FLIGHT DELAY PREDICTION

GROUP 3 - CAPSTONE PROJECT

1. Amol Nikam H22002
2. Anindita Roy H22003
3. Anoushka Raj Rao H22004
4. Kartik Mudgal H22007
5. Manasi Ghodake H22010
6. Sidharth Khurana H22016
7. Srikrishna Srinivasan H22017

PGPDE 2022-23

INDEX

Background & Business Justification

Nowadays, the aviation industry plays a crucial role in the world's transportation sector, and a lot of businesses rely on various airlines to connect them with other parts of the world. One of the key business issues that airlines face is that the vital prices that are related to flights being delayed because of natural occurrences and operational shortcomings that is an upscale affair for the airlines, making issues in scheduling and operations for the end users therefore inflicting unhealthy name and client discontent

To solve this issue, accurately predicting these flight delays allows passengers to be well prepared for the deterrent caused to their journey and enables airlines to respond to the potential causes of the flight delays in advance to diminish the negative impact.

The purpose of this project is to look at the approaches used to build models for predicting flight delays that occur due to bad weather conditions.

In the first part of the project, we primarily focus on gathering a dataset from SQL, Cosmos db, and web API data. We will be using Azure Data factory for transformation - joins, and cleaning .In the second part of the project, we primarily focus on modeling of the data. The Machine Learning Model used for this problem statement is Decision Tree Regressor.
Azure ML Studio provides us with a No Code way of preparing our dataset and training our model.

PROBLEM STATEMENT

- The problem statement is to develop a data engineering pipeline for **Flight Delay Prediction**, which involves integrating and cleaning data from multiple sources, performing EDA, transforming the data, and building and deploying a machine learning model.
- The aim is to create a reliable and efficient system for predicting flight departure delays, using a consistent and accurate dataset that can inform the predictive model.
- A successful application of this model could lead to improved operational efficiency, customer satisfaction, and profitability for the airline industry.

Salient Points :

DATA INGESTION

Ingest data from different sources namely - SQL database, NoSQL database and Web API

DATA CLEANING

Perform a simple preprocessing of data, removing redundant columns, Treating missing values, etc.

DATA TRANSFORMATION

Join the different datasets which are in different formats using azure data flow

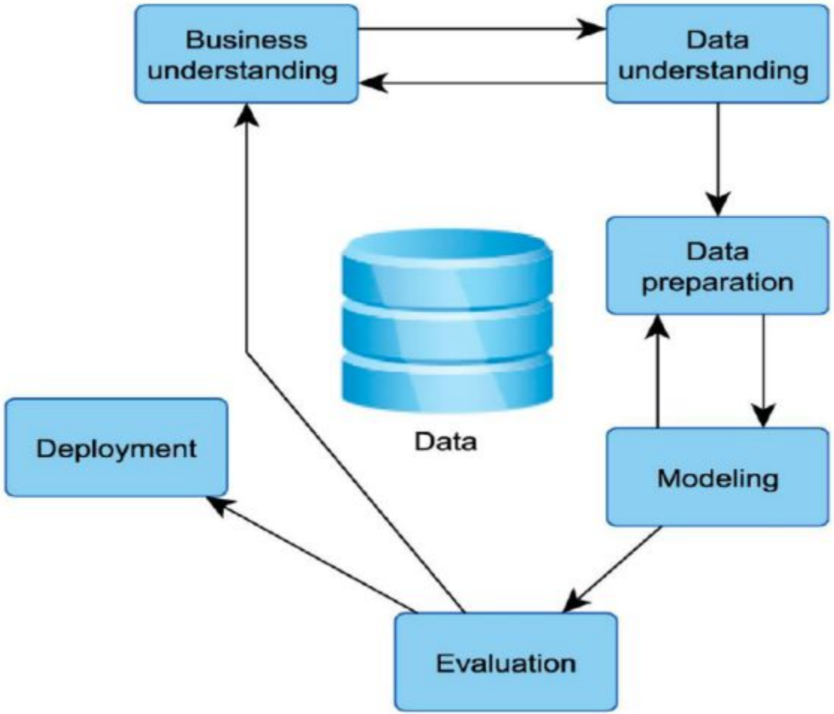
EDA

Perform extensive analysis of the combined master dataset which was made in azure data flow

DATA MODELLING

Prepare a ML pipeline for predicting departure delays and automate from ingestion to modelling in ADF.

ARCHITECTURE:



ABOUT THE DATASET

The data used for this project is a comprehensive collection of flight information from US airports. It is divided into 4 datasets, as follows:

Dataset : https://drive.google.com/drive/folders/1KYK2Ip6fNSc247zQ6QwkDdcQ6L7CSCLe

The Flights data, Airports Data and Airlines data were initially in CSV file, but to simulate how a organization receives data from multiple sources , we have changed then saved the files in SQL and Cosmos db

1. Flights Data - this transactional dataset contains over 6 million records, covering flight schedules and actual departure and arrival times- SQL database
2. Airports Data - this dataset contains IATA airport codes, names and other geographic details.- Cosmos db
3. Airlines Data - this dataset contains IATA airline codes and names.- Cosmos db
4. Weather Data - this dataset contains details of weather conditions during each scheduled flight- Raw data of Web API data stored data lake

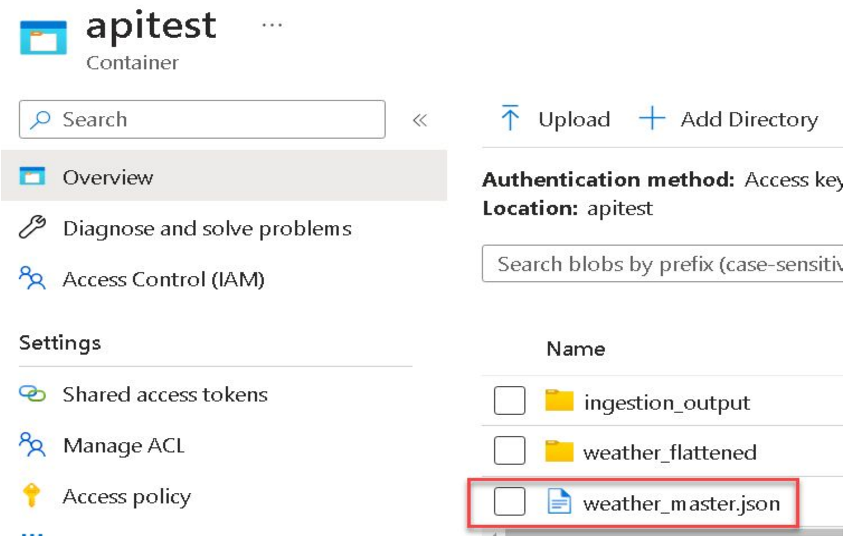
METHODOLOGY - DATA INGESTION

Data Ingestion Pipeline

- Sources - Azure SQL DB, CosmosDB, web API
- Tool used - Azure Data Factory
- Steps - data collection, transformation - joins, cleaning

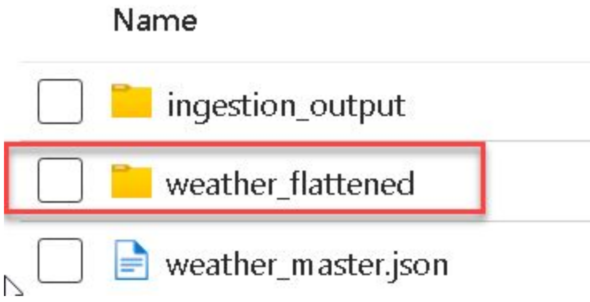
Data ingestion(Web api)

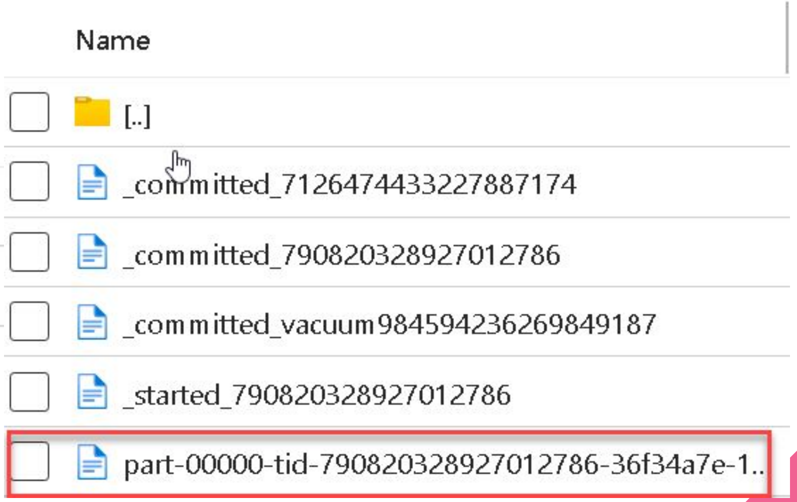
The weather data is fetched for the year of 2015 from a web api which has limits on the number of calls per minute, hence the data was gathered locally and raw output from the api was combined and imported into the data lake.



Data ingestion(Web api)

The initial schema of the raw json file is not suitable for joining with the other relatively structured datasets. Hence a databricks notebook in the pipeline processes the json file using spark and stores it back into the data lake as a single partition csv.





Data ingestion(Web api)

Schema before

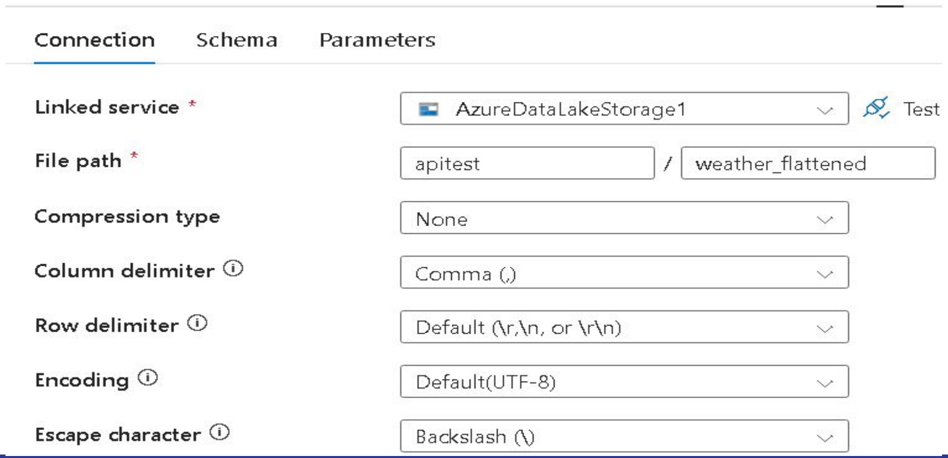
- [ ] JSON
  - { } 0
    - { } ABE
      - [ ] time
      - [ ] temperature_2m_max
      - [ ] temperature_2m_min
      - [ ] precipitation_sum
      - [ ] rain_sum
      - [ ] snowfall_sum
      - [ ] precipitation_hours
      - [ ] windspeed_10m_max
      - [ ] windspeed_10m_min
      - [ ] shortwave_radiation_sum

Schema after

- [ ] JSON
  - { } 0
    - Airport : "ABE"
    - time : "2015-01-01"
    - temperature_2m_max : 2.2
    - temperature_2m_min : -5.5
    - precipitation_sum : 0
    - rain_sum : 0
    - snowfall_sum : 0
    - precipitation_hours : 0
    - windspeed_10m_max : 19.4
    - windspeed_10m_min : 12
    - shortwave_radiation_sum : 9.41
  - { } 1
    - Airport : "ABE"
    - time : "2015-01-02"
    - temperature_2m_max : 4.8
    - temperature_2m_min : -2.9
    - precipitation_sum : 0
    - rain_sum : 0
    - snowfall_sum : 0
    - precipitation_hours : 0
    - windspeed_10m_max : 17.4
    - windspeed_10m_min : 6.2
    - shortwave_radiation_sum : 8.79

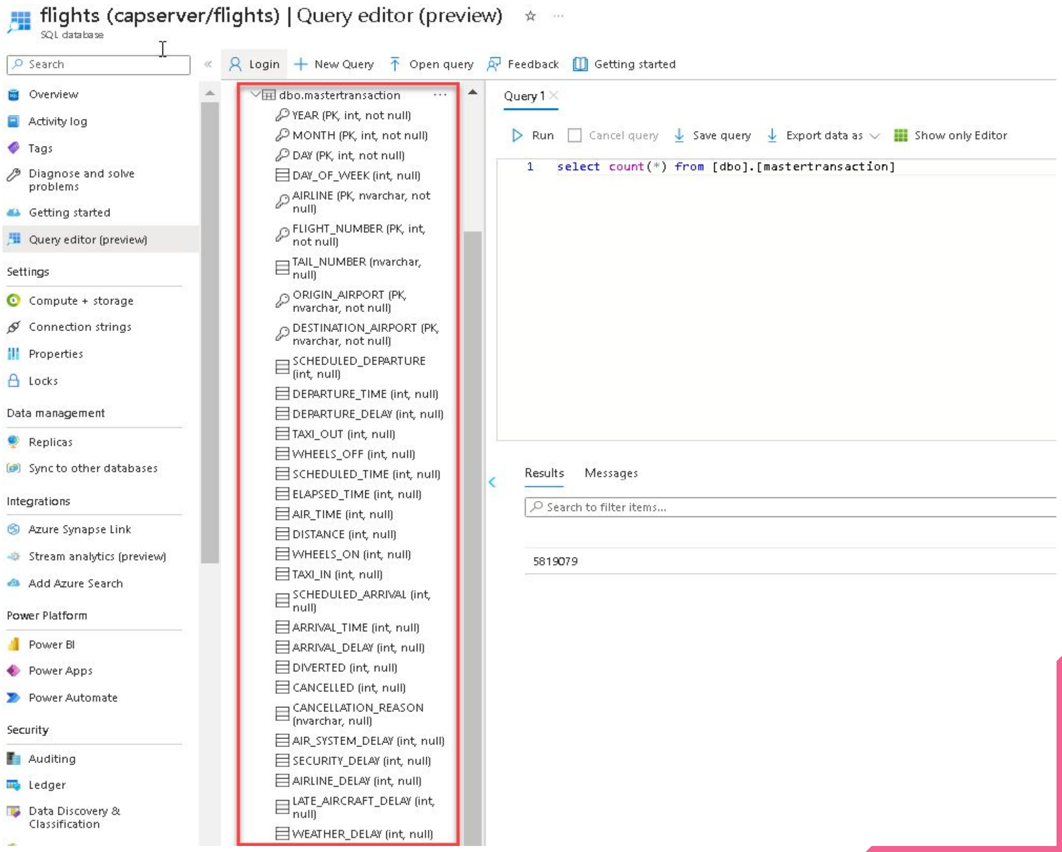
Data ingestion(Web api)

A dataset is created in azure data factory that points to the processed csv file present in data lake.



