
                origin varchar(3) not null,
                dest varchar(3) not null,
                hub varchar(3),
                airport_group nvarchar(60),
                reporting_carrier nvarchar(3),
                total_pax int,
                pax_perday int,
                market_coupons int
        );

        -- populate the table from db1b market
        insert                                      into                                        db1b
(origin,dest,hub,airport_group,reporting_carrier,total_pax,pax_perday,market_coupons)

        select
                b.origin, b.dest,
                case when len(b.airport_group) > 7
                        then -- this is a dirty trick to get the hub! remember to do it better
later
                                substring(
                                        right(b.airport_group,
                                        len(b.airport_group)-charindex(':',b.airport_group)),
                                        1,
                                        charindex(':',b.airport_group)-1)
                        else '-'
                end as hub,
                b.airport_group,
                b.reporting_carrier,
                        --b.year,
                        --b.quarter,
                b.ten_percent_total_pax_quaterly *10,
                b.avg_pax_perday_perairline_rounded,b.market_coupons
                --,b.avg_pax_perday_perairline
        from
        (
                select
                        a.market_coupons,a.origin,a.dest,
                        a.airport_group,a.reporting_carrier,
                        a.year,a.quarter,
                        sum(a.passengers) as ten_percent_total_pax_quaterly,
                        (sum(a.passengers)*10/(3*30)) as avg_pax_perday_perairline_rounded
                from db1b_market a
                where
                        a.market_coupons <=2
                        and a.year = @year
                        and a.quarter = @quarter
                group                                                                           by
a.origin,a.dest,a.airport_group,a.reporting_carrier,a.year,a.quarter,a.market_coupons

        )b
        where
                b.avg_pax_perday_perairline_rounded > 0.5
        order by origin, dest, hub, reporting_carrier;

        -- create indexes
        create index IDX_ORIGIN  on db1b(origin);
        create index IDX_DEST    on db1b(DEST);
        create index IDX_HUB     on db1b(hub);
        create index IDX_AIRLINES        on db1b(reporting_carrier);
        create index IDX_MARKET_COUPONS  on db1b(MARKET_COUPONS);

END
GO
```

5) Create stored procedure Create Final which create the table Final,to store the final itinerary/PTD results

```
USE [PaxDelay]
GO

/****** Object:  StoredProcedure [dbo].[Create_Final]    Script Date: 3/15/2015 7:19:07 PM ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

-- =============================================
-- Author:              <Sanja Avramovic>
-- Create date: <10/15/2014>
-- Description:  create the table Final,
-- to store the final itinerary/PTD results
-- Based on Guermo's and Ashwin's code
-- =============================================
create PROCEDURE  [dbo].[Create_Final]
        -- Add the parameters for the stored procedure here

AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;

        -- drop the table if it already exists
        IF OBJECT_ID('dbo.Final', 'U') IS NOT NULL
    DROP TABLE dbo.Final;

        -- create the table
        create table Final(
                ORIGIN varchar (3) not null,
                DEST varchar(3) not null,
                HUB varchar(3),
                AIRLINE varchar(3) not null,
                MAX_PAX smallint,
                FL_1_NUM smallint,
                FL_1_SCH_DEP smalldatetime,
                FL_1_SCH_ARR smalldatetime,
                FL_1_ACT_DEP smalldatetime,
                FL_1_ACT_ARR smalldatetime,
                FL_1_CANCELLED bit,
                FL_1_DIVERTED bit,
                FL_1_AVG_SEAT smallint,
                FL_1_LF float,

                FL_2_NUM smallint,
                FL_2_SCH_DEP smalldatetime,
                FL_2_SCH_ARR smalldatetime,
                FL_2_ACT_DEP smalldatetime,
                FL_2_ACT_ARR smalldatetime,
                FL_2_CANCELLED bit,
                FL_2_DIVERTED bit,
                FL_2_AVG_SEAT smallint,
                FL_2_LF float,

                PTD int,
                t100_max_pax smallint
        );

END
GO
```

6) Create stored procedure Create MasterTable, which contains itinerary and PTD information.

```
USE [PaxDelay]
GO

/****** Object:  StoredProcedure [dbo].[Create_MasterTable]    Script Date: 3/15/2015 7:24:03 PM
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

-- =============================================
-- Author:              <Sanja Avramovic>
-- Create date: <10/15/2014>
-- Description: create table MasterTable,
-- which contains itinerary and PTD info.
-- Requires a date, and the number of days to run.
-- Based on Guermo's and Ashwin's code
-- =============================================
create PROCEDURE  [dbo].[Create_MasterTable]
        @start_date smalldatetime,
        @num_days tinyint

AS
BEGIN

        SET NOCOUNT ON;

        -- set the end date based on the start date and the number of days to run
        declare @end_date as smalldatetime
        declare @days as int
        set @end_date = dateadd(dd,@num_days,@start_date)
        set @days = @num_days --DATEDIFF(dd, @start_date, @end_date) -- Set the number of days
according to the data needed to be collected

        -- a parameter specifying the number of dummy pax to add
        declare @additional_pax as int
        set @additional_pax = 0

        -- drop the table if it already exists
        IF OBJECT_ID('dbo.MasterTable', 'U') IS NOT NULL
    DROP TABLE dbo.MasterTable;

        -- create the master table
        create table MasterTable(
                ORIGIN varchar (3) not null, DEST varchar(3) not null, HUB varchar(3), AIRLINE
varchar(3) not null, MAX_PAX smallint,
                FL_1_NUM  smallint,  FL_1_SCH_DEP  smalldatetime,  FL_1_SCH_ARR  smalldatetime,
FL_1_ACT_DEP smalldatetime, FL_1_ACT_ARR smalldatetime,
                FL_1_CANCELLED bit, FL_1_DIVERTED bit, FL_1_AVG_SEAT smallint, FL_1_LF float,
                FL_2_NUM  smallint,  FL_2_SCH_DEP  smalldatetime,  FL_2_SCH_ARR  smalldatetime,
FL_2_ACT_DEP smalldatetime, FL_2_ACT_ARR smalldatetime,
                FL_2_CANCELLED bit, FL_2_DIVERTED bit, FL_2_AVG_SEAT smallint, FL_2_LF float,
                PTD int, t100_max_pax smallint
        );

        -- populate all connecting itineraries by merging AOTP and DB1B
        insert into MasterTable
        (
                ORIGIN,DEST,HUB,AIRLINE,MAX_PAX,

        FL_1_NUM,FL_1_SCH_DEP,FL_1_SCH_ARR,FL_1_ACT_DEP,FL_1_ACT_ARR,FL_1_CANCELLED,FL_1_DIVERTED,
FL_1_AVG_SEAT,FL_1_LF,

        FL_2_NUM,FL_2_SCH_DEP,FL_2_SCH_ARR,FL_2_ACT_DEP,FL_2_ACT_ARR,FL_2_CANCELLED,FL_2_DIVERTED,
FL_2_AVG_SEAT,FL_2_LF
        )
        select
                d.origin, d.dest, d.hub, d.reporting_carrier, pax_perday as max_pax,
                d.Flight_1_number,  d.Flight_1_Sch_Dep,  d.Flight_1_Sch_Arr,  d.Flight_1_Act_Dep,
d.Flight_1_Act_Arr,
                d.Flight_1_Status, d.Flight_1_Div, d.Flight_1_Avg_Seat, d.Flight_1_LF,
                d.Flight_2_number,  d.Flight_2_Sch_Dep,  d.Flight_2_Sch_Arr,  d.Flight_2_Act_Dep,
d.Flight_2_Act_Arr,
                d.Flight_2_Status, d.Flight_2_Div, d.Flight_2_Avg_Seat, d.Flight_2_LF
        from
        (
                select
                        db1b.origin, db1b.hub, db1b.dest, db1b.reporting_carrier,

                        aotp.Flight_1_number,
                        aotp.Flight_1_Sch_Dep,    aotp.Flight_1_Sch_Arr,    aotp.Flight_1_Act_Dep,
aotp.Flight_1_Act_Arr,
                        aotp.Flight_1_Status, aotp.Flight_1_div,
                        aotp.Flight_1_Avg_Seat, aotp.Flight_1_lf,

                        aotp.Flight_2_number,
                        aotp.Flight_2_Sch_Dep,    aotp.Flight_2_Sch_Arr,    aotp.Flight_2_Act_Dep,
aotp.Flight_2_Act_Arr,
                        aotp.Flight_2_Status, aotp.Flight_2_div,
                        aotp.Flight_2_Avg_Seat, aotp.Flight_2_lf ,

                        db1b.pax_perday
                from
                (
                        select origin,hub,dest,reporting_Carrier,pax_perday
                        from db1b
                        where market_coupons = 2
                ) db1b,
                        ##aotp aotp
                where
                        db1b.origin = aotp.Origin
                        and db1b.dest =  aotp.Dest
                        and db1b.hub = aotp.Hub
                        and db1b.reporting_carrier = aotp.airline

        )d;


-------------------------------------------------------------------------------------------------
--------------------

----------------------  DIRECT FLIGHTS-----------------------------------------------------------
--------------------

        -- add all flights without a connection
        insert into MasterTable
        (
                ORIGIN, DEST, HUB, AIRLINE, MAX_PAX,
                FL_1_NUM, FL_1_SCH_DEP, FL_1_SCH_ARR, FL_1_ACT_DEP, FL_1_ACT_ARR,
                FL_1_CANCELLED, FL_1_DIVERTED, FL_1_AVG_SEAT, FL_1_LF,
                FL_2_NUM, FL_2_SCH_DEP, FL_2_SCH_ARR, FL_2_ACT_DEP, FL_2_ACT_ARR,
                FL_2_CANCELLED, FL_2_DIVERTED, FL_2_AVG_SEAT, FL_2_LF
        )
        select
                db1b.origin, db1b.dest, null, db1b.reporting_carrier, (pax_perday/total_flights) +
@additional_pax as max_pax,

                aotp.Flight_number   as   Flight_1_number,   aotp.sch_dep   as   Flight_1_Sch_Dep,
aotp.sch_arr as Flight_1_Sch_Arr,
                aotp.act_dep as Flight_1_Act_Dep, aotp.act_arr as Flight_1_Act_Arr,
                aotp.Flight_1_Status, aotp.Flight_1_Div, aotp.Flight_1_Avg_Seat, aotp.Flight_1_LF,

                null, null, null, null, null, null, null, null, null
        from
        (
                select *
                from db1b
                where market_coupons = 1

        ) db1b,
        (
                select
                        m.origin as Origin, m.dest,m.airline, m.flight_num as Flight_number,
                        m.sch_dep  as  Sch_Dep,m.sch_arr  as  Sch_Arr,m.act_dep  as  Act_Dep,m.act_arr
as Act_Arr,
                        m.cancelled as Flight_1_Status,m.div_delay as Flight_1_div,
                        m.avgseat as Flight_1_Avg_Seat, m.lf as Flight_1_LF
                from ##ontime_lf m
                group by m.origin,m.dest,m.airline,m.flight_num,

        m.sch_dep,m.sch_arr,m.act_dep,m.act_arr,m.avgseat,m.avgpax,m.lf,m.cancelled,m.div_delay

        ) aotp,
        (
                --select m.origin, m.dest, m.airline,(count (*)/(3*30)) as total_flights
                select m.origin, m.dest, m.airline,(count (*)/@days) as total_flights
                from ##ontime_lf m
                group by m.origin, m.dest, m.airline

        )aotp_1
        where
                db1b.origin = aotp.Origin
                and db1b.dest =  aotp.Dest
                and db1b.reporting_carrier = aotp.airline
                and db1b.origin = aotp_1.origin
                and db1b.dest =  aotp_1.dest
                and db1b.reporting_carrier = aotp_1.airline
                and aotp_1.total_flights > 0;


        -- create indexes on the table, once it has been populated
        create index IDX_ORIGIN on MasterTable(origin);
        create index IDX_HUB on MasterTable(hub);
        create index IDX_DEST on MasterTable(DEST);
        create index IDX_AIRLINES on MasterTable(airline);
        create index IDX_FLNUM_1 on MasterTable(fl_1_num)
        create index IDX_FL_1_SCH_DEP on MasterTable(fl_1_sch_dep);
        create index IDX_FL_1_SCH_ARR on MasterTable(fl_1_sch_arr);
        create index IDX_FLNUM_2 on MasterTable(fl_2_num)
        create index IDX_FL_2_SCH_DEP on MasterTable(fl_2_sch_dep);
        create index IDX_FL_2_SCH_ARR on MasterTable(fl_2_sch_arr);
        create index IDX_T100_MAX_PAX    on MasterTable(t100_max_pax);

END
GO
```

7) Stored Procedure Create Ontime_lf: Creates a temp table with on time statistics and load factors

```
USE [PaxDelay]
GO

/****** ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:              <Sanja Avramovic>
-- Create date: <10/15/2015>
-- Description:  <create table ontime_lf>
-- Table contains seat/LF and flight status of
-- each flight.
-- Based on Guermo's and Ashwin's code
-- =============================================
create PROCEDURE [dbo].[Create_ontime_lf]
/*I PUT WN/AA as an airline FOR TESTING*/

        @start_date smalldatetime,
        @num_days tinyint

AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;

        -- determine the year and the time range based on start date.
        declare @year as smallint
        declare @end_date as smalldatetime
        set @year = year(@start_date)
        set @end_date = dateadd(dd,@num_days,@start_date)
        -- create the table.
                create table ##ontime_lf
                        (
                        origin varchar(3) not null,dest varchar(3) not null,airline varchar(6) not 
null,
                        flight_num int not null,
                        sch_dep smalldatetime,act_dep smalldatetime,
                        sch_arr smalldatetime,act_arr smalldatetime,
                        sch_arr_1 smalldatetime,sch_arr_2 smalldatetime,
                        cancelled bit not null,div_delay bit not null,
                        avgseat int ,avgpax int ,lf float
                        )

        -- use ontime and load factor data to populate the table.
        insert into ##ontime_lf
                (
                        origin,dest,airline,
                        flight_num,
                        sch_dep,act_dep,
                        sch_arr,act_arr,
                        sch_arr_1,sch_arr_2,
                        cancelled,div_delay,
                        avgseat,avgpax,lf

                )
                select  a.origin,a.dest,a.airline,
                        a.flight_num,
                        a.sch_dep,dateadd (minute, a.dep_delay, a.sch_dep),
                        a.sch_arr,dateadd (minute, a.arr_delay, a.sch_arr),
                        dateadd (minute, 30, sch_arr),dateadd (minute, 180, sch_arr),
                        a.cancelled,a.div_delay,
                        b.avgseat,b.avgpax, b.avgpax/(1.0*b.avgseat)

                from ON_TIME a, LOAD_FACTORS b
                where
                --Comment this line if you want to run for the all airlines, or change 'AA' of the 
other airline. It is set for American Airlines:
                a.AIRLINE ='AA' and b.CARRIER = 'AA' and
                        a.sch_dep >= @start_date
                        and     a.sch_dep <  @end_date
        --              and a.origin  = 'LGA'
                        and a.origin = b.origin
                        and a.dest = b.dest
                        and a.airline = b.carrier
                        and international = 0
                        and b.year = @year
                        and b.month = month(a.sch_dep)
                        and b.AVGSEAT!=0

                order by a.origin,a.dest,a.airline,a.flight_num,a.sch_dep,a.sch_arr

        -- build indexes.
        create index IDX_ORIGIN on ##ontime_lf(origin);

        create index IDX_DEST on ##ontime_lf(DEST);

        create index IDX_AIRLINES on ##ontime_lf(airline);

        create index IDX_FLNUM on ##ontime_lf(flight_num)

        create index IDX_SCH_DEP on ##ontime_lf(sch_dep);

        create index IDX_SCH_ARR on ##ontime_lf(sch_arr);
```

```
END
GO
```

8) Stored procedure

```
USE [PaxDelay]
GO

/****** Object:  StoredProcedure [dbo].[Create_OptionsBase]    Script Date: 5/27/2015
12:29:59 AM ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO



-- =============================================
--  Author:              <Sanja Avramovic>
-- Create date: <3/6/2015>
-- Description: <create table options table>
-- Create a table of alternate flight options
-- for each itinerary.
-- Based on Guermo's and Ashwin's code
-- =============================================
CREATE PROCEDURE [dbo].[Create_OptionsBase]
        @startdate smalldatetime,
        @numdays int,
        @max_wait int = 600,
        @lf_adj float = 0
AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;

    -- check if the table exists already, and drop it
        if OBJECT_ID('tempdb..##options_base') is not null drop table ##options_base;

        -- create the table, before populating it
        create table ##options_base
                (
                Itin_status smallint not null,origin varchar(3) not null,dest  varchar(3)
not null,Original_Hub  varchar(3) null,New_Hub  varchar(3),
                Total_Pax_on_Flight  int,
                Remaining_pax int,
                Original_Airline  varchar(10) not null,New_Airline  varchar(10),
                Original_Flight_Status  bit not null,
                Original_Flight_1  smallint not null,Original_Sch_Dep  smalldatetime not
null,Original_Flight_1_Sch_Arr smalldatetime,Original_Flight_1_Act_Arr smalldatetime,
                Original_Flight_2  smallint,Original_Flight_2_sch_dep
smalldatetime,Original_Sch_Arr  smalldatetime,
                Alternate_Flight_1  smallint ,Alternate_Flight_1_Sch_Dep
smalldatetime,Alternate_Flight_1_Sch_Arr  smalldatetime, Alternate_Flight_1_LF
float,Alternate_Flight_1_TotalSeats smallint ,Alternate_Flight_1_AvaSeats
smallint,Initial_Alternate_Flight_1_AvaSeats  smallint, Alternate_Flight_1_Act_Arr
smalldatetime,
                Alternate_Flight_2  smallint, Alternate_Flight_2_Sch_Dep
smalldatetime,Alternate_Flight_2_Sch_Arr  smalldatetime, Alternate_Flight_2_LF
float,Alternate_Flight_2_TotalSeats smallint ,Alternate_Flight_2_AvaSeats
smallint,Initial_Alternate_Flight_2_AvaSeats  smallint, Alternate_Flight_2_Act_Arr
smalldatetime,
                Total_Rebooked_Pax int, PTD_in_mins int, unit_trip_delay int,
unit_trip_time int, unit_sch_time int,
                alt_priority smallint, -- the priority level of this alterative option
(for example, alts on the same airline have better priority than those that don't)
        priority_metric smallint, -- reordering of alt_priority, used for ordering results
                alt1dep_metric int, -- ordering metric based on the departure time of
alternate flight 1
                alt2dep_metric int  -- ordering metric based on the departure time of
alternate flight 1
                )

        -- populate the table for each class of flight disruption
        EXEC  [dbo].[Populate_Cancellations] @startdate, @numdays, @max_wait, @lf_adj ;
        EXEC  [dbo].[Populate_Diverted] @startdate, @numdays, @max_wait, @lf_adj;
        EXEC  [dbo].[Populate_Delayed] @startdate, @numdays, @max_wait, @lf_adj;
        EXEC [dbo].[Populate_Cancellations2]  @startdate, @numdays, @max_wait, @lf_adj;

