

Airline Data Management and Analysis Using Power BI

Submitted By: - Aman Kumar Yadav

Course: - Data Science Placement Guarantee Course

Submission Date: - 13th July 2025

Table of Contents: -

Introduction.....	3
Data Preparation and Cleaning	4
Data Modelling	8
Enhanced Data Insights	9
DAX Calculations.....	10
Visualisation and Interactive features	12
Final Dashboard and Power BI Service	16

Introduction

The airline industry generates and relies heavily on large volumes of data, including flight schedules, passenger bookings, ticket status, and operational metrics. Managing and analysing this data effectively can significantly improve decision-making, resource allocation, and customer satisfaction.

In this project, we focus on analysing airline data using Power BI. The goal is to visualize operational patterns and uncover key insights by cleaning, transforming, and modelling datasets containing information about flights, passengers, and tickets.

By creating an interactive and dynamic Power BI dashboard, we aim to help airline operations teams understand trends, monitor performance, and identify areas of improvement.

Objectives of the Project:

- Analyse flight, passenger, and ticket data
- Clean and transform raw data using Power Query
- Build a data model with appropriate relationship
- Create interactive reports and dashboards
- Generate insights for operational improvements

Tools Used:

- Microsoft Power BI
- Power Query Editor
- DAX (Data analysis Expressions)

Data Preparation and Cleaning

In this step, I imported all three datasets: - Flight_Information, Ticket_Information, and Passenger_Information – into Power BI using Power Query Editor.

The Data preparation involved the following actions: -

- **Removed duplicate records** to ensure data integrity.
- **Eliminated blank rows** that could affect data modelling
- **Promoted headers** to convert the first row into column names.
- **Checked column validity** to confirm all values are in proper format (e.g., FlightID is numeric, FlightNumber is text).
- **Previewed data profiling** to ensure completeness and consistency

The following Screenshots shows the cleaned datasets in the Power Query Editor after all transformations were applied: -

