# 2015 Flight Delays and Cancellations

## Big-Data Architecture and Governance - DAMG 7250

Group 2

Aishwarya Lamture

Dwithika Shetty

Vini Wategaonkar

This project consists of building data dashboards for visual representation using PowerBI by loading data using Neo4J. The corresponding datasets document a variety of airline information including delays, cancellations, dates, and destinations. The dataset is from 2015 for major airlines and airports. The goal of the dashboards are to visualize which airlines have the most delays and potential causes. This information could potentially be useful in the improvement and optimization of airlines,which could improve their branding, customer experience/satisfaction.

## Columns used for dimensions

**Dimensions**

ON_TIME
ARRIVED_EARLY
ARRIVED_LATE
IS_AIR_SYSTEM_DELAY
IS_SECURITY_DELAY
IS_AIRLINE_DELAY
IS_LATE_AIRCRAFT_DELAY
IS_WEATHER_DELAY

**How would you generate new dimension?**

To create ON_TIME, ARRIVED_EARLY, ARRIVED_LATE dimensions we have use ARRIVAL_DELAY column and applied condition on ARRIVAL_DELAY to get desire result.

Using AIR_SYSTEM_DELAY we generated IS_AIR_SYSTEM_DELAY dimension to check whether air system delay occured or not

Using SECURITY_DELAY we generated IS_SECURITY_DELAY dimension to check whether security delay occured or not

Using AIRLINE_DELAY we generated IS_AIRLINE_DELAY dimension to check whether delay due to airline occured or not

Using LATE_AIRCRAFT_DELAY we generated IS_LATE_AIRCRAFT_DELAY dimension to check whether delay due to aircraft occured or not

Using WEATHER_DELAY we generated IS_WEATHER_DELAY dimension to check whether delay due to weather took place or not

## Data Profiling Instructions

1) Download Python (follow instructions on https://www.python.org/downloads/ based on operating system

2) Install pandas library by running pip install pandas

3) Install pandas profiling to be able to create an html profile report by running the command pip install pandas-profiling

4) Since we had 3 csv files, for profiling we first checked all the 3 csv's and merged those files into one file.

5) Run profiling.ipynb script and output.html file get generated

6) Open output.html on browser to view data profile report

## Data Wrangling/Cleansing Instructions

1) Make sure pandas library is installed (should have been installed during data profiling)

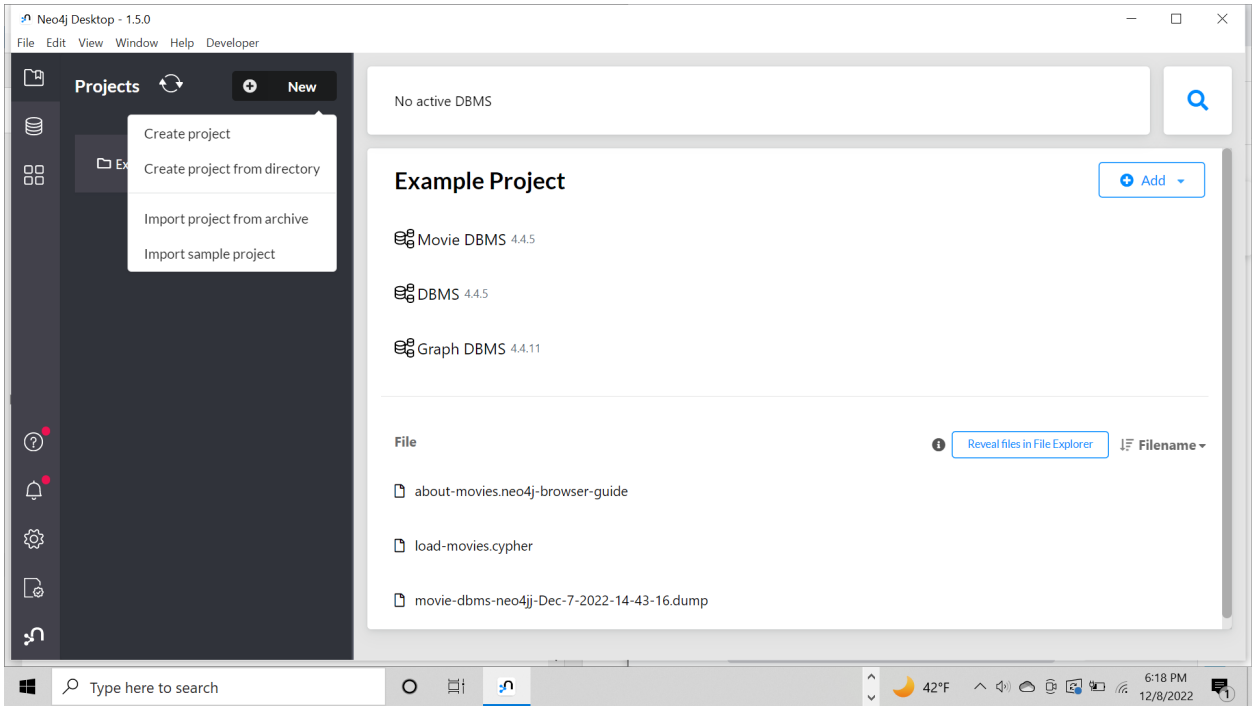
2) Run Data cleansing script

3) Once script is finished running, "wrangled_dataset.csv" file will be created

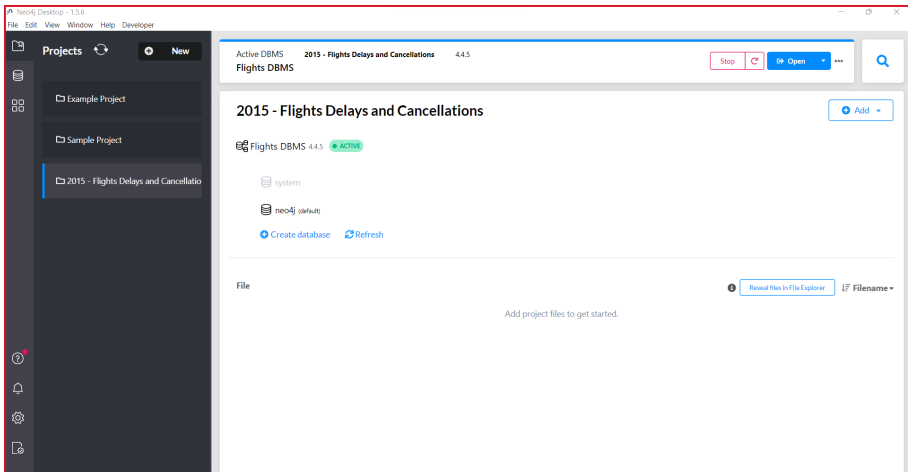
4) Use this new filtered dataset for database/visualization

## Neo4j Instructions

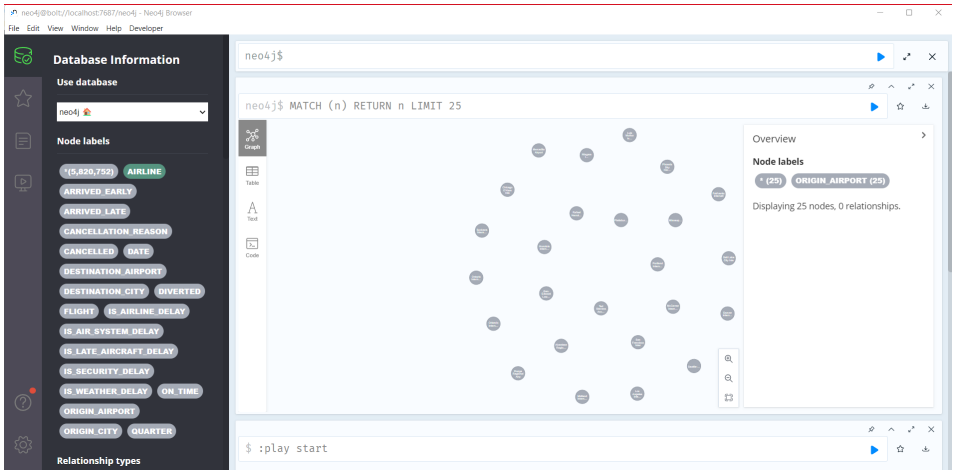
Graph Database End-User Instructions (Database creation and load):

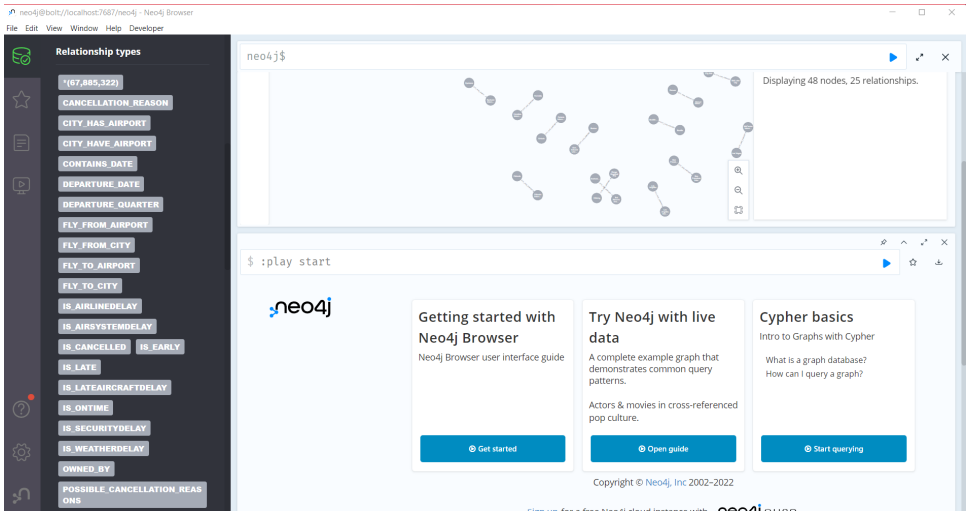



- Create a new database for project in Neo4j
- Copy the CSV file into the dbms, import folder of the database created




- Click on Start, to start the database and click on open to open the database, here you can see nodes and relationships are created




