FLIGHT ANALYSIS AND OPTIMIZATION FOR US AIRLINES

Team :

- Shubham Ghate
- Sai Charan Komati
- Akhil Kumar

PROJECT OVERVIEW:

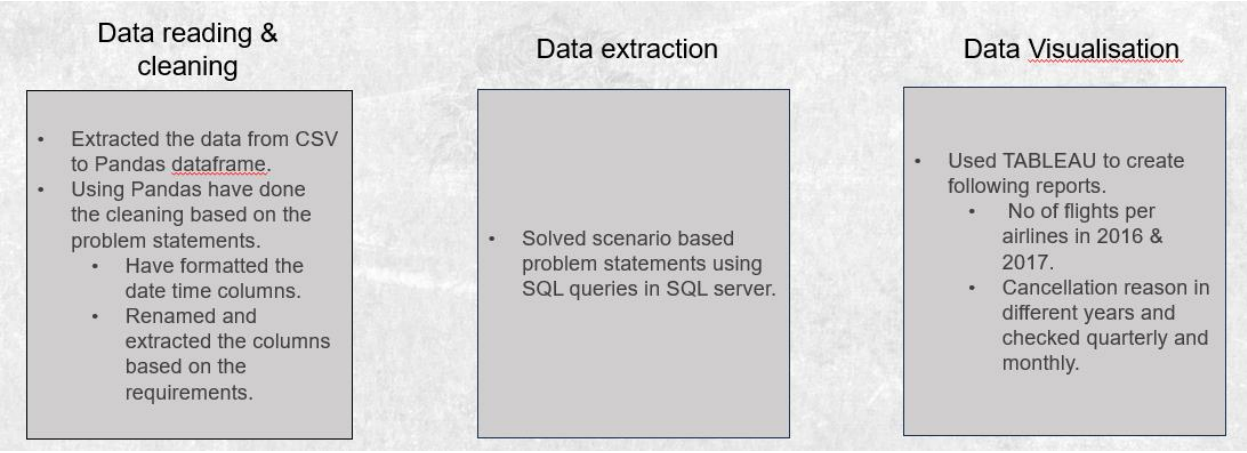
Understand and analyse various aspects of the US airlines 2015, 2016 & 2017 data using Python and SQL. Provide meaningful insights and reports to stakeholders in the airlines department. -Improve decision-making processes by identifying patterns, trends, and relationships within the airlines data.

Gain hands-on experience in Python and SQL query writing, and data analysis within the airlines domain.

Tools, Technologies & Libraries Used :

1) **Python**
    a. **Pandas** : For the added advantage for describing the
        i. the data , working with null value and
        ii. various data manipulation capabilities .
    b. **Numpy** : For mathematical calculation and applying various formulas .
    c. **Datetime** : For date data manipulation
2) **SQL Server:**
    a. Data Extraction
3) **Tableau:**
    a. Data Visualisation

PROJECT WORKFLOW:



Problem Statements

Problem Statement 1:

Create a data frame to store the processed data from the raw file. This processed data is to be uploaded to a SQL database in a cloud server.Read the 2015 flight data and find the Flight date in YYYY-MM-DD format. flight_date (YYYY-MM-DD format) and derive other data in the given format.

```
origin_iata (IATA_CODE of the origin Airport)
destination_iata (IATA_CODE of the destination Airport)
departure_time_delay (in minutes)
airline_iata(IATA_CODE of the Airline)
air_time (in minutes)
distance (in miles)
```

```
# Generating flight date using the given columns
df_2015['FLIGHT_DATE'] = pd.to_datetime(df_2015[['YEAR', 'MONTH', 'DAY']])

# Creating the new df for required data and renaming the columns as required
PS1 = df_2015[['FLIGHT_DATE', 'ORIGIN_AIRPORT', 'DESTINATION_AIRPORT', 'DEPARTURE_DELA
PS1.columns = ['FLIGHT_DATE', 'ORIGIN_IATA', 'DESTINATION_IATA', 'DEPARTURE_TIME_DELAY
PS1
```

Problem Statement 2:

The Scheduled departure time is required to be stored in a timestamp format which can be used in SQL. Find scheduled_departure_time (TIMESTAMP - format: YYYY-MM-DD HH:MI:SS) The scheduled date of departure is the same as the flight date. And the time can be derived from SCHEDULED_DEPARTURE of the raw data. However, the scheduled departure time is in a format such the rightmost two numbers signify the minutes and the leftmost two numbers signify the hour.

