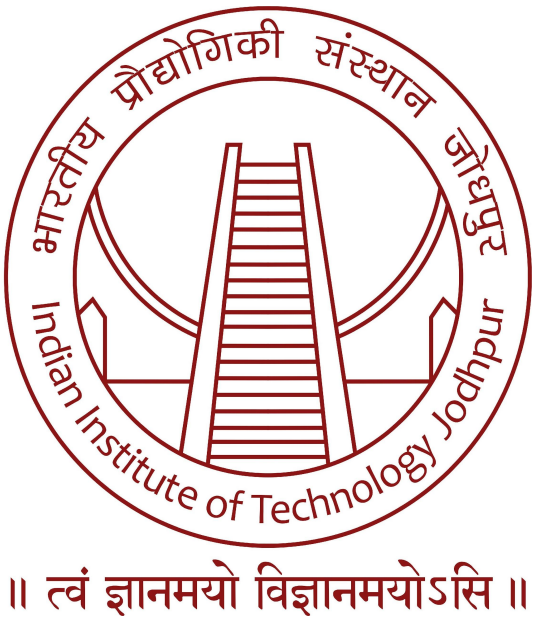
Capstone Project:

Airline Flight Delay Analysis Project Report

GROUP 2:-

Kalpit Sharma(G23AI2065)
Dhanshree Hood (G23AI2132)
Priyank Bhardwaj (G23AI2038)
Shubhankit Dutt (G23AI2070)
Nihar Dave (G23AI2097)
Ankush Jha (G23AI2007)



॥ त्वं ज्ञानमयो विज्ञानमयोऽसि ॥

Department of AIDE

Indian Institute of Technology Jodhpur

2024

1. Project Overview

1.1 Objective

The primary goal of this project is to analyze airline flight delays, identify patterns, reduce data redundancy, and build predictive models to forecast arrival delays. Additionally, the project leverages automated pipelines for data ingestion, cleaning, and visualization using tools such as AWS RDS, MySQL Workbench, Python, and Power BI.

1.2 Key Outcomes

1. An automated pipeline to load, clean, and process the flight delay data.
2. Data stored in an AWS RDS MySQL database for scalability.
3. Interactive visualizations using Power BI to extract insights.
4. Predictive modeling to estimate arrival delays and cluster flight delays into meaningful groups.
5. Feature engineering and dimensionality reduction to improve data quality for analysis.

1.3 Tools and Technologies

Component	Technology
Data Storage	Amazon RDS (MySQL)
Data Processing	Python (Pandas, NumPy)
Visualization	Power BI
Machine Learning	Scikit-Learn (PCA, Random Forest, K-Means)
Automation	Python Schedule Library

2. Data Ingestion and Exploration

2.1 Data Sources

- **Flight Delay Dataset**: A CSV file containing records of flight delays.
- **Column Definitions**: An Excel file (Download_Column_Definitions.xlsx) describing each column's purpose, datatype, and definition.

2.2 Steps to Implement Data Ingestion

Step 1: Load Data

We load both the flight delay dataset and column definitions into a Pandas DataFrame.

Python code-

```
import pandas as pd
# Load flight delay dataset
data = pd.read_csv("flight_delay_data.csv")
# Load column definitions
column_definitions = pd.read_excel("Download_Column_Definitions.xlsx")
```

Step 2: Data Exploration

1. **Check the Structure of the Data**

   - This involves checking the column names, data types, and overall size.

     Python code-

     ```
     print("Data Overview:\n", data.info())print("First Few Rows:\n",
     data.head())
     ```

2. **Check for Missing Values**

   - Identifying columns with missing values is crucial for data cleaning.

     Python code-

     ```
     print("Missing Values:\n", data.isnull().sum())
     ```

3. **Descriptive Statistics**

   - Generate basic statistical summaries for numeric features:

     Mean, Median, Standard Deviation, Minimum, Maximum, Count

Python code-

```
print("Descriptive Statistics:\n", data.describe())
```

3. Data Cleaning

3.1 Column Type Conversion

- Convert numeric fields to appropriate datatypes using the formula:

  Numeric Column=pd.to_numeric(column, errors="coerce")

  Python code-

```
data["arr_delay"] = pd.to_numeric(data["arr_delay"], errors="coerce")
data["carrier_delay"] = pd.to_numeric(data["carrier_delay"], errors="coerce")
```

3.2 Handling Missing Values

A. For Numeric Columns: Replace missing values with the mean.

New Value=Sum of Column Values / Total Count

Python code-

```
data.fillna(data.mean(), inplace=True)
```

B. For Categorical Columns: Replace missing values with the mode.

Python code-

```
for col in ["carrier", "airport"]:
   data[col].fillna(data[col].mode()[0], inplace=True)
```

3.3 Outlier Detection and Removal

1. **Z-Score Method**: Calculate Z-scores and filter out rows with $Z > 3$.