        -- update the trip delays per pax per alt itinerary
        update ##options_base
        set unit_trip_delay =
datediff(mi,alternate_flight_2_sch_arr,alternate_flight_2_act_arr)-15
        where alternate_flight_2_act_arr is not null and
datediff(mi,alternate_flight_2_sch_arr,alternate_flight_2_act_arr)>15;

        update ##options_base
        set unit_trip_delay =
datediff(mi,alternate_flight_1_sch_arr,alternate_flight_1_act_arr)-15
        where alternate_flight_2_act_arr is null and
datediff(mi,alternate_flight_1_sch_arr,alternate_flight_1_act_arr)>15;

        update ##options_base
        set unit_trip_time =
datediff(mi,alternate_flight_1_sch_dep,alternate_flight_2_act_arr)
        where alternate_flight_1_sch_dep is not null
        and alternate_flight_2_act_arr is not null
        and alternate_flight_1_sch_dep<alternate_flight_2_act_arr

        update ##options_base
        set unit_trip_time =
datediff(mi,alternate_flight_1_sch_dep,alternate_flight_1_act_arr)
        where alternate_flight_1_sch_dep is not null
        and alternate_flight_2_act_arr is null
        and alternate_flight_1_sch_dep<alternate_flight_1_act_arr

        update ##options_base
        set unit_trip_time =
datediff(mi,alternate_flight_2_sch_dep,alternate_flight_2_act_arr)
        where alternate_flight_1_sch_dep is null
        and alternate_flight_2_act_arr is not null
        and alternate_flight_2_sch_dep<alternate_flight_2_act_arr

        update ##options_base
        set unit_sch_time =
datediff(mi,alternate_flight_1_sch_dep,alternate_flight_2_sch_arr)
        where alternate_flight_1_sch_dep is not null
        and alternate_flight_2_sch_arr is not null
        and alternate_flight_1_sch_dep<alternate_flight_2_sch_arr

        update ##options_base
        set unit_sch_time =
datediff(mi,alternate_flight_1_sch_dep,alternate_flight_1_sch_arr)
        where alternate_flight_1_sch_dep is not null
        and alternate_flight_2_sch_arr is null
        and alternate_flight_1_sch_dep<alternate_flight_1_sch_arr

        update ##options_base
        set unit_sch_time =
datediff(mi,alternate_flight_2_sch_dep,alternate_flight_2_sch_arr)
        where alternate_flight_1_sch_dep is null
        and alternate_flight_2_sch_arr is not null
        and alternate_flight_2_sch_dep<alternate_flight_2_sch_arr

/* stranded pax was initially tracked in its own table, before it was merged into
   The main results:
        -- check if the table exists already, and drop it
        if OBJECT_ID('tempdb..##Stranded_pax') is not null drop table ##Stranded_pax;

    -- create a table for keeping track of the number of stranded pax
        select year, month, day, sum(Stranded_pax) as Stranded_pax, sum(Stranded_pax) as
Remaining_pax into ##Stranded_pax
        from
        (
                select year(Original_Sch_Dep) as year, month(Original_Sch_Dep) as month,
day(Original_Sch_Dep) as day, Stranded_pax
                from
                (
                        select Original_Flight_1, Original_Flight_2, Original_Sch_Dep,
Original_Sch_Arr, max(Remaining_pax) as Stranded_pax
                        from ##options_temp
                        group by Original_Flight_1, Original_Flight_2, Original_Sch_Dep,
Original_Sch_Arr
                ) a
        ) a
        group by year, month, day;
*/
        create index IDX_ITIN_STATUS on ##options_base(itin_status);
        create index IDX_ORIGIN on ##options_base(origin);
        create index IDX_OG_HUB on ##options_base(original_hub);
        create index IDX_NW_HUB on ##options_base(new_hub);
        create index IDX_DEST on ##options_base(DEST);
        create index IDX_AIRLINES on ##options_base(original_airline);
        create index IDX_OG_FL_1 on ##options_base(original_flight_1)
        create index IDX_OG_FL_1_SCH_DEP on ##options_base(original_sch_dep);
        create index IDX_OG_FL_1_SCH_ARR on ##options_base(Original_Flight_1_Sch_Arr);
        create index IDX_OG_FL_1_ACT_ARR on ##options_base(Original_Flight_1_Act_Arr);
        create index IDX_OG_FL_2 on ##options_base(original_flight_2);
        create index IDX_OG_FL_2_SCH_DEP on ##options_base(Original_Flight_2_sch_dep);
        create index IDX_OG_FL_2_SCH_ARR on ##options_base(Original_Sch_Arr);
        create index IDX_ALT_FL_1 on ##options_base(alternate_flight_1);
        create index IDX_ALT_FL_1_SCH_DEP on ##options_base(Alternate_Flight_1_Sch_Dep);
        create index IDX_ALT_FL_1_ACT_ARR on ##options_base(Alternate_Flight_1_Act_Arr);
        create index IDX_ALT_FL_2 on ##options_base(alternate_flight_2);
        create index IDX_ALT_FL_2_SCH_DEP on ##options_base(Alternate_Flight_2_Sch_Dep);
        create index IDX_ALT_FL_2_SCH_ARR on ##options_base(Alternate_Flight_2_Sch_Arr);
        create index IDX_ALT_FL_2_ACT_ARR on ##options_base(Alternate_Flight_2_Act_Arr);
        create index IDX_ALT_PRIORITY on ##options_base(alt_priority);
END

GO
```

9) Stored procedure

```
USE [PaxDelay]
GO

/****** Object:  StoredProcedure [dbo].[Create_OptionsTemp]    Script Date: 5/27/2015 12:31:49 AM
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO



-- =============================================
--  Author:             <Sanja Avramovic>
-- Create date: <10/23/2014>
-- Description: <create table options table>
-- Make a working copy of the options base table
-- in ##options_temp.
-- Based on Guermo's and Ashwin's code
-- =============================================
CREATE PROCEDURE [dbo].[Create_OptionsTemp]
/*      @startdate smalldatetime,
        @numdays int,
        @max_wait int = 600,
        @lf_adj float = 0
*/
AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;

    -- check if the table exists already, and drop it
        if OBJECT_ID('tempdb..##options_temp') is not null drop table ##options_temp;
        select * into ##options_temp from ##options_base;
/*
        -- create the table, before populating it
        create table ##options_temp
                (
                Itin_status smallint not null,origin varchar(3) not null,dest  varchar(3) not
null,Original_Hub  varchar(3) null,New_Hub  varchar(3),
                Total_Pax_on_Flight  int,
                Remaining_pax int,
                Original_Airline  varchar(10) not null,New_Airline  varchar(10),
                Original_Flight_Status  bit not null,
                Original_Flight_1  smallint not null,Original_Sch_Dep  smalldatetime not
null,Original_Flight_1_Sch_Arr smalldatetime,Original_Flight_1_Act_Arr smalldatetime,
                Original_Flight_2  smallint,Original_Flight_2_sch_dep
smalldatetime,Original_Sch_Arr  smalldatetime,
                Alternate_Flight_1  smallint ,Alternate_Flight_1_Sch_Dep
smalldatetime,Alternate_Flight_1_Sch_Arr  smalldatetime, Alternate_Flight_1_LF
float,Alternate_Flight_1_TotalSeats smallint ,Alternate_Flight_1_AvaSeats
smallint,Initial_Alternate_Flight_1_AvaSeats  smallint, Alternate_Flight_1_Act_Arr smalldatetime,
                Alternate_Flight_2  smallint, Alternate_Flight_2_Sch_Dep
smalldatetime,Alternate_Flight_2_Sch_Arr  smalldatetime, Alternate_Flight_2_LF
float,Alternate_Flight_2_TotalSeats smallint ,Alternate_Flight_2_AvaSeats
smallint,Initial_Alternate_Flight_2_AvaSeats  smallint, Alternate_Flight_2_Act_Arr  smalldatetime,
                Total_Rebooked_Pax int, PTD_in_mins int, unit_trip_delay int, unit_trip_time int,
unit_sch_time int,
                alt_priority smallint, -- the priority level of this alterative option (for
example, alts on the same airline have better priority than those that don't)
                early bit default 0  -- if the alternative is used for early rebooking rather
than normal rebooking (flights up to 30 min after the flight still count as early rebooking,
because advanced warning is needed for them)
```

```
                )

        -- populate the table for each class of flight disruption
        EXEC  [dbo].[Populate_Cancellations] @startdate, @numdays, @max_wait, @lf_adj ;
        EXEC  [dbo].[Populate_Diverted] @startdate, @numdays, @max_wait, @lf_adj;
        EXEC  [dbo].[Populate_Delayed] @startdate, @numdays, @max_wait, @lf_adj;
        EXEC [dbo].[Populate_Cancellations2]  @startdate, @numdays, @max_wait, @lf_adj;

        -- update the trip delays per pax per alt itinerary
        update ##options_temp
        set unit_trip_delay = datediff(mi,alternate_flight_2_sch_arr,alternate_flight_2_act_arr)-
15
        where alternate_flight_2_act_arr is not null and
datediff(mi,alternate_flight_2_sch_arr,alternate_flight_2_act_arr)>15;

        update ##options_temp
        set unit_trip_delay = datediff(mi,alternate_flight_1_sch_arr,alternate_flight_1_act_arr)-
15
        where alternate_flight_2_act_arr is null and
datediff(mi,alternate_flight_1_sch_arr,alternate_flight_1_act_arr)>15;

        update ##options_temp
        set unit_trip_time = datediff(mi,alternate_flight_1_sch_dep,alternate_flight_2_act_arr)
        where alternate_flight_1_sch_dep is not null
        and alternate_flight_2_act_arr is not null
        and alternate_flight_1_sch_dep<alternate_flight_2_act_arr

        update ##options_temp
        set unit_trip_time = datediff(mi,alternate_flight_1_sch_dep,alternate_flight_1_act_arr)
        where alternate_flight_1_sch_dep is not null
        and alternate_flight_2_act_arr is null
        and alternate_flight_1_sch_dep<alternate_flight_1_act_arr

        update ##options_temp
        set unit_trip_time = datediff(mi,alternate_flight_2_sch_dep,alternate_flight_2_act_arr)
        where alternate_flight_1_sch_dep is null
        and alternate_flight_2_act_arr is not null
        and alternate_flight_2_sch_dep<alternate_flight_2_act_arr

        update ##options_temp
        set unit_sch_time = datediff(mi,alternate_flight_1_sch_dep,alternate_flight_2_sch_arr)
        where alternate_flight_1_sch_dep is not null
        and alternate_flight_2_sch_arr is not null
        and alternate_flight_1_sch_dep<alternate_flight_2_sch_arr

        update ##options_temp
        set unit_sch_time = datediff(mi,alternate_flight_1_sch_dep,alternate_flight_1_sch_arr)
        where alternate_flight_1_sch_dep is not null
        and alternate_flight_2_sch_arr is null
        and alternate_flight_1_sch_dep<alternate_flight_1_sch_arr

        update ##options_temp
        set unit_sch_time = datediff(mi,alternate_flight_2_sch_dep,alternate_flight_2_sch_arr)
        where alternate_flight_1_sch_dep is null
        and alternate_flight_2_sch_arr is not null
        and alternate_flight_2_sch_dep<alternate_flight_2_sch_arr

*/
        create index IDX_ITIN_STATUS on ##options_temp(itin_status);
        create index IDX_ORIGIN on ##options_temp(origin);
        create index IDX_OG_HUB on ##options_temp(original_hub);
        create index IDX_NW_HUB on ##options_temp(new_hub);
        create index IDX_DEST on ##options_temp(DEST);
        create index IDX_AIRLINES on ##options_temp(original_airline);
        create index IDX_OG_FL_1 on ##options_temp(original_flight_1)
        create index IDX_OG_FL_1_SCH_DEP on ##options_temp(original_sch_dep);
        create index IDX_OG_FL_1_SCH_ARR on ##options_temp(Original_Flight_1_Sch_Arr);
        create index IDX_OG_FL_1_ACT_ARR on ##options_temp(Original_Flight_1_Act_Arr);
        create index IDX_OG_FL_2 on ##options_temp(original_flight_2);
        create index IDX_OG_FL_2_SCH_DEP on ##options_temp(Original_Flight_2_sch_dep);
        create index IDX_OG_FL_2_SCH_ARR on ##options_temp(Original_Sch_Arr);
        create index IDX_ALT_FL_1 on ##options_temp(alternate_flight_1);
        create index IDX_ALT_FL_1_SCH_DEP on ##options_temp(Alternate_Flight_1_Sch_Dep);
        create index IDX_ALT_FL_1_ACT_ARR on ##options_temp(Alternate_Flight_1_Act_Arr);
        create index IDX_ALT_FL_2 on ##options_temp(alternate_flight_2);
        create index IDX_ALT_FL_2_SCH_DEP on ##options_temp(Alternate_Flight_2_Sch_Dep);
        create index IDX_ALT_FL_2_SCH_ARR on ##options_temp(Alternate_Flight_2_Sch_Arr);
        create index IDX_ALT_FL_2_ACT_ARR on ##options_temp(Alternate_Flight_2_Act_Arr);
        create index IDX_ALT_PRIORITY on ##options_temp(alt_priority);
END



GO
```

10) Stored procedure

```
USE [PaxDelay]
GO

/****** Object:  StoredProcedure [dbo].[Create_OriginDest]    Script Date: 5/27/2015 12:33:14 AM
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO





-- =============================================
-- Author:               <Sanja Avramovic>
-- Create date: <11/18/2014>
-- Description:  <create table OriginDest>
-- Creates a table with all possible origin-dest pairs
-- =============================================
CREATE PROCEDURE [dbo].[Create_OriginDest]

AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;

        -- make sure the table does not already exist
    IF OBJECT_ID('dbo.OriginDest', 'U') IS NOT NULL
    DROP TABLE dbo.OriginDest;

        -- create the table (origin-dest pair, plus an id used as a key for each row)
        create table OriginDest(
                id int not null,
                origin varchar(3) not null,
                dest   varchar(3) not null
        );

        -- create a list of all airports that are used, using origin/dest/hub info from DB1B
        select distinct x.airport into #airports from
```

```
	(
		select distinct a.origin as airport from dbo.db1b a
		union
		select distinct a.dest as airport from dbo.db1b a
		union
		select distinct a.hub as airport from dbo.db1b a
	) x
	where x.airport is not null;

	-- populate the table, by generating a list of all airport-airport pairs
	insert into OriginDest (id,origin,dest)
	select row_number() over (order by a.airport,b.airport) as id, a.airport as origin,
b.airport as dest
	from #airports a, #airports b;

	-- clean up
	drop table #airports;

	-- create indexes for the table
	create unique index IDX_ID on OriginDest(id);
	create index IDX_ORIGIN  on OriginDest(origin);
	create index IDX_DEST    on OriginDest(dest);

END

GO
```

11) Stored procedure

```
USE [PaxDelay]
GO

/****** Object:  StoredProcedure [dbo].[Populate_Cancellations]    Script Date: 5/27/2015 12:34:03
AM ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO


-- =============================================
--  Author:		<Sanja Avramovic>
-- Create date: <10/23/2014>
-- Description:  <populate table options table - cancellations>
-- Based on Guermo's and Ashwin's code
-- TESTING NOTE: the range of flights to select was temporarily changed from a @max_wait minutes
range to a +-1 day range.  Change commented lines to change back.
-- =============================================
CREATE PROCEDURE [dbo].[Populate_Cancellations]
	@startdate smalldatetime,
	@numdays int,
	@max_wait int = 600,
	@lf_adj float = 0
	--, @early_max_wait = 600

AS
BEGIN
	-- SET NOCOUNT ON added to prevent extra result sets from
	-- interfering with SELECT statements.
	SET NOCOUNT ON;
	declare @enddate smalldatetime
	declare @wait_after_cancelled int
	set @wait_after_cancelled = 30
	set @enddate = dateadd(dd, @numdays,@startdate)
```

```
        declare @lf_mult float;
        set @lf_mult = 1+@lf_adj;


        --insert into  ##options_temp(
        insert into ##options_base(

                Itin_status,
                origin,dest,Original_Hub,New_Hub,
                Total_Pax_on_Flight,
                Remaining_pax,
                Original_Airline,New_Airline,

        Original_Flight_Status,Original_Flight_1,Original_Sch_Dep,Original_Flight_1_Sch_Arr,Origin
al_Flight_1_Act_Arr,
                Original_Flight_2,Original_Flight_2_sch_dep,Original_Sch_Arr,

        Alternate_Flight_1,Alternate_Flight_1_Sch_Dep,Alternate_Flight_1_Sch_Arr,Alternate_Flight_
1_LF,Alternate_Flight_1_TotalSeats,Alternate_Flight_1_AvaSeats,Initial_Alternate_Flight_1_AvaSeats
,Alternate_Flight_1_Act_Arr,

        Alternate_Flight_2,Alternate_Flight_2_Sch_Dep,Alternate_Flight_2_Sch_Arr,Alternate_Flight_
2_LF,Alternate_Flight_2_TotalSeats,Alternate_Flight_2_AvaSeats,Initial_Alternate_Flight_2_AvaSeats
,
                Alternate_Flight_2_Act_Arr,
                Total_Rebooked_Pax,PTD_in_mins,unit_trip_delay,unit_trip_time,unit_sch_time,
                alt_priority,
                priority_metric,alt1dep_metric,alt2dep_metric

                )
        select 1,--------If the Itin has flight 1 as cancelled and rebooking is needed
           a.origin,a.dest,
           a.hub as Original_Hub,b.hub as New_Hub,
           a.max_pax as Total_Pax_on_Flight,
           a.max_pax as Remaining_Pax,
           a.airline as Original_Airline,b.airline as New_Airline,
           a.fl_1_cancelled as Original_Flight_Status,a.fl_1_num as
Original_Flight_1,a.fl_1_sch_dep as Original_Sch_Dep,a.fl_1_sch_arr as
Original_Flight_1_Sch_Arr,null,
           a.fl_2_num as Original_Flight_2,null,a.fl_2_sch_arr as Original_Sch_Arr,

           b.fl_1_num as Alternate_Flight_1,b.fl_1_sch_dep as
Alternate_Flight_1_Sch_Dep,b.fl_1_sch_arr as Alternate_Flight_1_Sch_Arr, b.fl_1_lf as
Alternate_Flight_1_LF,b.fl_1_avg_seat*@lf_mult as Alternate_Flight_1_TotalSeats,
          (1-b.fl_1_lf)*b.fl_1_avg_seat*@lf_mult as Alternate_Flight_1_AvaSeats,(1-
b.fl_1_lf)*b.fl_1_avg_seat*@lf_mult as Initial_Alternate_Flight_1_AvaSeats,b.fl_1_act_arr as
Alternate_Flight_1_Act_Arr,
           b.fl_2_num as Alternate_Flight_2,null,b.fl_2_sch_arr as
Alternate_Flight_2_Sch_Arr,b.fl_2_lf as Alternate_Flight_2_LF,b.fl_2_avg_seat*@lf_mult as
Alternate_Flight_2_TotalSeats,
                 isnull((1-b.fl_2_lf)*b.fl_2_avg_seat*@lf_mult,30000) as
Alternate_Flight_2_AvaSeats,
                 isnull((1-b.fl_2_lf)*b.fl_2_avg_seat*@lf_mult,30000) as
Initial_Alternate_Flight_2_AvaSeats,
           b.fl_2_act_arr as Alternate_Flight_2_Act_Arr,
           0 as Total_Rebooked_Pax,
           0 as PTD_in_mins,
           0 as unit_trip_delay,
           0 as unit_trip_time,
           0 as unit_sch_time,

           /*
            1st - rebook on flights from the same airline (and it's subsidiaries) on that day
```

```
                2nd rebook on flights from other airlines on that day
                3rd rebook on flights for the same airline (and it's subsidiaries) on the next
day
                4th rebook on flights from other airlines on the next day.
                5th passengers not rebooked after any of these, would get assigned a 900 minute
delay
       */

           case when b.fl_1_sch_dep is null then -1      else
                        case when datediff(mi, a.fl_1_sch_dep, b.fl_1_sch_dep) <
@wait_after_cancelled then
                                (case when day(a.fl_1_sch_dep) != day (b.fl_1_sch_dep) and
datediff(mi,a.fl_1_sch_dep,b.fl_1_sch_dep)<0

                                                then (case when a.airline=b.airline then 1 else 2
end)-- 1. early, day before,same airline, 2. early, not same day, different airline

                                                else (case when a.airline=b.airline then 3 else 4
end) end)--  3. early, same day, same airline, 4. early, same day,different airline
                        else
                                (case when day(a.fl_1_sch_dep) = day (b.fl_1_sch_dep)     then
(case when a.airline=b.airline then 5 else 6 end)-- 5. normal, same day, same air,  6.  normal,
same day, different air.