## Constraint Creation

```
// Uniqueness constraints.
CREATE CONSTRAINT ON (airline:AIRLINE) ASSERT airline.iataCode IS UNIQUE;
CREATE CONSTRAINT ON (flight:FLIGHT) ASSERT flight.FLIGHTID IS UNIQUE;
CREATE CONSTRAINT ON (originAirport:ORIGIN_AIRPORT) ASSERT originAirport.originAirport IS UNIQUE;
CREATE CONSTRAINT ON (originCity:ORIGIN_CITY) ASSERT originCity.originCity IS UNIQUE;
CREATE CONSTRAINT ON (destinationAirport:DESTINATION_AIRPORT) ASSERT destinationAirport.destinationAirport IS UNIQUE;
CREATE CONSTRAINT ON (destinationCity:DESTINATION_CITY) ASSERT destinationCity.destinationCity IS UNIQUE;
CREATE CONSTRAINT ON (date:DATE) ASSERT date.date IS UNIQUE;
CREATE CONSTRAINT ON (quarter:QUARTER) ASSERT quarter.quarter IS UNIQUE;
CREATE CONSTRAINT ON (onTime:ON_TIME) ASSERT onTime.isOnTime IS UNIQUE;
CREATE CONSTRAINT ON (arrivedEarly:ARRIVED_EARLY) ASSERT arrivedEarly.isArrivedEarly IS UNIQUE;
CREATE CONSTRAINT ON (arrivedLate:ARRIVED_LATE) ASSERT arrivedLate.isArrivedLate IS UNIQUE;
CREATE CONSTRAINT ON (isAirSystemDelay:IS_AIR_SYSTEM_DELAY) ASSERT isAirSystemDelay.isAirSystemDelay IS UNIQUE;
CREATE CONSTRAINT ON (isSecurityDelay:IS_SECURITY_DELAY) ASSERT isSecurityDelay.isSecurityDelay IS UNIQUE;
CREATE CONSTRAINT ON (isLateAircraftDelay:IS_LATE_AIRCRAFT_DELAY) ASSERT isLateAircraftDelay.isLateAircraftDelay IS UNIQUE;
CREATE CONSTRAINT ON (isWeatherDelay:IS_WEATHER_DELAY) ASSERT isWeatherDelay.isWeatherDelay IS UNIQUE;
CREATE CONSTRAINT ON (isDiverted:DIVERTED) ASSERT isDiverted.isDiverted IS UNIQUE;
CREATE CONSTRAINT ON (isCancelled:CANCELLED) ASSERT isCancelled.isCancelled IS UNIQUE;
CREATE CONSTRAINT ON (cancellationReason:CANCELLATION_REASON) ASSERT cancellationReason.cancellationReason IS UNIQUE;
```

## Node Creation

```
// Create Airline node
:auto USING PERIODIC COMMIT 1000
LOAD CSV With HEADERS FROM 'file:///wrangled_dataset.csv' AS row
MERGE
  (airline:AIRLINE{iataCode:row.IATA_CODE})
   ON CREATE SET airline.AIRLINENAME = row.AIRLINE;

//
// Create Flight Node
:auto USING PERIODIC COMMIT 1000
LOAD CSV With HEADERS FROM 'file:///wrangled_dataset.csv' AS row
MERGE
  (flight:FLIGHT{FLIGHTID:row.FLIGHT})
      ON CREATE SET flight.flight_number=row.FLIGHT_NUMBER,
                                        flight.TAIL_NUMBER=row.TAIL_NUMBER,
                                        flight.SCHEDULED_DEPARTURE=row.SCHEDULED_DEPARTURE,
                                        flight.DEPARTURE_TIME=row.DEPARTURE_TIME,
                                        flight.DEPARTURE_DELAY=row.DEPARTURE_DELAY,
                                        flight.TAXI_OUT=row.TAXI_OUT,
                                        flight.WHEELS_OFF =row.WHEELS_OFF ,
                                        flight.SCHEDULED_TIME=row.SCHEDULED_TIME,
                                        flight.ELAPSED_TIME=row.ELAPSED_TIME,
                                        flight.AIR_TIME=row.AIR_TIME,
                                        flight.DISTANCE=row.DISTANCE,
                                        flight.WHEELS_ON=row.WHEELS_ON,
                                        flight.TAXI_IN=row.TAXI_IN,
                                        flight.SCHEDULED_ARRIVAL=row.SCHEDULED_ARRIVAL,
                                        flight.ARRIVAL_TIME=row.ARRIVAL_TIME,
                                        flight.ARRIVAL_DELAY=row.ARRIVAL_DELAY,
                                        flight.AIR_SYSTEM_DELAY=row.AIR_SYSTEM_DELAY,
                                        flight.SECURITY_DELAY=row.SECURITY_DELAY,
                                        flight.LATE_AIRCRAFT_DELAY=row.LATE_AIRCRAFT_DELAY,
```

**Relationship Creation**

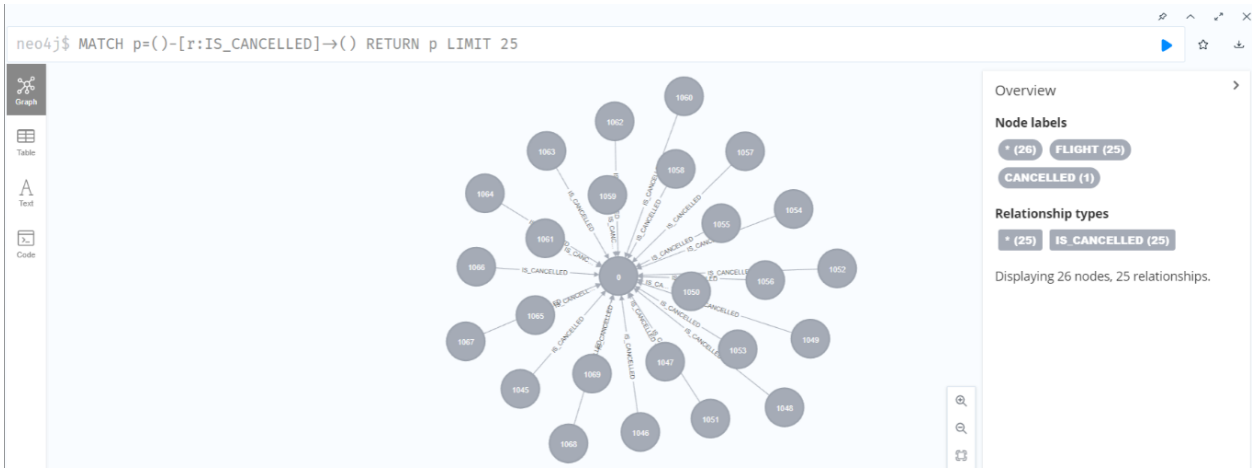
```
//      DONE
// Create Constrains  build reltionship
:auto USING PERIODIC COMMIT 1000
LOAD CSV With HEADERS FROM 'file:///wrangled_dataset.csv' AS row
     MATCH (flight:FLIGHT{FLIGHTID:row.FLIGHT})
     MATCH (quarter:QUARTER {QUARTER:row.Quarter})
     MERGE (flight)-[:DEPARTURE_QUARTER]->(quarter);

//      DONE
// Create Constrains  build reltionship
:auto USING PERIODIC COMMIT 1000
LOAD CSV With HEADERS FROM 'file:///wrangled_dataset.csv' AS row
     MATCH (flight:FLIGHT{FLIGHTID:row.FLIGHT})
     MATCH (date:DATE {DATE:row.DATE})
     MERGE (flight)-[:DEPARTURE_DATE]->(date);

// DONE
// Create Constrains  build reltionship
:auto USING PERIODIC COMMIT 1000
LOAD CSV With HEADERS FROM 'file:///wrangled_dataset.csv' AS row
     MATCH (date:DATE {DATE:row.DATE})
     MATCH (quarter:QUARTER {QUARTER:row.Quarter})
     MERGE (quarter)-[:CONTAINS_DATE]->(date);

//      DONE
// Create Constrains  build reltionship
:auto USING PERIODIC COMMIT 1000
LOAD CSV With HEADERS FROM 'file:///wrangled_dataset.csv' AS row
     MATCH (flight:FLIGHT{FLIGHTID:row.FLIGHT})
     MATCH (originCity:ORIGIN_CITY {ORIGIN_CITY:row.ORIGIN_CITY})
     MERGE (flight)-[:FLY_FROM_CITY]->(originCity);
```

- Copy the cypher code of 2015 Flight Delays and Cancellations to create the constraints, nodes and relationship between them
- Run the code to generate graph
- The script will take time to run due to large dataset size
- The graph database will be created and can be tested by queries. Below is our graph model generated

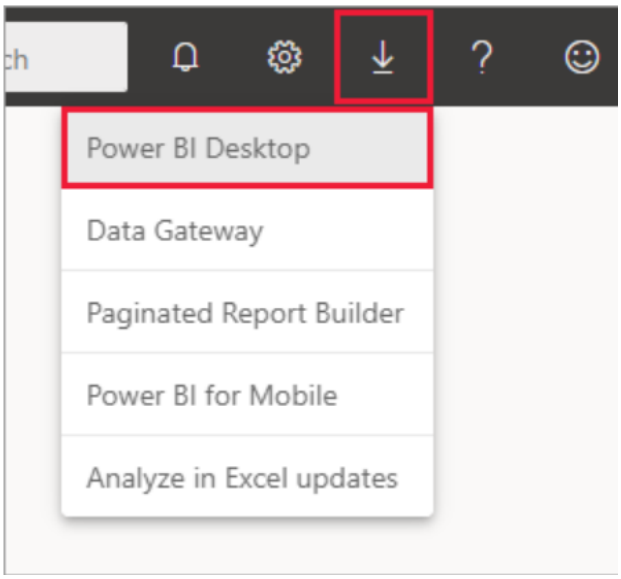



**Analystics dashboard documentation**

There are two ways to download Power BI Desktop

**Method 1:**

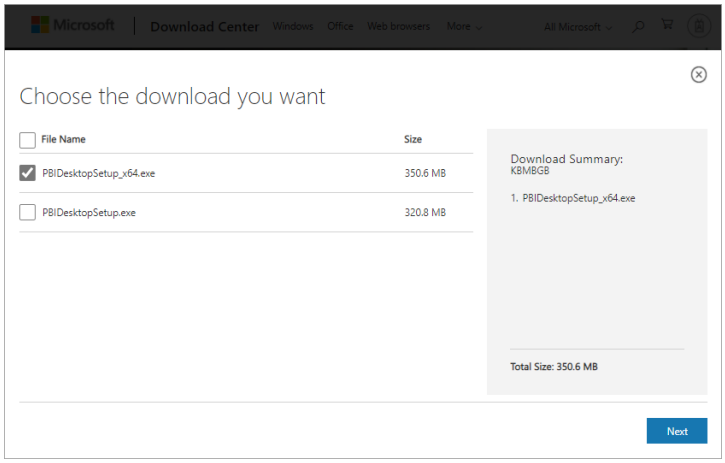
- Open a browser and go directly to the Power BI Desktop page of the Microsoft Store.
- From the Power BI service, in the upper right corner, select the Download icon and then choose Power BI Desktop.