Example:

Raw Data (SCHEDULED_DEPARTURE)	Derived time (HH:MI:SS format)
1042	10:42:00
559	05:59:00
35	00:35:00

Create a function named get_processed_time to process the time so that it can be reused. The DEPARTURE_TIME, SCHEDULED_ARRIVAL, ARRIVAL_TIME are in similar format and the function can be reused to process the time for that as well.

```
# Functiong for generating TimeStamp
def get_processed_time(df, column_name):
    df['MINUTES'] = df[[column_name]] % 100
    df['HOURS'] = df[[column_name]] // 100
    return pd.to_datetime(df[['YEAR', 'MONTH', 'DAY', 'HOURS', 'MINUTES']])
```

```
# Generating SCHEDULED_DEPARTURE_TIME as Time Stamp
df = df_2015.copy()
df_2015['SCHEDULED_DEPARTURE_TIME'] = get_processed_time(df, 'SCHEDULED_DEPARTURE')
df_2015['SCHEDULED_DEPARTURE'] = df_2015['SCHEDULED_DEPARTURE_TIME'].astype('str').str
```

Problem Statement 3:

Find the actual_departure_time (TIMESTAMP - format: YYYY-MM-DD HH:MI:SS). The actual departure date isn't necessarily the same as the flight date. In case there is a delay in the flight departure, the date may change. Write a function that takes the scheduled_departure_time, actual_departure_time and departure_time_delay argument and returns True if the date may have changed due to delay. Use the above function to find the actual departure date. Use the get_processed_time function to find the processed time. From the processed data deduce the actual_departure_time.

```
# Function to add ACTUAL_DEPARTURE_DELAY
def add_delay(df, SCHEDULED_TIME, TIME_DELAY):
    return df[SCHEDULED_TIME] + pd.to_timedelta(df[TIME_DELAY], unit='m')
```

```
# Generating ACTUAL_DEPARTURE_TIME with adding DEPARTURE_DELAY
df_2015['ACTUAL_DEPARTURE_TIME'] = add_delay(df_2015, 'SCHEDULED_DEPARTURE_TIME', 'DEP
```

Problem Statement 4:

Find the scheduled_arrival_time (TIMESTAMP - format: YYYY-MM-DD HH:MI:SS) Use the functions defined earlier if/when required. The date of arrival may change due to various reasons like different time zones and time taken for the flight. Create a logic to find if the date may have changed. Find actual_arrival_time (TIMESTAMP - format: YYYY-MM-DD HH:MI:SS). It is similar to the scheduled_arrival_time however the date may change for an additional reason, i.e., arrival time delay.

```
# Generating SCHEDULED_ARRIVAL_TIME as Time Stamp
df_2015['SCHEDULED_ARRIVAL_TIME'] = get_processed_time(df, 'SCHEDULED_ARRIVAL')
```

```
# Generating ACTUAL_DEPARTURE_TIME with adding DEPARTURE_DELAY
df_2015['ACTUAL_ARRIVAL_TIME'] = df_2015['SCHEDULED_ARRIVAL_TIME'] + pd.to_timedelta(d
```

Problem Statement 5:

I. Create a flight_id for each flight such that the ID starts with US150000000000 and goes on like US150000000001, US150000000002 ....

II. Find cancellation_code (A = Carrier, B = Weather, C = National Air System, D = Security, N = Not Cancelled) using the CANCELLED and CANCELLATION_REASON attributes of the raw data.

III. Create a table in the database named Flights2015 and upload the data to it. The table should have the following column names.

```
# I.Create a flight_id for each flight such that the ID starts with
#US160000000000 and goes on like US160000000001, US160000000002 …

start_id = 150000000000
df_2015['flight_id'] = ['US' + str(start_id + i) for i in range(len(df_2015))]
```

```
df_2015['CANCELLATION_REASON'].value_counts()
```

Problem Statement 6:

Create a data frame to store the processed data from the raw file. This processed data is to be uploaded to a SQL database in a cloud server. Read the 2016 flight data and find the attributes in the given format

```
# Converting FL_DATE into datetime object
df_2016['FL_DATE'] = pd.to_datetime(df_2016['FL_DATE'])

# Creating the new df for required data and renaming the columns as required
PS6 = df_2016[['FL_DATE', 'ORIGIN', 'DEST', 'DEP_DELAY', 'OP_CARRIER', 'AIR_TIME', 'DI
PS6.columns = ['FLIGHT_DATE', 'ORIGIN_IATA', 'DESTINATION_IATA', 'DEPARTURE_TIME_DELAY
PS6
```

Problem Statement 7:

The Scheduled departure time is required to be stored in a timestamp format which can be used in SQL. Find scheduled_departure_time (TIMESTAMP - format: YYYY-MM-DD HH:MI:SS) The scheduled date of departure is the same as the flight date. And the time can be derived from CRS_DEP_TIME of the raw data. However, the scheduled departure time is in a format such the rightmost two numbers signify the minutes and the leftmost two numbers signify the hour.

```
# Generating SCHEDULED_DEPARTURE_TIME as Time Stamp
df_2016['YEAR'] = df_2016['FL_DATE'].dt.year
df_2016['MONTH'] = df_2016['FL_DATE'].dt.month
df_2016['DAY'] = df_2016['FL_DATE'].dt.day

df = df_2016.copy()

df_2016['SCHEDULED_DEPARTURE_TIME'] = get_processed_time(df, 'CRS_DEP_TIME')
df_2016['CRS_DEP_TIME'] = df_2016['SCHEDULED_DEPARTURE_TIME'].astype('str').str[11:]
```