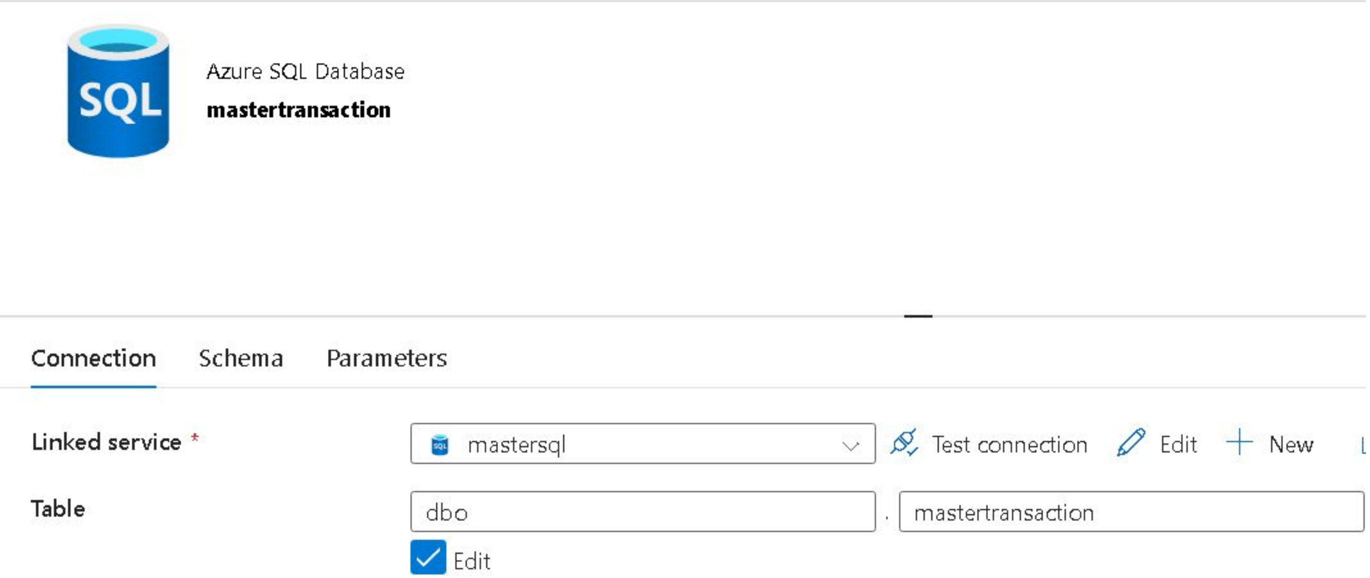
Data ingestion(SQL database)

The sql database contains transactional data on flight departures and arrivals.



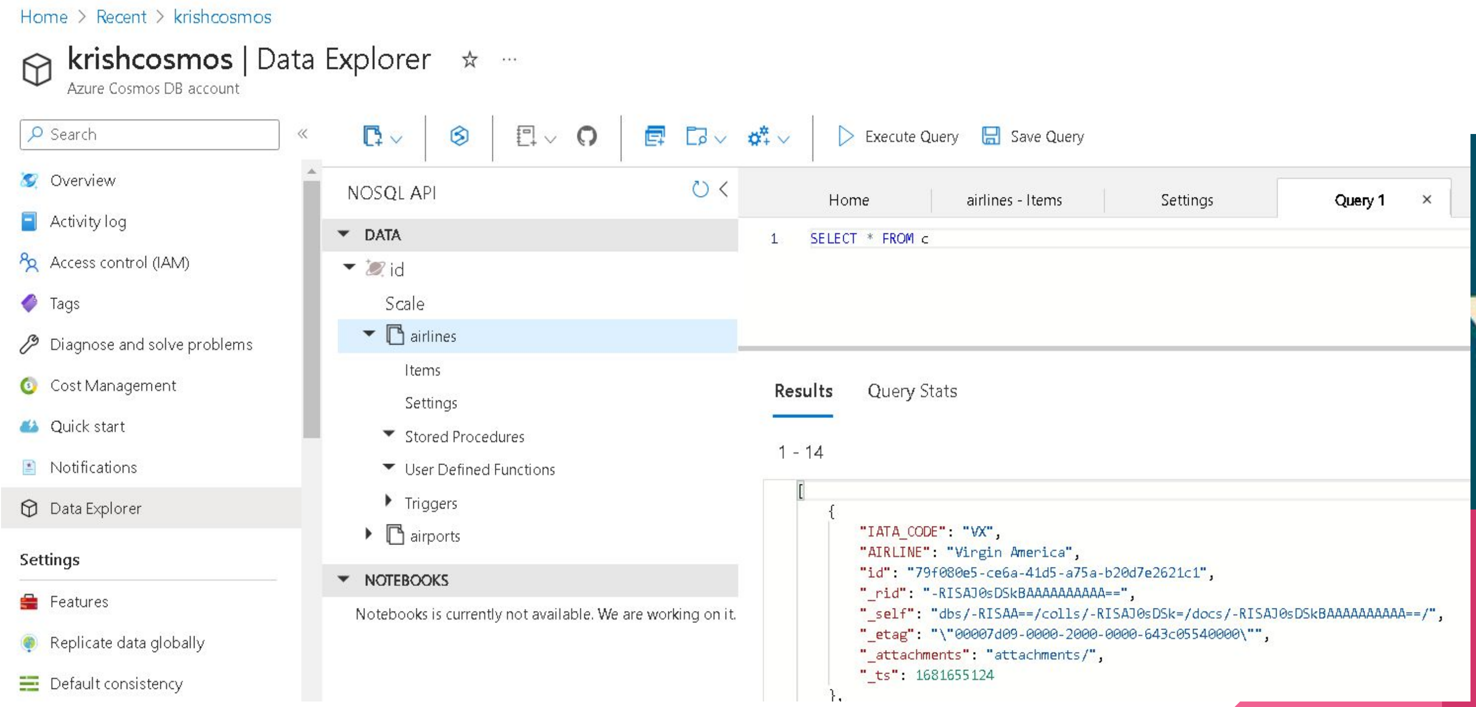
Data ingestion(SQL database)

A dataset pointing to the SQL database is created in azure data factory



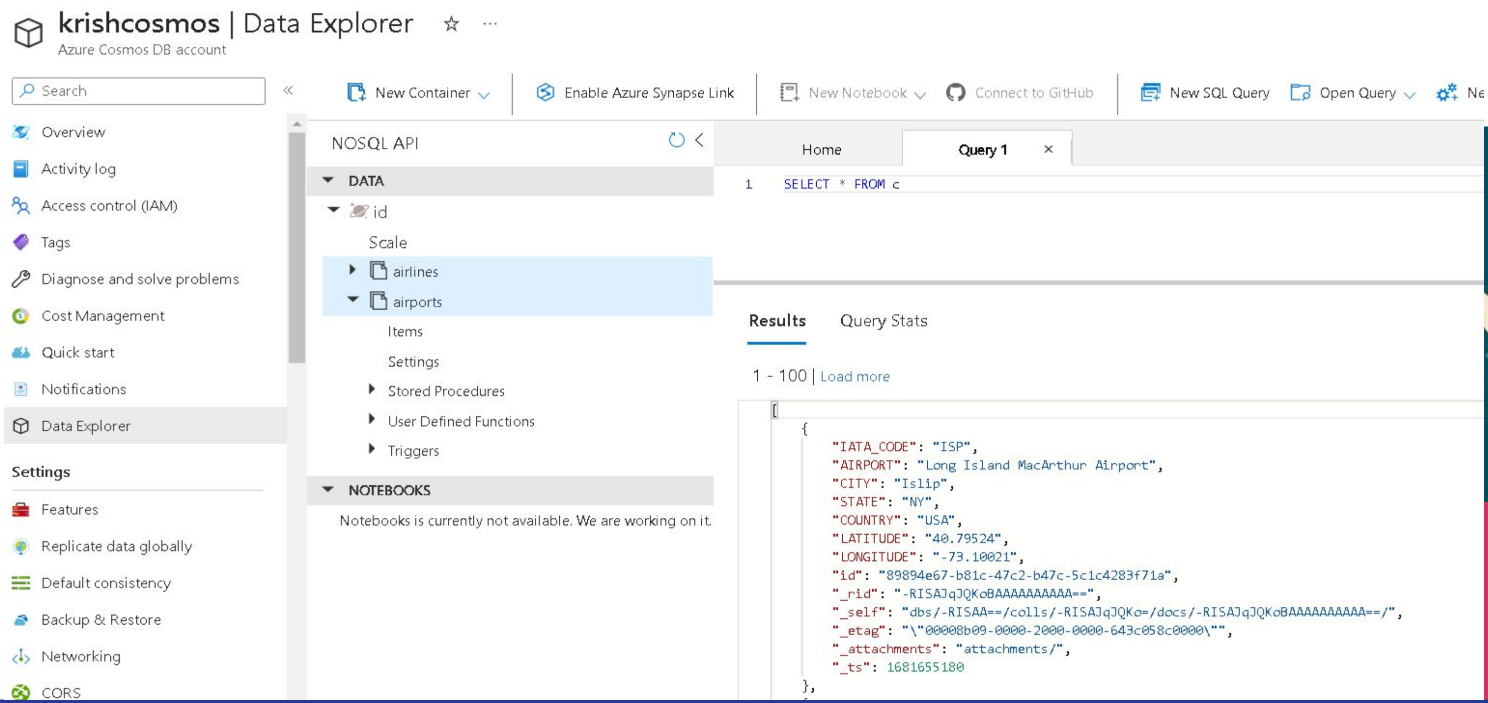
Data ingestion(nosql database)

In azure's Cosmos db the container contains non transactional details on airlines with name of airline and corresponding IATA codes.



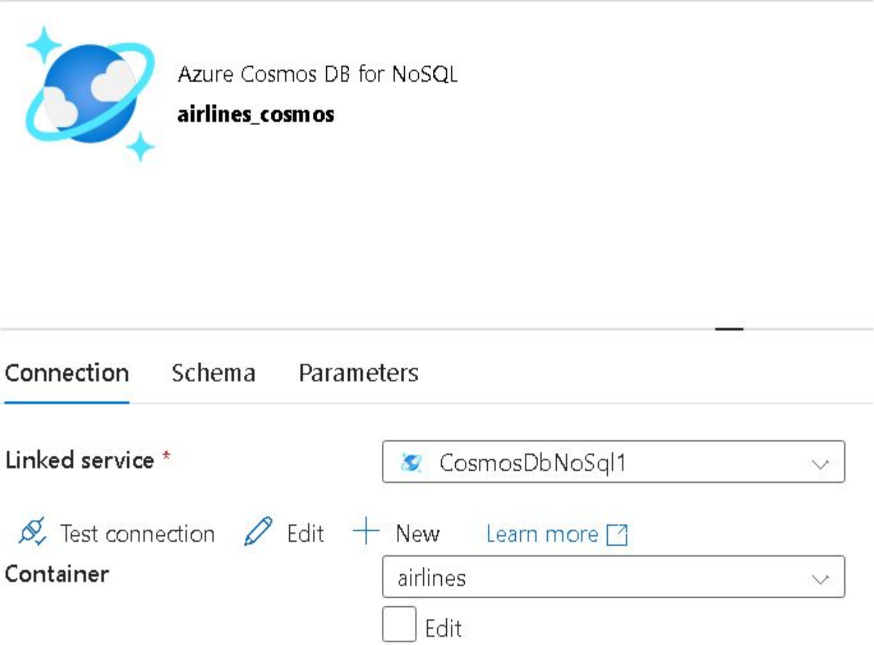
Data ingestion(nosql database)

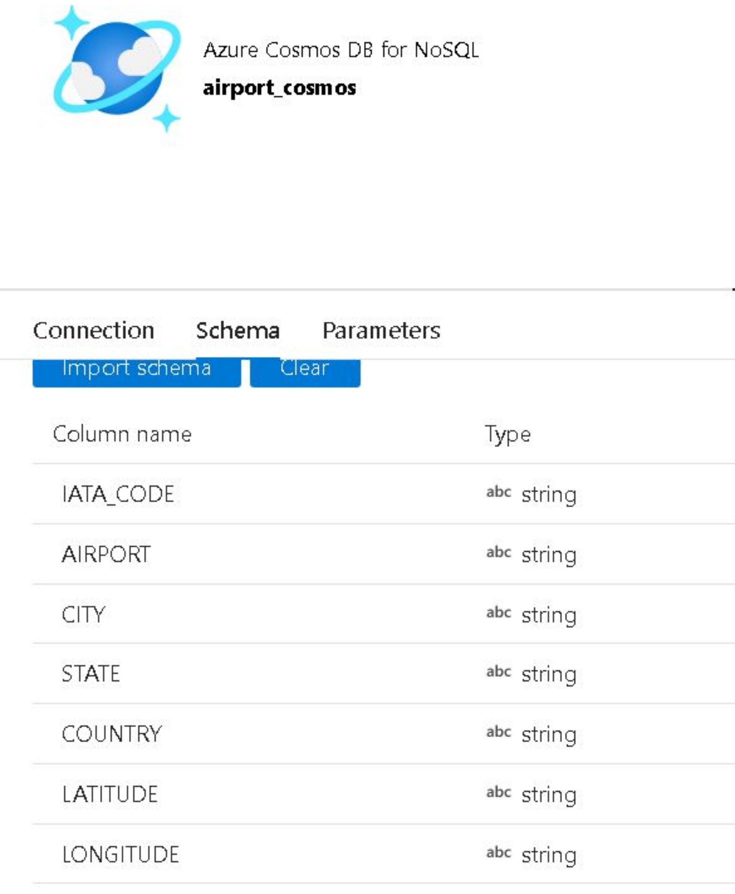
In azure's Cosmos db the container contains non transactional details on airports geographical location and IATA_codes.



Data ingestion(nosql database)

2 datasets pointing to the cosmos database container are created in azure data factory

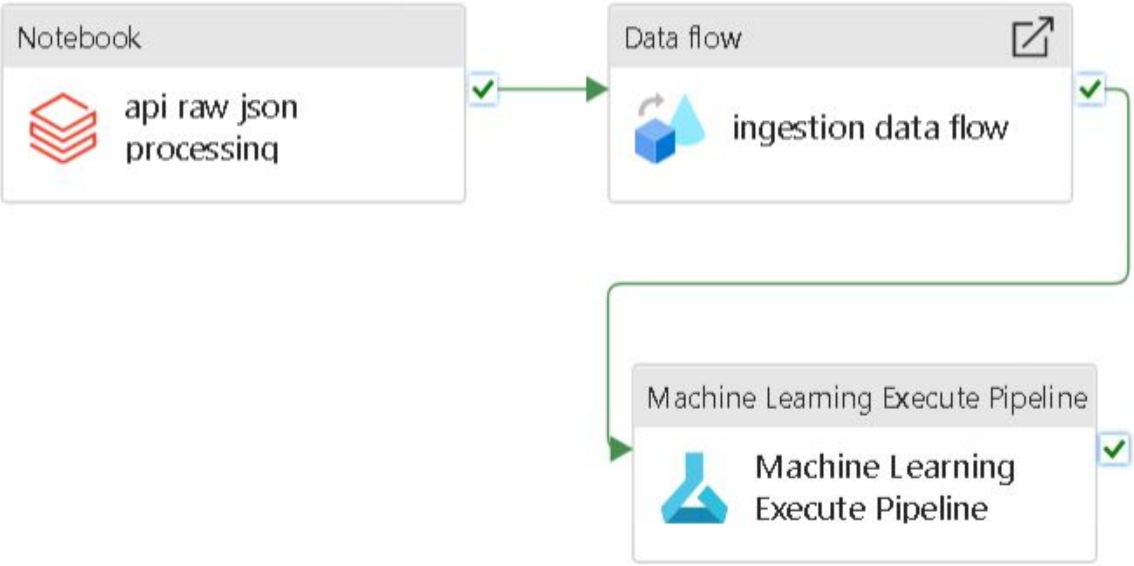




Data ingestion flow pipeline in ADF

Master pipeline overview

The master pipeline consists of 3 different components as shown below.