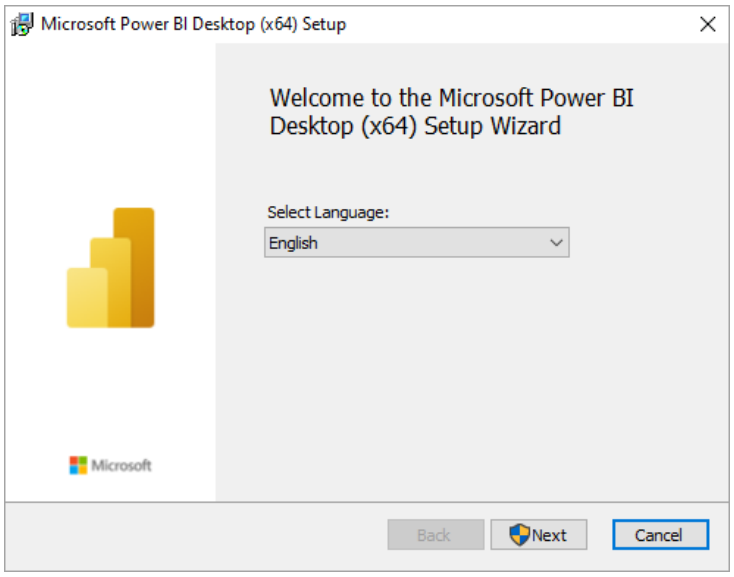

- Go to the Power BI Desktop product page, and then select Download free
- After you've landed on the Power BI Desktop page of the Microsoft Store, select install


**Method 2:**

- Download Power BI Desktop directly
- To download the Power BI Desktop executable from the Download Center, select Download from the Download Center page. Then, specify a 32-bit or 64-bit installation file to download.

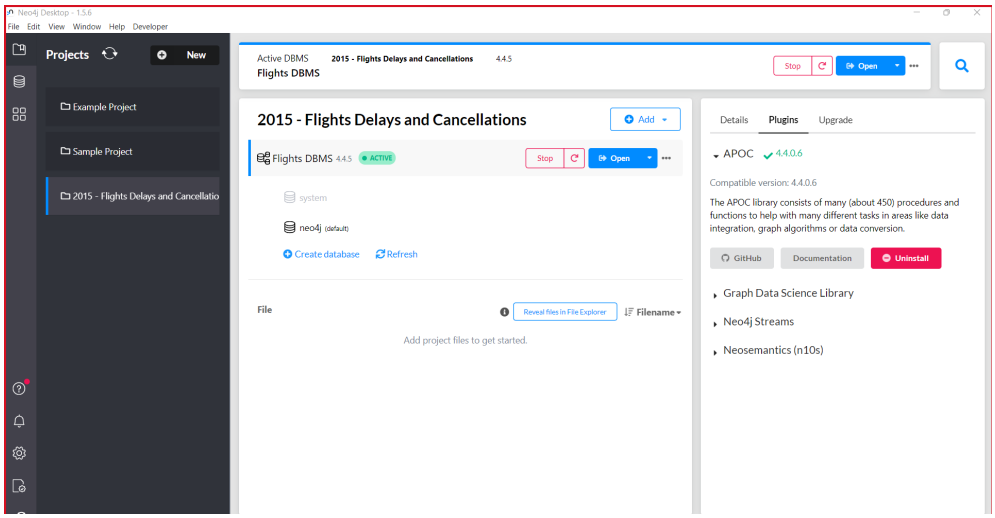



- Install Power BI Desktop after downloading it
- You're prompted to run the installation file after you've finished downloading it.
- After you launch the installation package, Power BI Desktop installs as an application and runs on your desktop.




**Connection between Neo4j and Power BI desktop**

- Download Neo4j-BI-Connector-ODBC-1.0.2-windows.zip and install ODBC driver needed for connection between Neo4j and PowerBI
- Install APOC plugin in neo4j needed for connection




- Enter username and password of database created in neo4j while connecting it with Power BI, connection is done between neo4j and Power BI

## Business Metadata

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | **Fields** | **Data Type** | **Description** | | | |
| 2 | FLIGHT | integer | Flight Id | | | |
| 3 | IATA_CODE | String | Short form of airline name | | | |
| 4 | FLIGHT_NUMBER | integer | Flight Identifier | | | |
| 5 | TAIL_NUMBER | String | Aircraft Identifier | | | |
| 6 | SCHEDULED_DEPARTURE | integer | Planned Departure Time | | | |
| 7 | DEPARTURE_TIME | Float | WHEEL_OFF - TAXI_OUT | | | |
| 8 | DEPARTURE_DELAY | Float | Total Delay on Departure | | | |
| 9 | TAXI_OUT | Float | The time duration elapsed between departure from the origin airport gate and wheels off | | | |
| 10 | WHEELS_OFF | Float | The time point that the aircraft's wheels leave the ground | | | |
| 11 | SCHEDULED_TIME | Float | Planned time amount needed for the flight trip | | | |
| 12 | ELAPSED_TIME | Float | AIR_TIME + TAXI_IN + TAXI_OUT | | | |
| 13 | AIR_TIME | Float | The time duration between wheels_off and wheels_on time | | | |
| 14 | DISTANCE | integer | Distance between two airports | | | |
| 15 | WHEELS_ON | Float | The time point that the aircraft's wheels touch on the ground | | | |
| 16 | TAXI_IN | Float | The time duration elapsed between wheels-on and gate arrival at the destination airport | | | |
| 17 | SCHEDULED_ARRIVAL | integer | Planned arrival time | | | |
| 18 | ARRIVAL_TIME | Float | WHEELS_ON + TAXI_IN | | | |
| 19 | ARRIVAL_DELAY | Float | ARRIVAL_TIME - SCHEDULED_ARRIVAL | | | |
| 20 | DIVERTED | integer | Aircraft landed on airport that out of schedule | | | |
| 21 | CANCELLED | integer | Flight Cancelled (1 = cancelled) | | | |
| 22 | CANCELLATION_REASON | String | Reason for Cancellation of flight: A - Airline/Carrier; B - Weather; C - National Air System; D - Security | | | |
| 23 | AIR_SYSTEM_DELAY | Float | Delay caused by air system | | | |

| | | | |
|---|---|---|---|
| 24 | SECURITY_DELAY | Float | Delay caused by security |
| 25 | AIRLINE_DELAY | Float | Delay caused by the airline |
| 26 | LATE_AIRCRAFT_DELAY | Float | Delay caused by aircraft |
| 27 | WEATHER_DELAY | Float | Delay caused by weather |
| 28 | AIRLINE | String | Airline Identifier |
| 29 | ORIGIN_AIRPORT | String | Starting Airport |
| 30 | ORIGIN_CITY | String | Starting City |
| 31 | DESTINATION_AIRPORT | String | Destination Airport |
| 32 | DESTINATION_CITY | String | Destination City |
| 33 | DATE | Datetime | Date |

## Testing and Validation

### System Integration Testing and User Acceptance Testing

### Data Wrangling/Cleansing

Test 1: Checking Number of Null Values

Number of null values accessed before and after data wrangling. Null values have not been removed, but have been modified for use in data analysis.

Before handling null values

```
IATA_CODE                  0
FLIGHT_NUMBER              0
TAIL_NUMBER            14721
SCHEDULED_DEPARTURE        0
DEPARTURE_TIME         86153
DEPARTURE_DELAY        86153
TAXI_OUT               89047
WHEELS_OFF             89047
SCHEDULED_TIME             6
ELAPSED_TIME          105071
AIR_TIME              105071
DISTANCE                   0
WHEELS_ON              92513
TAXI_IN                92513
SCHEDULED_ARRIVAL          0
ARRIVAL_TIME           92513
ARRIVAL_DELAY         105071
DIVERTED                   0
CANCELLED                  0
CANCELLATION_REASON   5729195
AIR_SYSTEM_DELAY      4755640
SECURITY_DELAY        4755640
AIRLINE_DELAY         4755640
LATE_AIRCRAFT_DELAY   4755640
WEATHER_DELAY         4755640
AIRLINE                    0
ORIGIN_AIRPORT        486165
ORIGIN_CITY           486165
DESTINATION_AIRPORT   486165
DESTINATION_CITY      486165
DATE                       0
Quarter                    0
dtype: int64
```

After handling null values

```
IATA_CODE              0
FLIGHT_NUMBER          0
TAIL_NUMBER            0
SCHEDULED_DEPARTURE    0
DEPARTURE_TIME         0
DEPARTURE_DELAY        0
TAXI_OUT               0
WHEELS_OFF             0
SCHEDULED_TIME         0
ELAPSED_TIME           0
AIR_TIME               0
DISTANCE               0
WHEELS_ON              0
TAXI_IN                0
SCHEDULED_ARRIVAL      0
ARRIVAL_TIME           0
```

Test 2: Replacing null values

These metrics were used in our data analysis and we wanted to account for Nan values instead of just removing them from the dataset.

In [16]:
```
#replace null values with 'NaN' for all elements

flights_data1['TAIL_NUMBER'] = flights_data1['TAIL_NUMBER'].fillna('NaN')
flights_data1['TAXI_OUT'] = flights_data1['TAXI_OUT'].fillna('NaN')
flights_data1['WHEELS_OFF'] = flights_data1['WHEELS_OFF'].fillna('NaN')
flights_data1['SCHEDULED_TIME'] = flights_data1['SCHEDULED_TIME'].fillna('NaN')
flights_data1['DEPARTURE_TIME'] = flights_data1['DEPARTURE_TIME'].fillna('NaN')
flights_data1['TAXI_OUT'] = flights_data1['TAXI_OUT'].fillna('NaN')
flights_data1['WHEELS_OFF'] = flights_data1['WHEELS_OFF'].fillna('NaN')
flights_data1['AIRLINE_DELAY'] = flights_data1['AIRLINE_DELAY'].fillna('NaN')
flights_data1['AIR_SYSTEM_DELAY'] = flights_data1['AIR_SYSTEM_DELAY'].fillna('NaN')
flights_data1['SECURITY_DELAY'] = flights_data1['SECURITY_DELAY'].fillna('NaN')
flights_data1['LATE_AIRCRAFT_DELAY'] = flights_data1['LATE_AIRCRAFT_DELAY'].fillna('NaN')
flights_data1['WEATHER_DELAY'] = flights_data1['WEATHER_DELAY'].fillna('NaN')
flights_data1['ELAPSED_TIME'] = flights_data1['ELAPSED_TIME'].fillna('NaN')
flights_data1['WHEELS_ON'] = flights_data1['WHEELS_ON'].fillna('NaN')
flights_data1['TAXI_IN'] = flights_data1['TAXI_IN'].fillna('NaN')
flights_data1['ARRIVAL_TIME'] = flights_data1['ARRIVAL_TIME'].fillna('NaN')
flights_data1['DEPARTURE_DELAY'] = flights_data1['DEPARTURE_DELAY'].fillna('NaN')
flights_data1['AIR_TIME'] = flights_data1['AIR_TIME'].fillna('NaN')
flights_data1['CANCELLATION_REASON'].fillna('Unknown', inplace=True)
```

Test 3: Dropping Duplicates

No duplicates were found in our dataset. The number of rows and columns were unchanged after checking it with duplicate dropfunction.