                                                else (case when a.airline=b.airline then 7 else 8
end) end) --7. normal, next day, same air , 8. normal, next day, different air
                        end
                        end as alt_priority,


                /* ordering metrics */
           case when b.fl_1_sch_dep is null then -1      else
                        case when datediff(mi, a.fl_1_sch_dep, b.fl_1_sch_dep) <
@wait_after_cancelled then
                                (case when day(a.fl_1_sch_dep) != day (b.fl_1_sch_dep) and
datediff(mi,a.fl_1_sch_dep,b.fl_1_sch_dep)<0

                                                then (case when a.airline=b.airline then 3 else 4
end)-- 1. early, day before,same airline, 2. early, not same day, different airline

                                                else (case when a.airline=b.airline then 1 else 2
end) end)--  3. early, same day, same airline, 4. early, same day,different airline
                        else
                                (case when day(a.fl_1_sch_dep) = day (b.fl_1_sch_dep)     then
(case when a.airline=b.airline then 5 else 6 end)-- 5. normal, same day, same air,  6.  normal,
same day, different air.

                                                else (case when a.airline=b.airline then 7 else 8
end) end) --7. normal, next day, same air , 8. normal, next day, different air
                        end
                        end as priority_metric,

                case when b.fl_1_sch_dep is null then null else
                        case when datediff(mi, a.fl_1_sch_dep, b.fl_1_sch_dep) <
@wait_after_cancelled then
                                -datediff(mi,a.fl_1_sch_dep,b.fl_1_sch_dep)
                        else datediff(mi,a.fl_1_sch_dep,b.fl_1_sch_dep)+(5*24*60)
                        end
                        end as alt1dep_metric,

                case when b.fl_2_sch_dep is null then null else
                        case when datediff(mi, a.fl_1_sch_dep, b.fl_1_sch_dep) <
@wait_after_cancelled then
                                -datediff(mi,a.fl_2_sch_dep,b.fl_2_sch_dep)
                        else datediff(mi,a.fl_2_sch_dep,b.fl_2_sch_dep)+(5*24*60)
                        end
                        end as alt2dep_metric


/*
           case when a.airline=b.airline    and day(a.fl_1_sch_dep) = day (b.fl_1_sch_dep) then 1
                when a.airline != b.airline and day(a.fl_1_sch_dep) = day (b.fl_1_sch_dep) then 2
                        when a.airline=b.airline
                                            then 3

                                                                   else 4
                        end as alt_priority,

                case when datediff(mi, a.fl_1_sch_dep, b.fl_1_sch_dep) < @wait_after_cancelled
then 1 else 0 end as early
*/
        from
                (select *
                from Final
                -- Set of cancelled flights
                where
                fl_1_cancelled = 1
                -- ADD START AND END DATE with sch dep and sch arr times comparison
                        --and fl_1_diverted = 0
                )a left join
                -- Set of possible alternate flights (candidate) where pax can be rebooked
                (select *
                        from Final
                        where
                -- ADD START AND END DATE with sch dep and sch arr times comparison
                        fl_1_cancelled = 0
                        and     ((fl_2_cancelled = 0) or (fl_2_cancelled is null))
                        and ((dateadd(minute,30,fl_1_act_arr) <= fl_2_sch_dep) or (fl_2_sch_dep
is null)) -- Actual Arrival of the first flight should be less than Sch Dep of second flight - 30
minutes(connection time)
                        and (fl_1_avg_seat*(1-fl_1_lf)) > 0  -- Sets the condition to check if
the flight is full or not
                        and (((fl_2_avg_seat*(1-fl_2_lf)) > 0) or (fl_2_avg_seat is null))
                        and fl_1_diverted = 0
                        and (fl_2_diverted = 0 or fl_2_diverted is null)
                        )b

                        on
                        a.FL_1_SCH_DEP >=@startdate and a.FL_1_SCH_DEP < @enddate and
                        a.origin = b.origin
                        and a.dest = b.dest
                        --and a.airline = b.airline
                        --and b.fl_1_sch_dep >=
dateadd(minute,@wait_after_cancelled,a.fl_1_sch_dep) -- need to wait 30 minutes before the next
flight
                /*if the time range is 24 hours:*/
                        --and datediff(mi, a.fl_1_sch_dep, b.fl_1_sch_dep) / @max_wait = 0 -- the
flight must be within @max_wait minutes, before or after the original, to be considered
       /*if the time range is the whole prev. day*/
                and
abs(datediff(dd,convert(smalldatetime,convert(date,a.fl_1_sch_dep)),b.fl_1_sch_dep))<=1
                        -- and datediff(mi, a.fl_1_sch_dep, b.fl_1_sch_dep) <= @max_wait and
datediff(mi, a.fl_1_sch_dep, b.fl_1_sch_dep) > @early_max_wait
                        --and b.fl_1_sch_dep <=dateadd(minute,@max_wait,a.fl_1_sch_dep)
                        --and year (a.fl_1_sch_dep) = year (b.fl_1_sch_dep)
                        --and month(a.fl_1_sch_dep) = month(b.fl_1_sch_dep)
                        --and day(a.fl_1_sch_dep) = day (b.fl_1_sch_dep)
                        --and b.fl_1_sch_dep
<=dateadd(minute,@max_waitForAlternateFlt+@wait_after_cancelled,a.fl_1_sch_dep)
                        --and b.fl_2_sch_arr >=a.fl_2_sch_arr
        order by a.fl_1_sch_dep,a.fl_2_sch_arr,a.fl_1_num,a.fl_2_num,b.fl_1_sch_dep,b.fl_2_sch_arr

END



GO
```

12) Stored procedure

```
USE [PaxDelay]
GO

/****** Object:  StoredProcedure [dbo].[Populate_Cancellations2]    Script Date: 5/27/2015
12:34:52 AM ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO


-- =============================================
--  Author:              <Sanja Avramovic>
-- Create date: <10/23/2014>
-- Description: <populate table options table - cancellations on the second flight>
-- Based on Guermo's and Ashwin's code
-- TESTING NOTE: the range of flights to select was temporarily changed from a @max_wait minutes
range to a +-1 day range.  Change commented lines to change back.
-- =============================================
CREATE PROCEDURE [dbo].[Populate_Cancellations2]
        @startdate smalldatetime,
        @numdays int,
        @max_wait int = 600,
        @lf_adj float = 0
AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;
        declare @enddate smalldatetime
        declare @waitAfterCancel int
        set @waitAfterCancel = 30
        set @enddate = dateadd(dd, @numdays,@startdate)

        declare @lf_mult float;
        set @lf_mult = 1+@lf_adj;

                --insert into ##options_temp
                insert into ##options_base
                (
                        Itin_Status,
                        origin,dest,Original_Hub,New_Hub,
                        Total_Pax_on_Flight,
                        Remaining_pax,
                        Original_Airline,New_Airline,

        Original_Flight_Status,Original_Flight_1,Original_Sch_Dep,Original_Flight_1_Sch_Arr,Origin
al_Flight_1_Act_Arr,
                        Original_Flight_2,Original_Flight_2_sch_dep,Original_Sch_Arr,

        Alternate_Flight_1,Alternate_Flight_1_Sch_Dep,Alternate_Flight_1_Sch_Arr,Alternate_Flight_
1_LF,Alternate_Flight_1_TotalSeats,Alternate_Flight_1_AvaSeats,Initial_Alternate_Flight_1_AvaSeats
,Alternate_Flight_1_Act_Arr,

        Alternate_Flight_2,Alternate_Flight_2_Sch_Dep,Alternate_Flight_2_Sch_Arr,Alternate_Flight_
2_LF,Alternate_Flight_2_TotalSeats,Alternate_Flight_2_AvaSeats,Initial_Alternate_Flight_2_AvaSeats
,
                        Alternate_Flight_2_Act_Arr,

        Total_Rebooked_Pax,PTD_in_mins,unit_trip_delay,unit_trip_time,unit_sch_time,
                        alt_priority,
                        priority_metric,alt1dep_metric,alt2dep_metric

                )

                 select 2, -- Flight 2 is cancelled. So itin_status is auto cancelled
                            a.origin,a.dest,
                                a.hub as Hub,null,
                                a.max_pax as Total_Pax_on_Flight,
                                a.max_pax as Remaining_Pax,
                                a.airline as Original_Airline,b.airline as New_Airline,
                                a.fl_1_diverted as Original_Flight_Status,a.fl_1_num as
Original_Flight_1,a.fl_1_sch_dep as Original_Sch_Dep,null,a.fl_1_act_arr as
Original_Flight_1_Act_Arr,
                                a.fl_2_num as Original_Flight_2,a.fl_2_sch_dep as
Original_Flight_2_sch_dep,a.fl_2_sch_arr as Original_Sch_Arr,
                            NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,

                            b.fl_2_num as Alternate_Flight_2,b.fl_2_sch_dep as
Alternate_Flight_2_Sch_Dep,b.fl_2_sch_arr as Alternate_Flight_2_Sch_Arr,b.fl_2_lf as
Alternate_Flight_2_LF,b.fl_2_avg_seat*@lf_mult as Alternate_Flight_2_TotalSeats,
                             (1-b.fl_2_lf)*b.fl_2_avg_seat*@lf_mult as
Alternate_Flight_2_AvaSeats,
                             (1-b.fl_2_lf)*b.fl_2_avg_seat*@lf_mult as
Initial_Alternate_Flight_2_AvaSeats,
                            b.fl_2_act_arr as Alternate_Flight_2_Act_Arr,
                            0 as Total_Rebooked_Pax,
                            0 as PTD_in_mins,
                                0 as unit_trip_delay,
                                0 as unit_trip_time,
                                0 as unit_sch_time,

                        /*
                                1st - rebook on flights from the same airline (and it's
subsidiaries) on that day
                                2nd rebook on flights from other airlines on that day
                                3rd rebook on flights for the same airline (and it's
subsidiaries) on the next day
                                4th rebook on flights from other airlines on the next day.
                                5th passengers not rebooked after any of these, would get
assigned a 900 minute delay
                        */

           case when b.fl_2_sch_dep is null then -1 else
                        case when
                                datediff(mi, a.fl_2_sch_dep, b.fl_2_sch_dep) < @waitAfterCancel
then
                                (case when day(a.fl_2_sch_dep) != day (b.fl_2_sch_dep) and
datediff(mi,a.fl_2_sch_dep,b.fl_2_sch_dep)<0

                                                then (case when a.airline=b.airline then 1 else 2
end)-- 1. early, day before,same airline, 2. early, not same day, different airline

                                                else (case when a.airline=b.airline then 3 else 4
end) end)--  3. early, same day, same airline, 4. early, same day,different airline
                        else
```

```
                                (case when day(a.fl_2_sch_dep) = day (b.fl_2_sch_dep)    then
(case when a.airline=b.airline then 5 else 6 end)-- 5. normal, same day, same air,  6.  normal,
same day, different air.

                                                else (case when a.airline=b.airline then 7 else 8
end) end) --7. normal, next day, same air , 8. normal, next day, different air
                        end
                        end as alt_priority,

                        /*      case when a.airline=b.airline    and day(a.fl_2_sch_dep) = day
(b.fl_2_sch_dep) then 1
                                        when a.airline != b.airline and day(a.fl_2_sch_dep) = day
(b.fl_2_sch_dep) then 2
                                        when a.airline=b.airline
                                                            then 3

                                                                                            else
4
                                end as alt_priority,
                        */

                /* ordering metrics */
            case when b.fl_2_sch_dep is null then -1      else
                        case when datediff(mi, a.fl_2_sch_dep, b.fl_2_sch_dep) < @waitAfterCancel
then
                                (case when day(a.fl_2_sch_dep) != day (b.fl_2_sch_dep) and
datediff(mi,a.fl_2_sch_dep,b.fl_2_sch_dep)<0

                                                then (case when a.airline=b.airline then 3 else 4
end)-- 1. early, day before,same airline, 2. early, not same day, different airline

                                                else (case when a.airline=b.airline then 1 else 2
end) end)--  3. early, same day, same airline, 4. early, same day,different airline
                        else
                                (case when day(a.fl_2_sch_dep) = day (b.fl_2_sch_dep)    then
(case when a.airline=b.airline then 5 else 6 end)-- 5. normal, same day, same air,  6.  normal,
same day, different air.

                                                else (case when a.airline=b.airline then 7 else 8
end) end) --7. normal, next day, same air , 8. normal, next day, different air
                        end
                        end as priority_metric,

                null as alt1dep_metric,

                case when b.fl_2_sch_dep is null then null else
                        case when datediff(mi, a.fl_2_sch_dep, b.fl_2_sch_dep) < @waitAfterCancel
then
                                -datediff(mi,a.fl_2_sch_dep,b.fl_2_sch_dep)
                        else datediff(mi,a.fl_2_sch_dep,b.fl_2_sch_dep)+(5*24*60)
                        end
                        end as alt2dep_metric


                from

                (
                -- SET OF ALL ITINERARIES WITH FLIGHT 2 CANCELLED
                        select *
                        from Final
                        where
                        --      fl_1_diverted = 0
                                fl_1_cancelled = 0
                                and     fl_2_cancelled = 1
                                --and dateadd(minute,30,fl_1_act_arr) > fl_2_act_dep
                )a left join

                (
                -- SET OF FLIGHTS in which Flight 2 is not cancelled or diverted and have seats
remaining
                        select distinct
fl_2_num,airline,fl_2_sch_dep,fl_2_sch_arr,origin,dest,hub,
                        fl_2_lf,fl_2_avg_seat,fl_2_act_arr
                        from Final
                        --from pax_trip_delay
                        where
                                fl_2_diverted = 0 -- FLight 2 cannot be diverted
                                and fl_2_cancelled = 0 -- Flight 2 cannot be cancelled
                                and fl_2_lf < 1 -- Flight 2 cannot be full
                )b
                on
                a.FL_1_SCH_DEP >=@startdate and a.FL_1_SCH_DEP < @enddate and
                a.origin = b.origin
                and a.dest = b.dest
                and a.hub = b.hub
                --and a.airline = b.airline
                -- the alternate flight 2 should be scheduled atleast 30 minutes after the
arrival of the cancelled flight:
                and b.fl_2_sch_dep >= dateadd(minute,@waitAfterCancel,a.fl_1_act_arr)
        /*if the time range is 24 hours:*/
                --and datediff(mi, a.fl_2_sch_dep, b.fl_2_sch_dep) <= @max_wait
        /*if the time range is the whole prev. day*/
              and
abs(datediff(dd,convert(smalldatetime,convert(date,a.fl_2_sch_dep)),b.fl_2_sch_dep))<=1
                --and year (a.fl_2_sch_dep) = year (b.fl_2_sch_dep)
                --and month(a.fl_2_sch_dep) = month(b.fl_2_sch_dep)
                --and day(a.fl_2_sch_dep) = day (b.fl_2_sch_dep)


END

GO
```

13) Stored procedure

```
USE [PaxDelay]
GO

/****** Object:  StoredProcedure [dbo].[Populate_Delayed]    Script Date: 5/27/2015 12:35:41 AM
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO




-- =============================================
--  Author:             <Sanja Avramovic>
-- Create date: <10/23/2014>
-- Description: < Populate for Delayed flights>
-- Based on Guermo's and Ashwin's code
-- TESTING NOTE: the range of flights to select was temporarily changed from a @max_wait minutes
range to a +-1 day range.  Change commented lines to change back.
-- =============================================
CREATE PROCEDURE  [dbo].[Populate_Delayed]
```

```
        @startdate smalldatetime,
        @numdays int,
        @max_wait int = 600,
        @lf_adj float = 0

AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;
        declare @enddate smalldatetime
        declare @waitAfterDelay int
        set @waitAfterDelay = 30
        set @enddate = dateadd(dd, @numdays,@startdate)

        declare @lf_mult float;
        set @lf_mult = 1+@lf_adj;

        --insert into ##options_temp
        insert into ##options_base
                (
                        Itin_status,
                        origin,dest,Original_Hub,New_Hub,
                        Total_Pax_on_Flight,
                        Remaining_pax,
                        Original_Airline,New_Airline,

        Original_Flight_Status,Original_Flight_1,Original_Sch_Dep,Original_Flight_1_Sch_Arr,Origin
al_Flight_1_Act_Arr,
                        Original_Flight_2,Original_Flight_2_sch_dep,Original_Sch_Arr,

        Alternate_Flight_1,Alternate_Flight_1_Sch_Dep,Alternate_Flight_1_Sch_Arr,Alternate_Flight_
1_LF,Alternate_Flight_1_TotalSeats,Alternate_Flight_1_AvaSeats,Initial_Alternate_Flight_1_AvaSeats
,Alternate_Flight_1_Act_Arr,

        Alternate_Flight_2,Alternate_Flight_2_Sch_Dep,Alternate_Flight_2_Sch_Arr,Alternate_Flight_
2_LF,Alternate_Flight_2_TotalSeats,Alternate_Flight_2_AvaSeats,Initial_Alternate_Flight_2_AvaSeats
,
                        Alternate_Flight_2_Act_Arr,

        Total_Rebooked_Pax,PTD_in_mins,unit_trip_delay,unit_trip_time,unit_sch_time,
                        alt_priority,
                        priority_metric,alt1dep_metric,alt2dep_metric
                )

                select  0,-- For missed connections
                                a.origin,a.dest,
                                a.hub as Hub,null,
                                a.max_pax as Total_Pax_on_Flight,
                                a.max_pax as Remaining_Pax,
                                a.airline as Original_Airline,b.airline as New_Airline,
                                a.fl_1_diverted as Original_Flight_Status,a.fl_1_num as
Original_Flight_1,a.fl_1_sch_dep as Original_Sch_Dep,null,a.fl_1_act_arr as
Original_Flight_1_Act_Arr,
                                a.fl_2_num as Original_Flight_2,a.fl_2_sch_dep as
Original_Flight_2_sch_dep,a.fl_2_sch_arr as Original_Sch_Arr,
                                NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,

                                b.fl_2_num as Alternate_Flight_2,b.fl_2_sch_dep as
Alternate_Flight_2_Sch_Dep,b.fl_2_sch_arr as Alternate_Flight_2_Sch_Arr,b.fl_2_lf as
Alternate_Flight_2_LF,b.fl_2_avg_seat*@lf_mult as Alternate_Flight_2_TotalSeats,
                                 (1-b.fl_2_lf)*b.fl_2_avg_seat*@lf_mult as
Alternate_Flight_2_AvaSeats,
                                 (1-b.fl_2_lf)*b.fl_2_avg_seat*@lf_mult as
Initial_Alternate_Flight_2_AvaSeats,
                                b.fl_2_act_arr as Alternate_Flight_2_Act_Arr,

                                0 as Total_Rebooked_Pax,
                                0 as PTD_in_mins,
                                0 as unit_trip_delay,
                                0 as unit_trip_time,
                                0 as unit_sch_time,

                        /*
                                1st - rebook on flights from the same airline (and it's
subsidiaries) on that day
                                2nd rebook on flights from other airlines on that day
                                3rd rebook on flights for the same airline (and it's
subsidiaries) on the next day
                                4th rebook on flights from other airlines on the next day.
                                5th passengers not rebooked after any of these, would get
assigned a 900 minute delay
                        */

           case when b.fl_2_sch_dep is null then -1 else
                        case when datediff(mi, a.fl_2_sch_dep, b.fl_2_sch_dep) < @waitAfterDelay
then
                                (case when day(a.fl_2_sch_dep) != day (b.fl_2_sch_dep) and
datediff(mi,a.fl_2_sch_dep,b.fl_2_sch_dep)<0

                                        then (case when a.airline=b.airline then 1 else 2
end)-- 1. early, day before,same airline, 2. early, not same day, different airline

                                        else (case when a.airline=b.airline then 3 else 4
end) end)--  3. early, same day, same airline, 4. early, same day,different airline
                        else
                                (case when day(a.fl_2_sch_dep) = day (b.fl_2_sch_dep)    then
(case when a.airline=b.airline then 5 else 6 end)-- 5. normal, same day, same air,  6.  normal,
same day, different air.