Api data processing

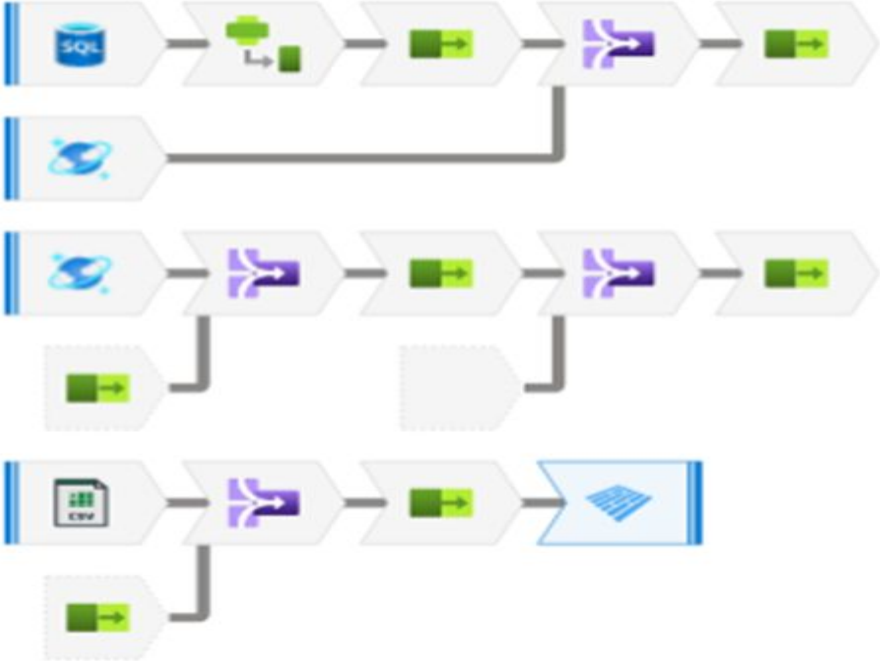
```
import pandas as pd
weather_df = pd.DataFrame()
for i in range(len(weather_data)):
  for key in weather_data[i].keys():
    weather_df = pd.concat([weather_df,pd.DataFrame(weather_data[i][key])])

airport_df = pd.DataFrame()
for i in range(len(weather_data)):
  for key in weather_data[i].keys():
    d = {'Airport':[key]*365}
    airport_df = pd.concat([airport_df,pd.DataFrame(d)])

weather_df['Airport'] = airport_df['Airport']
```

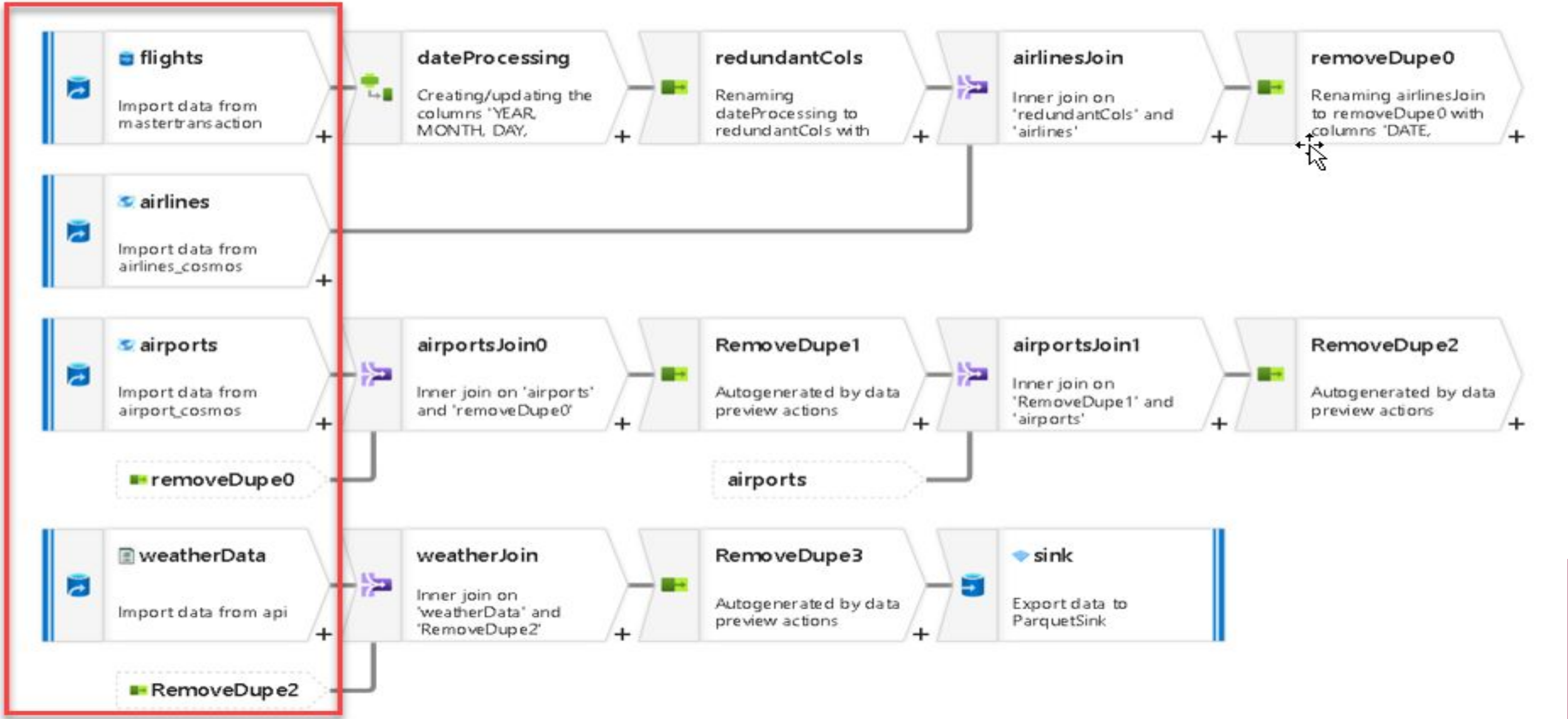
Data flow pipeline overview

The data flow pipeline purpose is to join all the 4 input datasets from different sources, clean them and store the combined dataset into a sink in the data lake.