Below is the screenshot of the count before performing duplicate function.

```
In [12]: flights_df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 5819079 entries, 0 to 5819078
Data columns (total 32 columns):
```

Below is the screenshot of the count after performing duplicate function.

```
In [15]: '''Drop duplicated rows'''

flights_data1 = flights_df.drop_duplicates()
print(flights_data1.shape)
flights_data1.head(2)
```

```
(5819079, 32)
```

Out[15]:

| | IATA_CODE | FLIGHT_NUMBER | TAIL_NUMBER | SCHEDULED_DEPARTURE | DEPARTURE_TIME | DEPARTURE_DELAY | TAXI_OUT |
|---|---|---|---|---|---|---|---|
| **0** | AS | 98 | N407AS | 5 | 2354.0 | -11.0 | 21.0 |
| **1** | AA | 2336 | N3KUAA | 10 | 2.0 | -8.0 | 12.0 |

2 rows × 32 columns

Test 4: Creating new columns for categorising flights as per arrival status and for categoring delayed flights

***Creating new columns for categorising flights as per arrival status*** ¶

```
In [22]: flights_data1["ON_TIME"]=flights_data1["ARRIVAL_DELAY"].apply(lambda row: 1 if row==0 else " ")
flights_data1["ARRIVED_EARLY"]=flights_data1["ARRIVAL_DELAY"].apply(lambda row: 1 if row<0 else " ")
flights_data1["ARRIVED_LATE"]=flights_data1["ARRIVAL_DELAY"].apply(lambda row: 1 if row>0 else " ")
```

***Creating new columns for categorising delayed flights***

```
In [24]: flights_data1["IS_AIR_SYSTEM_DELAY"] = flights_data1["AIR_SYSTEM_DELAY"].apply(lambda row:' ' if row == 0 or row == 'NaN' else
flights_data1["IS_SECURITY_DELAY"] = flights_data1["SECURITY_DELAY"].apply(lambda row:' ' if row == 0 or row == 'NaN' else 1)
flights_data1["IS_AIRLINE_DELAY"] = flights_data1["AIRLINE_DELAY"].apply(lambda row:' ' if row == 0 or row == 'NaN' else 1)
flights_data1["IS_LATE_AIRCRAFT_DELAY"] = flights_data1["LATE_AIRCRAFT_DELAY"].apply(lambda row:' ' if row == 0 or row == 'NaN
flights_data1["IS_WEATHER_DELAY"] = flights_data1["WEATHER_DELAY"].apply(lambda row:' ' if row == 0 or row == 'NaN' else 1)
```

Adding date column using the year, month and day columns

```
'''Adding date column using the year, month and day columns'''

flights_df['DATE'] = pd.to_datetime(flights_df[['YEAR','MONTH', 'DAY']])
```

Adding new column Quarter for data visualization purpose

```
'''Adding new column Quarter for data visualization purpose'''

flights_df['Quarter'] = pd.PeriodIndex(flights_df.DATE, freq='Q')
```

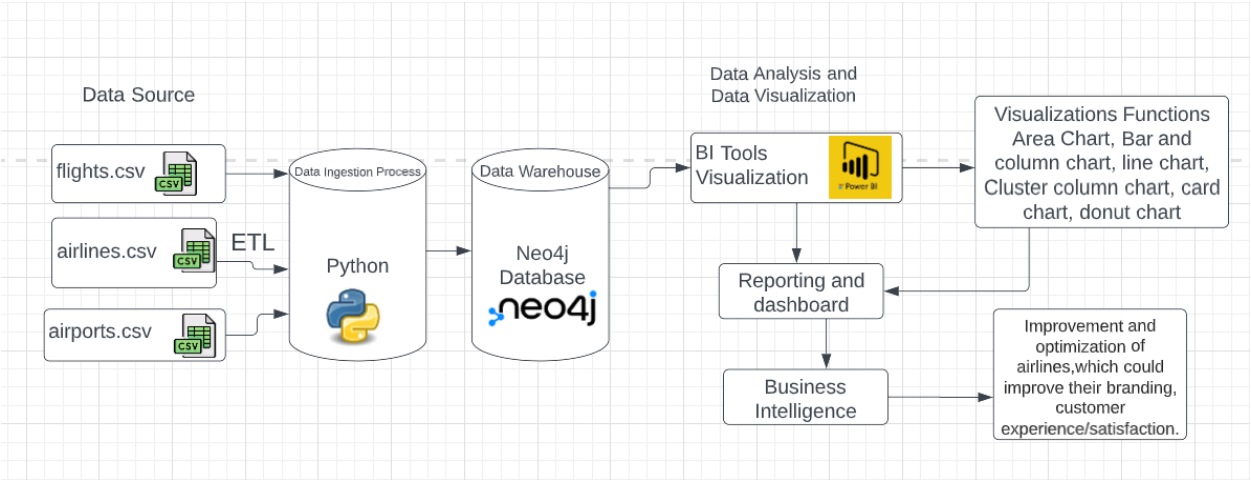
**Describe challenges encountered and how you resolved them?**

| Challenges | Solution |
|---|---|
| % of null values was greater than 80% in some of the important columns | Analyzed the impact of null values during data wrangling and rectified most of the data by replacing it with appropriate values |
| Faced performance issues while loading the data due to volume | Uploaded data in batches to avoid bottleneck |
| Few columns had complex data in the dataset for usability purposes | Analyzed the columns and created new measures to make better visualization |

**Risks/Issues of project**

| Risk/Issue | Description | Mitigation |
|---|---|---|
| Risk | Many columns has missing details which might be an analytical risk leading to biased visualization | Deriving new columns by understanding existing entities and attributes to generate meaningful reports |
| Risk | Amount of time was high to deploy the dataset in the system which led to database issues | Creating backup files in all the systems to avoid any data deletion. |
| Issue | Merging data from different data sources(Excel) | Performed multiple steps of data merging and multiple data validations to avoid data conflict. |

## Vision/Architectural Diagram




## Who would use this Dashboard and how would they benefit from it?

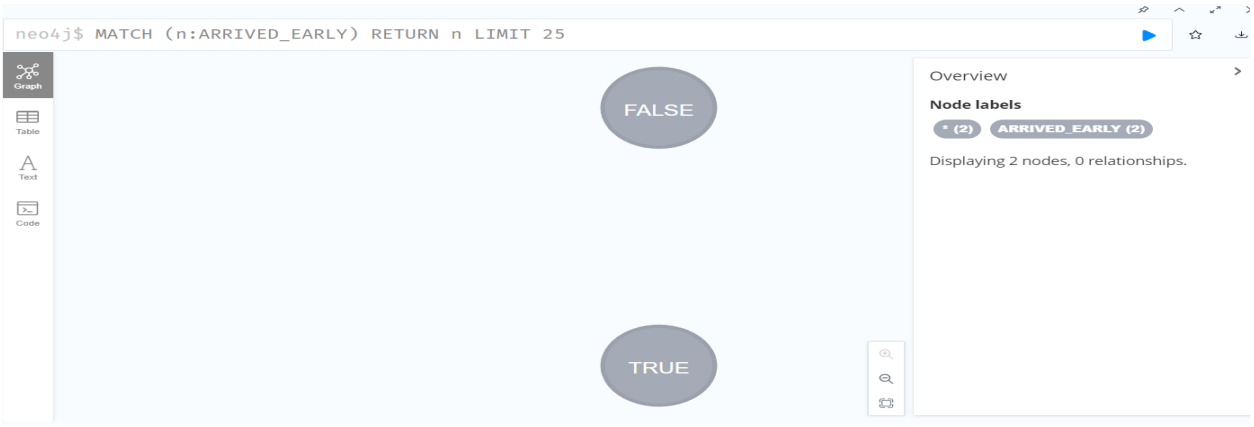
End-Users : Travel agents can use this to monitor the flight delays and book the tickets for customers accordingly. Especially when an important trip is planned with a tight schedule the bookings can be done with the flight details and the weather conditions.

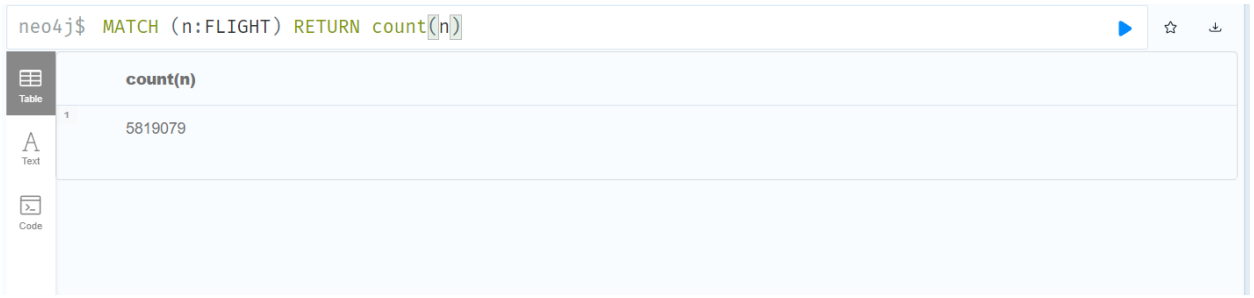
Airlines: Airlines can use the dashboard to understand the metrics and improve their branding.