   $Z=(X-\mu) / \sigma$

   Python code-

```
from scipy.stats import zscore

z_scores = data.select_dtypes(include="number").apply(zscore)
data_clean = data[(abs(z_scores) < 3).all(axis=1)]
```

3.4 Date Standardization

Standardize all dates into a consistent YYYY-MM-DD format:

Python code-

```
data_clean["date"] = pd.to_datetime(data_clean["date"], format="%Y-%m-%d")
```

4. Exploratory Data Analysis (EDA)

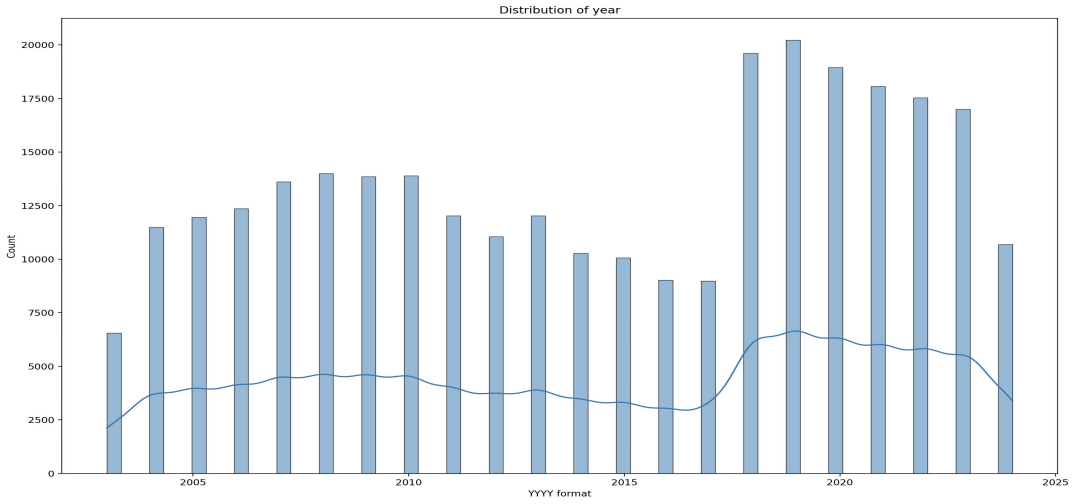
4.1 Visualizing Feature Distributions

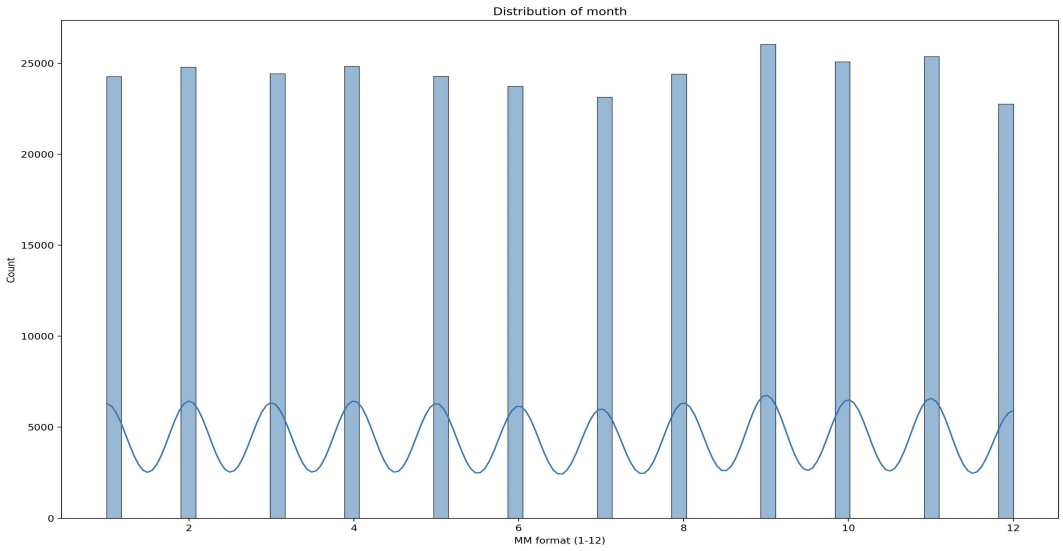
Step: Plot Histograms

Python code-

```
import matplotlib.pyplot as plt

data_clean["arr_delay"].hist(bins=30)
plt.title("Arrival Delay Distribution")
plt.xlabel("Delay (minutes)")
plt.ylabel("Frequency")
plt.show()
```