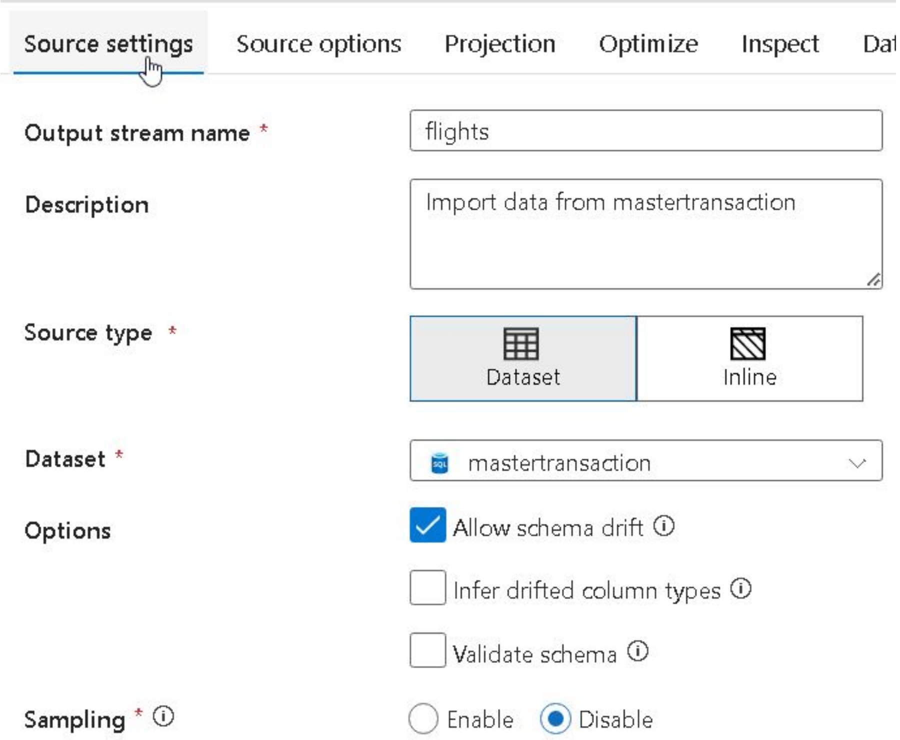


Data flow pipeline overview

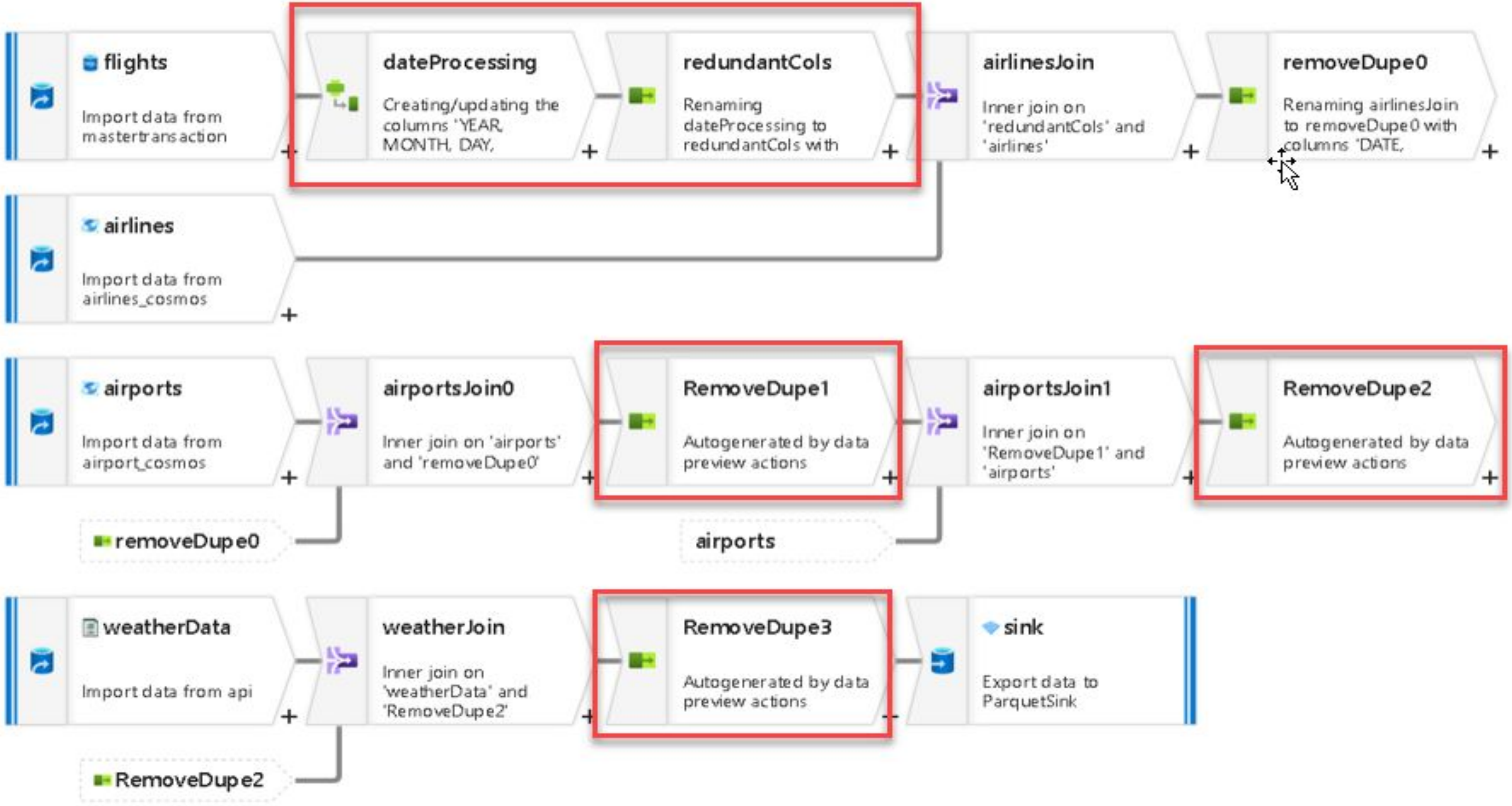
Data sources configuration



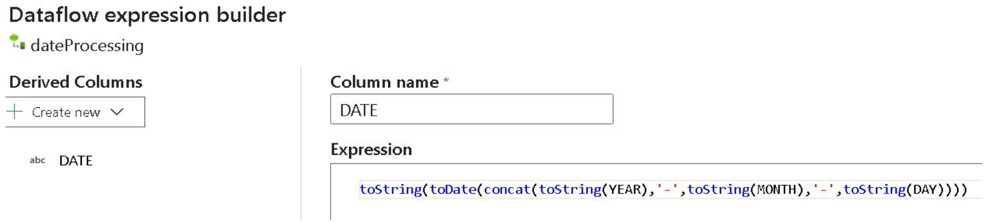
Data sources configuration

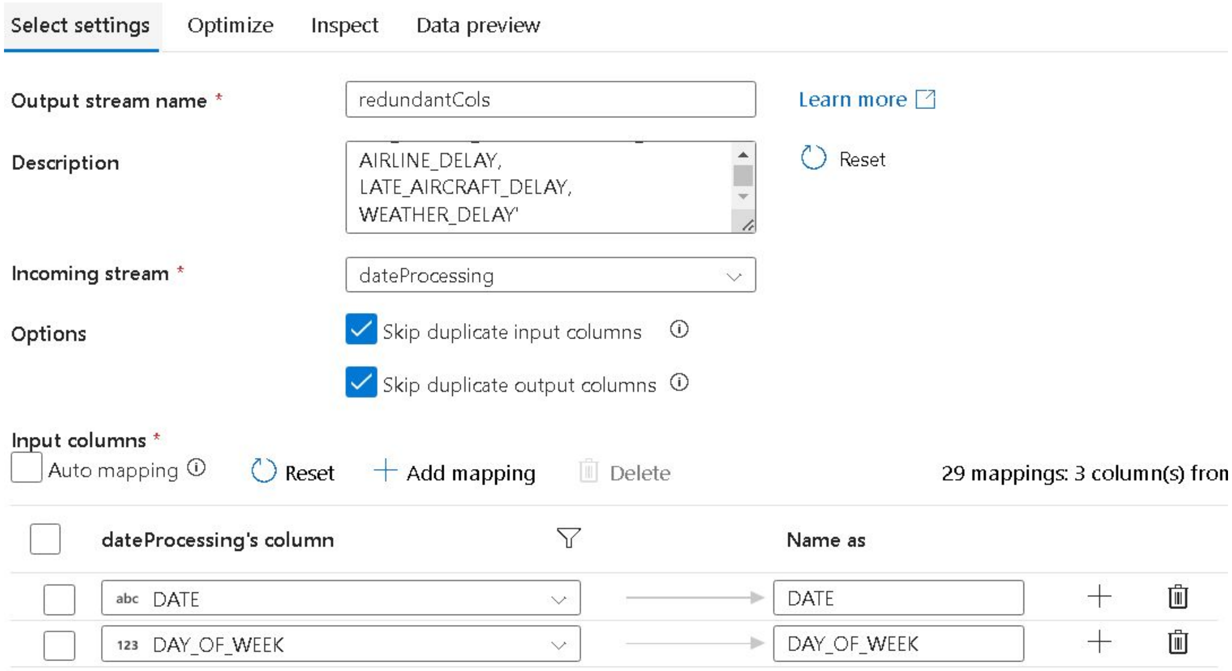


Data cleansing configuration

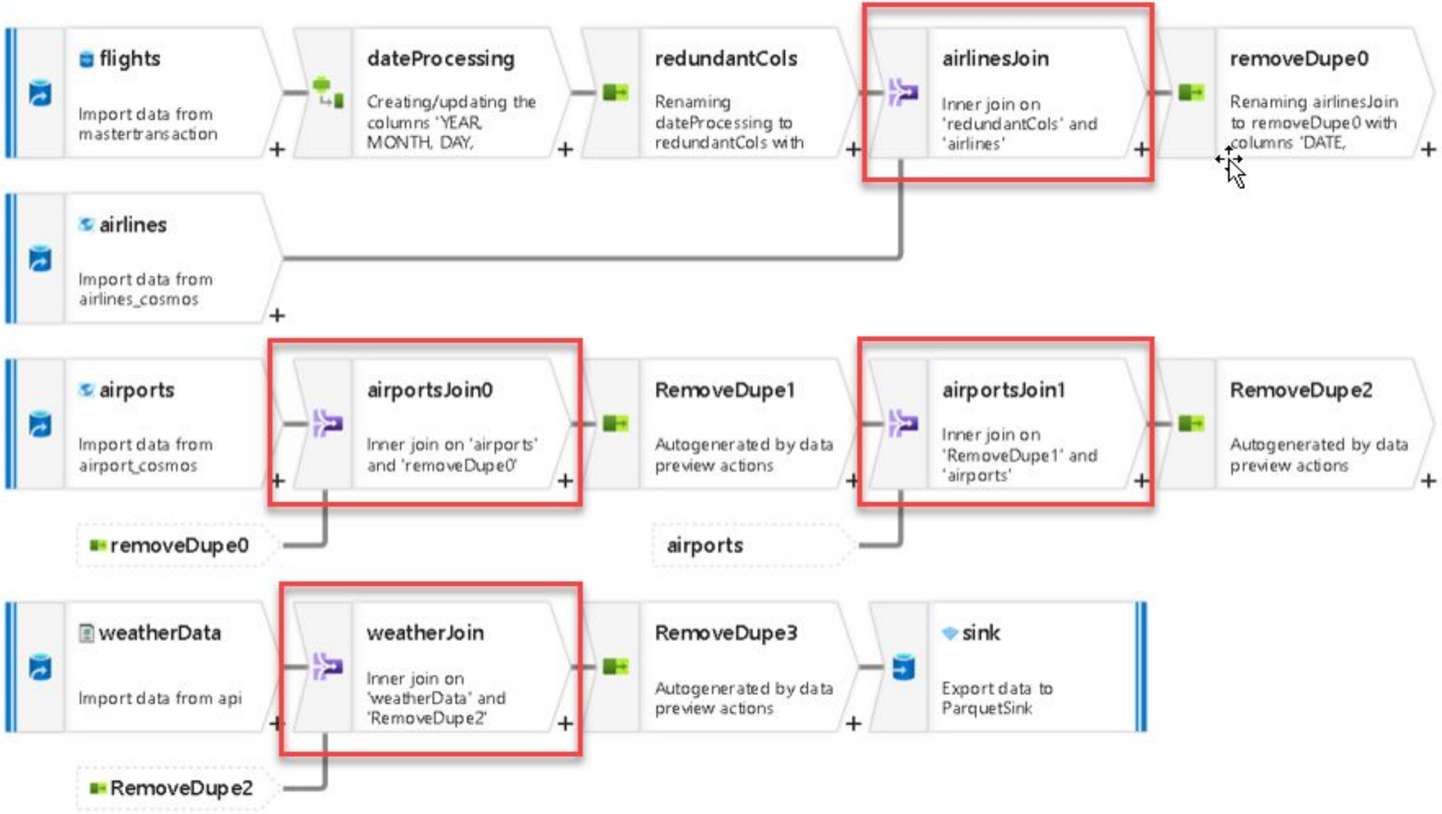


Data cleansing configuration

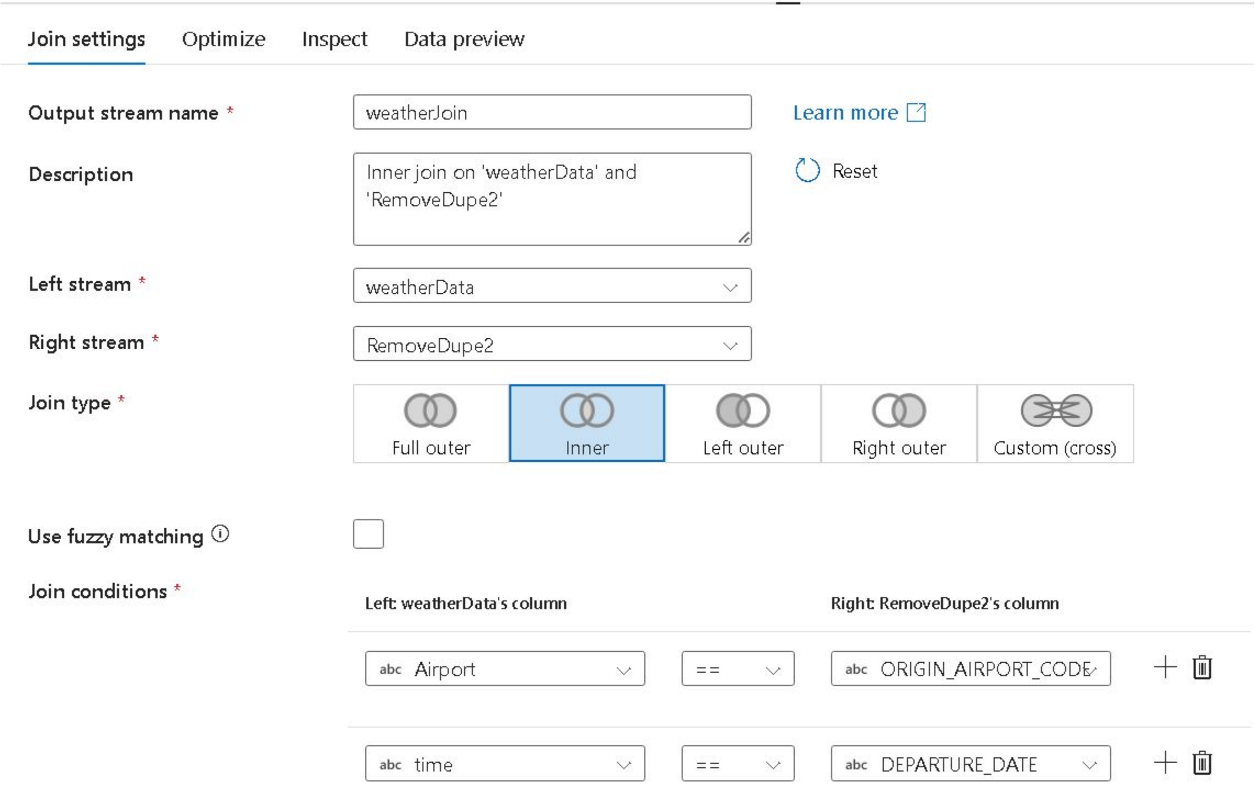




Data Joining configuration

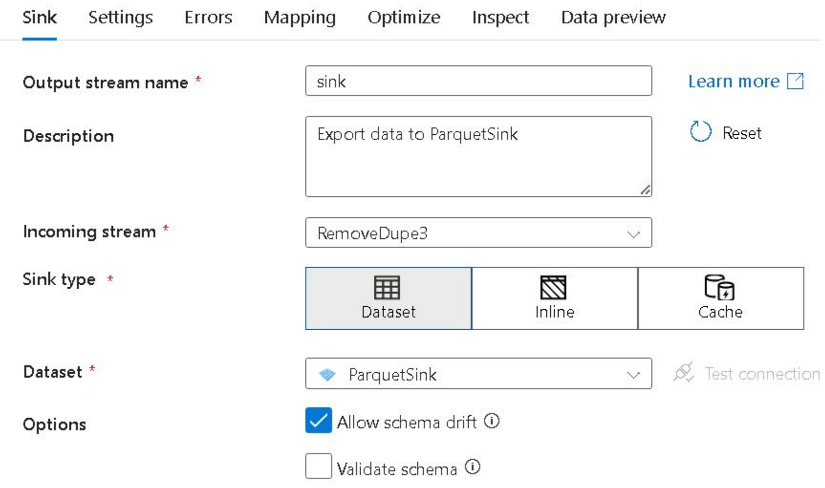


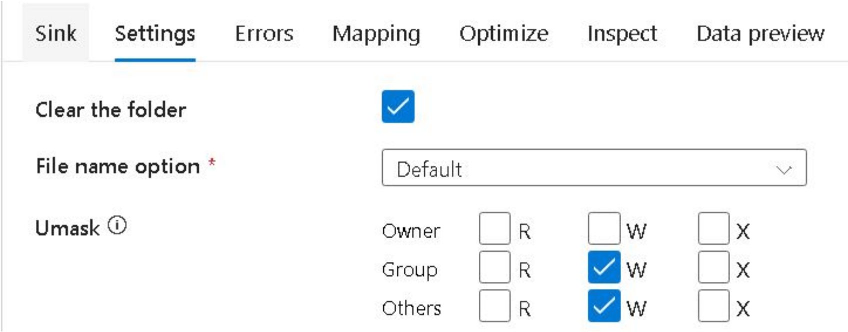
Data Joining configuration

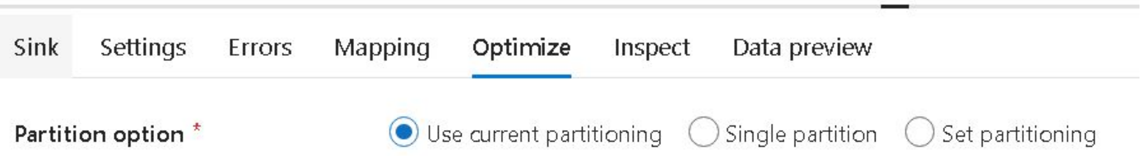


Data Sink configuration

Data Sink configuration







Data Sink configuration

Parquet
ParquetSink

Connection | Schema | Parameters

Linked service *	AzureDataLakeStorage1 (Test)
File path *	apitest / ingestion_output
Compression type	snappy

Exploratory Data Analysis (EDA)

METHODOLOGY - EDA

Exploratory Data Analysis

- This includes exploring the dataset to understand its characteristics and relationships between variables.
- It helps in identifying patterns, trends, and potential outliers that could affect the accuracy of the prediction model.
- Presentation of the analysis in a graphical or pictorial manner helps to visualize the features better, and make inferences based on correlations found in the data.

Missing Values in the Dataset

```
missing_df = df.isnull().sum(axis=0).reset_index()
missing_df.columns = ['variable', 'missing values']
missing_df['filling factor (%)']=(df.shape[0]-missing_df['missing values'])/df.shape[0]*100
missing_df.sort_values('filling factor (%)').reset_index(drop = True)
```

	variable	missing values	filling factor (%)
0	LATE_AIRCRAFT_DELAY	4325596	18.818597
1	WEATHER_DELAY	4325596	18.818597
2	AIRLINE_DELAY	4325596	18.818597
3	AIR_SYSTEM_DELAY	4325596	18.818597
4	SECURITY_DELAY	4325596	18.818597
5	ARRIVAL_DELAY	101740	98.090576
6	DEPARTURE_DELAY	83781	98.427625
7	SCHEDULED_TIME	6	99.999887
8	AIRLINE	0	100.000000
9	ORIGIN_windspeed_max	0	100.000000
10	ORIGIN_precipitation_hours	0	100.000000
11	ORIGIN_snowfall_sum	0	100.000000
12	ORIGIN_rain_sum	0	100.000000
13	ORIGIN_precipitation_sum	0	100.000000
14	ORIGIN_temperature_min	0	100.000000

Top 5 columns have 20% filling rate, which is quite low

They come into picture only when there is some delay, so these do not have missing data but rather context specific data.