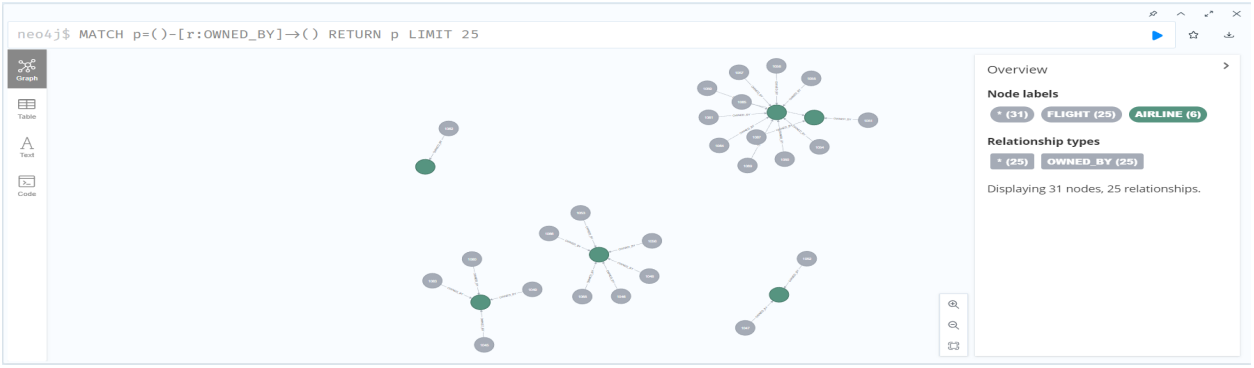
## Neo4j Testing screenshots using Cypher Queries

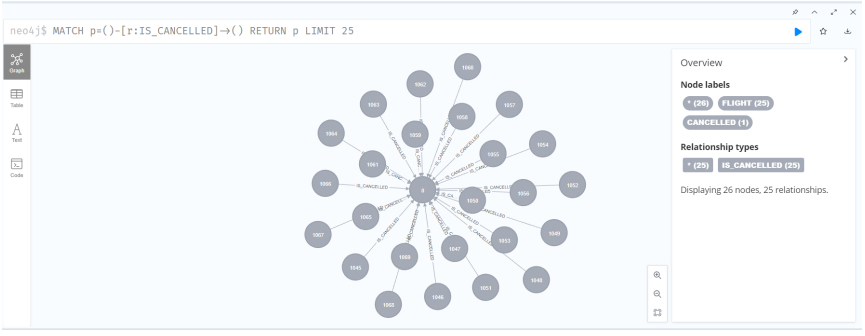
Steps for testing and verification of data using Neo4j:

- Copy the cypher code(neo4j_testing_script.txt) containing the test queries
- Run the code to view graphs and tables verifying data load to Neo4j
- The scripts will run one by one and will take a few minutes displaying the below results

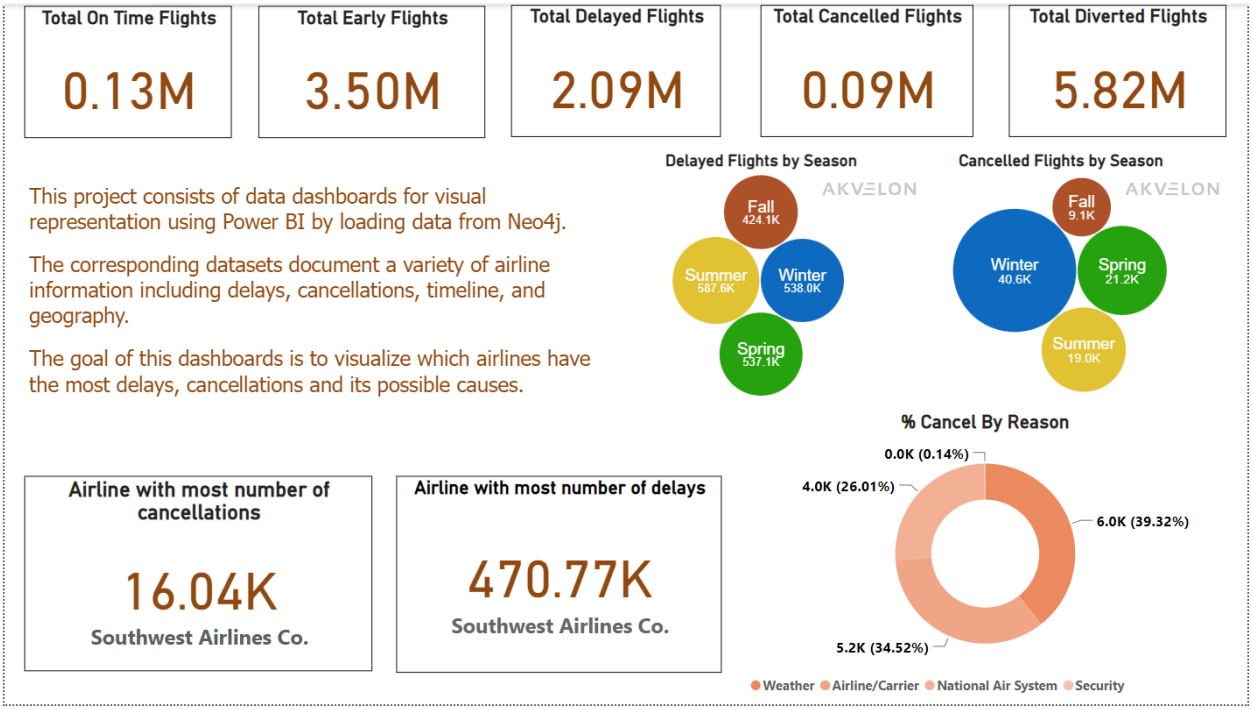

neo4j$ MATCH (n:ARRIVED_EARLY) RETURN n LIMIT 25
Graph
Table
Text
Code
FALSE
TRUE
Overview
Node labels
* (2)
ARRIVED_EARLY (2)
Displaying 2 nodes, 0 relationships.

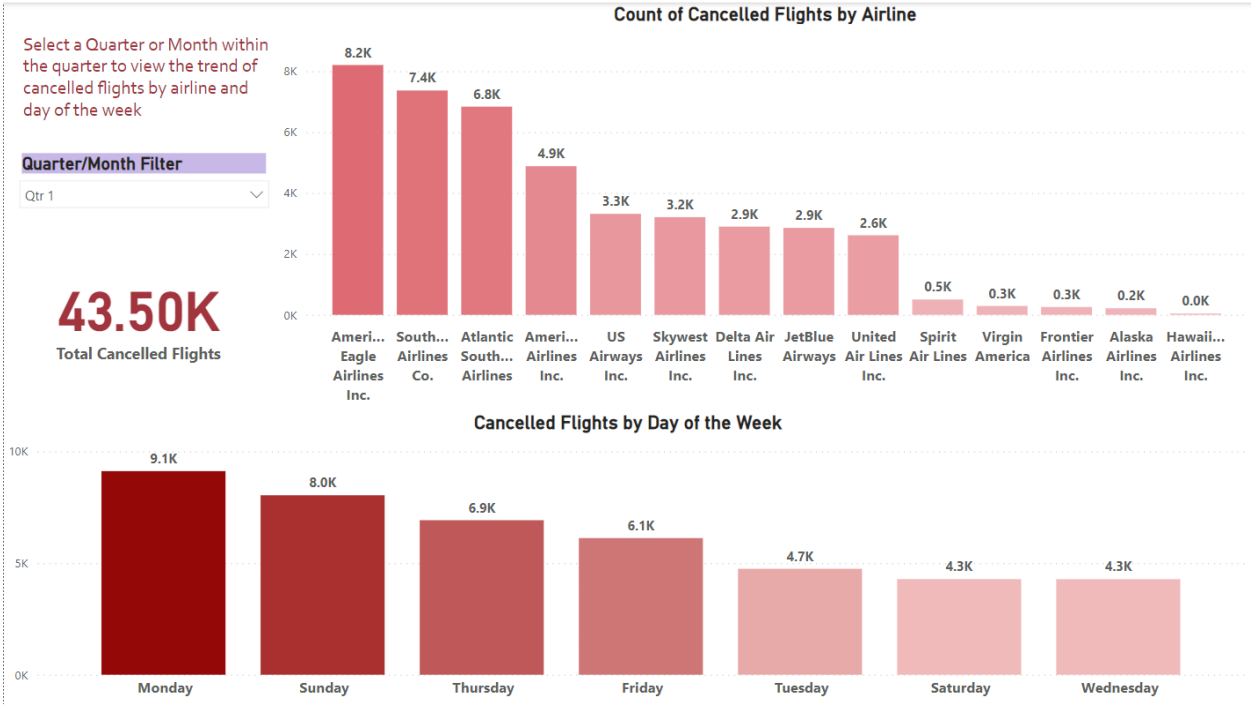

neo4j$ MATCH (n:FLIGHT) RETURN count(n)
Table
Text
Code
count(n)
1
5819079


neo4j$ MATCH p=()-[r:OWNED_BY]→() RETURN p LIMIT 25
Graph
Table
Text
Code
Overview
Node labels
* (31)
FLIGHT (25)
AIRLINE (6)
Relationship types
* (25)
OWNED_BY (25)
Displaying 31 nodes, 25 relationships.


neo4j$ MATCH p=()-[r:IS_CANCELLED]→() RETURN p LIMIT 25
Graph
Table
Text
Code
Overview
Node labels
* (26)
FLIGHT (25)
CANCELLED (1)
Relationship types
* (25)
IS_CANCELLED (25)
Displaying 26 nodes, 25 relationships.

# Dashboard Interpretation/Visualizations



Select a season within the quarter to view the trend of cancelled flights by airline and day of the week