4.2 Correlation Analysis

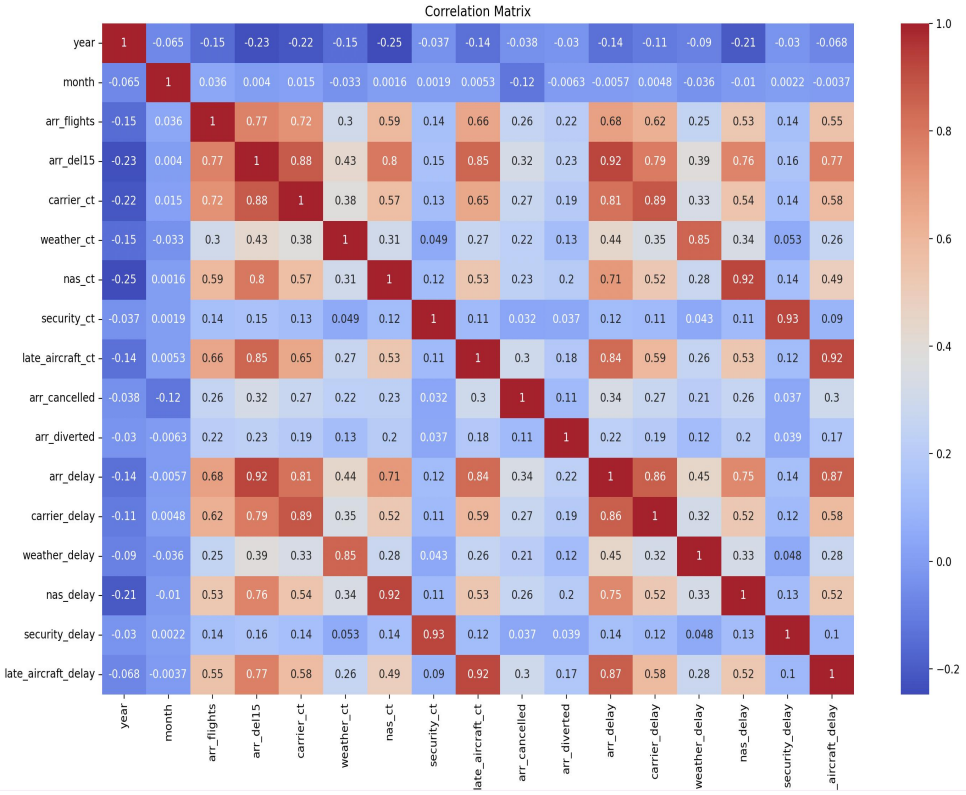
Step: Generate a Correlation Heatmap

- Correlation identifies relationships between numeric variables.

Python code-

```
import seaborn as sns

plt.figure(figsize=(12, 8))
sns.heatmap(data_clean.corr(), annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```



5. Data Storage (Amazon RDS)

5.1 MySQL Table Creation

Create a table for cleaned flight delay data in AWS RDS:

Sql code-

```
CREATE TABLE flight_delays (
   flight_number VARCHAR(10),
   arr_delay FLOAT,
   carrier_delay FLOAT,
   weather_delay FLOAT,
   flight_date DATE
);
```

5.2 Python Database Connection

Python code-

```
import mysql.connector

conn = mysql.connector.connect(
   host="rds-endpoint",
   user="username",
   password="password",
   database="flight_db"
)
cursor = conn.cursor()
# Insert Data into Tablefor _, row in data_clean.iterrows():
cursor.execute("INSERT INTO flight_delays VALUES (%s, %s, %s, %s, %s)",
tuple(row))
conn.commit()
```

6. Advanced Analysis and Modeling

6.1 Feature Engineering

Create Proportional Features

1. delay_per_flight: Arrival delay per flight.

   delay_per_flight=arr_delay / flight_count

   Python code-
   data_clean["delay_per_flight"] = data_clean["arr_delay"] /
   data_clean["flight_count"]

6.2 Dimensionality Reduction

PCA Implementation:

Python code-

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
pca_data = pca.fit_transform(data_clean.select_dtypes(include="number"))
```