                                        else (case when a.airline=b.airline then 7 else 8
end) end) --7. normal, next day, same air , 8. normal, next day, different air
                        end
                        end as alt_priority,

                        /*      case when a.airline=b.airline    and day(a.fl_2_sch_dep) = day
(b.fl_2_sch_dep) then 1
                                        when a.airline != b.airline and day(a.fl_2_sch_dep) = day
(b.fl_2_sch_dep) then 2
                                        when a.airline=b.airline
                                                           then 3

                                                                                                 else
4
                                end as alt_priority,*/


                /* ordering metrics */
           case when b.fl_2_sch_dep is null then -1      else
                        case when datediff(mi, a.fl_2_sch_dep, b.fl_2_sch_dep) < @waitAfterDelay
then
                                (case when day(a.fl_2_sch_dep) != day (b.fl_2_sch_dep) and
datediff(mi,a.fl_2_sch_dep,b.fl_2_sch_dep)<0

                                        then (case when a.airline=b.airline then 3 else 4
end)-- 1. early, day before,same airline, 2. early, not same day, different airline

                                        else (case when a.airline=b.airline then 1 else 2
end) end)--  3. early, same day, same airline, 4. early, same day,different airline
                        else
```

```
                                 (case when day(a.fl_2_sch_dep) = day (b.fl_2_sch_dep)     then
(case when a.airline=b.airline then 5 else 6 end)-- 5. normal, same day, same air,  6.  normal,
same day, different air.

                                                 else (case when a.airline=b.airline then 7 else 8
end) end) --7. normal, next day, same air , 8. normal, next day, different air
                         end
                         end as priority_metric,

                 null as alt1dep_metric,

                 case when b.fl_2_sch_dep is null then null else
                         case when datediff(mi, a.fl_2_sch_dep, b.fl_2_sch_dep) < @waitAfterDelay
then
                                 -datediff(mi,a.fl_2_sch_dep,b.fl_2_sch_dep)
                         else datediff(mi,a.fl_2_sch_dep,b.fl_2_sch_dep)+(5*24*60)
                         end
                         end as alt2dep_metric

                         from

                 (
                 -- SET OF ALL DELAYED FLIGHTS with missed connections-- Flight 1 is DELAYED
                         select *
                         from Final
                         where
                                 fl_1_diverted = 0
                                 and fl_1_cancelled = 0
                                 and dateadd(minute,30,fl_1_act_arr) > fl_2_act_dep
                 )a left join

                 (-- SET OF FLIGHTS in which Flight 2 is not cancelled or diverted and have seats
remaining
                         select distinct
fl_2_num,airline,fl_2_sch_dep,fl_2_sch_arr,origin,dest,hub,
                         fl_2_lf,fl_2_avg_seat,fl_2_act_arr
                         from Final
                         --from pax_trip_delay
                         where
                                 fl_2_diverted = 0 -- FLight 2 cannot be diverted
                                 and fl_2_cancelled = 0 -- Flight 2 cannot be cancelled
                                 and fl_2_lf < 1 -- Flight 2 cannot be full

                 )b
                 on
                     a.FL_1_SCH_DEP >=@startdate and a.FL_1_SCH_DEP < @enddate and
                         a.origin = b.origin
                         and a.dest = b.dest
                         and a.hub = b.hub
                         --and a.airline = b.airline
                         -- the alternate flight 2 should be scheduled atleast 30 minutes after
the arrival of the diverted flight:
                         and b.fl_2_sch_dep >= dateadd(minute,@waitAfterDelay,a.fl_1_act_arr)
                         /*if the time range is 24 hours:*/
                         --and datediff(mi, a.fl_2_sch_dep, b.fl_2_sch_dep) <= @max_wait
                          /*if the time range is the whole prev. day*/
                 and abs(
datediff(dd,convert(smalldatetime,convert(date,a.fl_2_sch_dep)),b.fl_2_sch_dep))<=1
                         --and year (a.fl_1_sch_dep) = year (b.fl_2_sch_dep)
                         --and month(a.fl_1_sch_dep) = month(b.fl_2_sch_dep)
                         --and day(a.fl_1_sch_dep) = day (b.fl_2_sch_dep)



END
```

```
GO
```

14) Stored procedure

```
USE [PaxDelay]
GO

/****** Object:  StoredProcedure [dbo].[Populate_Diverted]    Script Date: 5/27/2015 12:36:52 AM
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

--  =============================================
--  Author:             <Sanja Avramovic>
-- Create date: <10/23/2014>
-- Description: < Finding candidate flights for missed connections due to diversions and delays>
-- Based on Guermo's and Ashwin's code
-- TESTING NOTE: the range of flights to select was temporarily changed from a @max_wait minutes
range to a +-1 day range.  Change commented lines to change back.
-- =============================================
CREATE PROCEDURE  [dbo].[Populate_Diverted]
        @startdate smalldatetime,
        @numdays int,
        @max_wait int = 600,
        @lf_adj float = 0

AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;
        declare @enddate smalldatetime
        declare @waitAfterDiverted int
        set @waitAfterDiverted = 30
        set @enddate = dateadd(dd, @numdays,@startdate)

        declare @lf_mult float;
        set @lf_mult = 1+@lf_adj;

        --insert into  ##options_temp(
        insert into ##options_base(

                        Itin_status,
                        origin,dest,Original_Hub,New_Hub,
                        Total_Pax_on_Flight,
                        Remaining_pax,
                        Original_Airline,New_Airline,

        Original_Flight_Status,Original_Flight_1,Original_Sch_Dep,Original_Flight_1_Sch_Arr,Origin
al_Flight_1_Act_Arr,
                        Original_Flight_2,Original_Flight_2_sch_dep,Original_Sch_Arr,

        Alternate_Flight_1,Alternate_Flight_1_Sch_Dep,Alternate_Flight_1_Sch_Arr,Alternate_Flight_
1_LF,Alternate_Flight_1_TotalSeats,Alternate_Flight_1_AvaSeats,Initial_Alternate_Flight_1_AvaSeats
,Alternate_Flight_1_Act_Arr,

        Alternate_Flight_2,Alternate_Flight_2_Sch_Dep,Alternate_Flight_2_Sch_Arr,Alternate_Flight_
2_LF,Alternate_Flight_2_TotalSeats,Alternate_Flight_2_AvaSeats,Initial_Alternate_Flight_2_AvaSeats
,
                        Alternate_Flight_2_Act_Arr,
        Total_Rebooked_Pax,PTD_in_mins,unit_trip_delay,unit_trip_time,unit_sch_time,
                        alt_priority,
                        priority_metric,alt1dep_metric,alt2dep_metric

                )

                select 0,-- For missed connections
                   a.origin,a.dest,
                   a.hub as Hub,null,
                   a.max_pax as Total_Pax_on_Flight,
                   a.max_pax as Remaining_Pax,
                   a.airline as Original_Airline,b.airline as New_Airline,
                   a.fl_1_diverted as Original_Flight_Status,a.fl_1_num as
Original_Flight_1,a.fl_1_sch_dep as Original_Sch_Dep,null,a.fl_1_act_arr as
Original_Flight_1_Act_Arr,
                   a.fl_2_num as Original_Flight_2,a.fl_2_sch_dep as
Original_Flight_2_sch_dep,a.fl_2_sch_arr as Original_Sch_Arr,
                   NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,

                   b.fl_2_num as Alternate_Flight_2,b.fl_2_sch_dep as
Alternate_Flight_2_Sch_Dep,b.fl_2_sch_arr as Alternate_Flight_2_Sch_Arr,b.fl_2_lf as
Alternate_Flight_2_LF,b.fl_2_avg_seat*@lf_mult as Alternate_Flight_2_TotalSeats,
                        (1-b.fl_2_lf)*b.fl_2_avg_seat*@lf_mult as Alternate_Flight_2_AvaSeats,
                        (1-b.fl_2_lf)*b.fl_2_avg_seat*@lf_mult as
Initial_Alternate_Flight_2_AvaSeats,
                   b.fl_2_act_arr as Alternate_Flight_2_Act_Arr,
                   0 as Total_Rebooked_Pax,
                   0 as PTD_in_mins,
                   0 as unit_trip_delay,
                   0 as unit_trip_time,
                   0 as unit_sch_time,


                        /*
                                1st - rebook on flights from the same airline (and it's
subsidiaries) on that day
                                2nd rebook on flights from other airlines on that day
                                3rd rebook on flights for the same airline (and it's
subsidiaries) on the next day
                                4th rebook on flights from other airlines on the next day.
                                5th passengers not rebooked after any of these, would get
assigned a 900 minute delay
                        */
                   case when b.fl_2_sch_dep is null then -1 else
                                case when datediff(mi, a.fl_2_sch_dep, b.fl_2_sch_dep) <
@waitAfterDiverted then
                                (case when day(a.fl_2_sch_dep) != day (b.fl_2_sch_dep) and
datediff(mi,a.fl_2_sch_dep,b.fl_2_sch_dep)<0

                                                then (case when a.airline=b.airline then 1 else 2
end)-- 1. early, day before,same airline, 2. early, not same day, different airline

                                                else (case when a.airline=b.airline then 3 else 4
end) end)--  3. early, same day, same airline, 4. early, same day,different airline
                        else
                                (case when day(a.fl_2_sch_dep) = day (b.fl_2_sch_dep)    then
(case when a.airline=b.airline then 5 else 6 end)-- 5. normal, same day, same air,  6.  normal,
same day, different air.

                                                else (case when a.airline=b.airline then 7 else 8
end) end) --7. normal, next day, same air , 8. normal, next day, different air
                        end
                        end as alt_priority,
                        /*case when a.airline=b.airline    and day(a.fl_2_sch_dep) = day
(b.fl_2_sch_dep) then 1
                                when a.airline != b.airline and day(a.fl_2_sch_dep) = day
(b.fl_2_sch_dep) then 2
                                when a.airline=b.airline
                                                  then 3

                                                                                 else 4
                        end as alt_priority,*/


                /* ordering metrics */
           case when b.fl_2_sch_dep is null then -1      else
                        case when datediff(mi, a.fl_2_sch_dep, b.fl_2_sch_dep) <
@waitAfterDiverted then
                                (case when day(a.fl_2_sch_dep) != day (b.fl_2_sch_dep) and
datediff(mi,a.fl_2_sch_dep,b.fl_2_sch_dep)<0

                                                then (case when a.airline=b.airline then 3 else 4
end)-- 1. early, day before,same airline, 2. early, not same day, different airline

                                                else (case when a.airline=b.airline then 1 else 2
end) end)--  3. early, same day, same airline, 4. early, same day,different airline
                        else
                                (case when day(a.fl_2_sch_dep) = day (b.fl_2_sch_dep)     then
(case when a.airline=b.airline then 5 else 6 end)-- 5. normal, same day, same air,  6.  normal,
same day, different air.

                                                else (case when a.airline=b.airline then 7 else 8
end) end) --7. normal, next day, same air , 8. normal, next day, different air
                        end
                        end as priority_metric,

                null as alt1dep_metric,

                case when b.fl_2_sch_dep is null then null else
                        case when datediff(mi, a.fl_2_sch_dep, b.fl_2_sch_dep) <
@waitAfterDiverted then
                                -datediff(mi,a.fl_2_sch_dep,b.fl_2_sch_dep)
                        else datediff(mi,a.fl_2_sch_dep,b.fl_2_sch_dep)+(5*24*60)
                        end
                        end as alt2dep_metric

                from

                (-- SET OF ALL DIVERTED FLIGHTS with missed connections-- Flight 1 is Diverted
                select *
                --from #temp_pax_trip_delay
                from Final
                where
                        fl_1_diverted = 1
                        and fl_1_cancelled = 0
                        and dateadd(minute,30,fl_1_act_arr) > fl_2_act_dep

                )a left join

                (
                -- SET OF FLIGHTS in which Flight 2 is not cancelled or diverted and have seats
remaining
                select distinct fl_2_num,airline,fl_2_sch_dep,fl_2_sch_arr,origin,dest,hub,
                fl_2_lf,fl_2_avg_seat,fl_2_act_arr
                from Final
                --from pax_trip_delay
                where
                        fl_2_diverted = 0 -- FLight 2 cannot be diverted
```

```
                        and fl_2_cancelled = 0 -- Flight 2 cannot be cancelled
                        and fl_2_lf < 1 -- Flight 2 cannot be full
                )b
                on
                        a.FL_1_SCH_DEP >=@startdate and a.FL_1_SCH_DEP < @enddate and     a.origin
= b.origin
                and a.dest = b.dest
                and a.hub = b.hub
                --and a.airline = b.airline
                -- the alternate flight 2 should be scheduled atleast 30 minutes after the
arrival of the diverted flight:
                and b.fl_2_sch_dep >= dateadd(minute,@waitAfterDiverted,a.fl_1_act_arr)
                /*if the time range is 24 hours:*/
                --and datediff(mi, a.fl_2_sch_dep, b.fl_2_sch_dep) <= @max_wait
                 /*if the time range is the whole prev. day*/
               and
abs(datediff(dd,convert(smalldatetime,convert(date,a.fl_2_sch_dep)),b.fl_2_sch_dep))<=1
                --and year (a.fl_1_sch_dep) = year (b.fl_2_sch_dep)
                --and month(a.fl_1_sch_dep) = month(b.fl_2_sch_dep)
                --and day(a.fl_1_sch_dep) = day (b.fl_2_sch_dep)

END
GO
```

15) Stored procedure

```
USE [PaxDelay]
GO

/****** Object:  StoredProcedure [dbo].[Update_Avaseat]    Script Date: 5/27/2015 12:37:43 AM
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

-- =============================================
-- Author:              Guillermo Calderón-Meza
-- Create date: not recorded (Modified on May 18, 2009)
-- Description: Computes the number of available seats for the records in #on_time_tmp based on
load_factors data
-- =============================================
CREATE PROCEDURE [dbo].[Update_Avaseat]
        @year int,
        @cur_month int
AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;

        declare @start_date smalldatetime, @end_date smalldatetime

        set @start_date = dbo.parts2date(@year,@cur_month,1)
        set @end_date = dateadd(month,1,@start_date)

        update ##on_time_tmp set avaseat = null
                where sch_dep >= @start_date and
                        sch_dep < @end_date

        print 'computing available seats not cancelled'
        -- set value for column avaseat for non-cancelled flight, note I use carrier instead of
airline,
        -- so mq and aa has different seat or pax info
        update ##on_time_tmp set avaseat = (
                select avaseat from load_factors
                        where year = datepart(year,##on_time_tmp.sch_dep)
                                and month = datepart(month,##on_time_tmp.sch_dep)
                                and international = 0    -- consider only domestic
                                and origin= ##on_time_tmp.origin
                                and dest = ##on_time_tmp.dest
                                and carrier = ##on_time_tmp.airline)
        where cancelled <> 1
                and sch_dep >= @start_date
                -- Do not include @end_date in the range because it is actually out of the
intended range
                and sch_dep < @end_date

        print 'computing available seats not cancelled, still null'
        -- some carriers might not be included in load_factors table, then their avaseat value
will remain null,
        -- so their avaseat are set to be the avg for all carriers
        update ##on_time_tmp set avaseat = (
                select avg(avaseat) from load_factors
                        where year = datepart(year,##on_time_tmp.sch_dep)
                                and month = datepart(month,##on_time_tmp.sch_dep)
                                and international = 0    -- consider only domestic
                                and origin = ##on_time_tmp.origin
                                and dest = ##on_time_tmp.dest
                group by load_factors.origin, load_factors.dest)
        where cancelled <> 1
                and avaseat is null
                and sch_dep >= @start_date
                -- Do not include @end_date in the range because it is actually out of the
intended range
                and sch_dep < @end_date

        print 'Setting a default of 50 - 50 * 0.7 available seats to records with avaseat null
(assuming small airport)'
        update ##on_time_tmp set avaseat = 50 - 50 * 0.7
        where cancelled <> 1
                and avaseat is null
                and sch_dep >= @start_date
                -- Do not include @end_date in the range because it is actually out of the
intended range
                and sch_dep < @end_date

        print 'computing available seats cancelled, all = 0'
        -- set value for column avaseat for cancelled flight
        update ##on_time_tmp set avaseat = 0
        where cancelled = 1
                and sch_dep >= @start_date
                -- Do not include @end_date in the range because it is actually out of the
intended range
                and sch_dep < @end_date
END
GO
```

16) Stored procedure

```
USE [PaxDelay]
GO

/****** Object:  StoredProcedure [dbo].[Update_Avaseats1]    Script Date: 5/27/2015 12:38:40 AM
******/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO




-- =============================================
--  Author:             <Sanja Avramovic>
-- Create date: <11/29/2014>
-- Description:  update the number of available
-- seats on an alternative flight.
-- Based on Guermo's and Ashwin's code
-- =============================================
CREATE PROCEDURE [dbo].[Update_Avaseats1]

        @alt_fl_1_avaseats smallint,
        @alt_fl_1_lf float,
        @alt_fl_1_num smallint,
        @alt_fl_1_sch_dep smalldatetime

AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;

        update ##options_temp
        set Alternate_Flight_1_AvaSeats = @alt_fl_1_avaseats,Alternate_Flight_1_LF = @alt_fl_1_lf
        where
                Alternate_Flight_1 = @alt_fl_1_num
                and Alternate_Flight_1_Sch_Dep = @alt_fl_1_sch_dep;

END
GO
```

17) Stored procedure

```
USE [PaxDelay]
GO

/****** Object:  StoredProcedure [dbo].[Update_Avaseats2]    Script Date: 5/27/2015 12:39:22 AM
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO




-- =============================================
--  Author:             <Sanja Avramovic>
-- Create date: <11/29/2014>
-- Description:  update the number of available
-- seats on an alternative flight.
-- Based on Guermo's and Ashwin's code
-- =============================================
CREATE PROCEDURE [dbo].[Update_Avaseats2]

        @alt_fl_2_avaseats smallint,
        @alt_fl_2_lf float,
        @alt_fl_2_num smallint,
        @alt_fl_2_sch_arr smalldatetime
```

```
AS
BEGIN
	-- SET NOCOUNT ON added to prevent extra result sets from
	-- interfering with SELECT statements.
	SET NOCOUNT ON;

	update ##options_temp
	set Alternate_Flight_2_AvaSeats = @alt_fl_2_avaseats,Alternate_Flight_2_LF = @alt_fl_2_lf
	where
		Alternate_Flight_2 = @alt_fl_2_num
		and Alternate_Flight_2_Sch_Arr = @alt_fl_2_sch_arr
END

GO
```

18) Stored procedure

```
USE [PaxDelay]
GO

/****** Object:  StoredProcedure [dbo].[Update_Avgpax]    Script Date: 5/27/2015 12:40:00 AM
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO



-- =============================================
-- Author:		Guillermo Calderon-Meza
-- Create date: Not recorder (Modified on May 18, 2009)
-- Description:	Computes the average number of passengers (pax) in the flights based on the T100
information
--					It assumes that ther is a temprary table called
#on_time_tmp and that it was initialized
--						with some (most) of the values for avgpax, but
there are still some avgpax field with null value
--				Checking for dates is not necessary since #on_time_tmp is
correctly populated with the relevant records
-- =============================================
CREATE PROCEDURE [dbo].[Update_Avgpax]
AS
BEGIN
	-- SET NOCOUNT ON added to prevent extra result sets from
	-- interfering with SELECT statements.
	SET NOCOUNT ON;

	print 'computing average pax still null'
	-- for carrier not included in load_factors table, they use avg value of all carriers
	update ##on_time_tmp set avgpax = (
		select avg(avgpax)
		from load_factors
		where year = datepart(year,##on_time_tmp.sch_dep)
			and month = datepart(month,##on_time_tmp.sch_dep)
			and international = 0 -- consider only domestic
			and origin= ##on_time_tmp.origin
			and dest = ##on_time_tmp.dest
		group by origin, dest)
	where avgpax is null

	print 'Setting a default avgpax of 0.7 * 50 for the remaining flights (assuming they
correspond to small airports)'
	update ##on_time_tmp set avgpax = 0.7 * 50
```

```
        where avgpax is null
END


GO
```

19) Stored procedure

```
USE [PaxDelay]
GO

/****** Object:  StoredProcedure [dbo].[Update_DelDiv]    Script Date: 5/27/2015 12:40:35 AM
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO



-- =============================================
--  Author:              <Sanja Avramovic>
-- Create date: <10/29/2014>
-- Description:  <UPDATING TABLE FOR DELAYS AND DIVERSIONS-->
-- Based on Guermo's and Ashwin's code
-- =============================================
CREATE PROCEDURE [dbo].[Update_DelDiv]
        -- Add the parameters for the stored procedure here

@extraDivDelay int

AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;

        update Final
        set PTD = max_pax*(datediff(mi,fl_2_sch_arr,fl_2_act_arr)-15),
                trip_delay = max_pax*(datediff(mi,fl_2_sch_arr,fl_2_act_arr)-15),
                trip_time = cast(max_pax*datediff(mi,fl_1_sch_dep,fl_2_act_arr) as bigint),
                sch_time = cast(max_pax*datediff(mi,fl_1_sch_dep,fl_2_sch_arr) as bigint)


        where
        -- Setting the conditions for flights not being cancelled or diverted
                fl_1_cancelled = 0
                and fl_2_cancelled = 0
                and fl_1_diverted = 0
                and fl_2_diverted = 0
        --and fl_1_sch_dep >=@start_date
        --and fl_1_sch_dep <@end_date
                and fl_2_act_dep > dateadd(minute,30,fl_1_act_arr) -- Sets the condition for
catching a connecting flight
        -- A flight is termed a delayed flight only if its Actual time of arrival is more than 15
minutes than that of its scheduled arrival
                and fl_2_act_arr > dateadd(minute,15,fl_2_sch_arr) -- Sets the condition for
calculating delay beyond 15 minutes


---Delayed Flights
-- Direct
        update Final
        set PTD =  max_pax*(datediff(mi,fl_1_sch_arr,fl_1_act_arr)-15),
                trip_delay =  max_pax*(datediff(mi,fl_1_sch_arr,fl_1_act_arr)-15),
                trip_time = cast(max_pax*datediff(mi,fl_1_sch_dep,fl_1_act_arr) as bigint),
                sch_time = cast(max_pax*datediff(mi,fl_1_sch_dep,fl_1_sch_arr) as bigint)

        where
                hub is null
                and fl_1_cancelled = 0
                and fl_1_diverted = 0
                and datediff(mi,fl_1_sch_arr,fl_1_act_arr)-15 >=0


--Diverted Flights
--CONSIDERING THAT ONLY FLIGHT 2 (CONNECTING FLIGHTS) ARE DIVERTED----


        update Final
        set PTD =
max_pax*(cast(paxdelay.dbo.estimate_diverted_delay(fl_1_sch_dep,hub,dest,default,default,0,rand(),
default) as int)+@extraDivDelay),
                trip_delay =
max_pax*(cast(paxdelay.dbo.estimate_diverted_delay(fl_1_sch_dep,hub,dest,default,default,0,rand(),
default) as int)+@extraDivDelay),
                trip_time = cast(max_pax*datediff(mi,fl_1_sch_dep,fl_2_act_arr) as bigint),
                sch_time = cast(max_pax*datediff(mi,fl_1_sch_dep,fl_2_sch_arr) as bigint)
        where
                fl_1_cancelled = 0
                and fl_2_cancelled = 0
                and fl_1_diverted = 0
                and fl_2_diverted = 1
                --and fl_1_sch_dep >=@start_date
                --and fl_1_sch_dep <@end_date
                and fl_2_act_dep > dateadd(minute,30,fl_1_act_arr) -- Sets the condition for
catching a connecting flight

--Direct Flights

        update Final
        set PTD =
max_pax*(cast(paxdelay.dbo.estimate_diverted_delay(fl_1_sch_dep,origin,dest,default,default,0,rand
(),default) as int)+@extraDivDelay),
                trip_delay =
max_pax*(cast(paxdelay.dbo.estimate_diverted_delay(fl_1_sch_dep,origin,dest,default,default,0,rand
(),default) as int)+@extraDivDelay),
                trip_time = cast(max_pax*datediff(mi,fl_1_sch_dep,fl_1_act_arr) as bigint),
                sch_time = cast(max_pax*datediff(mi,fl_1_sch_dep,fl_1_sch_arr) as bigint)
        where
                fl_1_cancelled = 0
                and fl_1_diverted = 1
                and fl_2_num is null
                and fl_2_sch_dep is null

--On time flights

        update Final
        set trip_time = cast(max_pax*datediff(mi,fl_1_sch_dep,fl_1_act_arr) as bigint),
                sch_time = cast(max_pax*datediff(mi,fl_1_sch_dep,fl_1_sch_arr) as bigint)
        where
                fl_2_num is null
                and fl_2_sch_dep is null
                and fl_1_cancelled = 0
                and fl_1_diverted = 0
                and datediff(mi,fl_1_sch_arr,fl_1_act_arr)<=15;

        update Final
        set trip_time = cast(max_pax*datediff(mi,fl_1_sch_dep,fl_2_act_arr) as bigint),
                sch_time = cast(max_pax*datediff(mi,fl_1_sch_dep,fl_2_sch_arr) as bigint)
        where
                fl_1_cancelled = 0
                and fl_2_cancelled = 0
                and fl_1_diverted = 0
                and fl_2_diverted = 0
                and fl_2_act_dep > dateadd(minute,30,fl_1_act_arr)
                and fl_2_act_arr < dateadd(minute,15,fl_2_sch_arr);


END

GO
```

20) Stored procedure

```
USE [PaxDelay]
GO

/****** Object:  StoredProcedure [dbo].[Update_Itinerary_PTDs]    Script Date: 5/27/2015 12:41:12
AM ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO




-- =============================================
--  Author:             <Sanja Avramovic>
-- Create date: <10/29/2014>
-- Description: Update PTDs in Final table for a
-- chosen itinerary.
-- Based on Guermo's and Ashwin's code
-- =============================================
CREATE PROCEDURE [dbo].[Update_Itinerary_PTDs]
        -- Add the parameters for the stored procedure here


        -- Add the parameters for the stored procedure here
@last_fl_1_num smallint,
@last_fl_2_num smallint,
@last_fl_1_sch_dep smalldatetime,
@last_fl_2_sch_arr smalldatetime
AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;

                update Final
                                                set PTD =
                                                                (select sum(PTD_in_mins)
                                                                        from ##options_temp
                                                                        where @last_fl_1_num =
original_flight_1
                                                                          and @last_fl_1_sch_dep
= Original_Sch_Dep
                                                                          and (@last_fl_2_num =
original_flight_2 or (@last_fl_2_num is null and original_flight_2 is null))
                                                                          and
(@last_fl_2_sch_arr = original_sch_arr or (@last_fl_2_sch_arr is null and original_sch_arr is
null))
```

```
                                                        ),
                                                 trip_delay = (select
sum(total_rebooked_pax*unit_trip_delay)
                                                                 from ##options_temp
                                                                 where @last_fl_1_num =
original_flight_1
                                                                   and @last_fl_1_sch_dep
= Original_Sch_Dep
                                                                   and (@last_fl_2_num =
original_flight_2 or (@last_fl_2_num is null and original_flight_2 is null))
                                                                   and
(@last_fl_2_sch_arr = original_sch_arr or (@last_fl_2_sch_arr is null and original_sch_arr is
null))
                                                        ),
                                                 trip_time = (select
sum(cast(total_rebooked_pax*unit_trip_time as bigint))
                                                                 from ##options_temp
                                                                 where @last_fl_1_num =
original_flight_1
                                                                   and @last_fl_1_sch_dep
= Original_Sch_Dep
                                                                   and (@last_fl_2_num =
original_flight_2 or (@last_fl_2_num is null and original_flight_2 is null))
                                                                   and
(@last_fl_2_sch_arr = original_sch_arr or (@last_fl_2_sch_arr is null and original_sch_arr is
null))
                                                        ),
                                                 sch_time = (select
sum(cast(total_rebooked_pax*unit_sch_time as bigint))
                                                                 from ##options_temp
                                                                 where @last_fl_1_num =
original_flight_1
                                                                   and @last_fl_1_sch_dep
= Original_Sch_Dep
                                                                   and (@last_fl_2_num =
original_flight_2 or (@last_fl_2_num is null and original_flight_2 is null))
                                                                   and
(@last_fl_2_sch_arr = original_sch_arr or (@last_fl_2_sch_arr is null and original_sch_arr is
null))
                                                        )


                                         where fl_1_num = @last_fl_1_num
                                                   and fl_1_sch_dep = @last_fl_1_sch_dep
                                                   and (fl_2_num = @last_fl_2_num or
(@last_fl_2_num is null and fl_2_num is null))
                                                   and (fl_2_sch_arr = @last_fl_2_sch_arr
or (@last_fl_2_num is null and fl_2_sch_arr is null))

END
GO
```

21) Stored procedure

```
USE [PaxDelay]
GO

/****** Object:  StoredProcedure [dbo].[Update_MTable]    Script Date: 5/27/2015 12:41:49 AM
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
```

```
-- =============================================
--  Author:              <Sanja Avramovic>
-- Create date: <10/29/2014>
-- Description:  <Update table - MasterTable for the PTDs>
-- Based on Guermo's and Ashwin's code
-- =============================================
CREATE PROCEDURE [dbo].[Update_MTable]
        -- Add the parameters for the stored procedure here


        -- Add the parameters for the stored procedure here
@last_fl_1_num smallint,
@last_fl_2_num smallint,
@last_fl_1_sch_dep smalldatetime,
@last_fl_2_sch_arr smalldatetime
AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;

                update Final
                                                set PTD =
                                                                (select sum(PTD_in_mins)
                                                                 from ##options_temp
                                                                        where @last_fl_1_num =
original_flight_1
                                                                          and @last_fl_1_sch_dep
= Original_Sch_Dep
                                                                          and (@last_fl_2_num =
original_flight_2 or (@last_fl_2_num is null and original_flight_2 is null))
                                                                          and
(@last_fl_2_sch_arr = original_sch_arr or (@last_fl_2_sch_arr is null and original_sch_arr is
null))
                                                                )

                                                where fl_1_num = @last_fl_1_num
                                                          and fl_1_sch_dep = @last_fl_1_sch_dep
                                                          and (fl_2_num = @last_fl_2_num or
(@last_fl_2_num is null and fl_2_num is null))
                                                          and (fl_2_sch_arr = @last_fl_2_sch_arr
or (@last_fl_2_num is null and fl_2_sch_arr is null))

END



GO
```

22) Stored procedure

```
USE [PaxDelay]
GO

/****** Object:  StoredProcedure [dbo].[Update_PTDs]    Script Date: 5/27/2015 12:42:52 AM ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
```

```
-- =============================================
--  Author:              <Sanja Avramovic>
-- Create date: <10/29/2014>
-- Description:  <Update table - MasterTable for the PTDs>
-- Based on Guermo's and Ashwin's code
-- =============================================
CREATE PROCEDURE [dbo].[Update_PTDs]
        -- Add the parameters for the stored procedure here


@total_rebooked_pax int,
@PTD int,
@remaining_pax int,
@alt_fl_1_avaseats smallint,
@alt_fl_1_lf float,
@alt_fl_2_avaseats smallint,
@alt_fl_2_lf float,
@og_fl_1_num smallint,
@og_fl_1_sch_dep smalldatetime   ,
@alt_fl_1_num smallint,
@alt_fl_1_sch_dep smalldatetime,
@og_fl_2_num smallint,
@og_fl_2_sch_arr smalldatetime,
@alt_fl_2_num smallint,
@alt_fl_2_sch_arr smalldatetime



AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;

                        update ##options_temp
                                set Total_Rebooked_Pax = @total_rebooked_pax,
                                        PTD_in_mins = @PTD
                                where

                                         Original_Flight_1 = @og_fl_1_num
                                        and (Original_Flight_2 = @og_fl_2_num or
(@og_fl_2_num is null and Original_Flight_2 is null))
                                        and Original_Sch_Dep  = @og_fl_1_sch_dep
                                        and (Original_Sch_Arr  = @og_fl_2_sch_arr or
(@og_fl_2_num is null and Original_Sch_arr is null))
                                        and Alternate_Flight_1 = @alt_fl_1_num
                                        and (Alternate_Flight_2 = @alt_fl_2_num or
(@alt_fl_2_num is null and Alternate_Flight_2 is null))
                                        and Alternate_Flight_1_Sch_Dep =
@alt_fl_1_sch_dep
                                        and (Alternate_Flight_2_Sch_Arr =
@alt_fl_2_sch_arr or (@alt_fl_2_num is null and Alternate_flight_2_sch_arr is null))



                                update ##options_temp
                                        set Remaining_pax = @remaining_pax

                                        where

                                                Original_Flight_1 = @og_fl_1_num
```

```
                                                        and (Original_Flight_2 = @og_fl_2_num or
(@og_fl_2_num is null and Original_Flight_2 is null))
                                                        and Original_Sch_Dep  = @og_fl_1_sch_dep
                                                        and (Original_Sch_Arr  =
@og_fl_2_sch_arr or (@og_fl_2_num is null and Original_Sch_arr is null))

                                        update ##options_temp
                                                set Alternate_Flight_1_AvaSeats =
@alt_fl_1_avaseats, Alternate_Flight_1_LF = @alt_fl_1_lf
                                                where
                                                        Alternate_Flight_1 = @alt_fl_1_num
                                                        and Alternate_Flight_1_Sch_Dep =
@alt_fl_1_sch_dep

                                        update ##options_temp
                                                set Alternate_Flight_2_AvaSeats =
@alt_fl_2_avaseats,Alternate_Flight_2_LF = @alt_fl_2_lf
                                                where
                                                        Alternate_Flight_2 = @alt_fl_2_num
                                                        and Alternate_Flight_2_Sch_Arr =

@alt_fl_2_sch_arr
END
GO
```

23) Stored procedure

```
USE [PaxDelay]
GO

/****** Object:  StoredProcedure [dbo].[Update_PTDsII]    Script Date: 5/27/2015 12:43:39 AM
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO




-- =============================================
--  Author:              <Sanja Avramovic>
-- Create date: <10/29/2014>
-- Description: <Update table - MasterTable for the PTDs 2>
-- Based on Guermo's and Ashwin's code
-- =============================================
CREATE PROCEDURE [dbo].[Update_PTDsII]

@total_rebooked_pax int,
@PTD int,
@remaining_pax int,
@alt_fl_2_avaseats smallint,
@alt_fl_2_lf float,
@og_fl_1_num smallint,
@og_fl_1_sch_dep smalldatetime   ,
@og_fl_2_num smallint,
@og_fl_2_sch_arr smalldatetime,
@alt_fl_2_num smallint,
@alt_fl_2_sch_arr smalldatetime



AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;

                                         update ##options_temp
                                                 set Total_Rebooked_Pax = @total_rebooked_pax,
                                                         PTD_in_mins = @PTD
                                                 where
                                                         Original_Flight_1 = @og_fl_1_num
                                                 and Original_Flight_2 = @og_fl_2_num
                                                 and Original_Sch_Dep  = @og_fl_1_sch_dep
                                                 and Original_Sch_Arr  = @og_fl_2_sch_arr
                                                 and Alternate_Flight_2 = @alt_fl_2_num
                                                 and Alternate_Flight_2_Sch_Arr =
@alt_fl_2_sch_arr

                                         update ##options_temp
                                                 set Remaining_pax = @remaining_pax

                                                 where
                                                         Original_Flight_1 = @og_fl_1_num
                                                 and Original_Flight_2 = @og_fl_2_num
                                                 and Original_Sch_Dep  = @og_fl_1_sch_dep
                                                 and Original_Sch_Arr  = @og_fl_2_sch_arr