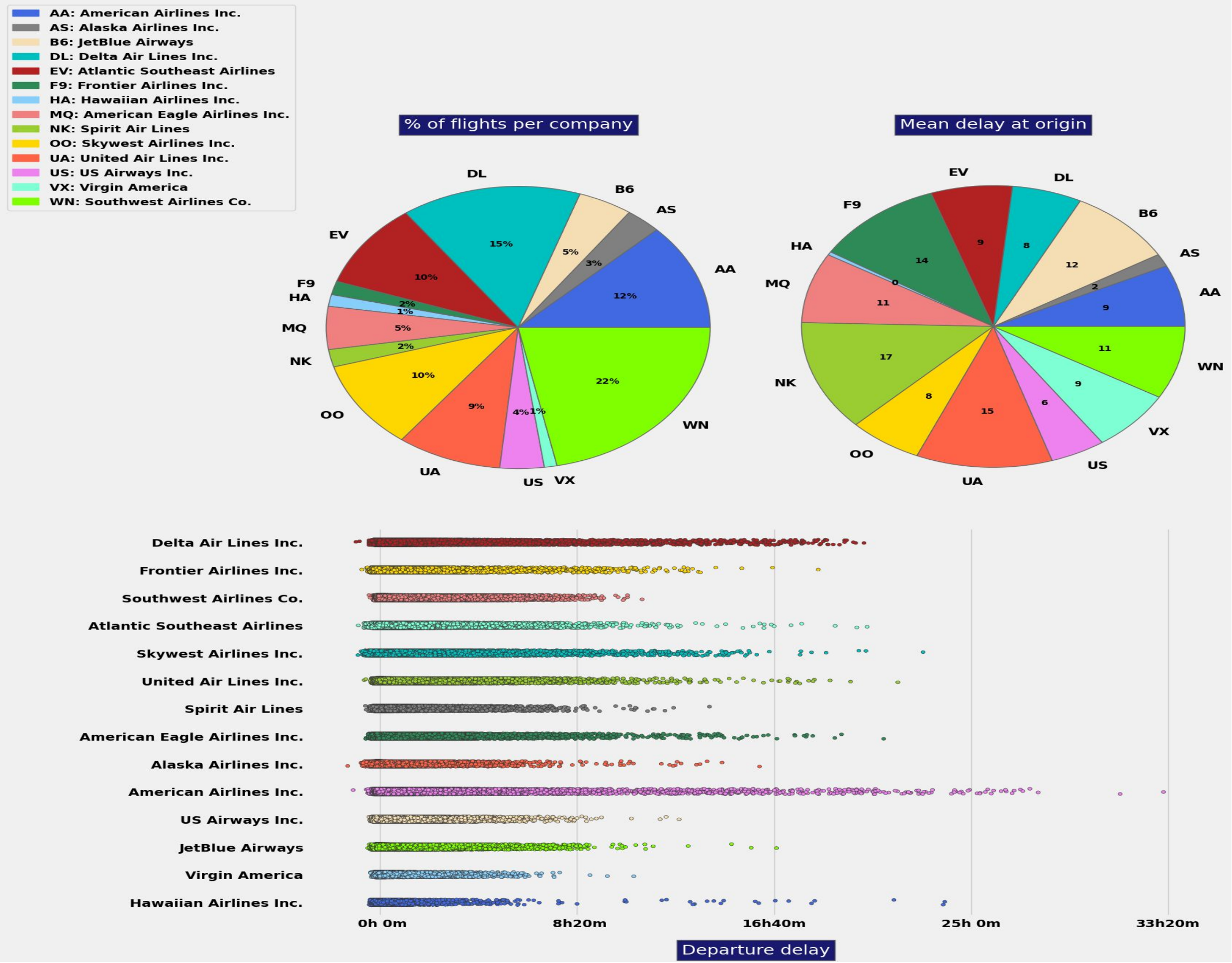
Need to remove rows with null values of Departure and Arrival Delays as these are variables to be predicted

AIRLINE	FLIGHT_COUNT	ON_TIME_COUNT	ON_TIME_PERCENT
Delta Air Lines Inc.	792941	558374	70.42
Alaska Airlines Inc.	157025	104574	66.60
American Airlines Inc.	636554	406985	63.94
United Air Lines Inc.	462086	287589	62.24
American Eagle Airlines Inc.	257130	158363	61.59
Southwest Airlines Co.	1136750	697063	61.32
Atlantic Southeast Airlines	508958	311215	61.15
Skywest Airlines Inc.	528328	322557	61.05
JetBlue Airways	240304	146455	60.95
US Airways Inc.	194223	117938	60.72
Virgin America	55813	33569	60.15
Hawaiian Airlines Inc.	69815	41881	59.99
Frontier Airlines Inc.	81861	43238	52.82
Spirit Air Lines	104781	52277	49.89

Ratio of flights ON TIME by each airline

Observations -

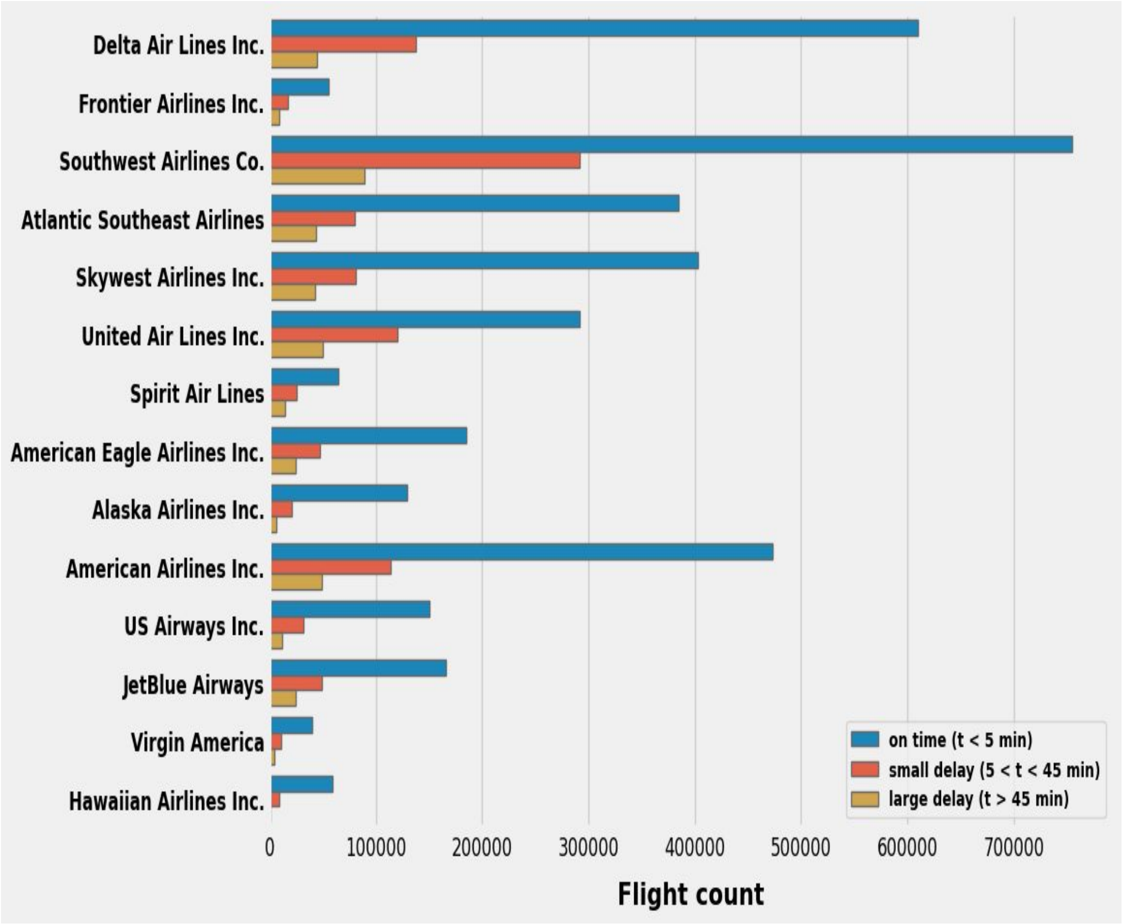
- Delta airlines seem to have the best record in terms of being on time
- Alaska Airlines has flown quite less compared to others, so their record is of a lower base.
- Southwest has flown the most, and has a decent record
- Spirit Airlines has a less than 50% record of completing the flight on time



Southwest Airlines accounts for ~ 20% of the flights which is equal to flights chartered by the 7 tiniest airlines.

~ 11 ± 7 minutes would correctly represent all mean delays.

Occasionally, we can face really large delays that can reach a few tens of hours !



Duration of delays per each airline

This figure gives a count of the delays as -

1. Less than 5 minutes,
2. 5 < t < 45 min
3. Greater than 45 minutes.

Delays greater than 45 minutes only account for a few percents.

In the case of SkyWest Airlines, the delays > 45 minutes are only lower by ~ 30% with respect to delays in the range 5 < t < 45 min.

Things are better for SouthWest Airlines since delays > 45 minutes are 3 times less frequent than delays in the range 5 < t < 45 min

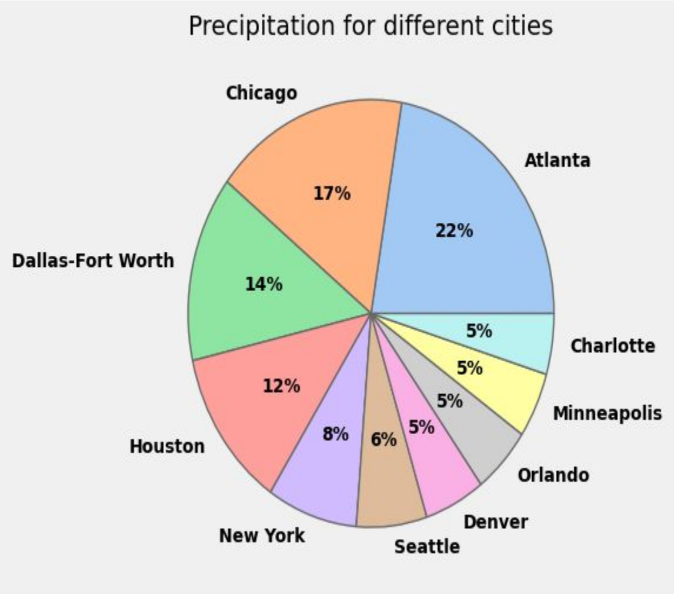
Analyzing total delays by each airport

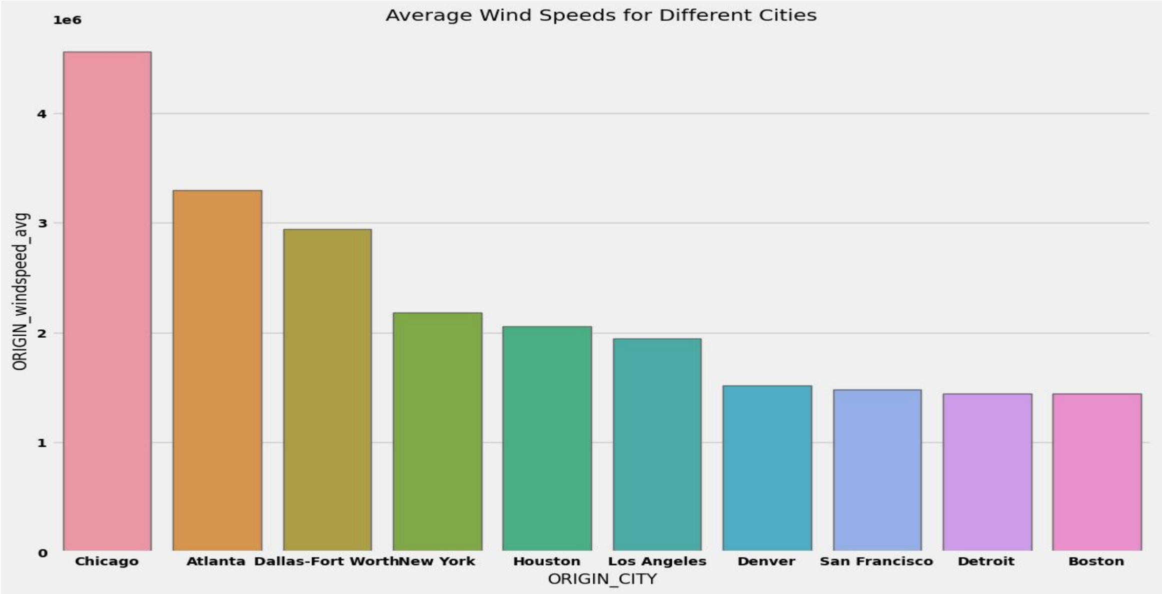
ORIGIN_AIRPORT_NAME	SECURITY_DELAY	AIR_SYSTEM_DELAY	AIRLINE_DELAY	WEATHER_DELAY	LATE_AIRCRAFT_DELAY	TOTAL_DELAYS
Chicago O'Hare International Airport	2697.0	947333.0	1260840.0	422593.0	1543993.0	4177456.0
Hartsfield-Jackson Atlanta International Airport	1334.0	634336.0	1207193.0	313491.0	1052413.0	3208767.0
Dallas/Fort Worth International Airport	5704.0	532221.0	1068866.0	256145.0	1042585.0	2905521.0
Denver International Airport	894.0	541548.0	757379.0	92635.0	1026488.0	2418944.0
Los Angeles International Airport	3215.0	436875.0	706498.0	32181.0	999620.0	2178389.0
George Bush Intercontinental Airport	2404.0	439734.0	614859.0	85568.0	643952.0	1786517.0
San Francisco International Airport	1783.0	287977.0	557758.0	26520.0	860377.0	1734415.0
McCarran International Airport	1054.0	303069.0	493739.0	31389.0	686130.0	1515381.0
LaGuardia Airport (Marine Air Terminal)	554.0	365359.0	354413.0	57349.0	703435.0	1481110.0
Orlando International Airport	1900.0	343101.0	410131.0	102521.0	586584.0	1444237.0
Phoenix Sky Harbor International Airport	5991.0	284333.0	515965.0	32599.0	530829.0	1369717.0
Newark Liberty International Airport	1699.0	278599.0	471903.0	57812.0	521971.0	1331984.0
John F. Kennedy International Airport (New York International Airport)	5882.0	335556.0	449859.0	92963.0	399769.0	1284029.0
Gen. Edward Lawrence Logan International Airport	2087.0	418765.0	348986.0	71672.0	419172.0	1260682.0
Detroit Metropolitan Airport	1170.0	299951.0	473649.0	43425.0	349062.0	1167257.0
Baltimore-Washington International Airport	2961.0	202666.0	410531.0	62754.0	412634.0	1091546.0
Charlotte Douglas International Airport	5435.0	335570.0	385763.0	42952.0	299302.0	1069022.0
Minneapolis-Saint Paul International Airport	641.0	288882.0	359620.0	35034.0	322811.0	1006988.0
Chicago Midway International Airport	428.0	150837.0	316097.0	57128.0	402415.0	926905.0
Miami International Airport	2574.0	215381.0	335073.0	53035.0	304223.0	910286.0

Chicago's airport clearly has the most amount of delays followed by airport of Atlanta and Dallas.