6.3 Clustering and Regression

A.K-Means Clustering

Python code-

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=3)
data_clean["cluster"] = kmeans.fit_predict(pca_data)
```

B.Random Forest Regressor

Python code-

```
from sklearn.ensemble import RandomForestRegressor

X = data_clean.drop("arr_delay", axis=1)
y = data_clean["arr_delay"]

model = RandomForestRegressor()
model.fit(X, y)
predictions = model.predict(X)
```

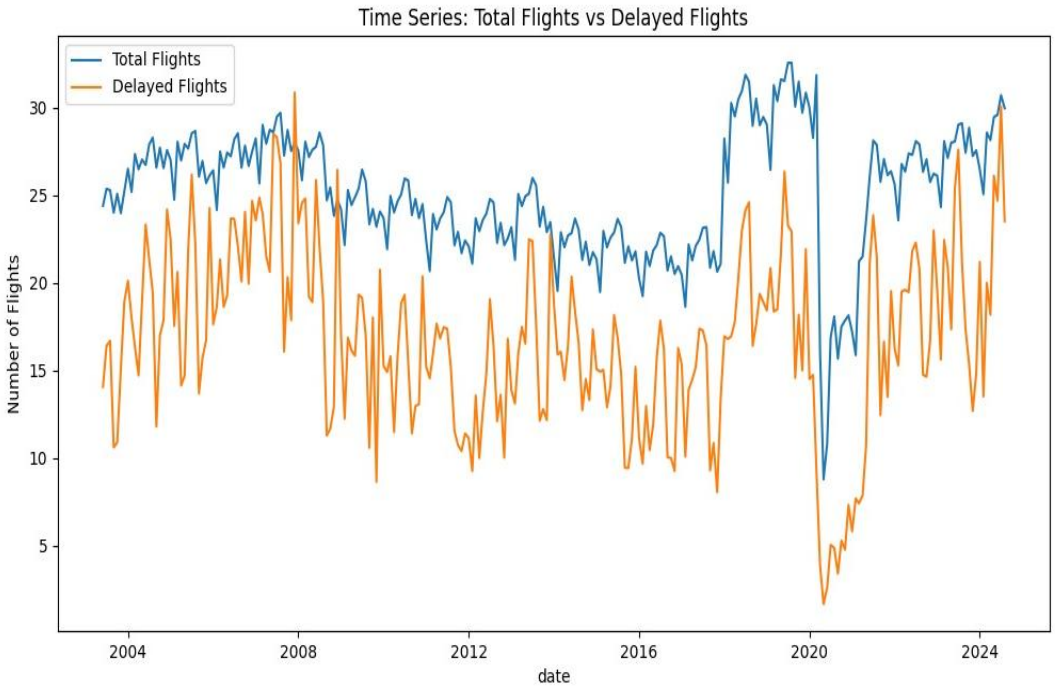
7. Data Visualization

1. **Monthly Delay Trends**: Line chart.
2. **Delay Causes by Proportion**: Bar chart for carrier_delay, weather_delay, etc.
3. **Cluster Visualizations**: Scatter plots showing grouped delays.

1. Time Series Analysis: Monthly trend: Line Chart:-

Why: Helps analyze trends over time, such as peak months for delays.

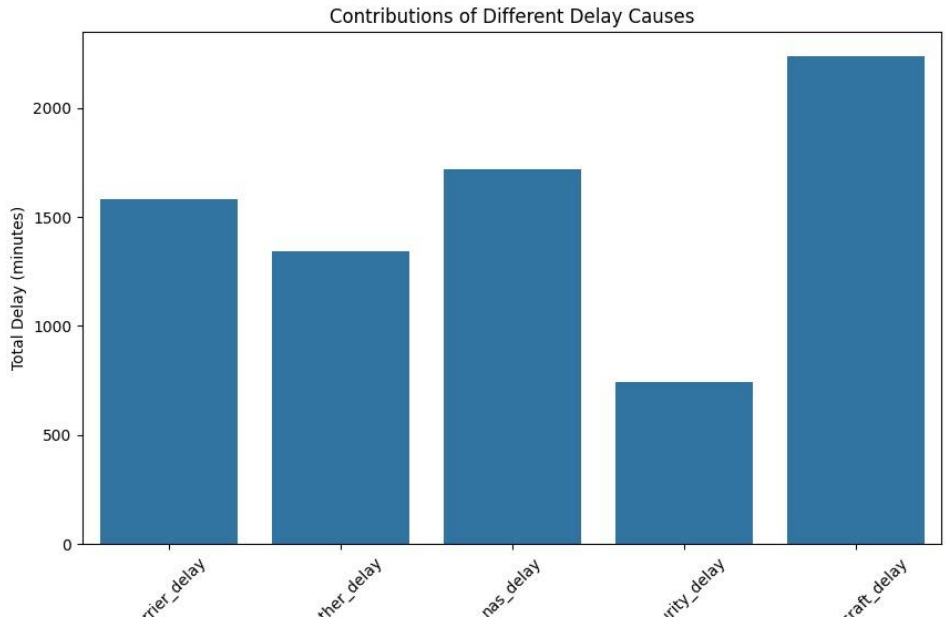
What: Compares total flights vs. delayed flights using a line plot.



2. **Contributions of Delay Causes: Delay Causes by Proportion:-**

Why: Identifies which causes (carrier, weather, NAS, security, etc.) are most responsible for delays.

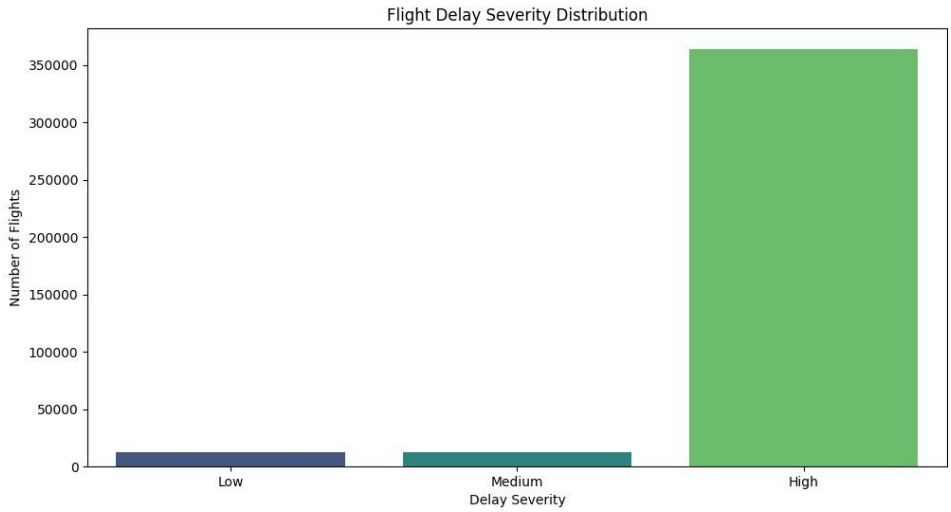
What: A bar chart showing delay contributions.



3. Flight Delay Severity:

Why: Categorizes delays into "Low," "Medium," or "High" and visualizes their proportions.

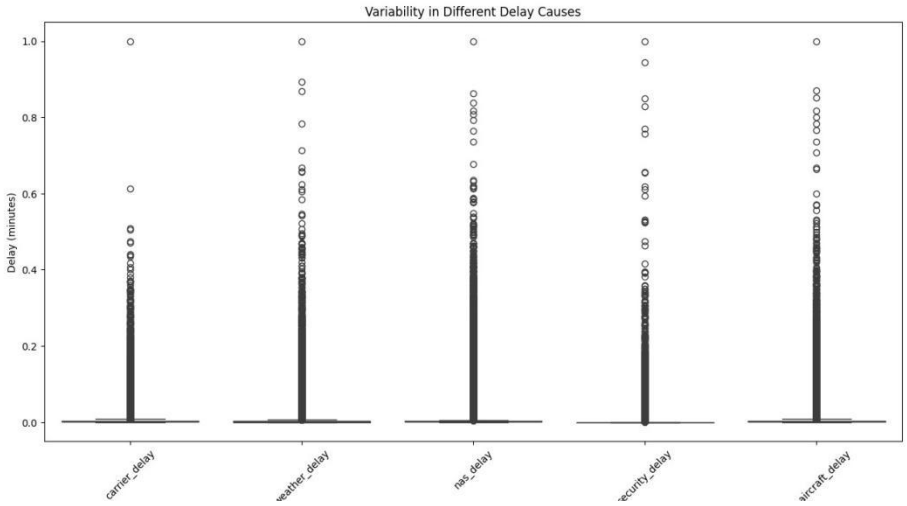
What: A count plot or pie chart.



4. **Boxplots for Delay Metrics:**

Why: Examines variability and outliers in delay causes.

What: Boxplots for carrier, weather, NAS, and other delay types.



Data Visualization with Microsoft Power BI:

1. Column Chart

Type: Bar Chart (Vertical Column Chart)

