Technical Report

Group Members:

1. Betsy Varghese
2. Sebastien Pavot
3. Subhamoy Dam

Introduction

Outlined below are the SQL queries we have generated for our analysis on the flights dataset along with the associated goals and logic.

Query 1

Aim:

These queries were performed in order to obtain the five most relevant rows regarding

- Avg delay routes (Max & Min)
- Avg delay per carrier (Max & min)

```
Proc SQL outobs = 5;
Create table final.top5_maxdelay_routes as
Select avg(dep_delay + arr_delay) as delay, origin, dest
From deep.flights as f
Group by origin, dest
Having count(flight) > 10
Order by delay desc;
quit;
run;
```

Process:

- Here, we use outobs = 5 to limit the number of rows displayed as 5.
- We select the average of the sum of departure delay and arrival delay in order to have the average of the overall delay.
- Origin and dest are the parameters that we will change depending on the above-mentioned objectives.
- We group by origin and dest to avoid repetition of data.
- To ensure that the top 5 results printed are relevant, we put a condition stating that the routes should have had at least 10 flights operations over the year.
- We then sort these results in descending order by using the 'order by' function on delay (desc).
- To obtain routes with least average delay, we carry out the same query but remove the desc in the order by part.

Detailed Queries:

```
/* TOP 5 avg max delay routes */

Proc SQL outobs = 5;
Create table final.top5_maxdelay_routes as
Select avg(dep_delay + arr_delay) as delay, origin, dest
From deep.flights as f
Group by origin, dest
Having count(flight) > 10
Order by delay desc;
Quit;
Run;

/* TOP 5 avg min delay routes */

Proc SQL outobs = 5;
Create table final.top5_mindelay_routes as
Select avg(dep_delay + arr_delay) as delay, origin, dest
From deep.flights as f
Group by origin, dest
Having count(flight) > 10
Order by delay;
Quit;
Run;

/*TOP 5 AVG carrier max delay*/

Proc SQL outobs =5;
Create table final.top5_avg_carrier_maxdelay as
Select avg(dep_delay + arr_delay) as delay, a.carrier, a.name as Airlines
From deep.flights as f,
deep.airlines as a
Where f.carrier = a.carrier
Group by a.carrier, a.name
Having count(flight) > 50
Order by delay desc;
Quit;
Run;

/*TOP 5 AVG carrier min delay*/

Proc SQL outobs =5;
Create table final.top5_avg_carrier_mindelay as
Select avg(dep_delay + arr_delay) as delay, a.carrier, a.name as Airlines
From deep.flights as f,
deep.airlines as a
Where f.carrier = a.carrier
Group by a.carrier, a.name
Having count(flight) > 50
Order by delay;
Quit;
Run;
```

```
/*AVG Delay per month*/

Proc SQL;
/* Create table final.avg_delay_month as */
Select avg(dep_delay + arr_delay) as delay, month, count(f.flight) as
nbr_flights_year
From deep.flights as f
Group by month
Order by delay desc;
Quit;
Run;

/* AVG Delay per airports with sum of all flight over the year*/

Proc SQL;
Create table final.avg_delay_airport_with_flights as
Select avg(dep_delay + arr_delay) as delay, f.origin, a.name, count(f.flight)
as nbr_flights_year
From deep.flights as f,
deep.airports as a
Where f.origin = a.faa
Group by f.origin, a.name
Order by delay desc;
Quit;
Run;
```

Query 2

Aim:

To analyze the delay based on weather conditions. Here, we use queries to determine the impact of one parameter on the average delay.

The results thus obtained would be used to determine if this weather parameter could have influenced the delay of a flight.

We used the following four parameters for analysis:

- Visibility
- Pressure
- Wind speed
- Precipitation

```
Proc sql;
Create table final.visibility_delay as
Select avg(dep_delay + arr_delay) as delay,
Case when visib < 2 Then "Very low visibility"
       when visib < 4 Then "Low visibility"
       when visib < 6 Then "Medium visibility"
       when visib < 8 Then "Good visiblity"
       when visib < 9 Then "Very good visibility"
       Else "Awesome visibility" end "Visibility"
```

```
From deep.weather as w,
deep.flights as f
Where f.origin = w.origin
And f.time_hour = w.time_hour
Group by 2
Order by delay;
Quit;
Run;
```

Process:

- In this query, we try to analyze the impact of visibility conditions on delay.
- First, we select the average delay as we explained in the previous query.
- Then, in the select, we use the "case when" function to define different categories of visibility (we defined the categories ourselves as we did another query to see the distinct values of visibility. The results showed that they were between 0-10).
- Thus, for each "when", we define the parameters.
- In the first "when", it's minus 2 and so the result will be "Very low visibility" if the visibility is under 2. As a case when verifies the code iteration by iteration (i.e. only if the first iteration is false will it go to the next one), we do not need to specify the intervals in our case when.
- We specify end "Visibility" to ensure that the "case when" will create a new column "Visibility" with the value defined in the case when.
- After our case when, we make sure to link visibility parameter to the delayed flights.
- As weather is nearly 30 000 rows and flights are more than 300 000 rows, we make sure to do an inner join on 2 conditions in order to make sure SAS returns only the matched rows as per our specified criteria.
- To finish, we group by 2 or in other words our "case when" to print out 2 columns, first with the average delay and the second with the category defined in the case when.
- We order by delay to check if there is correlation between low delay and high visibility or vice versa in order to understand which parameters is an influential factor to the flight delay.