We will analyze further probable reasons as to why these airports face these delays

Weather Conditions for different cities



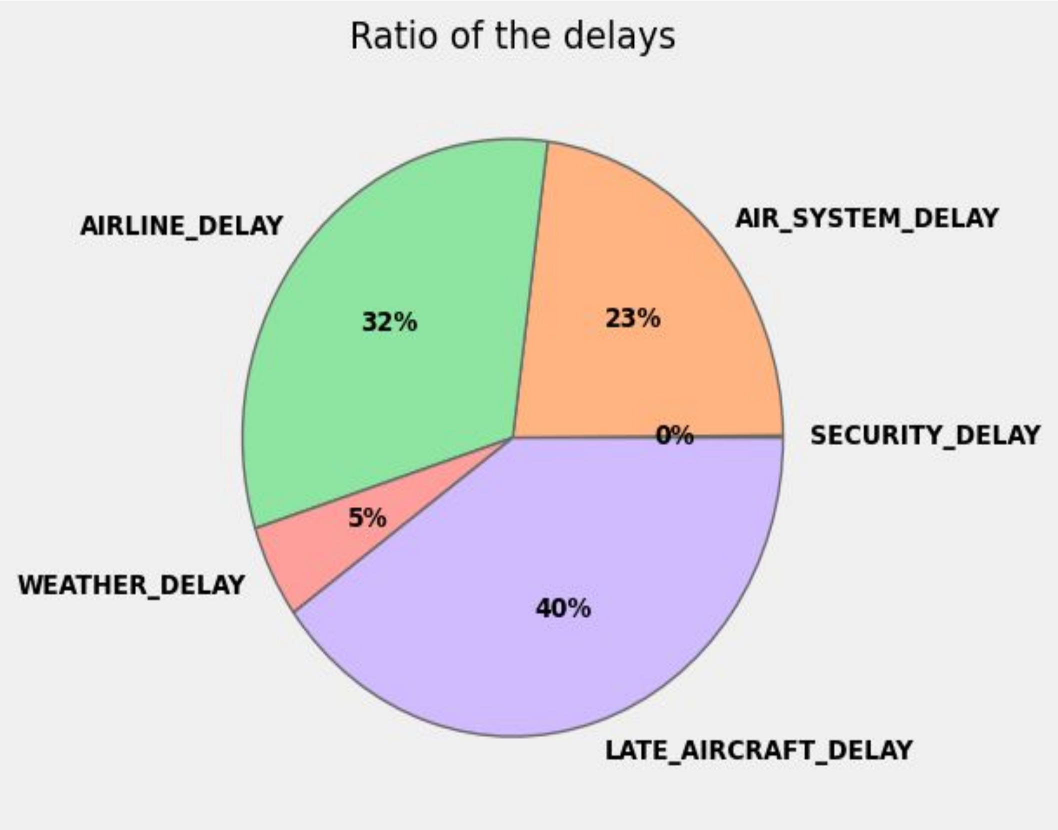


Observation -

Chicago, Atlanta and Dallas being on top of precipitation charts and wind charts and their airports on top of delay charts does not seem to be a coincidence.

The delays are probably due to the weather conditions of these cities.

Comparing different delays



1. LATE_AIRCRAFT_DELAY -> 23748821.0
2. AIRLINE_DELAY -> 18955762.0
3. AIR_SYSTEM_DELAY -> 13523157.0
4. WEATHER_DELAY -> 2987643.0
5. SECURITY_DELAY -> 77945.0

METHODOLOGY - DATA TRANSFORMATION

Data Cleaning

- After completing the EDA, it is necessary to perform data cleaning to remove or impute missing values, remove duplicates, and correct data inconsistencies.
- This step ensures that the data is clean and ready for modelling.

Data Manipulation

- The next step is to manipulate the data to create new variables or features that could be useful in predicting flight delays.
- This step could include feature engineering, such as extracting the week or hour of the day from the flight schedule data.

METHODOLOGY

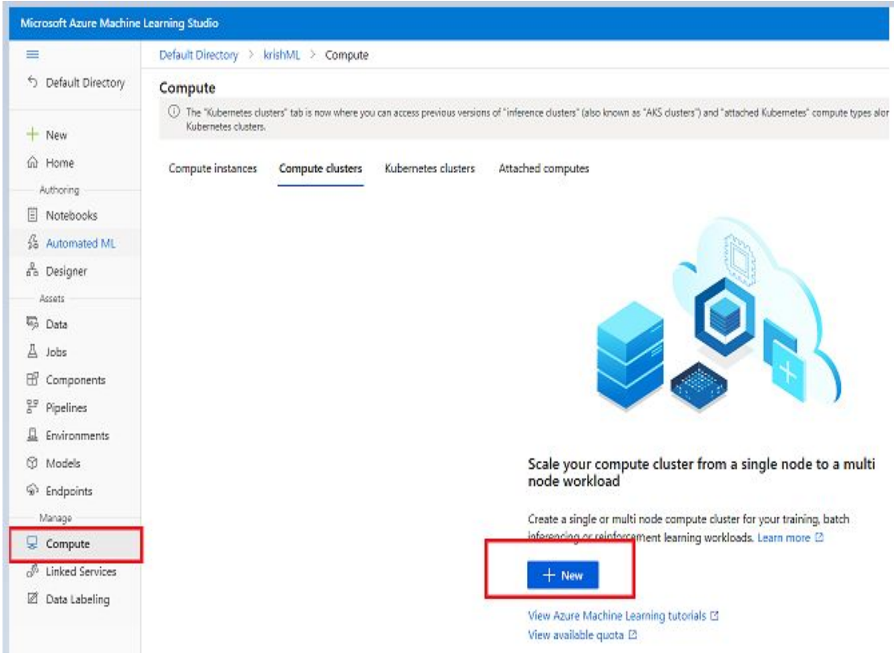
Modelling

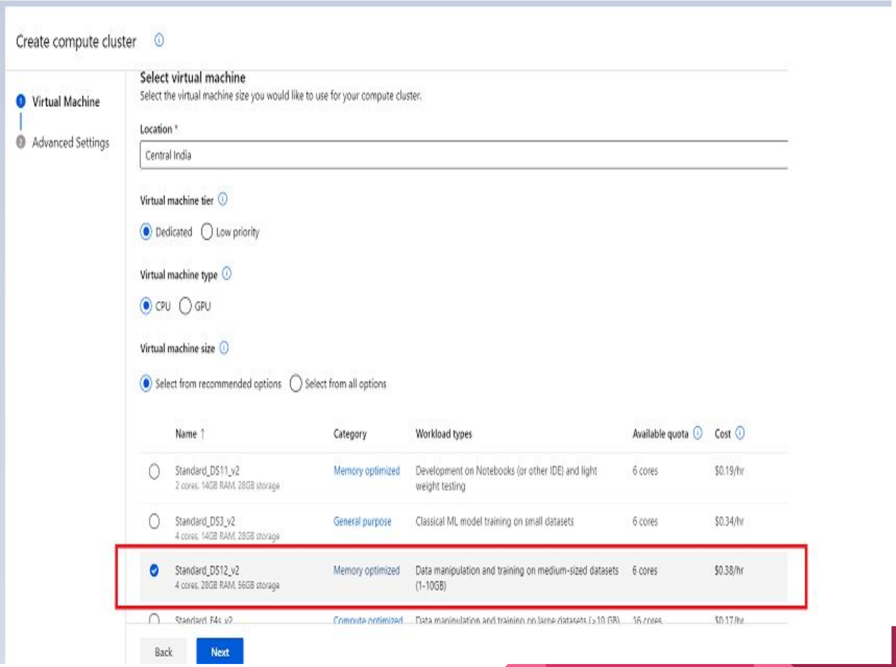
- The Machine Learning Model used for this problem statement is:
    - **Decision Tree Regressor**

- The decision tree regressor is a popular and effective algorithm for predicting airline delays because it can capture nonlinear relationships between features and the target variable, and it can handle categorical and numerical features.
- **Tool used -** Azure Machine Learning Studio
- Azure ML Studio provides us with a No Code way of preparing our dataset and training our model.

ML Studio Pipeline

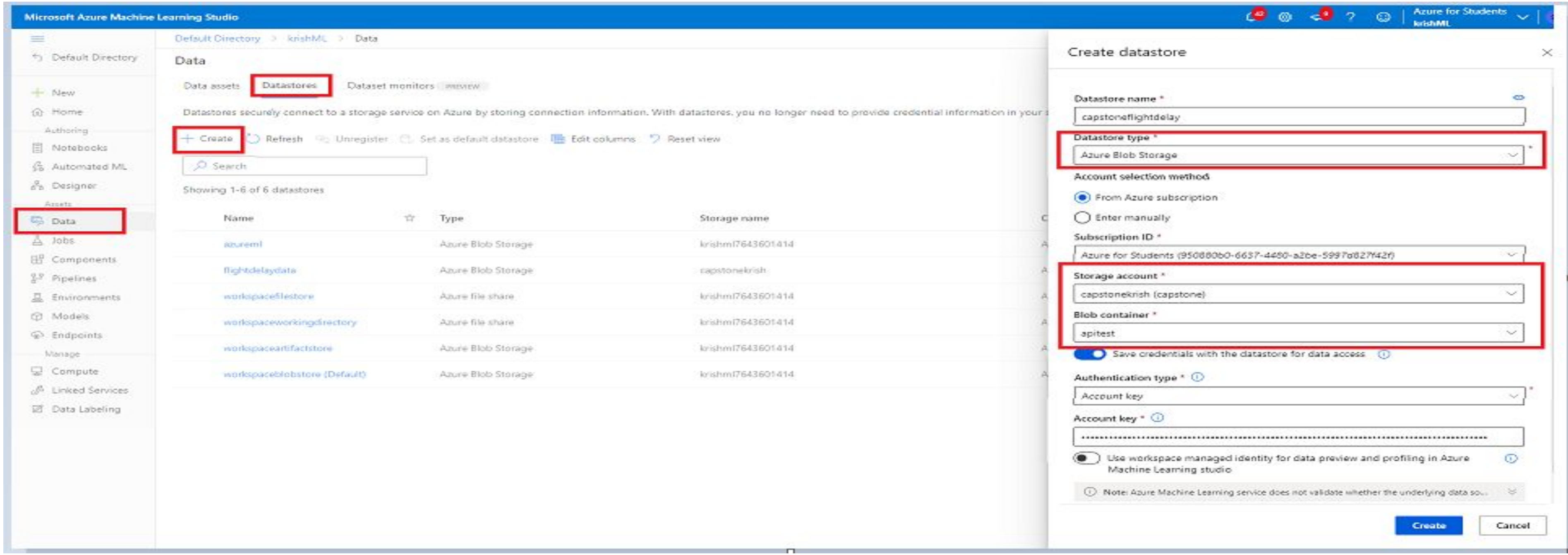
Create & Configure Compute Cluster





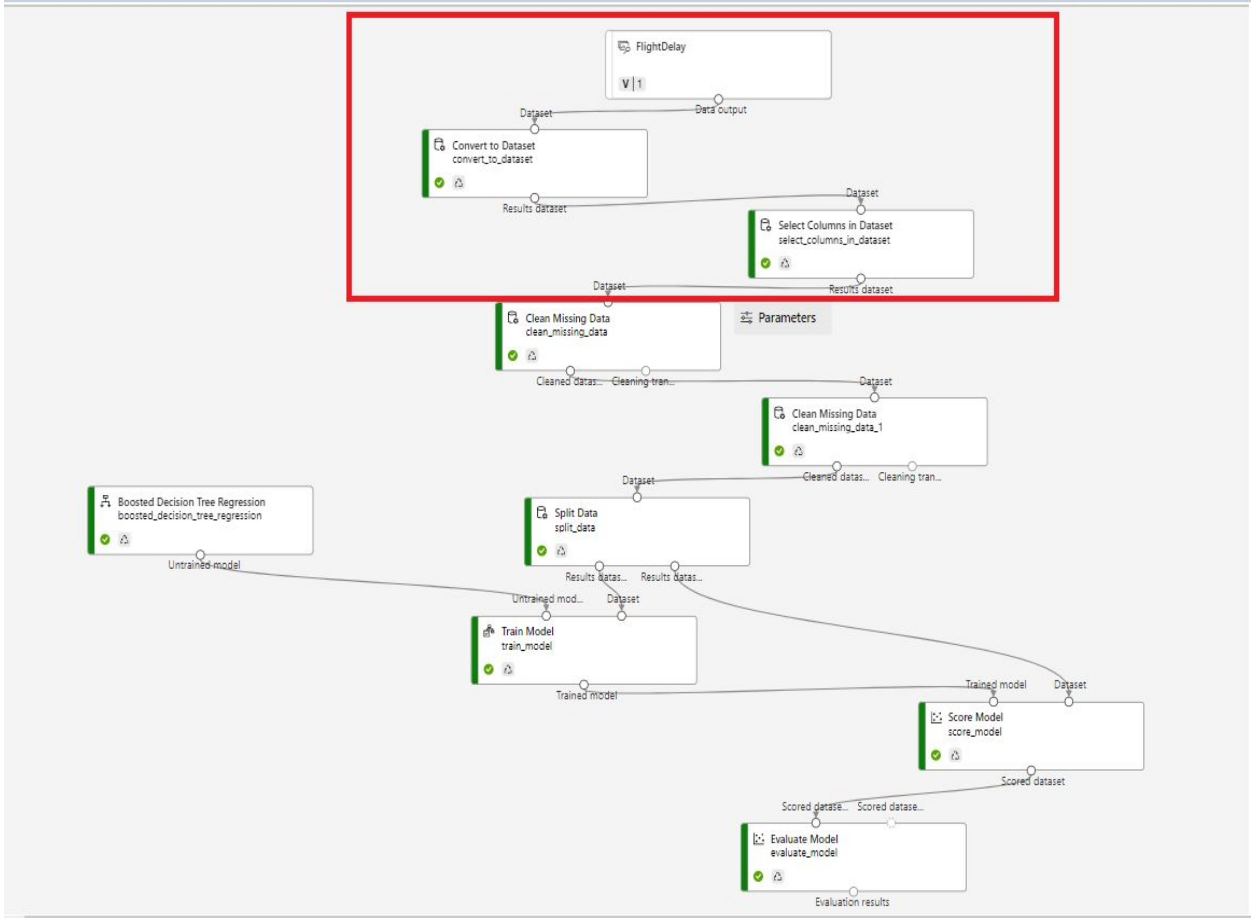
To execute the pipeline a compute is needed to be added and configured