Purpose:

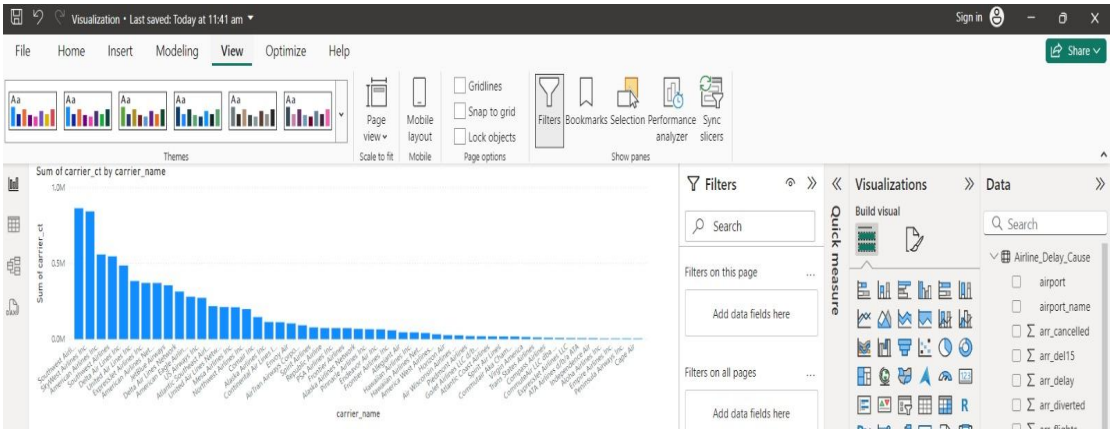
This chart visualizes the sum of carrier_ct (carrier delay count) for each carrier_name.

The X-axis represents carrier names (airline names), while the Y-axis represents the sum of carrier delay counts.

Key Observations:

The bars are sorted in descending order, where airlines with the highest carrier delays appear first.

The bar heights indicate the magnitude of carrier delays for each airline.



2. Treemap

Type: Treemap

Purpose:

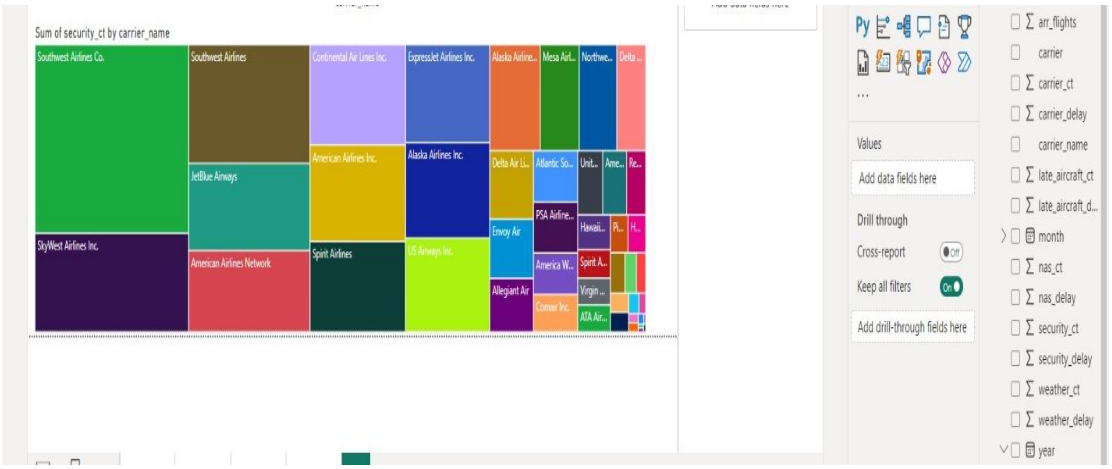
The treemap visualizes the sum of security_ct (security-related delay count) for each carrier_name.

Each rectangle represents a carrier (airline), and the size of the rectangle corresponds to the total security delays for that carrier.

Key Observations:

The largest rectangles represent carriers with the highest security-related delays.

Each rectangle is color-coded, making it easy to differentiate between carriers.



3. Horizontal Bar Chart

Type: Bar Chart (Horizontal Bars)

Purpose:

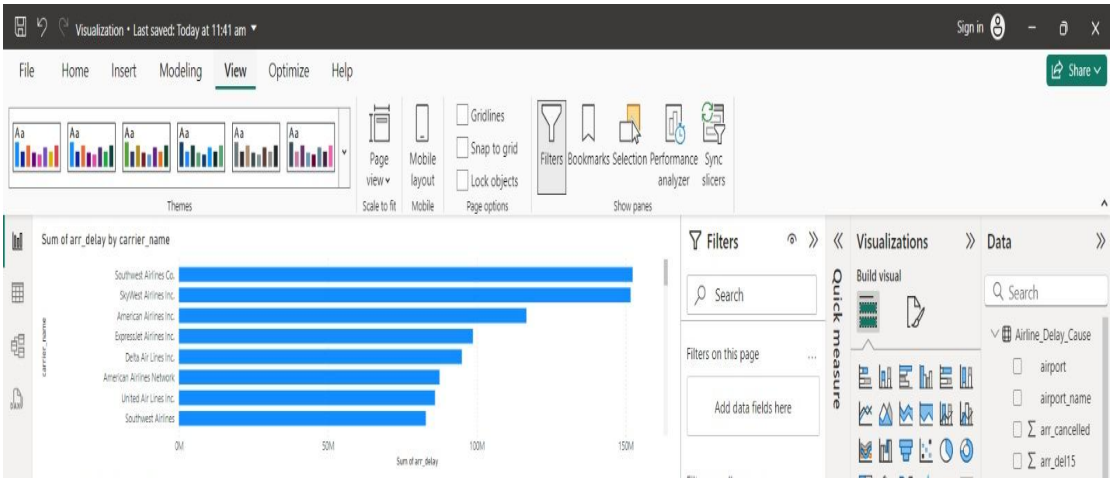
This chart visualizes the sum of arr_delay (arrival delays) for each carrier_name (airline name).

The X-axis shows the sum of arrival delays, while the Y-axis lists the airlines (carriers).

Key Observations:

The horizontal bars are sorted in descending order.

Airlines like Southwest Airlines Co. and SkyWest Airlines Inc. have the highest total arrival delays.



4. Line Chart

Type: Line Chart

Purpose:

This line chart visualizes the sum of arr_delay (arrival delays) for different airport_name (airport names).