                                         update ##options_temp
                                                 set Alternate_Flight_2_AvaSeats =
@alt_fl_2_avaseats,Alternate_Flight_2_LF = @alt_fl_2_lf
                                                 where
                                                         Alternate_Flight_2 = @alt_fl_2_num
                                                 and Alternate_Flight_2_Sch_Arr =
@alt_fl_2_sch_arr

END

GO
```

24) Stored procedure

```
USE [PaxDelay]
GO

/****** Object:  StoredProcedure [dbo].[Update_Rebook_PTD]    Script Date: 5/27/2015 12:44:14 AM
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO



-- =============================================
--  Author:             <Sanja Avramovic>
-- Create date: <11/29/2014>
-- Description:  Update options table to reflect
-- PTD as a result of rebooking
-- Based on Guermo's and Ashwin's code
-- =============================================
CREATE PROCEDURE [dbo].[Update_Rebook_PTD]

        @total_rebooked_pax int,
        @PTD int,
        @og_fl_1_num smallint,
        @og_fl_1_sch_dep smalldatetime   ,
        @alt_fl_1_num smallint,
        @alt_fl_1_sch_dep smalldatetime,
        @og_fl_2_num smallint,
        @og_fl_2_sch_arr smalldatetime,
        @alt_fl_2_num smallint,
        @alt_fl_2_sch_arr smalldatetime

AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;

        -- set the number of pax rebooked on the selected option, and the PTD resulting from
rebooking
        update ##options_temp
        set Total_Rebooked_Pax = @total_rebooked_pax,
                PTD_in_mins = @PTD

        where
                Original_Flight_1 = @og_fl_1_num
                and Original_Sch_Dep  = @og_fl_1_sch_dep
                and (Original_Flight_2 = @og_fl_2_num or (@og_fl_2_num is null and
Original_Flight_2 is null))
                and (Original_Sch_Arr  = @og_fl_2_sch_arr or (@og_fl_2_num is null and
Original_Sch_arr is null))
                and (Alternate_Flight_1 = @alt_fl_1_num or @alt_fl_1_num is null)
                and (Alternate_Flight_1_Sch_Dep = @alt_fl_1_sch_dep or @alt_fl_1_sch_dep is null)
                and (Alternate_Flight_2 = @alt_fl_2_num or (@alt_fl_2_num is null and
Alternate_Flight_2 is null))
                and (Alternate_Flight_2_Sch_Arr = @alt_fl_2_sch_arr or (@alt_fl_2_num is null and
Alternate_flight_2_sch_arr is null));

END
GO
```

25) Stored procedure

```
USE [PaxDelay]
GO

/****** Object:  StoredProcedure [dbo].[Update_Remaining_Pax]    Script Date: 5/27/2015 12:44:57
AM ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO


-- =============================================
--  Author:              <Sanja Avramovic>
-- Create date: <10/29/2014>
-- Description: Update the number of remaining
-- pax on an itinerary (after rebooking).
-- Based on Guermo's and Ashwin's code
-- =============================================
CREATE PROCEDURE [dbo].[Update_Remaining_Pax]

        @remaining_pax int,
        @og_fl_1_num smallint,
        @og_fl_1_sch_dep smalldatetime   ,
        @og_fl_2_num smallint,
        @og_fl_2_sch_arr smalldatetime

AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;

        -- update every entry which matches the original itinerary with the new remaining pax
value for that itinerary
        update ##options_temp
        set Remaining_pax = @remaining_pax
        where
                Original_Flight_1 = @og_fl_1_num
                and Original_Sch_Dep  = @og_fl_1_sch_dep
                and (Original_Flight_2 = @og_fl_2_num or (@og_fl_2_num is null and
Original_Flight_2 is null))
                and (Original_Sch_Arr  = @og_fl_2_sch_arr or (@og_fl_2_num is null and
Original_Sch_arr is null));

END
GO
```

26) Stored procedure

```
USE [PaxDelay]
GO

/****** Object:  StoredProcedure [dbo].[Update_Subsidiary]    Script Date: 5/27/2015 12:45:38 AM
******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO


-- =============================================
-- Author:               <Sanja Avramovic>
-- Create date: <11/08/2015>
-- Description:  <update subsidiary codes>
-- Modifies MasterTable to change all subsidiary
-- airlines to reflect the parent airline.
-- Based on Kevin's and Ashwin's code
-- =============================================
CREATE PROCEDURE [dbo].[Update_Subsidiary]
        -- Add the parameters for the stored procedure here

        @start_date smalldatetime
AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from
        -- interfering with SELECT statements.
        SET NOCOUNT ON;
        declare @newairline varchar(3)
        declare @oldairline varchar(3)
        declare @airport  varchar(3)
        declare @status int

        declare @year as smallint
        declare @quarter as tinyint

        set @year = year(@start_date)
        set @quarter = (month(@start_date) - 1) / 3 + 1
        declare subs cursor fast_forward
        for
        select newairline, oldairline, airport
        from Subsidiary
        where year=@year and quarter=@quarter

        open subs

        fetch next from subs into
        @newairline, @oldairline, @airport

        set @status = @@fetch_status

        while (@status = 0) begin
                update MasterTable
                set airline = @newairline
                where airline = @oldairline
                and (@airport is NULL or @airport = origin or @airport = hub or @airport = dest);

                fetch next from subs into
                @newairline, @oldairline, @airport
                set @status = @@fetch_status
        end

close subs
deallocate subs
END
GO
```

27) Main program

```
Set nocount on
Use PaxDelay

declare @startdate smalldatetime;
declare @enddate smalldatetime;
declare @numdays int;
declare @extradays int; -- number of "extra" days to put on either side in order to get alts, etc
declare @extstartdate smalldatetime;
declare @extenddate smalldatetime;
declare @extnumdays int;
declare @r int;
declare @preempt_same float;
declare @preempt_prev float;
declare @epoch int; -- number of times to run step 3 for each % choice
declare @lf_adj float; -- amount by which to adjust load factor for testing

set @numdays = 1;
set @startdate = '2012-07-18';
set @enddate = dateadd(dd,@numdays,@startdate);

set @extradays = 1;
set @extstartdate = dateadd(dd,-@extradays,@startdate);
set @extenddate = dateadd(dd,@extradays,@enddate);
set @extnumdays = @numdays + 2*@extradays;

set @epoch = 25;

set @preempt_prev =0   -- 0.1;  -- 10% rebooked prev. day
set @preempt_same =0 -- 0.4;  -- 40% rebooked earlier same day
set @lf_adj = -0.05 --0 --0.02 -- 0 no change to the default, so max 100% LF 0.02 is +2%, -0.02 is
-2%
```

```
declare @origin  varchar(3)
declare @dest  varchar(3)
declare @hub  varchar(3)
declare @airline  varchar(3)
declare @fl_num smallint
declare @pax  int
declare @status int
declare @totalpax int
declare @maxpaxperc float
declare @date smalldatetime
declare @t100mpax float
declare @prev_t100 float
declare @diff int
declare @lf float
declare @seats smallint

/*step 1 is in stored procedures (Create_DB1B, Create_ontime_lf, Create_aotp,
Create_MasterTable)*/

print('Start Stebp 1' +cast(getdate() as varchar))
RAISERROR('Start Step 1',0,1) WITH NOWAIT;
exec dbo.Create_DB1B @startdate;
print('Created DB1B' +cast(getdate() as varchar))
RAISERROR('Created DB1B' ,0,1) WITH NOWAIT;

-- create Final table (or recreate it, if it exists

exec dbo.Create_Final;

exec dbo.Create_ontime_lf @extstartdate,@extnumdays;
print('Created ontime_lf' +cast(getdate() as varchar))
RAISERROR('Created ontime_lf' ,0,1) WITH NOWAIT;
exec dbo.Create_aotp;
print('Created aotp' +cast(getdate() as varchar))

RAISERROR('Created aotp' ,0,1) WITH NOWAIT;
exec dbo.Create_MasterTable @extstartdate,@extnumdays;
print('Finish Step 1' +cast(getdate() as varchar))

/*step2.a*/
/**/
print('Start Step 2a' +cast(getdate() as varchar))
RAISERROR('step 2a' ,0,1) WITH NOWAIT;
exec dbo.Update_Subsidiary @startdate;
print('Finished Step 2a' +cast(getdate() as varchar))
print('Start Step 2b' +cast(getdate() as varchar))
RAISERROR('step2b' ,0,1) WITH NOWAIT;

print('2' +cast(getdate() as varchar))
RAISERROR('Start the cursor: step 2 ' ,0,1) WITH NOWAIT;
set @r=0
while @r<@extnumdays
begin
print('Finished day ' +cast((@r-@extradays) as varchar))
RAISERROR('Finished day %d - %d ' ,0,1, @r,@extradays) WITH NOWAIT;
exec  dbo.ConvertDB1BtoT100 @extstartdate,@r;
        set @r = @r+1;
end

--while  @r<@numdays
if not exists (SELECT * FROM sys.indexes WHERE name='IDX_ORIGIN' AND object_id =
OBJECT_ID('dbo.Final'))
begin
        create index IDX_ORIGIN                  on Final(origin);
        create index IDX_HUB                    on Final(hub);
        create index IDX_DEST                   on Final(DEST);
        create index IDX_AIRLINES               on Final(airline);
        create index IDX_FLNUM_1        on Final(fl_1_num)
        create index IDX_FL_1_SCH_DEP   on Final(fl_1_sch_dep);
        create index IDX_FL_1_SCH_ARR   on Final(fl_1_sch_arr);
        create index IDX_FLNUM_2        on Final(fl_2_num)
        create index IDX_FL_2_SCH_DEP   on Final(fl_2_sch_dep);
        create index IDX_FL_2_SCH_ARR   on Final(fl_2_sch_arr);
        create index IDX_T100_MAX_PAX   on Final(t100_max_pax);
end

print('Finish step 2' +cast(getdate() as varchar))
RAISERROR('Finished step2b' ,0,1) WITH NOWAIT;

-------------------------+++++++++++++++++++++++++++++++++++++++++ STEP 3
++++++++++++++++++++++++++++++++++++++++++++++++------------------------
print('Step 3' +cast(getdate() as varchar))
RAISERROR('Step3' ,0,1) WITH NOWAIT;
declare @rebooked int
declare @ptd int
declare @ptdperfl int
declare @avaseats int

set @rebooked = 0
set @ptd = 0
set @ptdperfl= 0
set @avaseats = 0

declare @pstatus int -- return status for preemptive_percent cursor
declare @spax int
declare @remainpax int
declare @istatus smallint
declare @priority smallint
declare @mpax int
declare @rpax int
declare @rpax_1 int -- rpax for prev day preemptive rebooking
declare @rpax_2 int -- rpax for same day preemptive rebooking
declare @rpax_3 int -- rpax for normal rebooking[dbo].[FinalDec1_2_600]
declare @rpax_orig int
declare @og_airline varchar(10)
declare @new_airline varchar(10)
declare @og_fl_1_num smallint
declare @og_fl_1_sch_dep smalldatetime
declare @og_fl_1_sch_arr smalldatetime
declare @og_fl_1_act_arr smalldatetime
declare @og_fl_2_num smallint
declare @og_fl_2_sch_dep smalldatetime
declare @og_fl_2_sch_arr smalldatetime
declare @alt_fl_1_num smallint
declare @alt_fl_1_sch_dep smalldatetime
declare @lf1 float
declare @totseats1 smallint
declare @avaseats1 smallint
declare @alt_fl_1_act_arr smalldatetime
declare @alt_fl_2_num smallint
declare @alt_fl_2_sch_dep smalldatetime
declare @alt_fl_2_sch_arr smalldatetime
declare @lf2 float
declare @totseats2 smallint
declare @avaseats2 smallint
declare @alt_fl_2_act_arr smalldatetime

-- table to store the results
if OBJECT_ID('tempdb..##results') is not null drop table ##results;
```

```
create table ##results
(
        runID int, status varchar(3), legs int,
        total_flight_itins int, total_pax int,
        PTD_in_Mins int, trip_delay int, trip_time bigint,sch_time bigint,
        alt_priority int,
        perc_prev float, perc_same float,
        year int, month int, day int
);

-- table of percents, to generate the preemptive percents
create table #percents (perc float);
insert into #percents (perc) values (0),(0.1),(0.3),(0.5),(0.7);--,(0.9),(1);

declare preemptive_percents cursor fast_forward
for
-- below selects when one of the two days is 0
select a.perc as prev, b.perc as same from #percents a, #percents b
where a.perc=0 or
b.perc=0;

-- below selects all combinations
/*select a.perc as prev, b.perc as same from #percents a, #percents b
where a.perc+b.perc <= 1;
*/
-- below selects only the preset value
--select @preempt_prev as prev, @preempt_same as same

exec dbo.Create_OptionsBase @startdate, @numdays, 1440, @lf_adj; -- the number is the window, in
minutes, to look for alt flights

open preemptive_percents;
fetch next from preemptive_percents into
        @preempt_prev, @preempt_same;

set @pstatus = @@fetch_status;

while @pstatus = 0 begin

        set @r=0;
        while @r < @epoch begin
                set @r = @r + 1;

                print('Create OptionsTemp' +cast(getdate() as varchar))
                RAISERROR('Create OptionsTemp' ,0,1) WITH NOWAIT;
                --exec dbo.Create_OptionsTemp @startdate, @numdays, 1440, @lf_adj; -- the number
is the window, in minutes, to look for alt flights
                exec dbo.Create_OptionsTemp;
                update Final set PTD=0, trip_delay=0, trip_time=0, sch_time=0;
                print('Created OptionsTemp ' +cast(getdate() as varchar) + '
prev='+cast(@preempt_prev as varchar) + ' same='+cast(@preempt_same as varchar)+ ' r='+cast(@r as
varchar));
                RAISERROR('Start step 3 cursor' ,0,1) WITH NOWAIT;


                declare rebooking cursor forward_only keyset read_only --fast_forward
                for
                select itin_status,     origin,
        dest,Total_Pax_on_Flight,Remaining_pax,Original_Airline,New_Airline,

        Original_Flight_1,Original_Sch_Dep,Original_Flight_1_Sch_Arr,Original_Flight_1_Act_Arr,Ori
ginal_Flight_2,Original_Flight_2_sch_dep,Original_Sch_Arr,

        Alternate_Flight_1,Alternate_Flight_1_Sch_Dep,Alternate_Flight_1_LF,Alternate_Flight_1_Tot
alSeats,Alternate_Flight_1_AvaSeats,Alternate_Flight_1_Act_Arr,
        Alternate_Flight_2,Alternate_Flight_2_Sch_Dep,Alternate_Flight_2_Sch_Arr,Alternate_Flight_
2_LF,   Alternate_Flight_2_TotalSeats,Alternate_Flight_2_AvaSeats,
                        Alternate_Flight_2_Act_Arr, Total_Rebooked_Pax, alt_priority from
##options_temp
                where alternate_flight_1 is not null or alternate_flight_2 is not null
                        --where alt_priority > 4
                /*order by Original_Flight_1,Original_Sch_Dep,Original_Flight_2,Original_Sch_Arr,

        alt_priority,Alternate_Flight_1_Sch_Dep,Alternate_Flight_2_Sch_Arr,Alternate_Flight_2_Sch_
Dep*/

                        --************************new ORDER BY to get reverse order of
flights******************
                        order by
Original_Flight_1,Original_Sch_Dep,Original_Flight_2,Original_Sch_Arr,
                        --case when alt_priority in (1,2) then alt_priority+2 when alt_priority
in (3,4) then alt_priority-2 else alt_priority end,
                   ---For rebooking, the code starts to rebook the pax from original departure
time in the past to the first available option:
                        --case when alt_priority<5 then -
datediff(mi,original_sch_dep,Alternate_Flight_1_Sch_Dep) else
datediff(mi,original_sch_dep,Alternate_flight_1_sch_dep)+(5*24*60) end asc,
                        --case when alt_priority<5 then -
datediff(mi,original_sch_dep,Alternate_Flight_2_Sch_Dep) else
datediff(mi,original_sch_dep,Alternate_flight_2_sch_dep)+(5*24*60) end asc,
                        --Alternate_Flight_1_Sch_Dep,
                        priority_metric,alt1dep_metric,alt2dep_metric,
                        Alternate_Flight_2_Sch_Arr

                open rebooking
                fetch next from rebooking into
                        @istatus, @origin, @dest, @mpax, @rpax_orig, @og_airline, @new_airline,
                        @og_fl_1_num, @og_fl_1_sch_dep, @og_fl_1_sch_arr, @og_fl_1_act_arr,
                        @og_fl_2_num, @og_fl_2_sch_dep, @og_fl_2_sch_arr,
                        @alt_fl_1_num, @alt_fl_1_sch_dep, @lf1, @totseats1, @avaseats1,
@alt_fl_1_act_arr,
                        @alt_fl_2_num, @alt_fl_2_sch_dep, @alt_fl_2_sch_arr, @lf2, @totseats2,
@avaseats2, @alt_fl_2_act_arr,
                        @rebooked,@priority

                set @status = @@fetch_status

                declare @last_fl_1 int
                declare @last_sch_dep smalldatetime
                declare @last_fl_2 int
                declare @last_sch_arr smalldatetime
                declare @last_priority smallint

                set @last_fl_1 = NULL
                set @last_sch_dep = NULL
                set @last_fl_2 = NULL
                set @last_sch_arr = NULL
                set @last_priority = -1
                set @rpax_1 = 0;
                set @rpax_2 = 0;
                set @rpax_3 = 0;
                --set @counter =0

                while (@status = 0) begin
                /*set @counter = @counter+1
                        if (@counter % 10000)=0  begin
                                RAISERROR('done counter step3 steps %d'  ,0,1,@counter) WITH
NOWAIT

                                print('counter step 3' +cast(getdate() as varchar))
                        end*/

                        -- check if the original itinerary for the current option is different
from the previous one
                        if (@last_fl_1 is null or (@last_fl_1 != @og_fl_1_num) or (@last_sch_dep
!= @og_fl_1_sch_dep)
                                or (@last_fl_2 != @og_fl_2_num) or (@last_sch_arr !=
@og_fl_2_sch_arr)
                                or (@last_fl_2 is null and @og_fl_2_num is not null)
                                or (@last_fl_2 is not null and @og_fl_2_num is null)
                                or (@last_sch_arr is null and @og_fl_2_sch_arr is not null)
                                or (@last_sch_arr is not null and @og_fl_2_sch_arr is null))
                        BEGIN