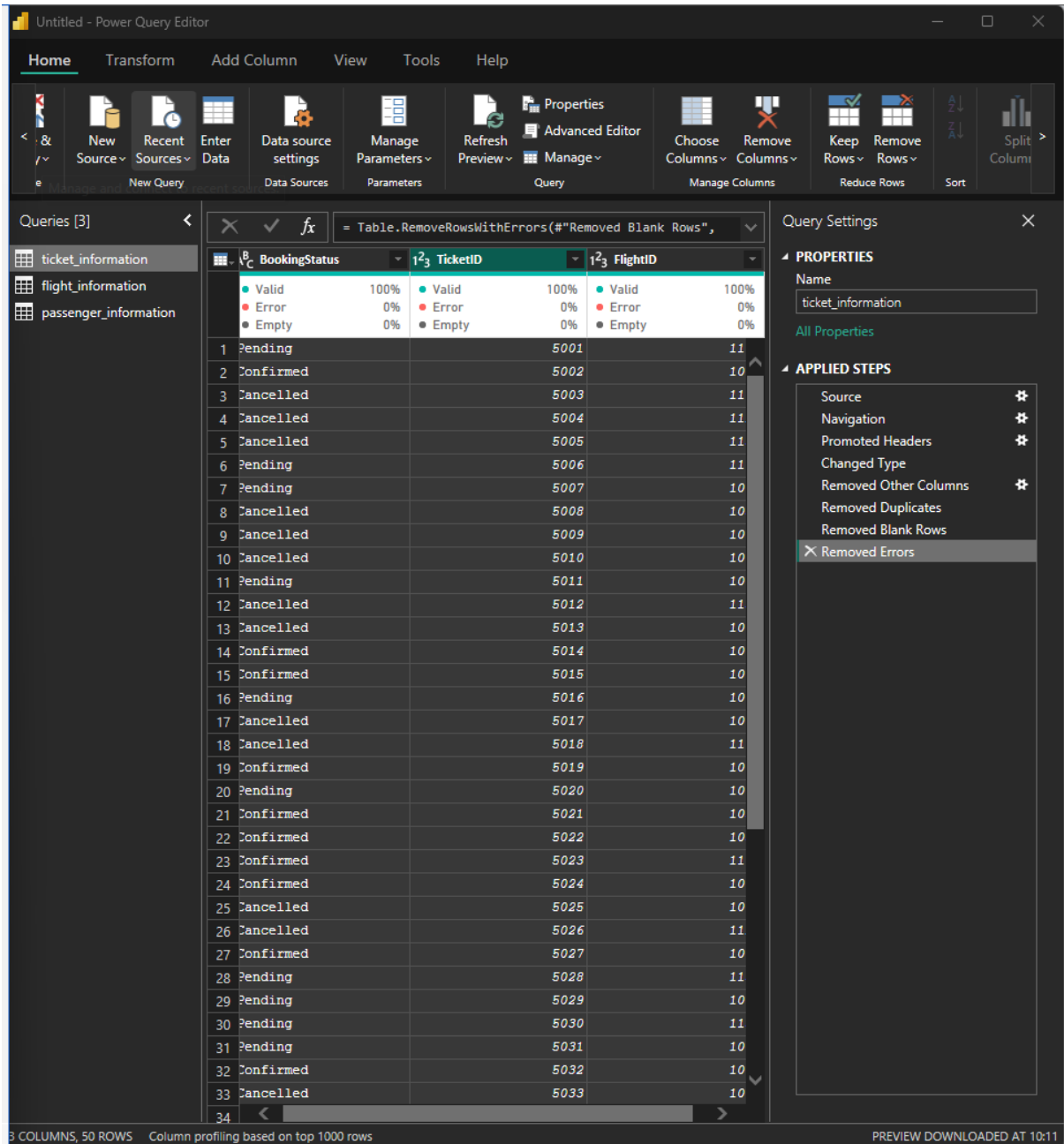


Figure 1: Cleaned Ticket_Information Dataset in Power Query

Untitled - Power Query Editor

Home Transform Add Column View Tools Help

New Source Recent Sources Enter Data Data source settings Manage Parameters Refresh Preview Advanced Editor Properties Choose Columns Remove Columns Keep Rows Remove Rows Sort Split Column

Queries [3]

- ticket_information
- flight_information
- passenger_information

fx = Table.RemoveRowsWithErrors(#"Removed Blank Rows",

	FlightID	FlightNumber	Airline
1	1001	FL1102	Airline D
2	1002	FL1435	Airline B
3	1003	FL1860	Airline A
4	1004	FL1270	Airline C
5	1005	FL1106	Airline C
6	1006	FL1071	Airline A
7	1007	FL1700	Airline C
8	1008	FL1020	Airline C
9	1009	FL1614	Airline A
10	1010	FL1121	Airline D
11	1011	FL1466	Airline A
12	1012	FL1214	Airline D
13	1013	FL1330	Airline C
14	1014	FL1458	Airline C
15	1015	FL1087	Airline C
16	1016	FL1372	Airline B
17	1017	FL1099	Airline D
18	1018	FL1871	Airline B
19	1019	FL1663	Airline B
20	1020	FL1130	Airline A
21	1021	FL1661	Airline B
22	1022	FL1308	Airline A
23	1023	FL1769	Airline A
24	1024	FL1343	Airline B
25	1025	FL1491	Airline D
26	1026	FL1413	Airline D
27	1027	FL1805	Airline D
28	1028	FL1385	Airline D
29	1029	FL1191	Airline D
30	1030	FL1955	Airline B
31	1031	FL1276	Airline B
32	1032	FL1160	Airline C
33	1033	FL1459	Airline D

Query Settings

PROPERTIES

Name: flight_information

APPLIED STEPS

- Source
- Navigation
- Promoted Headers
- Changed Type
- Removed Other Columns
- Removed Duplicates
- Removed Blank Rows
- Removed Errors

5 COLUMNS, 200 ROWS Column profiling based on top 1000 rows PREVIEW DOWNLOADED AT 10:10

Figure 2: - Cleaned Flight_Information Dataset in Power Query

Untitled - Power Query Editor

Home Transform Add Column View Tools Help

Close & Apply Close New Source New Query Recent Sources Enter Data Data source settings Data Sources Manage Parameters Parameters Refresh Preview Query Properties Advanced Editor Manage Choose Columns Manage Columns Remove Columns Remove Rows Reduce Rows Keep Rows Remove Rows Sort Split Column

Queries [3]