Create Dataset in Azure ML Studio



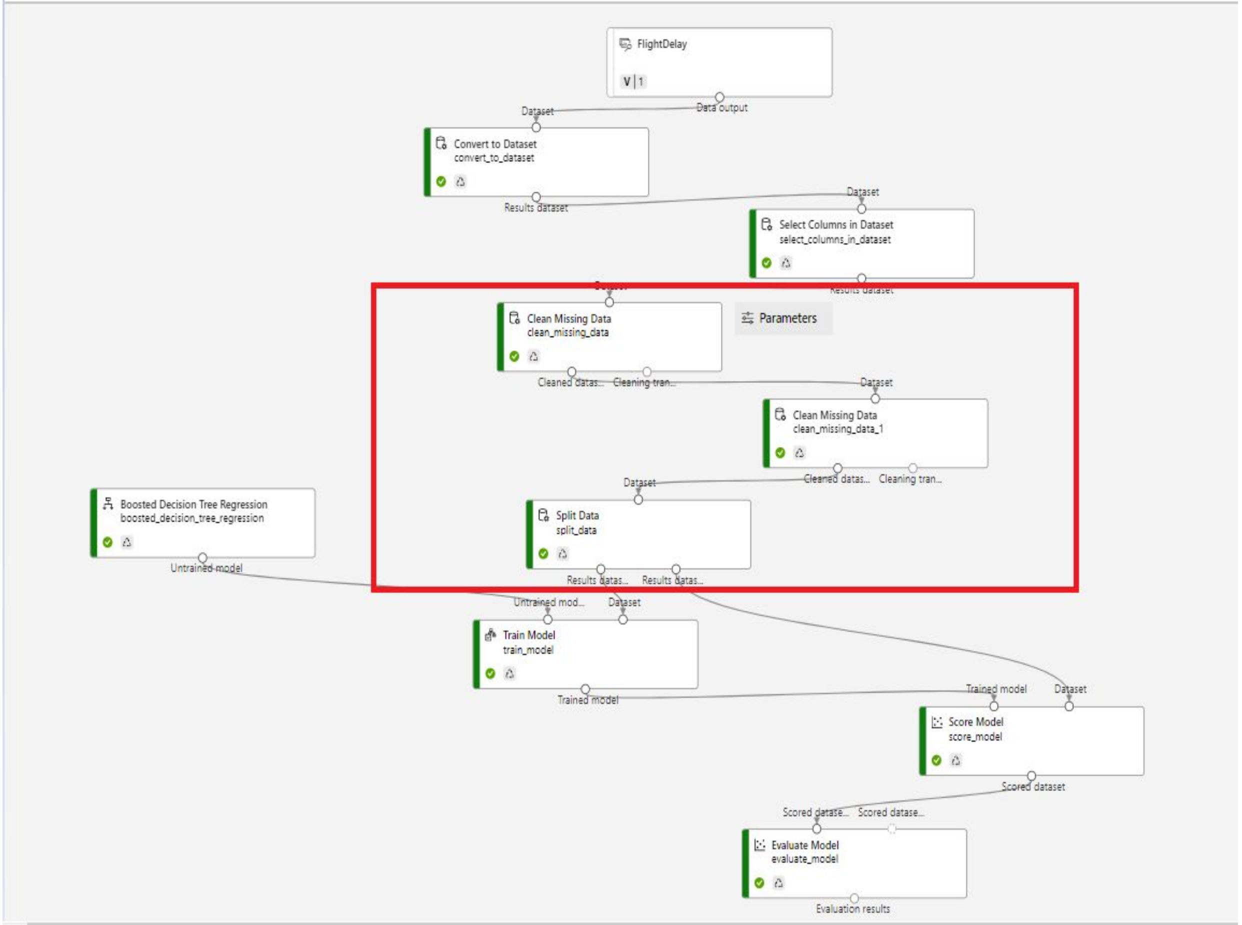
Go to Data > Datastore > + Create > Provide Datastore Type > Provide Storage Account > Provide the Account Key > Create

Design the ML Flow Pipeline



- Starting with flight delay dataset created in the previous slide
- Component added for Converting it to Dataset
- Select the columns of importance for the model

Design the ML Flow Pipeline contd..

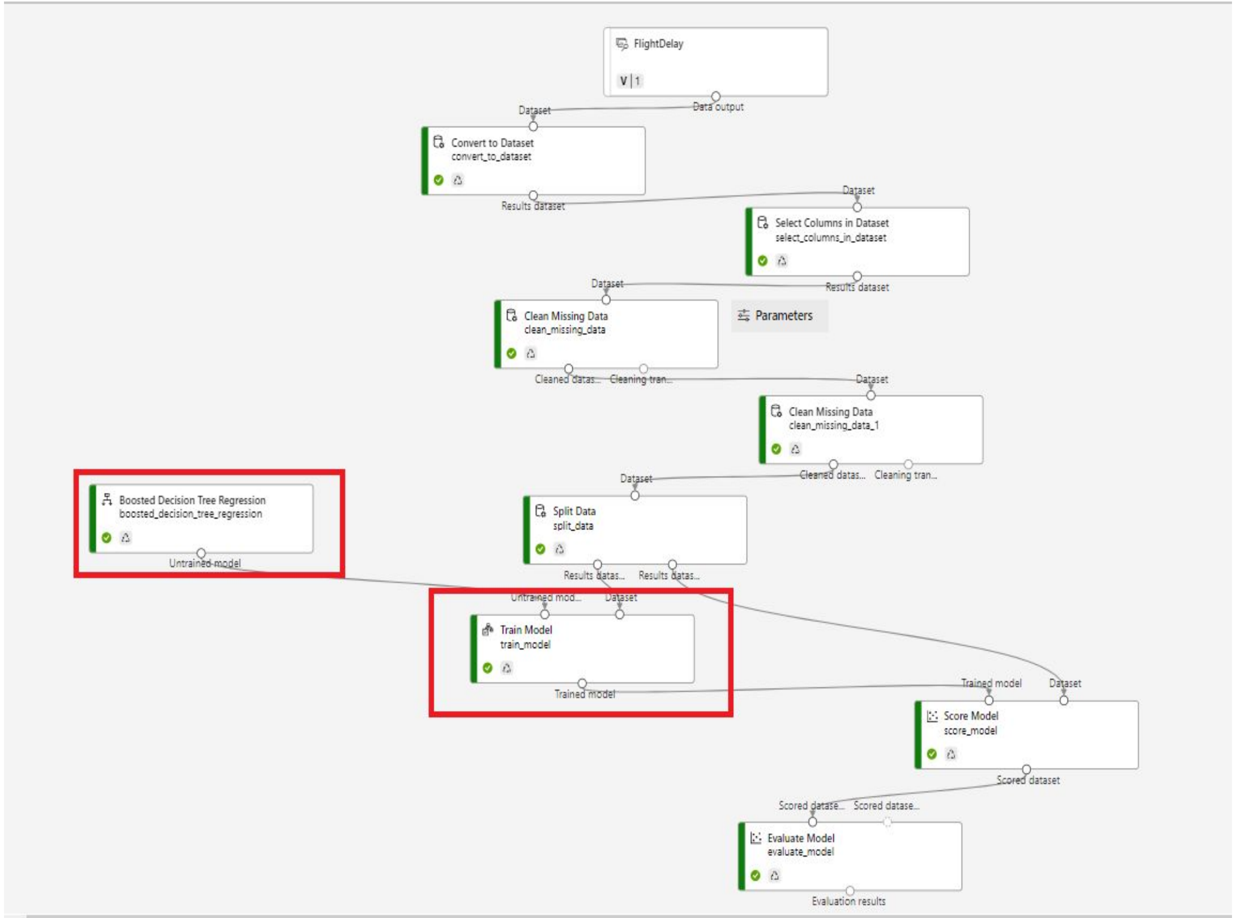


- First of two clean missing data components to remove null values for the dependent variable i.e Departure Delay

- Second of two clean data component to replace the null values of 5 types of delay with 0

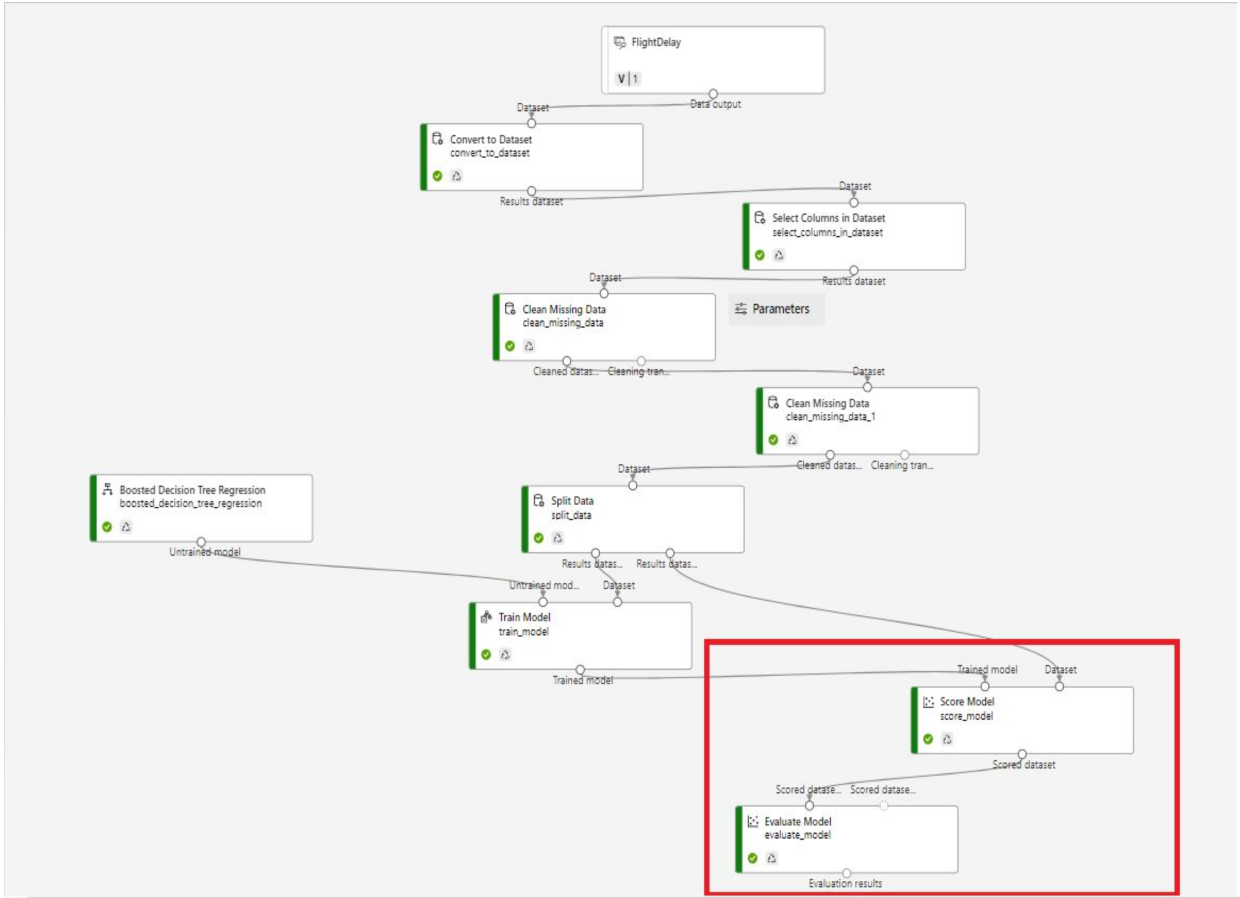
- Split data component added to split it into train and test data

Design the ML Flow Pipeline contd..



- Model component added. Boosted Decision Tree Regression Model
- Train Model component taking inputs from untrained model above and the train part of the split data component

Design the ML Flow Pipeline contd..

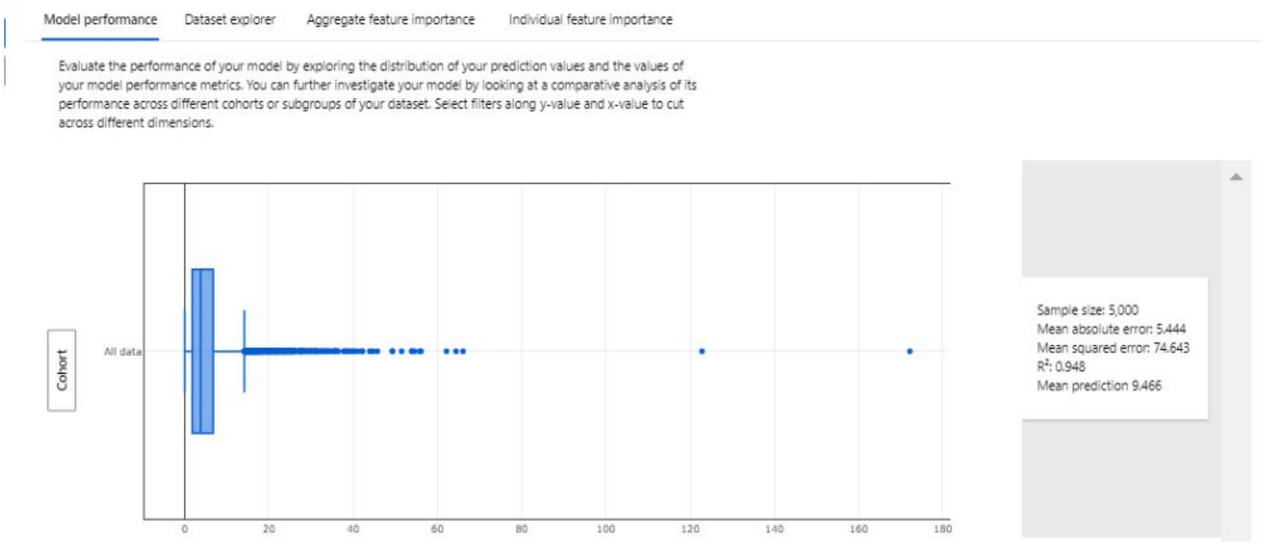


- Score model component taking inputs from Train model and Test dataset from Split data
- Score Model predicts the output on the test dataset
- Next Evaluate Model to check the model metrics and performance on the test dataset

Train Model Output

Feature Importance

ORIGIN_temperature_max
DESTINATION_AIRPORT_CODE
ORIGIN_AIRPORT_CODE
AIRLINE_IATA_CODE
ORIGIN_STATE
ORIGIN_CITY
WEATHER_DELAY
LATE_AIRCRAFT_DELAY
AIRLINE_DELAY
SECURITY_DELAY
DESTINATION_CITY
AIR_SYSTEM_DELAY
SCHEDULED_TIME
ORIGIN_shortwave_radiation_sum
ORIGIN_windspeed_min
ORIGIN_windspeed_max
ORIGIN_precipitation_hours
ORIGIN_snowfall_sum
ORIGIN_rain_sum
ORIGIN_precipitation_sum
ORIGIN_temperature_min
DISTANCE
DESTINATION_STATE

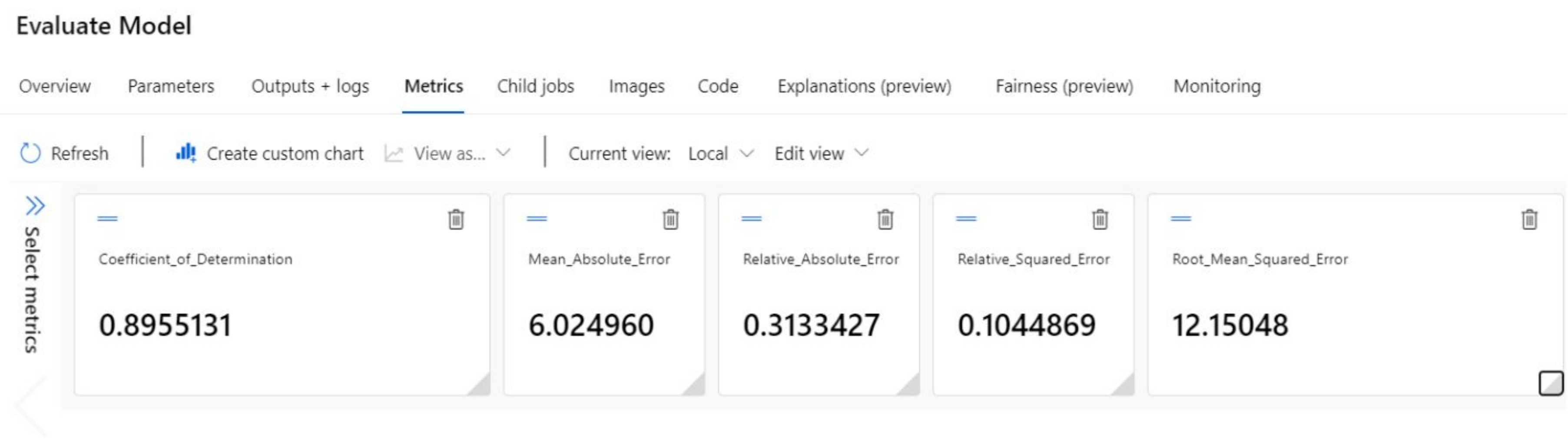


- Feature importance in descending order from the trained model (on the left)
- Model performance on trained dataset (on the right)

PERFORMANCE

1. **Mean Squared Error (MSE)**: Measures the average squared difference between predicted and actual values.
2. **Root Mean Squared Error (RMSE):** Measures the square root of MSE, giving a measure of the average absolute error.
3. **Mean Absolute Error (MAE):** Measures the average absolute difference between predicted and actual values.
4. **R-squared (R2):** Measures the proportion of variance in the target variable that can be explained by the model.