                                -- store the computed PTDs into the final table
                                exec dbo.Update_Itinerary_PTDs @last_fl_1,
@last_fl_2,@last_sch_dep,@last_sch_arr;
                                -- save the previous remaining pax value
                                --if @last_priority<3 set @rpax = @rpax+@rpax_2;
                                if @last_priority<5 set @rpax = @rpax+@rpax_3;
                                exec dbo.Update_Remaining_Pax @rpax, @last_fl_1, @last_sch_dep,
@last_fl_2, @last_sch_arr;
                                -- save the remaining pax value for the current itinerary,
including splitting rpax off for preemptive rebooking
                                --set @rpax_1 = floor(@preempt_prev*@rpax_orig);
                                exec @rpax_1 = dbo.binomial_rand @rpax_orig, @preempt_prev;
                                --set @rpax_2 = floor(@preempt_same*@rpax_orig);
                                exec @rpax_2 = dbo.binomial_rand @rpax_orig, @preempt_same;
                                set @rpax_3 = @rpax_orig - @rpax_1 - @rpax_2;
                                if @rpax_3 < 0 begin
                                        set @rpax_2 = @rpax_2 + @rpax_3;
                                        set @rpax_3 = 0;
                                end
                                set @rpax = @rpax_1+@rpax_2;
                                set @last_priority=0;

                        END -- if (original itinerary is different than prev original itinerary)

                        set @last_fl_1    = @og_fl_1_num;
                        set @last_sch_dep = @og_fl_1_sch_dep;
                        set @last_fl_2    = @og_fl_2_num;
                        set @last_sch_arr = @og_fl_2_sch_arr;

                        -- if the priority class changes to a new new preemptive rebooking
category, add all of the rpax from that category
                        if @priority in (1,2) and @last_priority in (-1,0,3,4) and @rpax>@rpax_1
                        begin
                            set @rpax_3 = @rpax_3 + (@rpax-@rpax_1);
                            set @rpax = @rpax_1;
                        end
                        -- if @priority>=3 and @last_priority<3 set @rpax = @rpax+@rpax_2;
                        if @priority>=5 and @last_priority<5 set @rpax = @rpax+@rpax_3;
                        set @last_priority=@priority;

                        set @spax=0;
                        set @remainpax=0;

                        if @istatus = 1  --FOR ITINS HAVING 1ST FLIGHT AS CANCELLED
                        begin
                                if @rpax > 0
                                begin
                                        set @rebooked = DBO.pairmin(@avaseats1,@avaseats2);
                                        if @rpax < @rebooked set @rebooked = @rpax;
                                        set @rpax = @rpax - @rebooked;
                                        -- if the alt flight is on the same day, then none of the
rebooked pax are stranded anymore
                                        if day(@og_fl_1_sch_dep) = day(@alt_fl_1_sch_dep) or
@alt_fl_1_sch_dep<@og_fl_1_sch_dep
                                                set @spax = -@rebooked;
                                        set @remainpax = -@rebooked;

                                        --/direct to direct option
                                        if (@og_fl_2_num is NULL and @alt_fl_2_num is NULL)
                                                set @ptd =
datediff(mi,@og_fl_1_sch_arr,@alt_fl_1_act_arr) -15;
                                        --direct to connecting option
                                        else if (@og_fl_2_num is NULL  and @alt_fl_2_num is not
NULL)
                                                set @ptd=
datediff(mi,@og_fl_1_sch_arr,@alt_fl_2_act_arr) -15;
                                                -- Connecting to Direct Option
                                        else if (@og_fl_2_num is not NULL and @alt_fl_2_num is
NULL)
                                                set @ptd=
datediff(mi,@og_fl_2_sch_arr,@alt_fl_1_act_arr) -15;
                                                --connecting to connecting
                                        else if (@og_fl_2_num is not NULL and @alt_fl_2_num is
not NULL)
                                                set @ptd=
datediff(mi,@og_fl_2_sch_arr,@alt_fl_2_act_arr) -15;

                                        if (@ptd >0)     set @ptd = @ptd * @rebooked;
                                                else  set @ptd = 0;

                                        set @avaseats1 = @avaseats1 - @rebooked;
                                        set @avaseats2  = @avaseats2 - @rebooked;
                                        set @lf1 = 1 -
convert(float,(1.0*@avaseats1)/@totseats1);
                                        set @lf2 = 1 -
convert(float,(1.0*@avaseats2)/@totseats2);

                                        --//    --- Need to assume that the same flight number
doesnt repeat in a day--wont work for more than one day
                                        --//    -- Need to adjust the load factors so that the
available seats gets adjusted accordingly

                                end--//if rpax > 0
                                else set @ptd = 0;

                                exec dbo.Update_Rebook_PTD @rebooked, @ptd, @og_fl_1_num,
@og_fl_1_sch_dep, @alt_fl_1_num, @alt_fl_1_sch_dep,

        @og_fl_2_num,@og_fl_2_sch_arr,@alt_fl_2_num,@alt_fl_2_sch_arr;
                                exec dbo.Update_Avaseats2
@avaseats2,@lf2,@alt_fl_2_num,@alt_fl_2_sch_arr;
                                exec dbo.Update_Avaseats1
@avaseats1,@lf1,@alt_fl_1_num,@alt_fl_1_sch_dep;

                        end -- // if istatus == 1
                        else if (@istatus = 2 or @istatus = 0)
                        begin
                                if (@rpax >0 )
                                begin
                                        set @rebooked = @avaseats2;
                                        if @rpax < @rebooked set @rebooked = @rpax;
                                        set @rpax = @rpax - @rebooked;

                                        -- if flight 2 and it's alternate are on the same day,
none of the rebooked pax are stranded anymore
```

```
                                        if day(@og_fl_2_sch_dep)=day(@alt_fl_2_sch_dep) or
@alt_fl_2_sch_dep<@og_fl_2_sch_dep
                                                set @spax = -@rebooked;
                                        set @remainpax = -@rebooked;

                                        set @ptd= datediff(mi,@og_fl_2_sch_arr,@alt_fl_2_act_arr)
-15;
                                        if (@ptd >0)     set @ptd = @ptd * @rebooked;
                                                else set @ptd = 0;

                                        set @avaseats2 = @avaseats2 - @rebooked;
                                        set @lf2 = 1 -
convert(float,(1.0*@avaseats2)/@totseats2);

                                        --- Need to assume that the same flight number doesnt
repeat in a day--wont work for more than one day
                                        -- Need to adjust the load factors so that the available
seats gets adjusted accordingly

                                end --//if rpax > 0
                                else set @ptd = 0;

                                exec dbo.Update_Rebook_PTD @rebooked, @ptd, @og_fl_1_num,
@og_fl_1_sch_dep, null, null,
        @og_fl_2_num,@og_fl_2_sch_arr,@alt_fl_2_num,@alt_fl_2_sch_arr;
                                exec dbo.Update_Avaseats2
@avaseats2,@lf2,@alt_fl_2_num,@alt_fl_2_sch_arr;

                        end-- if  (@istatus = 2 or @istatus = 0)

/*
                        if @remainpax != 0
                                update ##Stranded_pax
                                set Stranded_pax = Stranded_pax+@spax,
                                        Remaining_pax = Remaining_pax+@remainpax
                                where year=year(@og_fl_1_sch_dep) and
month=month(@og_fl_1_sch_dep) and day=day(@og_fl_1_sch_dep);
*/

                        fetch next from rebooking into
                                @istatus, @origin, @dest,  @mpax, @rpax_orig, @og_airline,
@new_airline,
                                @og_fl_1_num, @og_fl_1_sch_dep, @og_fl_1_sch_arr,
@og_fl_1_act_arr,
                                @og_fl_2_num, @og_fl_2_sch_dep, @og_fl_2_sch_arr,
                                @alt_fl_1_num, @alt_fl_1_sch_dep, @lf1, @totseats1, @avaseats1,
@alt_fl_1_act_arr,
                                @alt_fl_2_num, @alt_fl_2_sch_dep, @alt_fl_2_sch_arr, @lf2,
@totseats2, @avaseats2, @alt_fl_2_act_arr,
                                @rebooked,@priority

                        set @status = @@fetch_status

                end -- end while (@status = 0)

                close rebooking
                deallocate rebooking

                exec dbo.Update_Itinerary_PTDs @last_fl_1,
@last_fl_2,@last_sch_dep,@last_sch_arr;
                --if @last_priority<3 set @rpax = @rpax+@rpax_2;
                if @last_priority<5 set @rpax = @rpax+@rpax_3;
                exec dbo.Update_Remaining_Pax @rpax, @last_fl_1, @last_sch_dep, @last_fl_2,
@last_sch_arr;
```

```
                exec dbo.Update_DelDiv 180;

                exec dbo.Collect_results @preempt_prev, @preempt_same, @r;
        end -- while @r < ... loop

        fetch next from preemptive_percents into
                @preempt_prev, @preempt_same;

        set @pstatus = @@fetch_status;

end -- while @pstatus = 0

close preemptive_percents
deallocate preemptive_percents
drop table #percents
if OBJECT_ID('tempdb..##options_temp') is not null drop table ##options_temp;

print('Finished Step 3' +cast(getdate() as varchar))
RAISERROR('Finished step 3' ,0,1) WITH NOWAIT;
--select * from ##Stranded_pax
select * from ##results --where day = 27

/*clean up*/
/*
IF OBJECT_ID('dbo.Final', 'U') IS NOT NULL
  DROP TABLE dbo.Final;
select * from Final*/

drop table DB1B;
drop table MasterTable
drop table Final;
drop table ##options_temp;
drop table ##Stranded_pax
drop table ##ontime_lf;
drop table ##aotp;

--select * from ##Stranded_pax
--select * into results10 from ##results
```

28) Statistics

```
use PaxDelay

select distinct year,month,day,status, legs, perc_prev, perc_same, alt_priority,
avg(cast(PTD_in_mins as float)) as median_ptd into #median_ptd
from
(
        select status, legs, perc_prev, perc_same, alt_priority, PTD_in_mins,
year,month,day,n, max(n) over (partition by status, legs, perc_prev,
perc_same,alt_priority, year,month,day) maxn from
        (
                SELECT status,legs,perc_prev,perc_same, alt_priority,
PTD_in_mins,year,month,day,
                        ROW_NUMBER() OVER(PARTITION BY
status,legs,perc_prev,perc_same,alt_priority, year,month,day ORDER BY PTD_in_mins) AS n
                FROM ##results
        ) a
) a
where (maxn%2=1 and 2*n=(maxn+1)) or (maxn%2=0 and 2*n=maxn) or (maxn%2=0 and 2*(n-
1)=maxn)
group by status, legs, perc_prev, perc_same,alt_priority, year,month,day;
```

```
select distinct year,month,day,status, legs, perc_prev, perc_same, alt_priority,
avg(cast(trip_delay as float)) as median_trip_delay into #median_trip_delay
from
(
        select status, legs, perc_prev, perc_same, alt_priority, trip_delay,
year,month,day,n, max(n) over (partition by status, legs, perc_prev,
perc_same,alt_priority, year,month,day) maxn from
        (
                SELECT status,legs,perc_prev,perc_same, alt_priority,
trip_delay,year,month,day,
                        ROW_NUMBER() OVER(PARTITION BY
status,legs,perc_prev,perc_same,alt_priority, year,month,day ORDER BY trip_delay) AS n
                FROM ##results
        ) a
) a
where (maxn%2=1 and 2*n=(maxn+1)) or (maxn%2=0 and 2*n=maxn) or (maxn%2=0 and 2*(n-
1)=maxn)
group by status, legs, perc_prev, perc_same,alt_priority, year,month,day;

select distinct year,month,day,status, legs, perc_prev, perc_same, alt_priority,
avg(cast(trip_time as float)) as median_trip_time into #median_trip_time
from
(
        select status, legs, perc_prev, perc_same, alt_priority, trip_time,
year,month,day,n, max(n) over (partition by status, legs, perc_prev,
perc_same,alt_priority, year,month,day) maxn from
        (
                SELECT status,legs,perc_prev,perc_same, alt_priority,
trip_time,year,month,day,
                        ROW_NUMBER() OVER(PARTITION BY
status,legs,perc_prev,perc_same,alt_priority, year,month,day ORDER BY trip_time) AS n
                FROM ##results
        ) a
) a
where (maxn%2=1 and 2*n=(maxn+1)) or (maxn%2=0 and 2*n=maxn) or (maxn%2=0 and 2*(n-
1)=maxn)
group by status, legs, perc_prev, perc_same,alt_priority, year,month,day;

select distinct year,month,day,status, legs, perc_prev, perc_same, alt_priority,
avg(cast(sch_time as float)) as median_sch_time into #median_sch_time
from
(
        select status, legs, perc_prev, perc_same, alt_priority, sch_time,
year,month,day,n, max(n) over (partition by status, legs, perc_prev,
perc_same,alt_priority, year,month,day) maxn from
        (
                SELECT status,legs,perc_prev,perc_same, alt_priority,
sch_time,year,month,day,
                        ROW_NUMBER() OVER(PARTITION BY
status,legs,perc_prev,perc_same,alt_priority, year,month,day ORDER BY sch_time) AS n
                FROM ##results
        ) a
) a
where (maxn%2=1 and 2*n=(maxn+1)) or (maxn%2=0 and 2*n=maxn) or (maxn%2=0 and 2*(n-
1)=maxn)
group by status, legs, perc_prev, perc_same,alt_priority, year,month,day;

select distinct year,month,day,status, legs, perc_prev, perc_same, alt_priority,
avg(cast(total_pax as float)) as median_tpax into #median_tpax
from
(
        select status, legs, perc_prev, perc_same, alt_priority,
total_pax,year,month,day, n, max(n) over (partition by status, legs, perc_prev,
perc_same,alt_priority, year,month,day) maxn from
        (
                SELECT status,legs,perc_prev,perc_same, alt_priority,
total_pax,year,month,day,
                        ROW_NUMBER() OVER(PARTITION BY
status,legs,perc_prev,perc_same,alt_priority, year,month,day ORDER BY total_pax) AS n
                FROM ##results
        ) a
) a
where (maxn%2=1 and 2*n=(maxn+1)) or (maxn%2=0 and 2*n=maxn) or (maxn%2=0 and 2*(n-
1)=maxn)
group by status, legs, perc_prev, perc_same,alt_priority, year,month,day;

--The code is averiging results from AllSteps (binomial distribution ~25 runs)
select a.year,a.month,a.day,a.perc_prev, a.perc_same, a.status, a.legs, a.alt_priority,
        avg(cast(a.total_pax as float)) as total_pax, stdev(cast(a.total_pax as float))
as stdev_total_pax, n.median_tpax,
        avg(cast(a.PTD_in_mins as float)) as mean_ptd, stdev(cast(a.PTD_in_mins as
float)) as stdev_ptd, p.median_ptd,
        avg(cast(a.trip_delay as float)) as mean_trip_delay, stdev(cast(a.trip_delay as
float)) as stdev_trip_delay, t.median_trip_delay,
        avg(cast(a.trip_time as float))/60 as mean_trip_time, stdev(cast(a.trip_time as
float)/60) as stdev_trip_time, x.median_trip_time/60 as median_trip_time,
        avg(cast(a.sch_time as float))/60 as mean_sch_time, stdev(cast(a.sch_time as
float)/60) as stdev_sch_time, y.median_sch_time/60 as median_sch_time
into #summary
from ##results a, #median_tpax n, #median_ptd p, #median_trip_delay t, #median_trip_time
x, #median_sch_time y
where
                a.status=n.status and a.legs=n.legs and a.perc_prev=n.perc_prev and
a.perc_same=n.perc_same and a.year=n.year and a.month=n.month and a.day=n.day and
a.alt_priority=n.alt_priority
        and     a.status=p.status and a.legs=p.legs and a.perc_prev=p.perc_prev and
a.perc_same=p.perc_same and a.year=p.year and a.month=p.month and a.day=p.day and
a.alt_priority=p.alt_priority
        and a.status=t.status and a.legs=t.legs and a.perc_prev=t.perc_prev and
a.perc_same=t.perc_same and a.year=t.year and a.month=t.month and a.day=t.day and
a.alt_priority=t.alt_priority
        and a.status=x.status and a.legs=x.legs and a.perc_prev=x.perc_prev and
a.perc_same=x.perc_same and a.year=x.year and a.month=x.month and a.day=x.day and
a.alt_priority=x.alt_priority
        and a.status=y.status and a.legs=y.legs and a.perc_prev=y.perc_prev and
a.perc_same=y.perc_same and a.year=y.year and a.month=y.month and a.day=y.day and
a.alt_priority=y.alt_priority

group by a.status, a.legs, a.perc_prev, a.perc_same, a.alt_priority, a.year,a.month,a.day,
                n.median_tpax, p.median_ptd,
                t.median_trip_delay, x.median_trip_time, y.median_sch_time
order by a.year,a.month,a.day,a.perc_prev, a.perc_same, a.legs, a.status desc;