- ticket_information
- flight_information
- passenger_information

fx = Table.RemoveRowsWithErrors(#"Removed Blank Rows",

	123 FlightID	123 PassengerID	A ^B _C SeatNumber
	Valid 100%	Valid 100%	Valid 100%
	Error 0%	Error 0%	Error 0%
	Empty 0%	Empty 0%	Empty 0%
1	1161	1	38A
2	1157	2	24D
3	1141	3	30B
4	1046	4	17E
5	1035	5	29D
6	1134	6	10A
7	1082	7	10A
8	1115	8	20E
9	1197	9	34E
10	1047	10	2E
11	1153	11	43C
12	1194	12	48C
13	1010	13	47A
14	1056	14	23C
15	1030	15	16D
16	1109	16	40D
17	1005	17	25C
18	1119	18	32C
19	1033	19	27E
20	1118	20	32B
21	1065	21	19E
22	1146	22	5B
23	1177	23	28B
24	1011	24	22E
25	1085	25	6A
26	1026	26	5A
27	1063	27	12B
28	1086	28	46B
29	1059	29	49B
30	1027	30	45C
31	1177	31	9B
32	1161	32	47A
33	1098	33	22C
34			

Query Settings

PROPERTIES

Name

passenger_information

APPLIED STEPS

- Source
- Navigation
- Promoted Headers
- Changed Type
- Removed Other Columns
- Removed Duplicates
- Removed Blank Rows
- Removed Errors

3 COLUMNS, 100 ROWS Column profiling based on top 1000 rows PREVIEW DOWNLOADED AT 10:11

Figure 3: - Cleaned Passenger_Information dataset in Power Query

Data Modelling

After cleaning the data, I moved to the Model View in Power BI to establish relationships among the three datasets: -

1. Flight_Information
2. Passenger_Information
3. Ticket_Infomation

Relationships Created: -

- **Flight_id** is the **Primary key** in the **flight_Information** Table.
- **Passenger_Infomation** and **Ticket_Information** tables both use **FlightID** as a **Foreign key**.
- These relationships follow a one-to-many cardinality, where one flight can have many passengers and tickets.

Cross-filter Direction: s-

- I kept the default single filter direction to maintain a simpler model.
- No circular dependencies were present.

The following Screenshot shows the data model with relationships:-

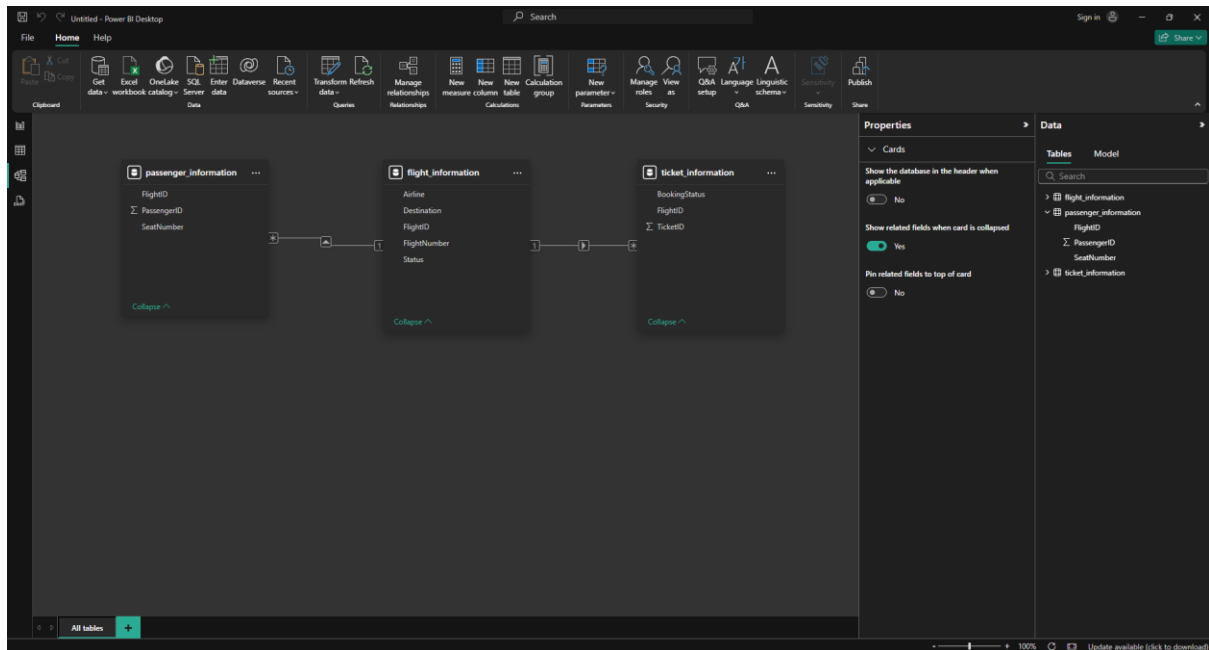


Figure 4: - Data Model Showing Relationships using FlightID

Enhanced Data Insights

To gain better insights from the flight data, I enhanced the dataset with two new columns using Power Query Editor.