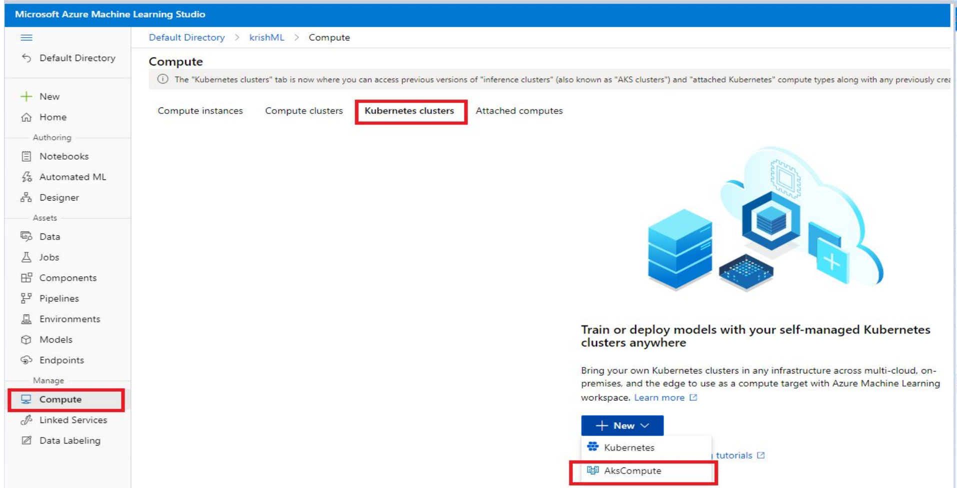
Test Data Metrics

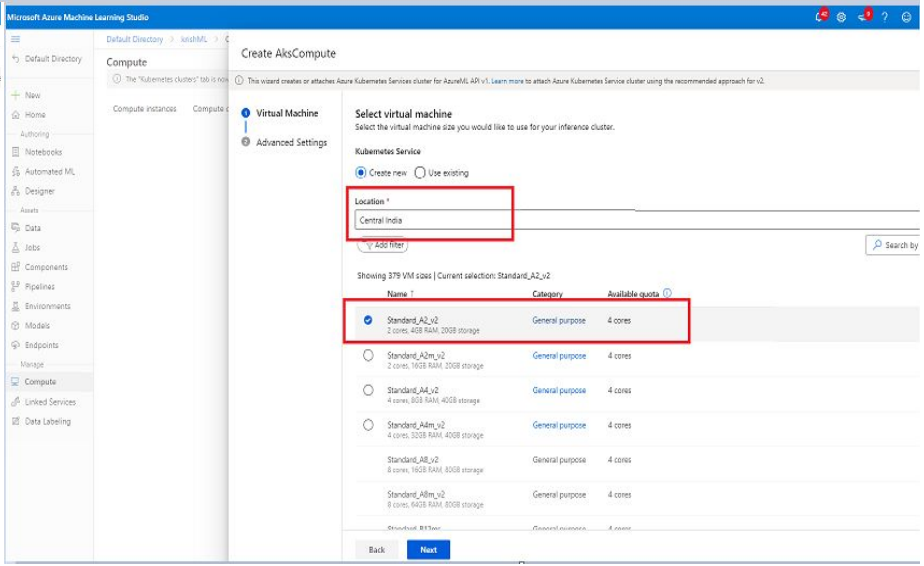


Here we have the Test Data metrics from Evaluate Model Component

Coefficient of Determination (R^2) ~ 0.89 reflects a good model output

Create Inference Cluster & Pipeline

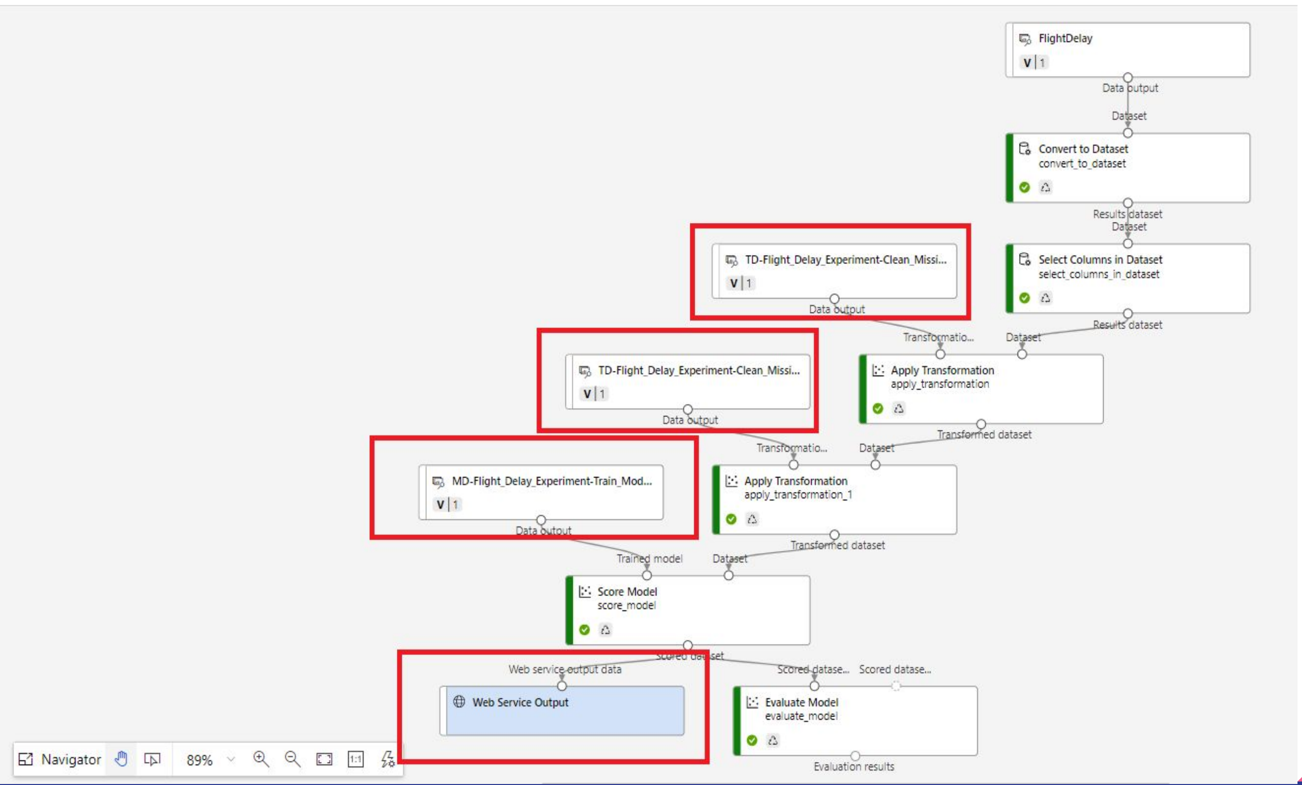




Create an Aks Compute Cluster and Configure it

This is being done to create an endpoint for our pipeline

Create Inference Cluster & Pipeline



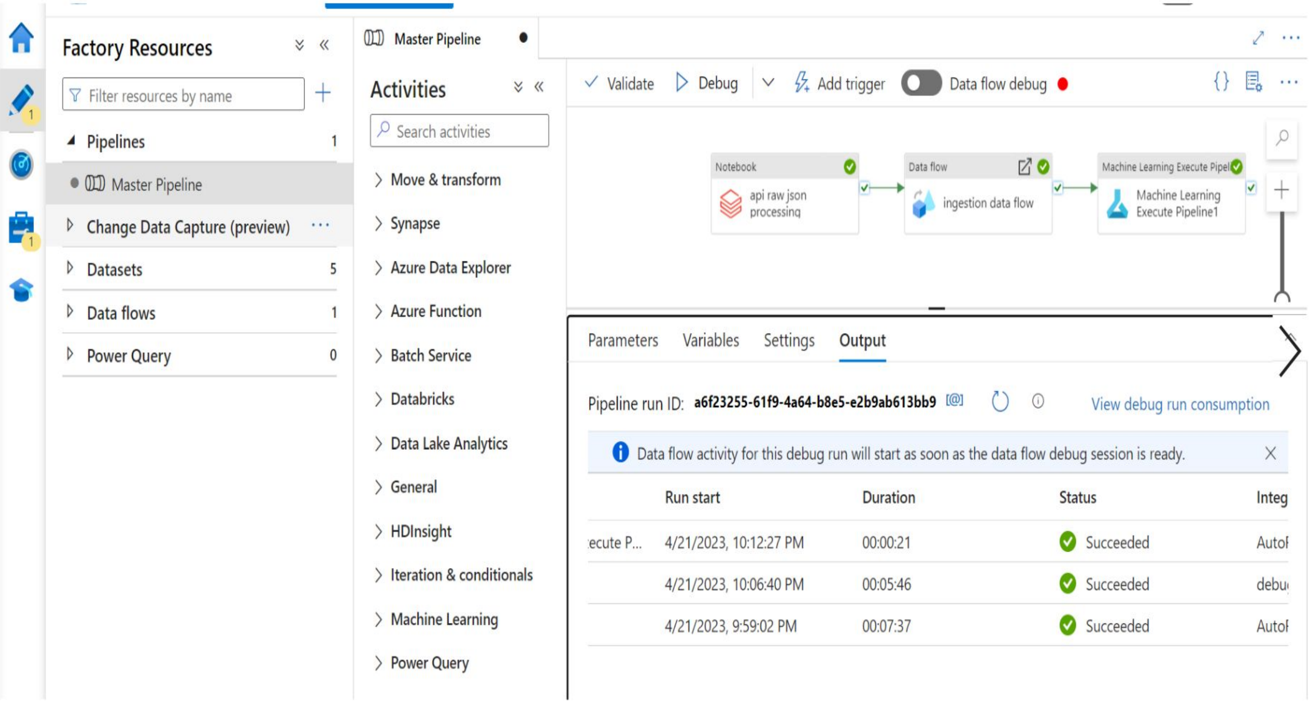
After creating an inference pipeline -

The Clean Data components along with Split Data and Model converted to transformations.

Web Service Output component auto added for REST API output which can be used externally.

Deploy it to the Aks Server for Endpoint

Master Pipeline Run



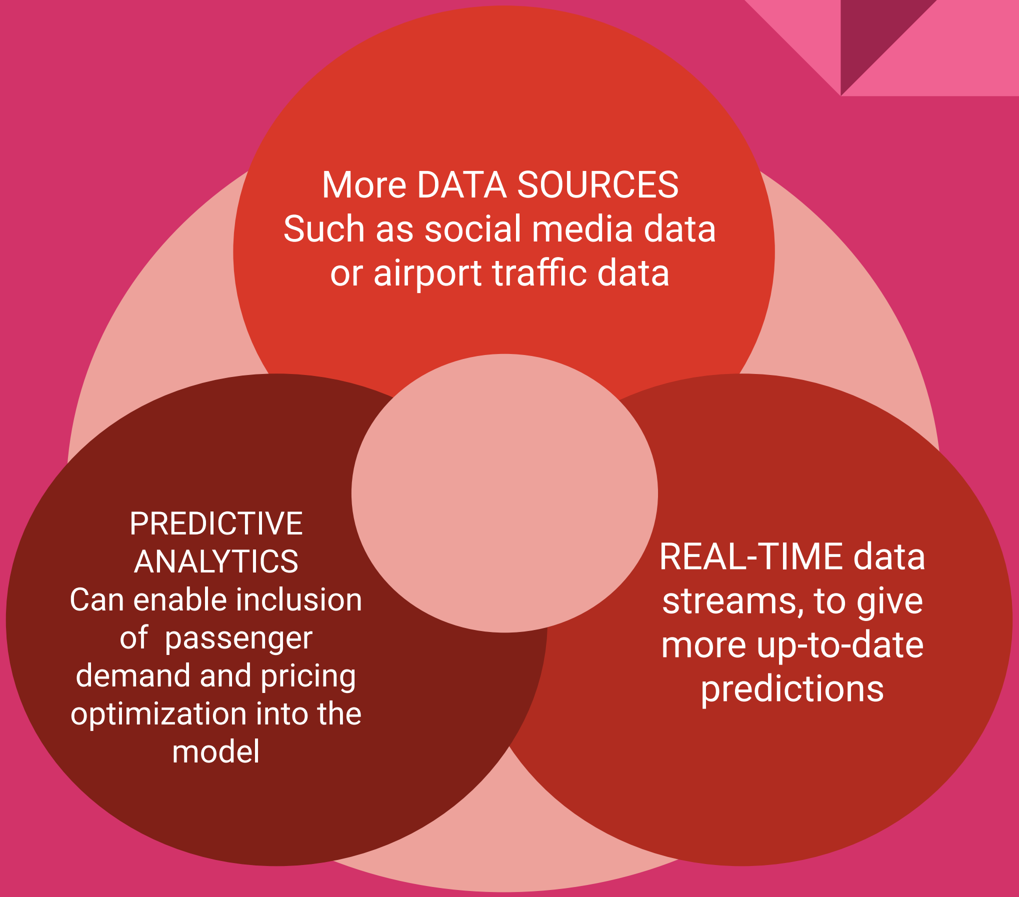
After successfully publishing our ML Studio Pipeline,

The Master ADF Pipeline was triggered to run the whole flow

IMPACT

- By analysing historical data using machine learning techniques, airlines can predict the likelihood of flight delays and take necessary measures to minimize or avoid them.
- This can lead to better operational efficiency, better management of customer expectations, and improved profitability by reducing customer dissatisfaction and operational costs for the airline industry.

Next Steps



More DATA SOURCES
Such as social media data or airport traffic data
PREDICTIVE ANALYTICS
Can enable inclusion of passenger demand and pricing optimization into the model
REAL-TIME data streams, to give more up-to-date predictions

Thank you!