The X-axis represents the airport names, and the Y-axis shows the total arrival delay.

Key Observations:

The line starts high on the left side, indicating that airports such as Chicago O'Hare and Atlanta have the highest delays.

As you move right, the delays decrease gradually for smaller airports.

5.Bar Chart:

X-Axis (Airport Names):

The x-axis represents different airport names, likely ordered by the sum of weather-related delays.

The airports with the highest delays are on the left, while those with lower delays are on the right.

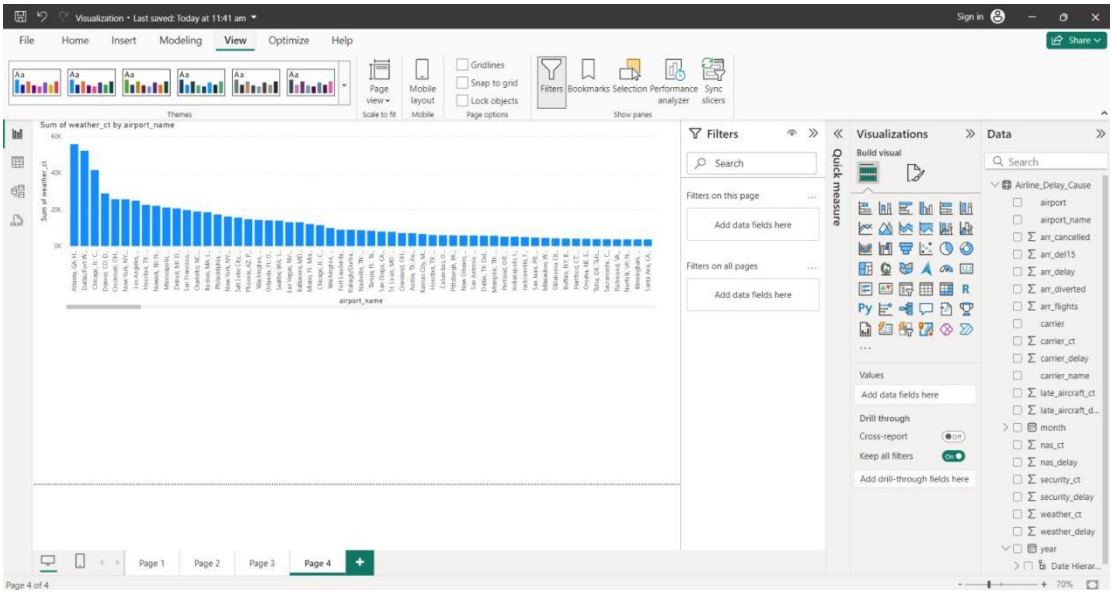
Y-Axis (Sum of Weather_ct):

The y-axis displays the total count of weather-related delays (weather_ct) for each airport.

The unit on the axis seems to be the number of delay events caused by weather, summing up to nearly 60K for the highest bar.

Top Airports with Weather Delays:

Airports like Atlanta (ATL) and Chicago (ORD) are likely the ones with the most weather-related delays, judging by the bar length.


Visualization • Last saved: Today at 11:41 am
Sign in
File
Home
Insert
Modeling
View
Optimize
Help
Share
Themes
Page view
Scale to fit
Mobile layout
Mobile
Gridlines
Snap to grid
Lock objects
Page options
Filters
Bookmarks
Selection
Performance analyzer
Sync slicers
Show panes
Sum of weather_ct by airport_name
Sum of weather_ct
airport_name
Filters
Search
Filters on this page
Add data fields here
Filters on all pages
Add data fields here
Quick measure
Visualizations
Build visual
Values
Add data fields here
Drill through
Cross-report
Keep all filters
Add drill-through fields here
Data
Search
Airline_Delay_Cause
airport
airport_name
arr_cancelled
arr_del15
arr_delay