1. Conditional Column: - FlightStatusLabel

I added a new column **FlightStatusLabel** in the **Flight_Information** table using the Conditional Column feature. The column classifies flights as:

- “Best” – if the flight status is On Time.
- “To Be Improved” – if the flight status is Delayed or Cancelled

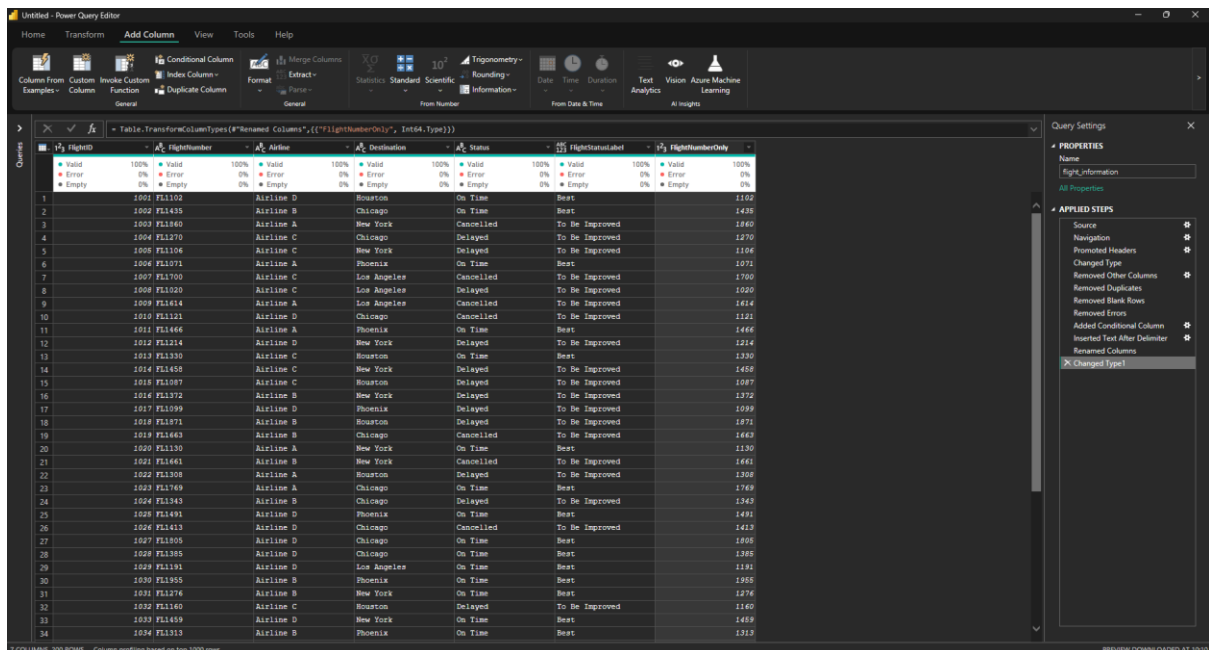
This allows better filtering and visualization of flight performance.

2. Extracted Flight Number: -

Using the “Column from Examples”, I created a new column **FlightNumberOnly** by extracting the numeric part of the FlightNumber field. For Example:-

- From FL1102 → Extracted 1102

This helps in sorting, filtering, or grouping flights based on numeric Flight IDs.