Problem Statement 8:

Find the actual_departure_time (TIMESTAMP - format: YYYY-MM-DD HH:MI:SS) The actual departure date isn't necessarily the same as the flight date. In case there is a delay in the flight departure, the date may change. Write a function that takes the scheduled_departure_time, actual_departure_time and departure_time_delay argument and returns True if the date may have changed due to delay. Use the above function to find the actual departure date. Use the

```
# Generating ACTUAL_DEPARTURE_TIME with adding DEPARTURE_DELAY
df_2016['ACTUAL_DEPARTURE_TIME'] = add_delay(df_2016, 'SCHEDULED_DEPARTURE_TIME', 'DEP
```

Problem Statement 9:

Find the scheduled_arrival_time (TIMESTAMP - format: YYYY-MM-DD HH:MI:SS) Use the functions defined earlier if/when required. The date of arrival may change due to various reasons like different time zones and time taken for the flight. Create a logic to find if the date may have changed. The

Find actual_arrival_time (TIMESTAMP - format: YYYY-MM-DD HH:MI:SS) It is similar to the scheduled_arrival_time however the date may change for an additional reason, i.e., arrival time delay.

```
# Generating SCHEDULED_ARRIVAL_TIME as Time Stamp
df_2016['SCHEDULED_ARRIVAL_TIME'] = get_processed_time(df, 'CRS_ARR_TIME')
```

```
# Generating ACTUAL_DEPARTURE_TIME with adding DEPARTURE_DELAY
df_2016['ACTUAL_ARRIVAL_TIME'] = df_2016['SCHEDULED_ARRIVAL_TIME'] + pd.to_timedelta(
```

Problem Statement 10:

I. Create a flight_id for each flight such that the ID starts with US160000000000 and goes on like US160000000001, US160000000002 ....

II. Find cancellation_code (A = Carrier, B = Weather, C = National Air System, D = Security, N = Not Cancelled) using the CANCELLED and CANCELLATION_CODE attributes of the raw data.

III. Create a table in the database named Flights2016 and upload the data to it. The table should have the following column names.

```
# I.Create a flight_id for each flight such that the ID starts with
#US160000000000 and goes on like US160000000001, US160000000002 …

start_id = 160000000000
df_2016['flight_id'] = ['US' + str(start_id + i) for i in range(len(df_2016))]
```

```
# II. Generating cancellation code
df_2016.loc[df_2016.CANCELLED==0, 'CANCELLATION_CODE'] = 'N'
Conditions = {'A':'Carrier','B':'Weather', 'C':'National Air System', 'D':'Security',
df_2016['CANCELLATION_REASON'] = df_2016['CANCELLATION_CODE'].apply(lambda x: Conditic
```

```
F_2016_Final_Data = df_2016[['flight_id', 'FL_DATE', 'OP_CARRIER', 'ORIGIN', 'DEST',
F_2016_Final_Data.rename(columns={'FL_DATE':'FLIGHT_DATE',
                                  'OP_CARRIER':'AIRLINE_IATA',
                                  'ORIGIN':'ORIGIN_IATA',
                                  'DEST':'DESTINATION_IATA',
                                  'DEP_DELAY':'DEPARTURE_TIME_DELAY',
                                  'ARR_DELAY':'ARRIVAL_TIME_DELAY',
                                  'CANCELLATION_REASON': 'CANCELLATION_CODE'},inplace=True)
```

Problem Statement 11:

The data/airport.csv file contains the details about different airports in USA, however some data is missing. As we may need those data, we can make use of another data file that contains the airport data from all over the world(data/iata_icao.csv) Find the missing data and upload the data to a table named Airports_usa with the following column names. airport_iata_code, airport_name, state, city, latitude, and logitude

```
df_iata=pd.read_csv('airports.csv')
df_iata_icao=pd.read_csv('iata_icao.csv')
df_airlines=pd.read_csv('airlines.csv')
df_states = pd.read_csv('states.txt', delimiter=':',header=0)
```

```
Airports_usa = pd.merge(df_iata, df_iata_icao, how='inner', left_on='IATA_CODE', right
Airports_usa['LATITUDE'] = Airports_usa['LATITUDE'].fillna(Airports_usa['latitude'])
Airports_usa['LONGITUDE'] = Airports_usa['LONGITUDE'].fillna(Airports_usa['longitude']
```

```
Airports_usa = Airports_usa[['IATA_CODE','AIRPORT','STATE','CITY','LATITUDE','LONGITUD
```

Problem Statement 13:

Write a SQL query that finds out for each airport, the airport name, the airport code, the name of the state it is located in, the state code(usps abbr), and the number of airports that exist in that state.

```
df_airports_states = pd.merge(df_iata, df_states, how='inner', left_on='STATE', right_
```

```
df_airports_states = df_airports_states[['IATA_CODE', 'AIRPORT', 'State Name', 'STATE'
```

```
df_group = df_airports_states.groupby(['State Name','STATE']).AIRPORT.count().reset_ir
```