Season Filter

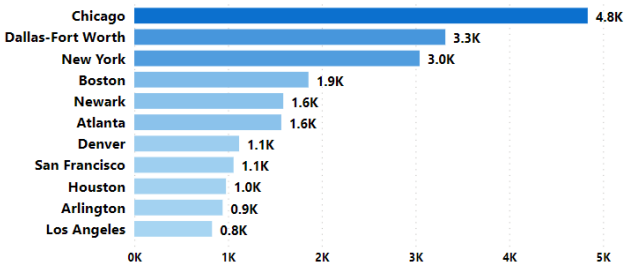
Winter

**40.56K**

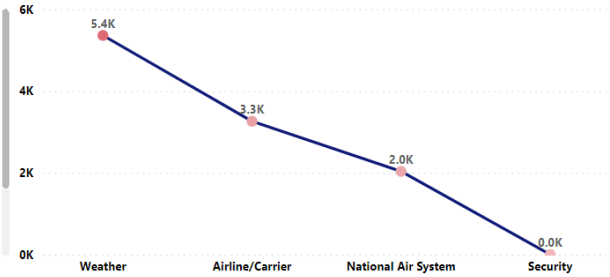
Total Cancelled Flights

## Count of Cancelled Flights by Airline


## Cancelled Flights per Top 15 Destination City




## Cancellation Count By Cancellation Reason




Quarter/Month Filter

Qtr 1

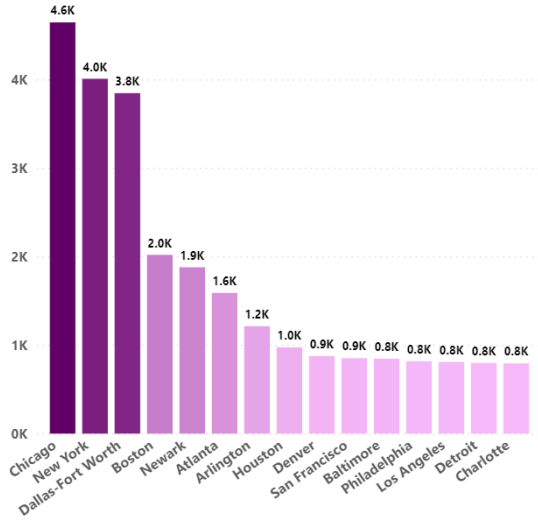
**8183**
Airline/Carrier Cancellations

**5980**
National Air System Cancell...

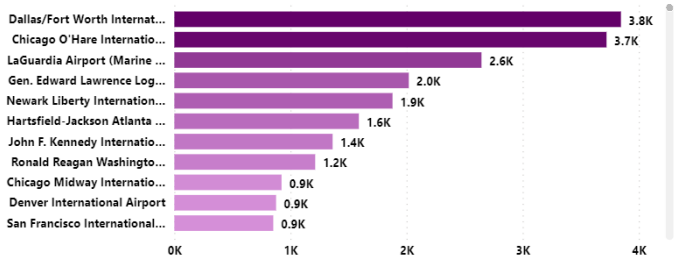
**29.33K**
Weather Cancellations

**7**
Security Cancellations

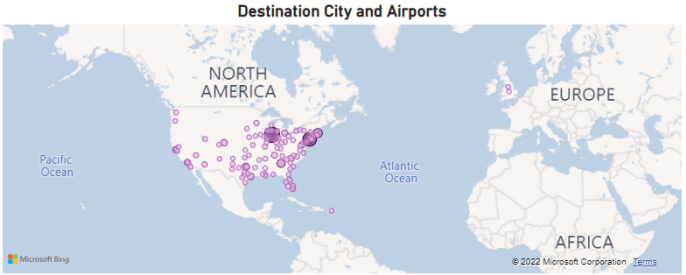
## Top 15 Destination City by Cancellations

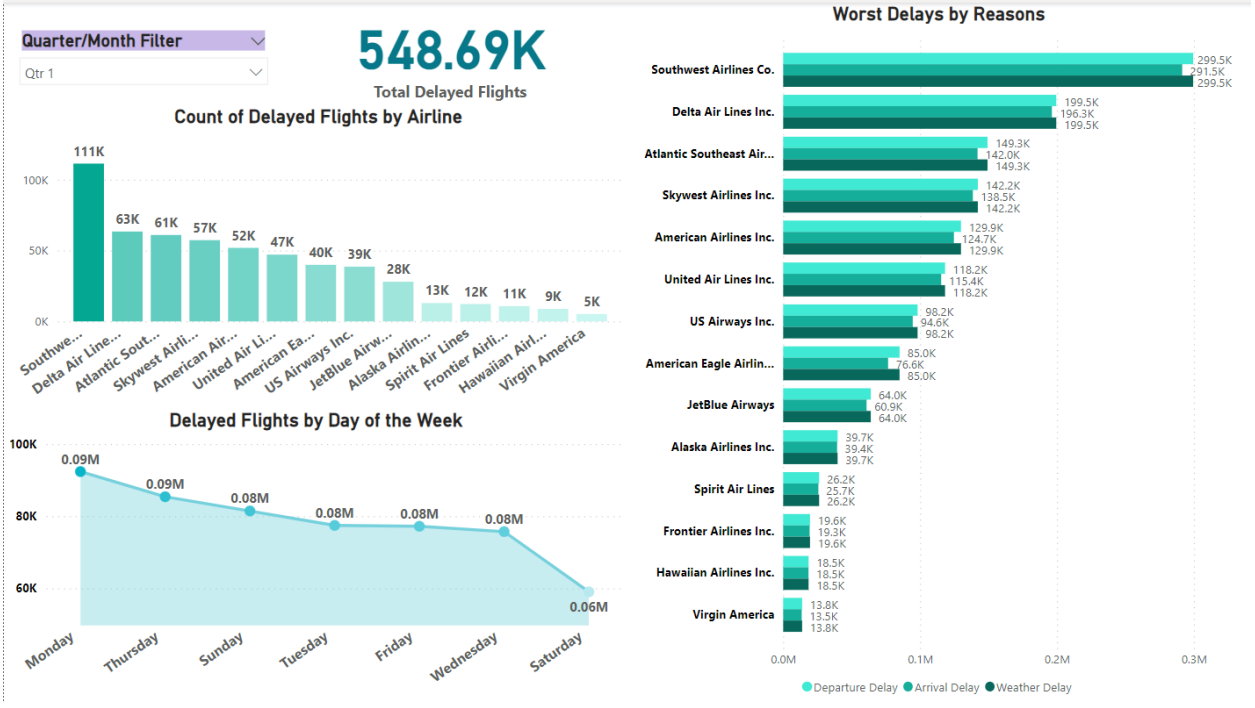


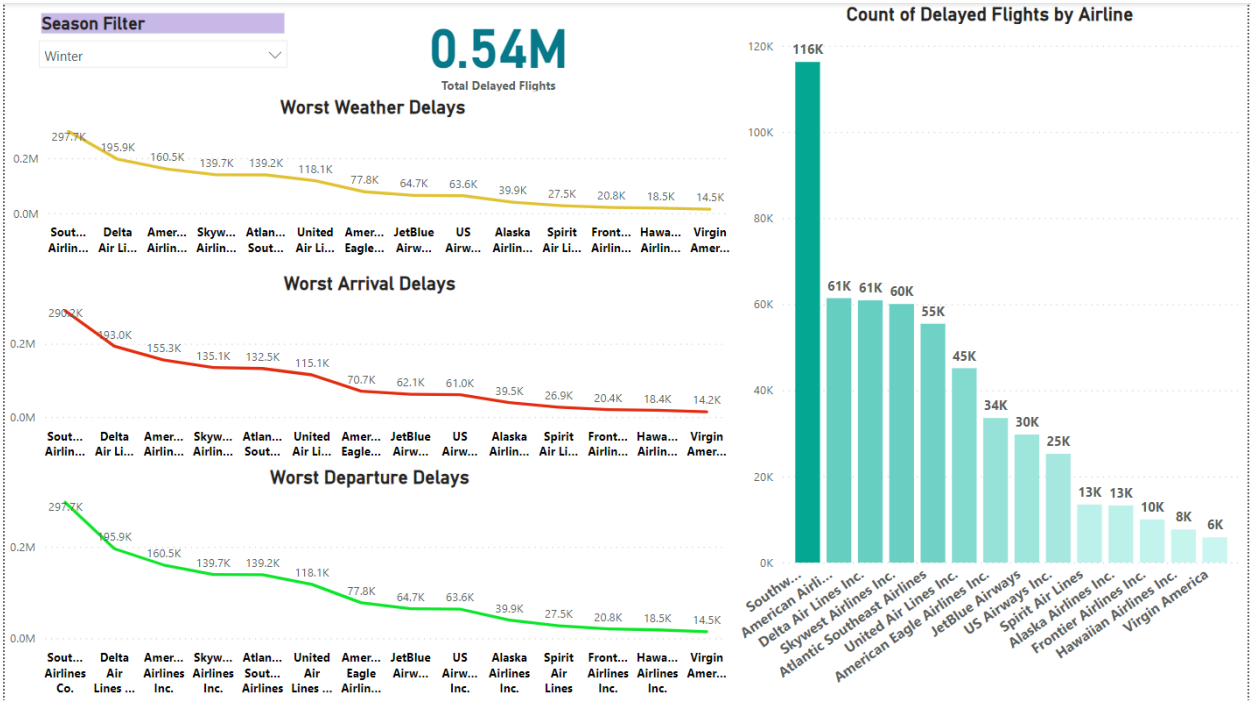

## Cancellations by Destination Airport




## Destination City and Airports



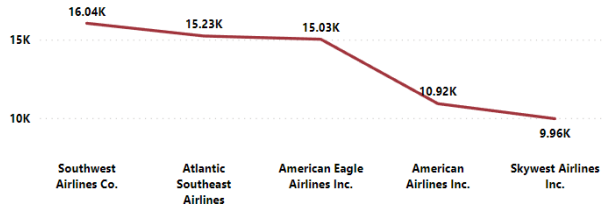

Quarter/Month Filter
Qtr 1
548.69K
Total Delayed Flights
Count of Delayed Flights by Airline
111K
63K
61K
57K
52K
47K
40K
39K
28K
13K
12K
11K
9K
5K
Delayed Flights by Day of the Week
0.09M
0.09M
0.08M
0.08M
0.08M
0.08M
0.06M
Monday
Thursday
Sunday
Tuesday
Friday
Wednesday
Saturday
Worst Delays by Reasons
Southwest Airlines Co.
Delta Air Lines Inc.
Atlantic Southeast Air...
Skywest Airlines Inc.
American Airlines Inc.
United Air Lines Inc.
US Airways Inc.
American Eagle Airlin...
JetBlue Airways
Alaska Airlines Inc.
Spirit Air Lines
Frontier Airlines Inc.
Hawaiian Airlines Inc.
Virgin America
Departure Delay
Arrival Delay
Weather Delay


Season Filter
Winter
0.54M
Total Delayed Flights
Worst Weather Delays
Worst Arrival Delays
Worst Departure Delays
Count of Delayed Flights by Airline
116K
61K
61K
60K
55K
45K
34K
30K
25K
13K
13K
10K
8K
6K