arr_diverted
arr_flights
carrier
carrier_ct
carrier_delay
carrier_name
late_aircraft_ct
late_aircraft_d...
month
nas_ct
nas_delay
security_ct
security_delay
weather_ct
weather_delay
year
Date Hierar...
Page 1
Page 2
Page 3
Page 4
Page 4 of 4
70%

8. Conclusion

The project successfully implemented a comprehensive pipeline to analyze airline flight delay data, integrating data ingestion, cleaning, transformation, storage, visualization, and advanced analytics.

Data was ingested from structured CSV and Excel files, with column definitions aiding in proper interpretation of the dataset's schema. Cleaning processes resolved missing values, handled outliers using statistical methods, and standardized date formats, ensuring high-quality and consistent data for analysis.

Exploratory Data Analysis (EDA) revealed patterns in flight delays, correlations between variables, and distribution anomalies, providing insights into potential root causes of delays. Delays were primarily categorized into causes like weather, carrier, and security delays, allowing for targeted analysis of each contributing factor.

The use of AWS RDS for optimized data storage ensured scalability, performance, and ease of querying for large datasets. Automation techniques were employed to enable periodic data ingestion and processing, reducing manual intervention and enhancing efficiency.

Feature engineering created derived metrics such as delay_per_flight and aggregated delay causes, enriching the dataset for predictive modeling. Principal Component Analysis (PCA) helped reduce data dimensionality, preserving key patterns while improving computational efficiency.

Predictive modeling using Random Forest Regressor provided reliable predictions for arrival delays, with metrics like Mean Squared Error (MSE) guiding performance evaluation.

Clustering using K-Means uncovered groups of airports with similar delay patterns, enabling targeted interventions for high-delay airports. Visualizations created in Power BI, such as bar charts and heatmaps, allowed stakeholders to easily interpret and act on insights.

One key insight from visualizations showed that specific airports, such as Albany G.H.L. and Charlotte, experience disproportionately high weather-related delays. Temporal trends indicated seasonal variations in flight delays, with certain months showing significantly higher delay rates.

Fact and dimension tables in the data warehouse facilitated efficient reporting and querying, supporting business intelligence needs.

Data governance practices, including validation checks, encryption, and role-based access, ensured data integrity, security, and compliance with regulations.

Hyperparameter tuning and cross-validation enhanced the accuracy and generalizability of the predictive models used in the project.

Performance optimization techniques like indexing and batching ensured smooth handling of large datasets and complex queries. The project demonstrated the ability to analyze and visualize large-scale flight delay data effectively, providing actionable insights for airlines and airports.

By integrating modern tools such as AWS RDS, MySQL Workbench, and Power BI, the project established a robust framework for ongoing flight delay analysis and future scalability.