Problem Statement 14:

The distance between two locations can be calculated(in km) with the following formula

```
acos(sin(lat1)*sin(lat2)+cos(lat1)*cos(lat2)*cos(lon2-lon1))*6371
(6371 is Earth radius in km.)
```

Where lat1, lon1 are the latitude and longitude of location1 and Lat2, lon2 are the latitude and longitude of location2. And 1 km, 0.6213711922 miles

Find the 3 closest airports to Waco Regional Airport(Texas).

```
df_lat = df_iata[df_iata.AIRPORT=='Waco Regional Airport']['LATITUDE']
```

```
df_lat = df_iata[df_iata.AIRPORT=='Waco Regional Airport']['LATITUDE'][6]
df_long = df_iata[df_iata.AIRPORT=='Waco Regional Airport']['LONGITUDE'][6]
df_lat, df_long
```

```
(31.61129, -97.23052)
```

```
df_iata['distance_WRATXAS'] = np.arccos(np.sin(np.deg2rad(df_iata['LATITUDE']))*np.sir
```

```
df_iata[df_iata['AIRPORT']!='Waco Regional Airport'][['AIRPORT','distance_WRATXAS']].s
```

Problem Statement 16:

Read the header file and associate it with the flight data with no header.

Find the Flight date in YYYY-MM-DD format.

```
# Generating flight date using the given columns
df_2017['FLIGHT_DATE'] = pd.to_datetime(df_2017[['YEAR', 'MONTH', 'DAY']])

# Creating the new df for required data and renaming the columns as required
PS16 = df_2017[['FLIGHT_DATE', 'ORIGIN_IATA', 'DESTINATION_IATA', 'DEP_DELAY_NEW', 'OF
PS16.columns = ['FLIGHT_DATE', 'ORIGIN_IATA', 'DESTINATION_IATA', 'DEPARTURE_TIME_DELA
PS16
```

Problem Statement 18:

The Scheduled departure time is required to be stored in a timestamp format which can be used in SQL. Find scheduled_departure_time (TIMESTAMP - format: YYYY-MM-DD HH:MI:SS)

The scheduled date of departure is the same as the flight date. And the time can be derived from CRS_DEP_TIME of the raw data. However, the scheduled departure time is in a format such the rightmost two numbers signify the minutes and the leftmost two numbers signify the hour.

```
# Generating SCHEDULED_DEPARTURE_TIME as Time Stamp
df_2017['HOURS'] = df_2017['CRS_DEP_TIME'].str[:2]
```

```
df_2017['MINUTES'] = df_2017['CRS_DEP_TIME'].str[3:5]
df_2017['SCHEDULED_DEPARTURE_TIME'] = pd.to_datetime(df_2017[['YEAR', 'MONTH', 'DAY',
```

Problem Statement 19:

Find the actual_departure_time (TIMESTAMP - format: YYYY-MM-DD HH:MI:SS)

The actual departure time can be found using the The actual departure date isn't necessarily the same as the flight date. In case there is a delay in the flight departure, the date may change.

```
# Generating ACTUAL_DEPARTURE_TIME with adding DEPARTURE_DELAY
df_2017['ACTUAL_DEPARTURE_TIME'] = add_delay(df_2017, 'SCHEDULED_DEPARTURE_TIME', 'DEP
```

Problem Statement 20:

Find the scheduled_arrival_time (TIMESTAMP - format: YYYY-MM-DD HH:MI:SS) Use the functions defined earlier if/when required.

The date of arrival may change due to various reasons like different time zones and time taken for the flight. Create a logic to find if the date may have changed. The

Find actual_arrival_time (TIMESTAMP - format: YYYY-MM-DD HH:MI:SS) It is similar to the scheduled_arrival_time however the date may change for an additional reason, i.e., arrival time delay.

```
# Generating SCHEDULED_ARRIVAL_TIME as Time Stamp
df_2017['SCHEDULED_ARRIVAL_TIME'] = get_processed_time(df, 'CRS_ARR_TIME')
```

Problem Statement 21:

I. Create a flight_id for each flight such that the ID starts with US170000000000 and goes on like US170000000001, US170000000002 ....

II. Find cancellation_code (A = Carrier, B = Weather, C = National Air System, D = Security, N = Not Cancelled) using the CANCELLED and CANCELLATION_CODE attributes of the raw data.