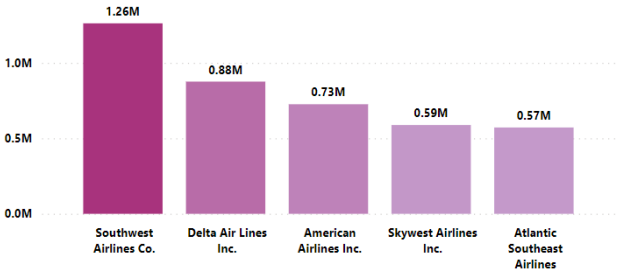
**Southwest Airlines Co.**

- Most Cancelled Flights
- Most Delayed Flights
- Most Diverted Flights

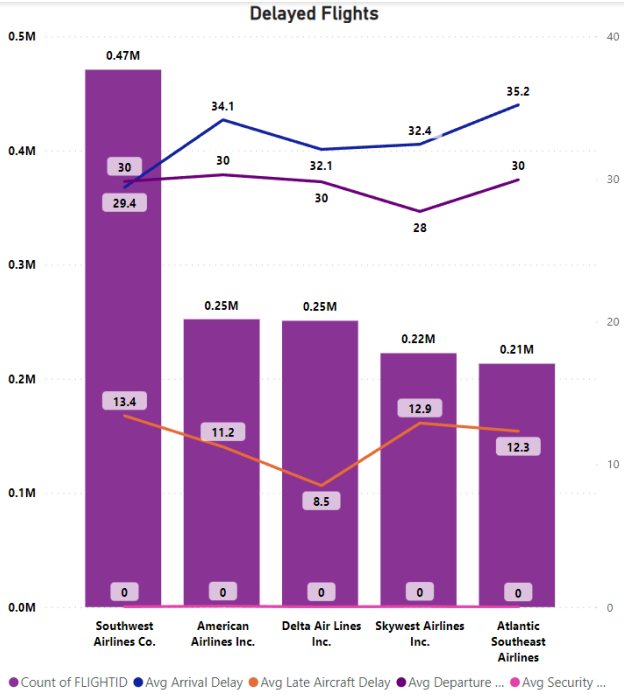
## Cancelled Flights




## Diverted Flights




## Delayed Flights

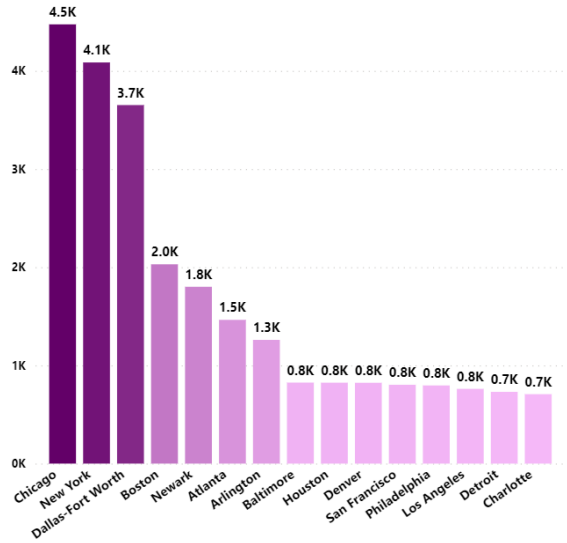



Quarter/Month Filter

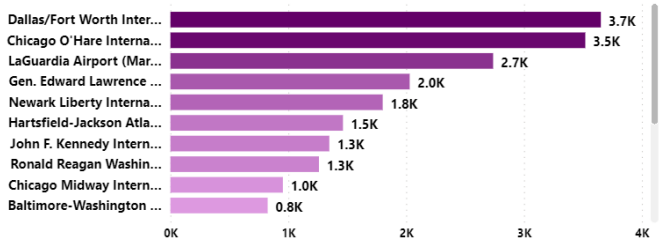
Qtr 1

| 8183 | 5980 | 29.33K | 7 |
|---|---|---|---|
| Airline/Carrier Cancellations | National Air System Cancell... | Weather Cancellations | Security Cancellations |