Detailed Queries:

```
/* Impact of visibility on delay */

Proc SQL;
Create table final.visibility_delay as
Select avg(dep_delay + arr_delay) as delay,
Case when visib < 2 Then "Very low visibility"
       when visib < 4 Then "Low visibility"
       when visib < 6 Then "Medium visibility"
       when visib < 8 Then "Good visiblity"
       when visib < 9 Then "Very good visibility"
       Else "Awesome visibility" end "Visibility"
From deep.weather as w,
deep.flights as f
Where f.origin = w.origin
```

```
And f.time_hour = w.time_hour
Group by 2
Order by delay
;
Quit;
Run;

/* Impact of pressure on delay */

Proc SQL;
Create table final.pressure_delay as
Select avg(dep_delay + arr_delay) as delay,
Case when pressure < 980 Then "Very low pressure"
       when pressure < 1000 Then "Low pressure"
       when pressure < 1020 Then "Medium pressure"
       when pressure < 1040 Then "High pressure"
       Else "Extreme pressure" end "Pressure"
From deep.weather as w,
deep.flights as f
Where f.origin = w.origin
And f.time_hour = w.time_hour
Group by 2
Order by delay
;
Quit;
Run;

/* Impact of windspeed on delay */

Proc SQL;
Create table final.windspeed_delay as
Select avg(dep_delay + arr_delay) as delay,
Case when wind_speed < 5 Then "Very low wind"
       when wind_speed < 15 Then "Low wind"
       when wind_speed < 25 Then "Medium wind"
       when wind_speed < 35 Then "High wind"
       Else "Extreme Wind" end "Wind Speed"
From deep.weather as w,
deep.flights as f
Where f.origin = w.origin
And f.time_hour = w.time_hour
Group by 2
Order by delay
;
Quit;
Run;

/* Impact of precip on delay */

Proc SQL;
Create table final.precip_delay as
Select avg(dep_delay + arr_delay) as delay,
Case when precip = 0 Then "No precipitations"
       when precip < 0.2 Then "Very Low precipitation"
       when precip < 0.4 Then "Low precipitation"
       when precip < 0.6 Then "Medium precipitation"
       when precip < 0.8 Then "High wind"
```

```
       Else "Extreme Precipitation" end "Wind Speed"
From deep.weather as w,
deep.flights as f
Where f.origin = w.origin
And f.time_hour = w.time_hour
Group by 2
Order by delay
;
Quit;
Run;


/* Aiport with the lowest quality weather */

Proc SQL;
Create table final.weather_airport as
Select a.faa, a.name, avg(pressure)as pressure, avg(wind_speed)as Wind_Speed,
avg(precip)as Precipitation,
avg(visib) as Visibility, avg(dep_delay + arr_delay) as Delay
From deep.airports as a,
deep.weather as w,
deep.flights as f
Where a.faa = w.origin
and w.origin = f.origin
and w.time_hour = f.time_hour
Group by a.name, a.faa
;
Quit;
Run;
```

Query 3

Aim:

To analyze the airtime for different routes and different airline companies. To also check if the assumption that greater airtime is correlated to a higher delay. (This has been graphically represented in our Tableau story)

Detailed Queries:

```
/* Routes with maximum air_time */

proc sql outobs = 5;
select origin, dest, avg(air_time) as Average_Air_Time
from deep.flights
group by 1, 2
order by 3 desc;
quit;
run;
```

```
/* Companies with highest air_time */

proc sql outobs = 5;
select A.name, sum(F.air_time) as Total_Air_Time
from deep.flights F left outer join deep.Airlines A
  on F.carrier = A.carrier
group by 1
order by 2 desc;
quit;
run;

/* Routes with maximum delays and air-time */

proc sql;
create table deep.avg_delays as
select origin, dest, avg(arr_delay + dep_delay) as Avg_Total_Delay
from deep.flights
group by 1, 2
order by 3 desc;
quit;
run;

proc sql;
select F.origin, F.dest, D.Avg_Total_Delay, sum(F.air_time) as
Total_Air_Time
from deep.avg_delays D left outer join deep.flights F
  on D.dest = F.dest
group by 1, 2
order by 3, 4;
quit;
run;
```

Query 4

Aim:

To get an overview of the flight operations by different carrier companies. To also find out which are the busiest airports in the USA based on the number of incoming flights.

Detailed Queries:

```
/* Busiest Airports based on air traffic (incoming flights) */

proc sql;
select dest, count(flight) as Total_Incoming
from deep.flights
group by 1
order by 2 desc;
quit;
run;
```

```
/*Total flight operations by different carriers */

proc sql;
create table group.Carrier_Operations as
select A.name, count(F.carrier) as Total_Operations
from deep.Airlines A, deep.flights as F
where A.carrier = F.carrier
group by 1
order by 2 desc;
quit;
run;
```