III. Create a table in the database named Flights2017 and upload the data to it. The table should have the following column names.

```
# I.Create a flight_id for each flight such that the ID starts with
#US160000000000 and goes on like US160000000001, US160000000002 …

start_id = 170000000000
df_2017['flight_id'] = ['US' + str(start_id + i) for i in range(len(df_2017))]
```

```
# II. Generating cancellation code
df_2017.loc[df_2017.CANCELLED==0, 'CANCELLATION_CODE'] = 'N'
Conditions = {'A':'Carrier','B':'Weather', 'C':'National Air System', 'D':'Security',
df_2017['CANCELLATION_REASON'] = df_2017['CANCELLATION_CODE'].apply(lambda x: Conditic
```

```
F_2017_Final_Data = df_2017[['flight_id', 'FLIGHT_DATE', 'OP_UNIQUE_CARRIER', 'ORIGIN_
F_2017_Final_Data.rename(columns={'OP_UNIQUE_CARRIER':'AIRLINE_IATA',
                                  'DEP_DELAY_NEW':'DEPARTURE_TIME_DELAY',
                                  'ARR_DELAY_NEW':'ARRIVAL_TIME_DELAY',
                                  'CANCELLATION_REASON': 'CANCELLATION_CODE'},inplace=True)
```

Scenario based questions: SQL

Scenario 1:

Problem Statement:

Imagine that you are a data analyst for an airline company. Your team has been tasked with identifying all flights that were cancelled due to weather in the years 2015 and 2016. Your objective is to compile a list of these flights and organize them by flight date in descending order.

```
(SELECT FLIGHT_DATE, CANCELLATION_CODE, AIRLINE_IATA, ORIGIN_IATA, DESTINATION_IATA FROM Flights2015
WHERE CANCELLATION_CODE = 'Weather'
UNION
SELECT FLIGHT_DATE, CANCELLATION_CODE, AIRLINE_IATA, ORIGIN_IATA, DESTINATION_IATA FROM Flights2016
WHERE CANCELLATION_CODE = 'Weather')
ORDER BY FLIGHT_DATE DESC;
```

Scenario 2:

Problem Statement:

Imagine that you are a data analyst for a travel agency. Your team has been tasked with identifying which airlines operated flights in both 2016 and 2017, and the number of flights operated by each airline in each year.

To accomplish this, you will need to access a database containing information about flights operated by various airlines. Specifically, you will need to extract data on the airline's name and IATA code, as well as the flight date for each flight.

```
SELECT ar.AIRLINE,ar.IATA_CODE, YEAR(FLIGHT_DATE) AS YEAR_OF_FLYING, COUNT(flight_id) AS NUMBER_OF_FLIGHTS_OPERATED FROM Flights2016 f16
LEFT JOIN df_airlines ar
ON  f16.AIRLINE_IATA = ar.IATA_CODE
GROUP BY ar.AIRLINE,ar.IATA_CODE,YEAR(FLIGHT_DATE)
UNION
SELECT ar.AIRLINE,ar.IATA_CODE, YEAR(FLIGHT_DATE) AS YEAR_OF_FLYING, COUNT(flight_id) AS NUMBER_OF_FLIGHTS_OPERATED FROM Flights2017 f17
LEFT JOIN df_airlines ar
ON  f17.AIRLINE_IATA = ar.IATA_CODE
GROUP BY ar.AIRLINE,ar.IATA_CODE,YEAR(FLIGHT_DATE)
ORDER BY 3;
```

Scenario 3:

Problem Statement:

Imagine that you are a data analyst for an airline company. Your team has been tasked with identifying all flights that flew between New York (JFK) and Los Angeles (LAX) with an air time between 5 and 6 hours, and were not cancelled. The objective is to compile a list of these flights and organize them by scheduled departure time.

To accomplish this, you will need to access a database containing information about all flights operated by the airline. Specifically, you will need to extract data on the scheduled departure time, airline IATA code, origin IATA code, destination IATA code, and air time for each flight.

```
SELECT flight_id,ar.AIRLINE,ar.IATA_CODE,SCHEDULED_DEPARTURE_TIME, AIRLINE_IATA, ORIGIN_IATA, DESTINATION_IATA, AIR_TIME FROM Flights2015 f15
LEFT JOIN df_airlines ar
ON  f15.AIRLINE_IATA = ar.IATA_CODE
WHERE
  ORIGIN_IATA = 'JFK'
  AND DESTINATION_IATA = 'LAX'
  AND AIR_TIME >= '300' AND AIR_TIME <= '360'
  AND CANCELLATION_CODE='Not Cancelled'
 UNION
SELECT flight_id,ar.AIRLINE,ar.IATA_CODE,SCHEDULED_DEPARTURE_TIME, AIRLINE_IATA, ORIGIN_IATA, DESTINATION_IATA, AIR_TIME FROM Flights2016 f16
LEFT JOIN df_airlines ar
ON  f16.AIRLINE_IATA = ar.IATA_CODE
WHERE
  ORIGIN_IATA = 'JFK'
  AND DESTINATION_IATA = 'LAX'
  AND AIR_TIME >= '300' AND AIR_TIME <= '360'
  AND CANCELLATION_CODE='Not Cancelled'
UNION
SELECT flight_id,ar.AIRLINE,ar.IATA_CODE,SCHEDULED_DEPARTURE_TIME, AIRLINE_IATA, ORIGIN_IATA, DESTINATION_IATA, AIR_TIME FROM Flights2017 f17
LEFT JOIN df_airlines ar
ON  f17.AIRLINE_IATA = ar.IATA_CODE
WHERE
  ORIGIN_IATA = 'JFK'
  AND DESTINATION_IATA = 'LAX'
  AND AIR_TIME >= '300' AND AIR_TIME <= '360'
  AND CANCELLATION_CODE='Not Cancelled'

ORDER BY SCHEDULED_DEPARTURE_TIME ASC;
```

Scenario 4:

Problem Statement:

Imagine that you are a data analyst for United Airlines. Your team has been tasked with identifying all flights that had a departure delay of more than 2 hours, but arrived on time, and were operated by United Airlines. Your objective is to compile a list of these flights and extract relevant attributes from the data.