## Top 15 Origin City by Cancellations

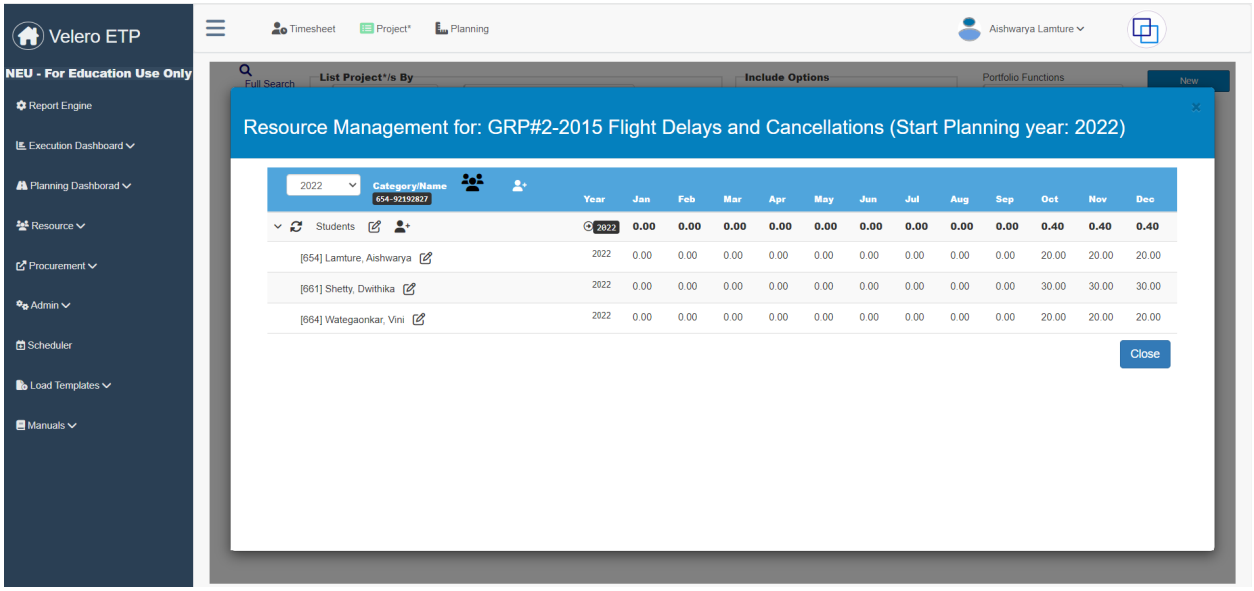


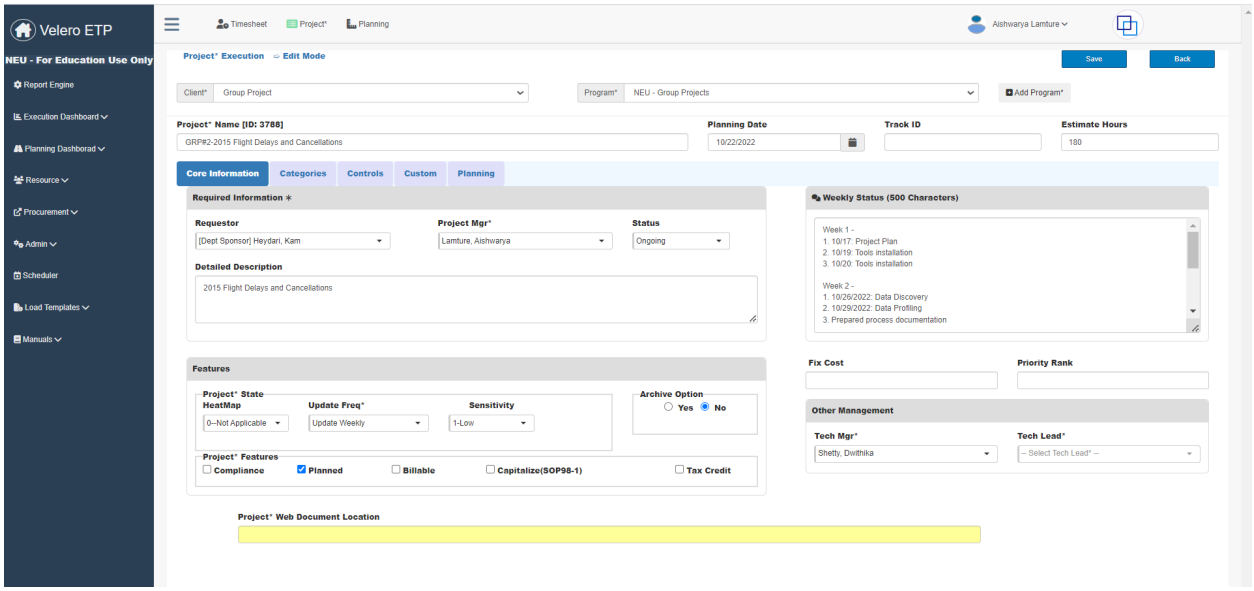

## Top 15 Origin Airport by Cancellations

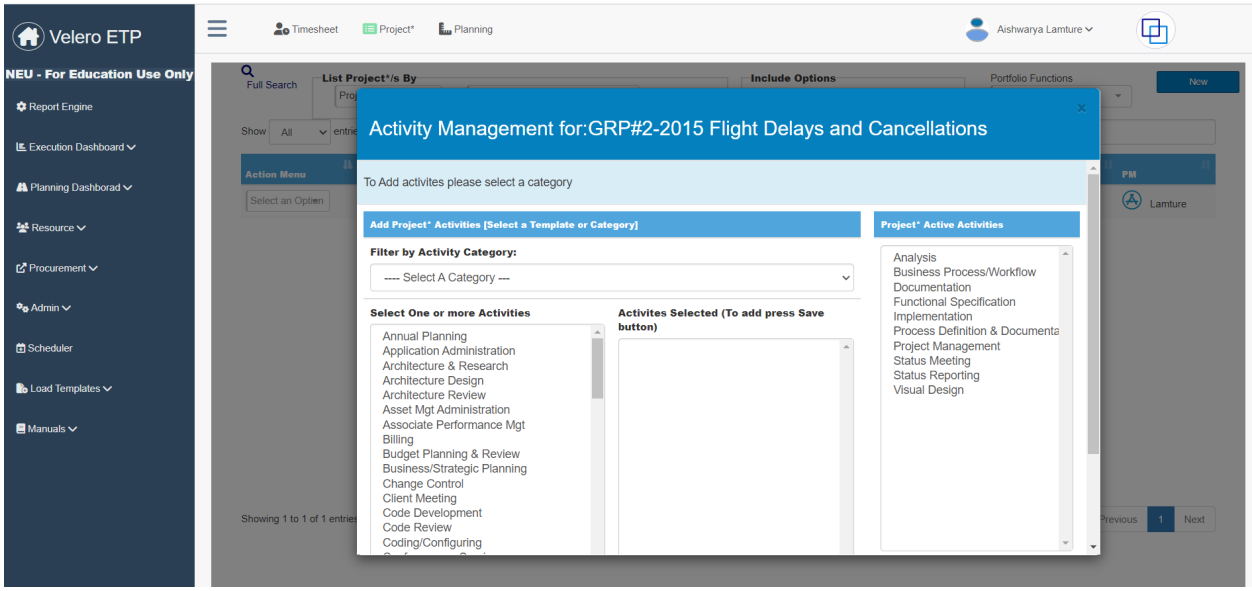



## Origin City and Airports

# Velero

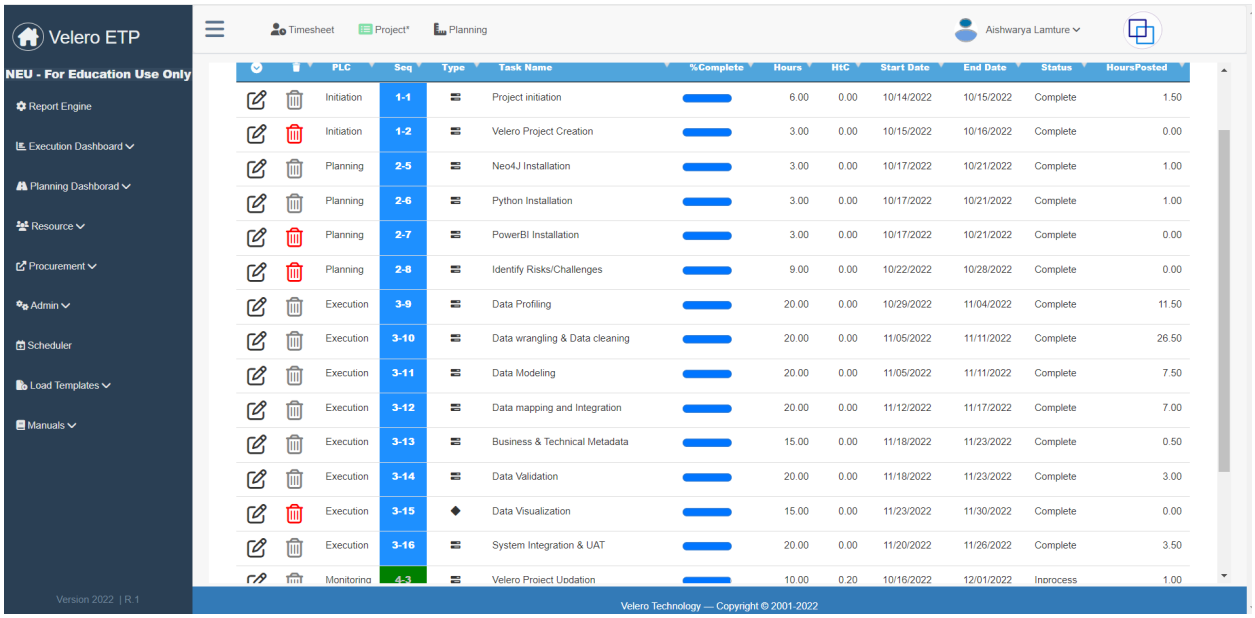

Velero ETP
NEU - For Education Use Only
Report Engine
Execution Dashboard
Planning Dashborad
Resource
Procurement
Admin
Scheduler
Load Templates
Manuals
Timesheet
Project*
Planning
Aishwarya Lamture
Full Search
List Project*/s By
Include Options
Portfolio Functions
New
Resource Management for: GRP#2-2015 Flight Delays and Cancellations (Start Planning year: 2022)
2022
Category/Name
654-92192827
Year Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Students 2022 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.40 0.40 0.40
[654] Lamture, Aishwarya 2022 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 20.00 20.00 20.00
[661] Shetty, Dwithika 2022 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 30.00 30.00 30.00
[664] Wategaonkar, Vini 2022 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 20.00 20.00 20.00
Close

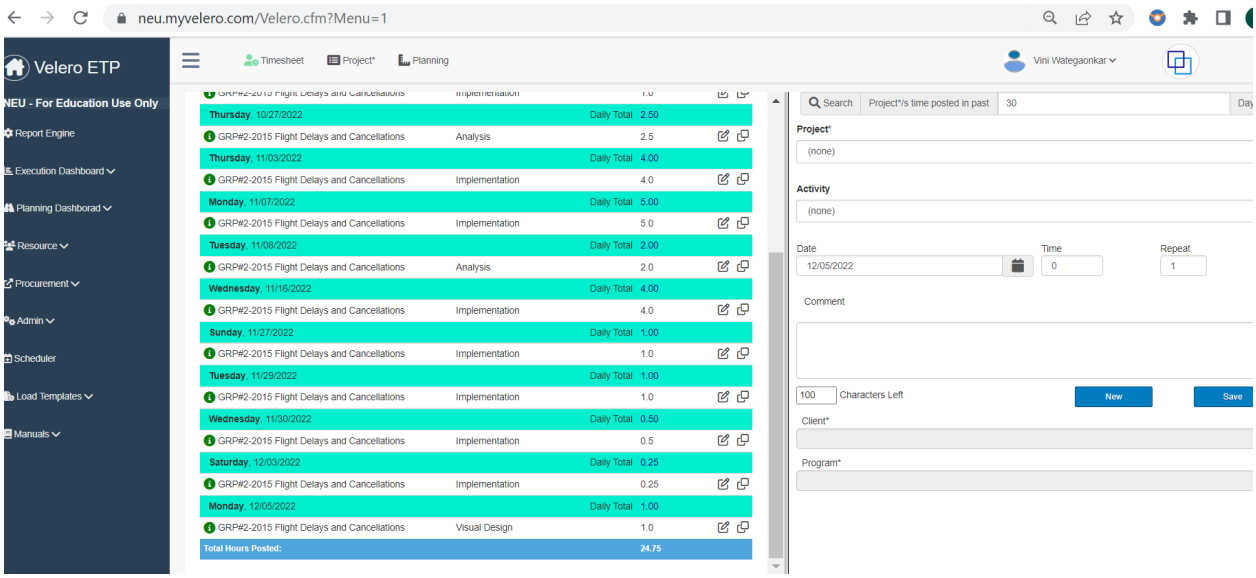

Velero ETP
NEU - For Education Use Only
Timesheet
Project*
Planning
Aishwarya Lamture
Project* Execution
Edit Mode
Save
Back
Client*
Group Project
Program*
NEU - Group Projects
Add Program*
Project* Name [ID: 3788]
GRP#2-2015 Flight Delays and Cancellations
Planning Date
10/22/2022
Track ID
Estimate Hours
180
Core Information
Categories
Controls
Custom
Planning
Required Information
Requestor
[Dept Sponsor] Heydari, Kam
Project Mgr*
Lamture, Aishwarya
Status
Ongoing
Detailed Description
2015 Flight Delays and Cancellations
Weekly Status (500 Characters)
Week 1 -
1. 10/17: Project Plan
2. 10/19: Tools installation
3. 10/20: Tools installation
Week 2 -
1. 10/26/2022: Data Discovery
2. 10/29/2022: Data Profiling
3. Prepared process documentation
Features
Project* State HeatMap
0--Not Applicable
Update Freq*
Update Weekly
Sensitivity
1-Low
Archive Option
Yes
No
Project* Features
Compliance
Planned
Billable
Capitalize(SOP98-1)
Tax Credit
Fix Cost
Priority Rank
Other Management
Tech Mgr*
Shetty, Dwithika
Tech Lead*
-- Select Tech Lead* --
Project* Web Document Location




| Date / Project* | activity | Hours |
|---|---|---|
| **Wednesday**, 10/19/2022 | Daily Total | 2.00 |
| GRP#2-2015 Flight Delays and Cancellations | Analysis | 2.0 |
| **Thursday**, 10/20/2022 | Daily Total | 2.00 |
| GRP#2-2015 Flight Delays and Cancellations | Implementation | 2.0 |
| **Saturday**, 10/29/2022 | Daily Total | 2.00 |
| GRP#2-2015 Flight Delays and Cancellations | Analysis | 2.0 |
| **Monday**, 10/31/2022 | Daily Total | 3.00 |
| GRP#2-2015 Flight Delays and Cancellations | Implementation | 3.0 |
| **Saturday**, 11/12/2022 | Daily Total | 3.00 |
| GRP#2-2015 Flight Delays and Cancellations | Implementation | 3.0 |
| **Wednesday**, 11/16/2022 | Daily Total | 4.00 |
| GRP#2-2015 Flight Delays and Cancellations | Implementation | 4.0 |
| **Sunday**, 11/27/2022 | Daily Total | 1.00 |
| GRP#2-2015 Flight Delays and Cancellations | Implementation | 1.0 |
| **Wednesday**, 11/30/2022 | Daily Total | 2.00 |
| GRP#2-2015 Flight Delays and Cancellations | Implementation | 2.0 |
| **Thursday**, 12/01/2022 | Daily Total | 0.50 |
| GRP#2-2015 Flight Delays and Cancellations | Process Definition & Documentation | 0.5 |
| **Monday**, 12/05/2022 | Daily Total | 3.00 |
| GRP#2-2015 Flight Delays and Cancellations | Visual Design | 3.0 |
| **Wednesday**, 12/07/2022 | Daily Total | 2.00 |
| GRP#2-2015 Flight Delays and Cancellations | Status Reporting | 2.0 |
| **Total Hours Posted:** | | **24.50** |

| Date / Project* | activity | Hours |
|---|---|---|
| **Friday**, 10/21/2022 | | Daily Total 2.50 |
| GRP#2-2015 Flight Delays and Cancellations | Analysis | 2.5 |
| **Wednesday**, 10/26/2022 | | Daily Total 2.00 |
| GRP#2-2015 Flight Delays and Cancellations | Implementation | 2.0 |
| **Friday**, 10/28/2022 | | Daily Total 1.00 |
| GRP#2-2015 Flight Delays and Cancellations | Analysis | 1.0 |
| **Tuesday**, 11/01/2022 | | Daily Total 2.50 |
| GRP#2-2015 Flight Delays and Cancellations | Analysis | 2.5 |
| **Wednesday**, 11/02/2022 | | Daily Total 3.00 |
| GRP#2-2015 Flight Delays and Cancellations | Implementation | 3.0 |
| **Thursday**, 11/03/2022 | | Daily Total 1.50 |
| GRP#2-2015 Flight Delays and Cancellations | Implementation | 1.5 |
| **Friday**, 11/04/2022 | | Daily Total 0.50 |
| GRP#2-2015 Flight Delays and Cancellations | Implementation | 0.5 |
| **Monday**, 11/07/2022 | | Daily Total 5.00 |
| GRP#2-2015 Flight Delays and Cancellations | Implementation | 5.0 |
| **Thursday**, 11/10/2022 | | Daily Total 1.00 |
| GRP#2-2015 Flight Delays and Cancellations | Analysis | 1.0 |
| **Monday**, 11/28/2022 | | Daily Total 1.00 |
| GRP#2-2015 Flight Delays and Cancellations | Implementation | 1.0 |
| **Thursday**, 12/01/2022 | | Daily Total 0.50 |
| GRP#2-2015 Flight Delays and Cancellations | Implementation | 0.5 |
| **Friday**, 12/02/2022 | | Daily Total 0.25 |
| GRP#2-2015 Flight Delays and Cancellations | Process Definition & Documentation | 0.25 |
| **Friday**, 12/09/2022 | | Daily Total 2.00 |
| GRP#2-2015 Flight Delays and Cancellations | Visual Design | 2.0 |
| **Total Hours Posted:** | | **22.75** |






## Conclusion

This visualizations could potentially be useful in the improvement and optimization of airlines,which could improve their branding, customer experience/satisfaction.

**Thank You!!**