/*
select * from #summary
order by year,month,day,perc_prev, perc_same, legs, status desc;
*/

drop table #median_ptd;
drop table #median_trip_delay;
drop table #median_trip_time;
drop table #median_sch_time;
drop table #median_tpax;

select year,month,day,perc_same,perc_prev,
    min(rpSeats) as min_rpSeats, max(rpSeats) as max_rpSeats,
        min(rpItins) as min_rpItins, max(rpItins) as max_rpItins,
        min(prevpax) as min_prevpax, max(prevpax) as max_prevpax,
        min(samepax) as min_samepax, max(samepax) as max_samepax,
        min(latepax) as min_latepax, max(latepax) as max_latepax,
        min(nextpax) as min_nextpax, max(nextpax) as max_nextpax
        into #minmax
from (
        select year, month, day, perc_same, perc_prev, runid,
                sum(iif(alt_priority in (1,2),total_pax,0)) as prevpax,
                sum(iif(alt_priority in (3,4),total_pax,0)) as samepax,
                sum(iif(alt_priority in (5,6),total_pax,0)) as latepax,
                sum(iif(alt_priority in (7,8),total_pax,0)) as nextpax,
                sum(iif(alt_priority in (0),total_pax,0))   as rpSeats,
                sum(iif(alt_priority in (-1),total_pax,0))  as rpItins
        from ##results
        where status='CNX'
        group by year,month,day,perc_same,perc_prev,runid
) a
group by year, month, day, perc_same, perc_prev

select * from (
select year,month,day,'same' as pretype, perc_same as perc,
    min_rpSeats, max_rpSeats,
        min_rpItins, max_rpItins,
        min_prevpax, max_prevpax,
        min_samepax, max_samepax,
        min_latepax, max_latepax,
        min_nextpax, max_nextpax
from #minmax
where perc_prev=0
union
select year,month,day,'prev' as pretype, perc_prev as perc,
    min_rpSeats, max_rpSeats,
        min_rpItins, max_rpItins,
        min_prevpax, max_prevpax,
        min_samepax, max_samepax,
        min_latepax, max_latepax,
        min_nextpax, max_nextpax

from #minmax
where perc_same=0
) a
where day=18
order by year,month,day,pretype,perc;

drop table #minmax


select * into #cancsummary
from (
        select
                year, month, day,
                'same' as pretype, perc_same as perc,
                sum(total_pax) as totpax, sum(stdev_total_pax) as sd_totpax,
                sum(iif(alt_priority in (1,2),total_pax,0)) as prevpax,
sum(iif(alt_priority in (1,2),stdev_total_pax,0)) as sd_prevpax,
                sum(iif(alt_priority in (3,4),total_pax,0)) as samepax,
sum(iif(alt_priority in (3,4),stdev_total_pax,0)) as sd_samepax,
                sum(iif(alt_priority in (5,6),total_pax,0)) as latepax,
sum(iif(alt_priority in (5,6),stdev_total_pax,0)) as sd_latepax,
                sum(iif(alt_priority in (7,8),total_pax,0)) as nextpax,
sum(iif(alt_priority in (7,8),stdev_total_pax,0)) as sd_nextpax,
                sum(iif(alt_priority<1,total_pax,0)) as rpax,
sum(iif(alt_priority<1,stdev_total_pax,0)) as sd_rpax,
                sum(iif(alt_priority=0,total_pax,0)) as rpaxNoSeat,
sum(iif(alt_priority=0,stdev_total_pax,0)) as sd_rpaxNoSeat,
                sum(iif(alt_priority<0,total_pax,0)) as rpaxNoItins,
sum(iif(alt_priority<0,stdev_total_pax,0)) as sd_rpaxNoItins,
                sum(mean_ptd) as ptd, sum(stdev_ptd) as sd_ptd,
                sum(mean_trip_time) as trip_time, sum(stdev_trip_time) as sd_trip_time,
                sum(mean_sch_time) as sch_time, sum(stdev_sch_time) as sd_sch_time
        from #summary
        where perc_prev=0 and status='CNX'
        group by year, month, day, perc_same

        union

        select
                year, month, day,
                'prev' as pretype, perc_prev as perc,
                sum(total_pax) as totpax, sum(stdev_total_pax) as sd_totpax,
                sum(iif(alt_priority in (1,2),total_pax,0)) as prevpax,
sum(iif(alt_priority in (1,2),stdev_total_pax,0)) as sd_prevpax,
                sum(iif(alt_priority in (3,4),total_pax,0)) as samepax,
sum(iif(alt_priority in (3,4),stdev_total_pax,0)) as sd_samepax,
                sum(iif(alt_priority in (5,6),total_pax,0)) as latepax,
sum(iif(alt_priority in (5,6),stdev_total_pax,0)) as sd_latepax,
                sum(iif(alt_priority in (7,8),total_pax,0)) as nextpax,
sum(iif(alt_priority in (7,8),stdev_total_pax,0)) as sd_nextpax,
                sum(iif(alt_priority<1,total_pax,0)) as rpax,
sum(iif(alt_priority<1,stdev_total_pax,0)) as sd_rpax,
                sum(iif(alt_priority=0,total_pax,0)) as rpaxNoSeat,
sum(iif(alt_priority=0,stdev_total_pax,0)) as sd_rpaxNoSeat,
                sum(iif(alt_priority<0,total_pax,0)) as rpaxNoItins,
sum(iif(alt_priority<0,stdev_total_pax,0)) as sd_rpaxNoItins,
                sum(mean_ptd) as ptd, sum(stdev_ptd) as sd_ptd,
                sum(mean_trip_time) as trip_time, sum(stdev_trip_time) as sd_trip_time,
                sum(mean_sch_time) as sch_time, sum(stdev_sch_time) as sd_sch_time
        from #summary
        where perc_same=0 and status='CNX'
        group by year, month, day, perc_prev

) a

/*
select
        year, month, day, pretype, perc, null as lf_bias,
        100*prepax/totpax as perc_prepax, 100*(totpax-prepax-rpax)/totpax as
perc_latepax, null as perc_overnight, ptd/(60*24) as paxdelay_days,
        377*prepax as rev_recoup,
        250*spax/2 as indirect_cost,
        sch_time/24  as sch_trip_time_days,
        trip_time/24 as act_trip_time_days
from #cancsummary
order by year,month,day,pretype desc,perc
*/

/*
select
        year, month, day, pretype, perc,
        totpax, prevpax, samepax, latepax, nextpax, rpax,
        377*(prevpax+samepax) as rev_recoup,
        ptd as tot_ptd, nextpax as overnight, rpax as unbooked, 250*(nextpax+rpax) as
ind_cost
from #cancsummary
where day =20
order by year,month,day,pretype desc,perc

select
        year, month, day, pretype, perc,
        (prevpax+samepax+latepax+nextpax) as num_rebooked_mean,
(sd_prevpax+sd_samepax+sd_latepax+sd_nextpax) as num_rebooked_sdev,
        (prevpax+samepax) as num_early_mean, (sd_prevpax+sd_samepax) as num_early_sdev,
        rpaxNoSeat,  sd_rpaxNoSeat, rpaxNoItins, sd_rpaxNoItins,
        nextpax as overnight_pax,
        377*(prevpax+samepax) as rev_recoup,
        ptd/(24.0*60.0) as ptd_days_mean, sd_ptd/(24*60) as ptd_days_sdev,
        ptd/(1.0*totpax) as avg_ptd_min
from #cancsummary
where day =20
order by year,month,day,pretype desc,perc
*/
/*
select
        pretype, perc,
        sum(totpax) as totpax, sum(prevpax) as prevpax, sum(samepax) as samepax,
sum(latepax) as latepax, sum(nextpax) as nextpax, sum(rpax) as rpax,
        377*sum(prevpax+samepax) as rev_recoup,
        sum(ptd) as tot_ptd, sum(nextpax) as overnight, sum(rpax) as unbooked,
250*sum(nextpax+rpax) as ind_cost
from #cancsummary
where day in (25,26)
group by pretype, perc
order by pretype desc,perc
*/
select pretype, perc, PercPrepax,PercSamepax, PercLate,PercNext,PercRPAX,  CorpTravCost,
rev_recoup, num_rebooked_mean, num_rebooked_sdev, num_early_mean, num_early_sdev,
overnight_pax, rev_recoup, ptd_days_mean, ptd_days_sdev, ptd_min/totpax as avg_ptd_min
from (
select
        pretype, perc,
        sum(prevpax+samepax+latepax+nextpax) as num_rebooked_mean,
sum(sd_prevpax+sd_samepax+sd_latepax+sd_nextpax) as num_rebooked_sdev,
        sum(prevpax+samepax) as num_early_mean, sum(sd_prevpax+sd_samepax) as
num_early_sdev,
        sum(nextpax) as overnight_pax,
        100*sum(prevpax)/sum(totpax) as PercPrepax,
        100*sum(samepax)/sum(totpax) as PercSamepax,
        100*sum(latepax)/sum(totpax) as PercLate,
        100*sum(nextpax)/sum(totpax) as PercNext,
        100*sum(rpax)/sum(totpax) as PercRPAX,
        377*sum(nextpax+rpax) as rev_recoup,
        250/2/2*sum(nextpax +rpax)+ 250/2/2*sum(prevpax) as CorpTravCost,
        sum(ptd)/(24.0*60.0) as ptd_days_mean,
        sum(sd_ptd)/(24*60) as ptd_days_sdev,
        sum(ptd) as ptd_min,
        sum(totpax) as totpax
from #cancsummary
where day = 18 --in (25,26)
group by pretype, perc
) a
order by pretype desc,perc

select avg(tpax) as tpax, legs from (
select runID,legs,sum(total_pax) as tpax from ##results where perc_prev=0 and perc_same=0
and status='CNX' and day=18 group by runID, legs) a group by legs

select * from #summary where day=18
select * from #cancsummary where day=18
drop table #cancsummary
drop table #summary;

select min(lf) as minlfCNXALL, max(lf) as maxlfCNXALL,avg(lf) avglfCNXALL from ##ontime_lf
where day(sch_dep) = 18   and cancelled =1
select min(lf) as minlfALL, max(lf) as maxlfALL,avg(lf) avglfALL from ##ontime_lf where
day(sch_dep) = 18
select min(lf) as minlfORCNX, max(lf) as maxlfORCNX,avg(lf) avglfORCNX from ##ontime_lf
where day(sch_dep) = 18 and (origin = 'EWR' or dest='EWR') and cancelled =1
```

```
select min(lf) as minlfOR, max(lf) as maxlfOR,avg(lf) avglfOR from ##ontime_lf where
day(sch_dep) = 18 and (origin = 'EWR' or dest='EWR')
select sum(lf50) as CNXlf50ALLCX, sum(lf50to60)as CNXlf50to60ALLCX, sum(lf60to70) as
CNXlf60to70ALLCX, sum(lf70to80) as CNXlf70to80ALLCX, sum(lf80to90) as lf80to90ALLCX,
sum(lf90) as lf90ALLCX from
(
select
CASE WHEN lf <=0.50 THEN 1 ELSE 0 END AS lf50,
CASE WHEN  lf > 0.50 and lf <=0.60 then 1 else 0 end as lf50to60,
case when lf > 0.60 and lf <=0.70 then 1 else 0 end as lf60to70 ,
case when lf > 0.70 and lf <=0.80 then 1 else 0 end as lf70to80,
case when lf>0.80 and lf <=0.90 then 1 else 0 end as lf80to90,
case when lf>0.90 then 1 else 0 end as lf90
        from ##ontime_lf where day(sch_dep) = 18  and cancelled = 1
        )a
        select sum(lf50) as  lf50ALL, sum(lf50to60)as  lf50to60ALL, sum(lf60to70) as
lf60to70ALL, sum(lf70to80) as  lf70to80ALL, sum(lf80to90) as  lf80to90ALL, sum(lf90) as
lf90ALL from
(
select
CASE WHEN lf <=0.50 THEN 1 ELSE 0 END AS lf50,
CASE WHEN  lf > 0.50 and lf <=0.60 then 1 else 0 end as lf50to60,
case when lf > 0.60 and lf <=0.70 then 1 else 0 end as lf60to70,
case when lf > 0.70 and lf <=0.80 then 1 else 0 end as lf70to80,
case when lf>0.80 and lf <=0.90 then 1 else 0 end as lf80to90,
case when lf>0.90 then 1 else 0 end as lf90
        from ##ontime_lf where day(sch_dep) = 18
)a

select sum(lf50) as CNXlf50ORDC, sum(lf50to60)as CNXlf50to60ORDC, sum(lf60to70) as
CNXlf60to70ORDC, sum(lf70to80) as CNXlf70to80ORDC, sum(lf80to90) as lf80to90ORDC,
sum(lf90) as lf90ORDC from
(
select
CASE WHEN lf <=0.50 THEN 1 ELSE 0 END AS lf50,
CASE WHEN  lf > 0.50 and lf <=0.60 then 1 else 0 end as lf50to60,
case when lf > 0.60 and lf <=0.70 then 1 else 0 end as lf60to70,
case when lf > 0.70 and lf <=0.80 then 1 else 0 end as lf70to80,
case when lf>0.80 and lf <=0.90 then 1 else 0 end as lf80to90,
case when lf>0.90 then 1 else 0 end as lf90
        from ##ontime_lf where day(sch_dep) = 18and (origin = 'EWR' or dest='EWR') and
cancelled = 1
        )a

select sum(lf50) as  lf50ORD, sum(lf50to60)as  lf50to60ORD, sum(lf60to70) as  lf60to70ORD,
sum(lf70to80) as  lf70to80ORD, sum(lf80to90) as  lf80to90ORD, sum(lf90) as lf90ORD from
(
select
CASE WHEN lf <=0.50 THEN 1 ELSE 0 END AS lf50,
CASE WHEN  lf > 0.50 and lf <=0.60 then 1 else 0 end as lf50to60,
case when lf > 0.60 and lf <=0.70 then 1 else 0 end as lf60to70,
case when lf > 0.70 and lf <=0.80 then 1 else 0 end as lf70to80,
case when lf>0.80 and lf <=0.90 then 1 else 0 end as lf80to90,
case when lf>0.90 then 1 else 0 end as lf90
        from ##ontime_lf where day(sch_dep) = 18and (origin = 'EWR' or dest='EWR')
        )a
```

REFERENCES

Ahmadbeygi, S., Cohn, A., & Lapp, M. (2010). *Decreasing airline delay propagation by re-allocating scheduled slack*. IIE transactions, 42(7), 478-489.

Ball, M., Barnhart, C., Dresner, M., Hansen, M., Neels, K., Odoni, A., ... & Zou, B. (2010). *Total delay impact study: a comprehensive assessment of the costs and impacts of flight delay in the United States.*

Barnhart, C., Belobaba, P., & Odoni, A. R. (2003). *Applications of operations research in the air transport industry*. Transportation science, 37(4), 368-391.

Barnhart, C., Fearing, D., & Vaze, V. (2010*). Analyzing passenger travel disruptions in the National Air Transportation System.*

Barnhart, C., Fearing, D., & Vaze, V. (2014). *Modeling passenger travel and delays in the National Air Transportation System*. Operations Research, Issue 3, Volume 62.

Belobaba, P. (2009). *Overview of airline economics, markets and demand*. The Global Airline Industry, 47-71.

Belobaba, P., Odoni, A., & Barnhart, C. (Eds.). (2009). *The global airline industry* (Vol. 23). John Wiley & Sons.

Bratu, S., & Barnhart, C. (2005). *An analysis of passenger delays using flight operations and passenger booking data. Air Traffic Control Quarterly*, 13(1). BTS, 2015 On Time Performance, T100, Db1b

Bratu, S., & Barnhart, C. (2006*). Flight operations recovery: New approaches considering passenger recovery*. Journal of Scheduling, 9(3), 279-298.

Bureau of Transportation and Statistics (2015), Air Fares, Available: http://www.rita.dot.gov/bts/airfares

Chiraphadhanakul, V., & Barnhart, C. (2013). *Robust flight schedules through slack re-allocation.* EURO Journal on Transportation and Logistics, 2(4), 277-306.

Clarke, M. D. D. (1998). *Irregular airline operations: a review of the state-of-the-practice in airline operations control centers*. Journal of Air Transport Management, 4(2), 67-76.

DoT (2015). *Percentage of Air Travel for Business vs Other Purposes*, Available: https://ntl.custhelp.com/app/answers/detail/a_id/252/~/percentage-of-air-travel-for-business-vs-other-purposes

Eggenberg, N., Salani, M., & Bierlaire, M. (2010). *Constraint-specific recovery network for solving airline recovery problems.* Computers & operations research, 37(6), 1014-1026.

Jafari, N., & Zegordi, S. H. (2011). *Simultaneous recovery model for aircraft and passengers.* Journal of the Franklin Institute, 348(7), 1638-1655.Le, Zhan,Wu (2008) Solving the Airlines Recovery Problem Considering Aircraft Rerouting and Passengers

Lan, S., Clarke, J. P., & Barnhart, C. (2006). *Planning for robust airline operations: Optimizing aircraft routings and flight departure times to minimize passenger disruptions.* Transportation science, 40(1), 15-28

Le, M., Zhan, C., & Wu, C. *Solving the Airlines Recovery Problem Considering Aircraft Rerouting and Passengers.*

Li, Tao, H. Baik, A.A. Trani (2013). *A method to estimate the historical US air travel demand. Journal of Advanced Transportation*. Special Issue: Public Transport Systems. Volume 47, Issue 3, pages 249–265, April 2013

McCarthy, M. B. (2010). *Air Transportation by Metropolitan Area: Different Measures of Airport Activity Yields* (Doctoral dissertation, University of North Carolina at Greensboro).

McCarty, L. A. & Cohn, A. M. (2012). *"Preemptive Rerouting of Airline Passenger Under Uncertain Delays*" targeted for Transportation Science.

McCarty, L. A. (2012). *Preemptive rerouting of airline passengers under uncertain delays* (Doctoral dissertation, The University of Michigan)

National Oceanic and Atmospheric Administration (NOAA) (2015), Storm Prediction Center, 20121225 Storm Reports, Available: http://www.spc.noaa.gov/climo/reports/121225_rpts .html

Raiteri, A (2015) personal conversation

Sherry, L. & Calderon-Meza, G. (2008) *Passenger Trip Delays in the U.S. Airline Transportation System*. International Conference on Research in Air Transportation (ICRAT-2008), Fairfax, VA, 2008

Sherry, L. (2011). *Modeling Passenger Trip Reliability: Why NextGen May Not Improve Passenger Delays.* Journal of Air Traffic Control, 53(3), 11.

Sherry, L. (2013) *A model for estimating airline passenger trip reliability metrics from system-wide flight simulations.* Journal of Transport Literature, vol. 7, n. 2, pp. 319-337.

Sherry, L. (2014) *A method for quantifying travel productivity for corporate travel managers.* Journal of Air Transport Management, Volume 42, January 2015, Pages 118–124

Sherry, L., & Donohue, G. (2008). *US airline passenger trip delay report (2007).* Center for Air Transportation Systems Research. George Mason University.

Sherry, L., Calderon-Meza, G., & Donohue, G. (2010). *Trends in airline passenger trip delays: exploring the design of the passenger air transportation service*. In Transportation Research Board 89th Annual Meeting, Washington DC (pp. 1-14).

Sherry, L., Samant, A., & Calderon-Meza, G. (2010, September). *Trends in airline passenger trip delays (2007): a multi-segment itinerary analysis.* In American Institute of Aeronautics and Astronautics: 10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference.

Sherry, L., Wang D. (2007*) Air Travel Consumer Protection: Metric for Passenger OnTime Performance*. Transportation Research Record, Transportation Research Board of the National Academies, Volume 2007, pages 22-27.

Sherry, L., Wang, D., & Donohue, G. (2007). *Air travel consumer protection: metric for passenger on-time performance.* Transportation Research Record: Journal of the Transportation Research Board, 2007(1), 22-27.

Southwest Airlines Co. Contract of Carriage – Passenger Thirteenth Revised Effective May 6, 2015 https://www.southwest.com/assets/pdfs/corporate-commitments/contract-of-carriage.pdf

Southwest, *Customer Service Commitment, Our Mission Statement*, Available: http://swamedia.com/media_storage/CSC_Revised_05-01-2014_English.pdf

United Airlines Co. Contract of Carriage, https://www.united.com/web/format/pdf/Contract_of_Carriage.pdf

United, *Flight delays and cancellations*, Available: http://www.united.com/CMS/en-US/travel/policy/Pages/FlightDelaysandCancellations.aspx

Wang, D., & Sherry, L. (2006). *Passenger Trip Metric for Air Transportation*. 2nd International Conference on Research in Air Transportation-ICRAT.

Wambsganss, M.C. (2001). *New Concepts and Methods in Air Traffic Management*. In Transportation Analysis: Collaborative Decision Making in Air Traffic Management. Editors: Lucio Bianco, Paolo Dell'Olmo, Amedeo R. Odoni. Springer Berlin Heidelberg: Germany. pp 1-15

Yan, C., Vaze, V., Vanderboll, A., & Barnhart, C. (2015). *Tarmac Delay Policies: A Passenger-Centric Analysis*, Transportation Research Part A: Policy and Practice, Manuscript Draft

Zhang, Y., & Hansen, M. (2008). *Real-time intermodal substitution: Strategy for airline recovery from schedule perturbation and for mitigation of airport congestion*. Transportation Research Record: Journal of the Transportation Research Board, 2052(1), 90-99.

BIOGRAPHY

Sanja Avramovic received her Bachelor of Arts and Masters of Science from University of Belgrade, Serbia in 2009. She was employed in Serbian Broadcasting Corporation for 12 years. After she moved to the U.S. she worked in ArchLab, George Mason University as a GRA on flying simulators, and after that in The Green Technology Group as a data analyst on the contract with Department of Veterans Affairs, working with Big Databases and performing data mining. She is currently working as a GRA in the Department of Health Administration and Policy GMU on the development of CDE ontology and SEER database.