To accomplish this, you will need to access a database containing information about all flights operated by United Airlines. Specifically, you will need to extract data on the airline

IATA code, scheduled departure time, actual departure time, departure time delay, actual arrival time, and arrival time delay for each flight.

Next, you will need to filter the data to include only flights that were operated by United Airlines, had a departure delay of more than 2 hours, but arrived on time. This means that the actual arrival time should match the scheduled arrival time or be earlier.

Once you have filtered the data, you will need to extract the relevant attributes for each flight, including the airline IATA code, scheduled departure time, actual departure time, departure time delay, actual arrival time, and arrival time delay.

Finally, you will need to compile a list of all flights that met the specified criteria and extract the relevant attributes. This information will be used by your team to analyze flight operations, identify potential areas for improvement in United Airlines' services, and develop strategies to minimize departure delays while ensuring timely arrival for passengers.

attributes required: airline_iata, scheduled_departure_time, actual_departure_time, departure_time_delay, actual_arrival_time, arrival_time_delay

```
select AIRLINE_IATA,
SCHEDULED_DEPARTURE_TIME,
ACTUAL_DEPARTURE_TIME,
DEPARTURE_TIME_DELAY,
ACTUAL_ARRIVAL_TIME,
ARRIVAL_TIME_DELAY
from F_2015_Final_Data
where DEPARTURE_TIME_DELAY>120
and AIRLINE_IATA='UA'
and ARRIVAL_TIME_DELAY=0
```

Scenario 5:

Problem Statement:

Imagine that you are a data analyst for an airport in JFK. Your team has been tasked with identifying all flights that were scheduled to depart from JFK in 2015, but were cancelled due to the carrier, and for which there was no other flight on the same day by the same airline to the same destination. Your objective is to compile a list of these flights.

To accomplish this, you will need to access a database containing information about all flights that departed from JFK in 2015. Specifically, you will need to extract data on the scheduled departure time, airline IATA code, origin IATA code, destination IATA code, and cancellation reason for each flight.

```
 SELECT
  FLIGHT_ID,
  FLIGHT_DATE,
  ORIGIN_IATA,
  CANCELLATION_CODE,
  AIRLINE_IATA
FROM
  Flights2015 f1
WHERE
  ORIGIN_IATA = 'JFK'
  AND YEAR(FLIGHT_DATE) = 2015
  AND CANCELLATION_CODE = 'Carrier'
  AND NOT EXISTS (
    SELECT 1
    FROM Flights2015 f2
    WHERE f2.ORIGIN_IATA = f1.ORIGIN_IATA
      AND f2.AIRLINE_IATA = f1.AIRLINE_IATA
      AND f2.DESTINATION_IATA = f1.DESTINATION_IATA
      AND f2.FLIGHT_DATE = f1.FLIGHT_DATE
      AND f2.CANCELLATION_CODE <> 'Carrier'
  )
```

Scenario 6:

Problem Statement:

Imagine that you are a data analyst for Delta Airlines. Your team has been tasked with identifying the average distance and airtime of all the flights operated by Delta Airlines in 2016, grouped by origin airport. Your objective is to compile a list of these flights and extract relevant attributes from the data.

To accomplish this, you will need to access a database containing information about all flights operated by Delta Airlines in 2016. Specifically, you will need to extract data on the airline IATA code, origin IATA code, distance, and airtime for each flight.

```
SELECT
  AIRLINE_IATA,
  ORIGIN_IATA,
  AVG(DISTANCE) AS average_distance,
  AVG(AIR_TIME) AS average_airtime
FROM
  Flights2016
WHERE
  AIRLINE_IATA = 'DL'
  AND YEAR(FLIGHT_DATE) = 2016
GROUP BY
  AIRLINE_IATA,
  ORIGIN_IATA;
```

Scenario 7:

Problem Statement:

Imagine that you are a data analyst for American Airlines. Your team has been tasked with identifying the ranking of flights operated by American Airlines in 2017, based on the departure delay, with ties included. Your objective is to compile a list of these flights and extract relevant attributes from the data.

To accomplish this, you will need to access a database containing information about all flights operated by American Airlines in 2017. Specifically, you will need to extract data on the airline IATA code, scheduled departure time, actual departure time, and departure time delay for each flight.

Next, you will need to filter the data to include only flights that were operated by American Airlines in 2017. Then, you will need to rank the flights based on departure delay, with ties included. This means that if two or more flights have the same departure delay, they should be ranked the same.

Once you have completed this analysis, you will have a comprehensive list of all flights operated by American Airlines in 2017, ranked based on departure delay with ties included. This information will be used by your team to identify potential issues in flight operations, assess the efficiency of American Airlines' services, and take steps to improve the airline's operations in the future.

attributes required: airline_iata, scheduled_departure_time, actual_departure_time, departure_time_delay

```sql
select AIRLINE_IATA,
SCHEDULED_DEPARTURE_TIME,
ACTUAL_DEPARTURE_TIME,
DEPARTURE_TIME_DELAY,
DENSE_RANK() over(order by departure_time_delay) as ranking
from F_2017_Final_Data
where AIRLINE_IATA='AA'
and DEPARTURE_TIME_DELAY<>0
order by ranking desc;
```

Scenario 9:

Problem Statement:

Imagine that you are a data analyst working for an airline. Your team has been tasked with creating a function that takes an airline IATA code and a date as input, and returns the total number of flights operated by that airline on that date. Your objective is to create this function.

To accomplish this, you will need to access a database containing information about all flights operated by the airline. Specifically, you will need to extract data on the airline IATA code, flight date, and flight ID for each flight.

Next, you will need to create a function that takes an airline IATA code and a date as input. Within the function, you will need to filter the flight data to include only flights operated by the specified airline on the specified date. Then, you will need to count the number of flights that meet these criteria and return this value.

Once you have created this function, it can be used to retrieve information quickly and easily about the total number of flights operated by a particular airline on a particular date. This information can be used by your team to analyze flight operations, track performance metrics, and identify areas for improvement.

attributes required: IATA_CODE, flight_date, flight_id

Z

```sql
create function no_of_flights(@iata varchar(20),@date date)
returns table as return
select AIRLINE_IATA,FLIGHT_DATE,
count(flight_id) as no_of_flights
from F_2015_Final_Data
where AIRLINE_IATA=@iata and FLIGHT_DATE=@date
group by AIRLINE_IATA,FLIGHT_DATE;

select * from no_of_flights('OO','2015-10-25');
```

90 %

Results | Messages

	AIRLINE_IATA	FLIGHT_DATE	no_of_flights
1	OO	2015-10-25	1608

Visualization

➢

Years vs Cancellation Reason

2015

Cancellation Code	
Weather	54.35%
Carrier	28.11%
National Air System	17.52%
Security	0.02%

2016

Cancellation Code	
Weather	52.33%
Carrier	30.79%
National Air System	16.84%
Security	0.04%

2017

Cancellation Code	
Weather	59.65%
National Air System	20.14%
Carrier	20.11%
Security	0.09%

No of flights cancelled per quater(weather)

Quarter of Flight Date	
Q1	60.04%
Q4	18.28%
Q2	16.16%
Q3	5.52%

Quarter of Flight D..	
Q1	46.07%
Q4	25.75%
Q2	16.22%
Q3	11.97%

Quarter of Flight Date	
Q3	48.33%
Q1	32.22%
Q2	10.74%
Q4	8.71%

No of flights cancelled per Month(weather)

Month of Flight Date	
February	31.62%
January	14.37%
March	14.05%
December	11.49%
June	6.81%
May	5.69%
November	4.79%
April	3.66%
August	2.68%
October	2.00%
July	1.81%
September	1.03%

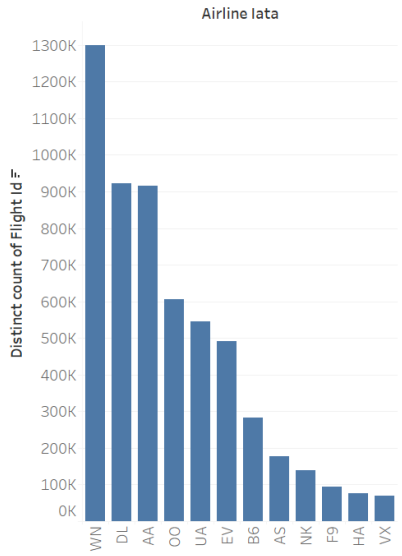
Month of Flight Date	
January	26.27%
December	14.15%
February	10.97%
October	10.11%
March	8.83%
April	7.02%
July	6.34%
June	5.86%
August	4.43%
May	3.33%
November	1.49%
September	1.20%

Month of Flight Date	
August	22.41%
September	20.39%
March	12.95%
January	9.78%
February	9.49%
July	5.53%
December	5.37%
April	4.72%
June	4.25%
October	2.30%
May	1.76%
November	1.04%

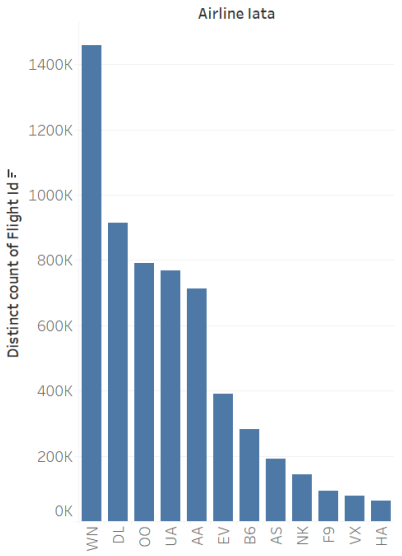
➢

Scenario 2: Airline IATA vs Numbers of Flights

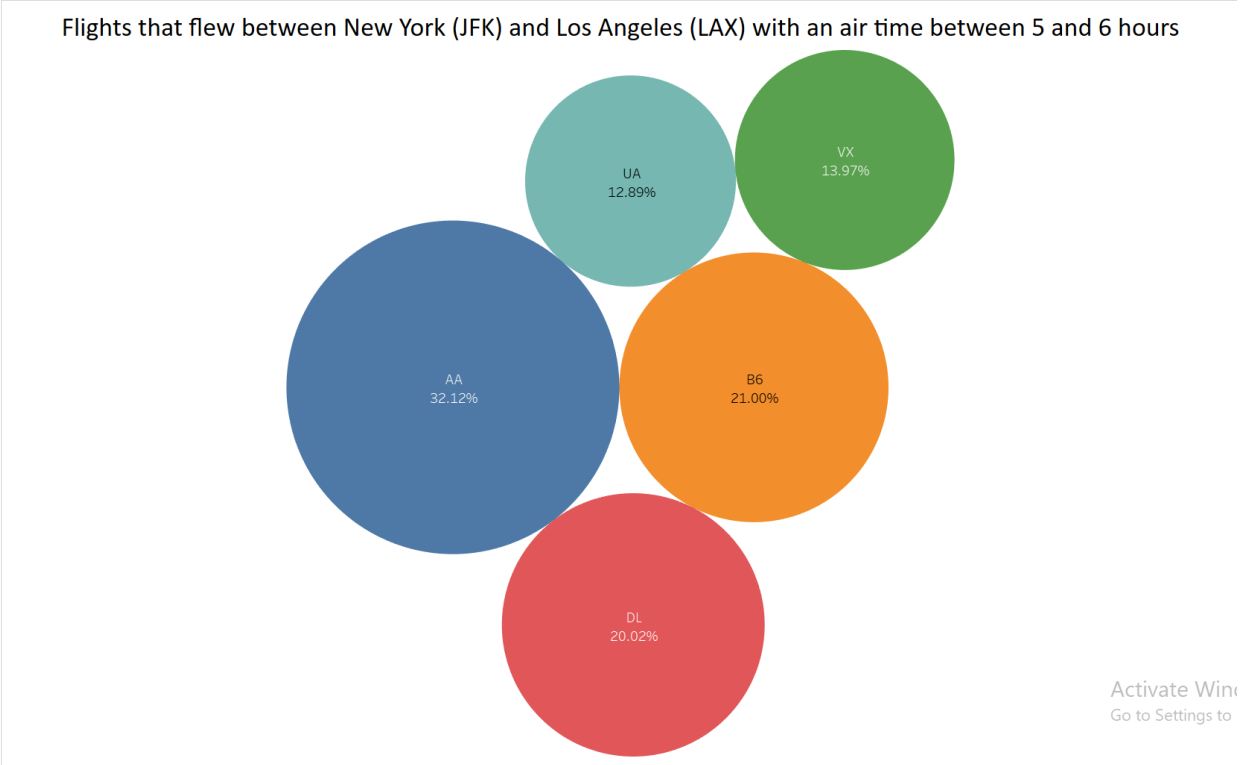
2016



2017



➢


Flights that flew between New York (JFK) and Los Angeles (LAX) with an air time between 5 and 6 hours
UA
12.89%
VX
13.97%
AA
32.12%
B6
21.00%
DL
20.02%
Activate Win
Go to Settings to

Insights:

1. There were many factors influencing flight cancellations, but the major one was weather, accounting for more than 50% of cancellations in 2015, 2016, and 2017.
2. In 2016 and 2017, Southwest Airlines Co.(WN) operated the most flights, with a total of 1,299,444 and 1,458,978, respectively. Additionally, in 2016, Virgin America(VX) operated the fewest flights, while in 2017, Hawaiian Airlines Inc.(HA) operated the fewest flights.
3. The maximum number of flights were operated by American Airlines (AA) from New York to Los Angeles, with an air time of 5 to 6 hours. On the other hand, the minimum number of flights were operated by United Airlines (UA) from New York to Los Angeles, also with an airtime of 5 to 6 hours.
4. Spirit Airlines (NK) had the highest number of delayed flights in 2015, while JetBlue Airways (B6) had the highest number of delayed flights in both 2016 and 2017.
5. In 2015, there were 235 flights operated by United Airlines that experienced a departure delay of more than 2 hours but arrived on time.
6. Customers traveling through certain airlines should plan accordingly due to the high number of delayed operations.