Flight	FlightNumber	Airline	Destination	Status	FlightStatusLabel	FlightNumberOnly
1	1001 FL1102	Airline D	Houston	On Time	Best	1102
2	1002 FL1450	Airline B	Chicago	On Time	Best	1450
3	1003 FL1060	Airline A	New York	Cancelled	To Be Improved	1060
4	1004 FL1270	Airline C	Chicago	Delayed	To Be Improved	1270
5	1005 FL1106	Airline C	New York	Delayed	To Be Improved	1106
6	1006 FL1071	Airline A	Phoenix	On Time	Best	1071
7	1007 FL1700	Airline C	Los Angeles	Cancelled	To Be Improved	1700
8	1008 FL1020	Airline C	Los Angeles	Delayed	To Be Improved	1020
9	1009 FL1614	Airline A	Los Angeles	Cancelled	To Be Improved	1614
10	1010 FL1121	Airline D	Chicago	Cancelled	To Be Improved	1121
11	1011 FL1466	Airline A	Phoenix	On Time	Best	1466
12	1012 FL1214	Airline D	New York	Delayed	To Be Improved	1214
13	1013 FL1350	Airline C	Houston	On Time	Best	1350
14	1014 FL1450	Airline C	New York	Delayed	To Be Improved	1450
15	1015 FL1087	Airline C	Houston	Delayed	To Be Improved	1087
16	1016 FL1372	Airline B	New York	Delayed	To Be Improved	1372
17	1017 FL1099	Airline D	Phoenix	Delayed	To Be Improved	1099
18	1018 FL1871	Airline B	Houston	Delayed	To Be Improved	1871
19	1019 FL1463	Airline B	Chicago	Cancelled	To Be Improved	1463
20	1020 FL1350	Airline A	New York	On Time	Best	1350
21	1021 FL1461	Airline B	New York	Cancelled	To Be Improved	1461
22	1022 FL1308	Airline A	Houston	Delayed	To Be Improved	1308
23	1023 FL1769	Airline A	Chicago	On Time	Best	1769
24	1024 FL1343	Airline B	Chicago	Delayed	To Be Improved	1343
25	1025 FL1401	Airline D	Phoenix	On Time	Best	1401
26	1026 FL1413	Airline D	Chicago	Cancelled	To Be Improved	1413
27	1027 FL1805	Airline D	Chicago	On Time	Best	1805
28	1028 FL1385	Airline D	Chicago	On Time	Best	1385
29	1029 FL1191	Airline D	Los Angeles	On Time	Best	1191
30	1030 FL1855	Airline B	Phoenix	On Time	Best	1855
31	1031 FL1276	Airline B	New York	On Time	Best	1276
32	1032 FL1160	Airline C	Houston	Delayed	To Be Improved	1160
33	1033 FL1459	Airline D	New York	On Time	Best	1459
34	1034 FL1313	Airline B	Phoenix	On Time	Best	1313

Figure 5: - Enhanced Flight Information with Conditional Column and Extracted Flight Number.

DAX Calculations

1. Total Passengers for a Specific Flight: -

To analyse the number of passengers on a particular flight, I created a DAX measure using the CALCULATE () and COUNTROWS () functions.

The goal was to calculate the total number of passengers who booked Flight FL1102 which corresponds to FlightID = 1001 in the dataset.

```
1 TotalPassengers_FL1102 = CALCULATE(  
2     COUNTROWS(passenger_information),  
3     passenger_information[FlightID] = 1001  
4 )  
5
```

Figure 6: - DAX Measure Calculating Total Passengers for FL1102

2. Total Confirmed Tickets: -

I created a DAX measure to count only the tickets with a BookingStatus marked as “Confirmed”, to understand the actual number of passengers who successfully booked flights.

```
Total Tickets Booked = CALCULATE(COUNTROWS(ticket_information),  
ticket_information[BookingStatus] = "Confirmed")
```

Figure 7: - DAX Measure for Total Confirmed Tickets

3. Filtered Table for Best Flights Only: -

To focus on well-performing flights, I created a new calculated table using DAX that filters the flight_information table for rows labelled “Best” in the FlightStatusLabel column (which was created in Step 3).

```
1 BestFlightsOnly =  
2 SELECTCOLUMNS(  
3     FILTER(  
4         flight_information,  
5         flight_information[FlightStatusLabel] = "Best"  
6     ),  
7     "FlightID", flight_information[FlightID],  
8     "FlightNumber", flight_information[FlightNumber],  
9     "Airline", flight_information[Airline],  
10    "Status", flight_information[Status],  
11    "FlightStatusLabel", flight_information[FlightStatusLabel]  
12 )  
13
```

Figure 8: - Best Flights Table Filtered using DAX

Visualisation and Interactive features

1. Passenger Count by Airline: -

A bar Chart was created to show the total number of passengers served by each airline. This Chart uses the Airline column on the axis and count of passengers using the TotalPassengers measure or row count.

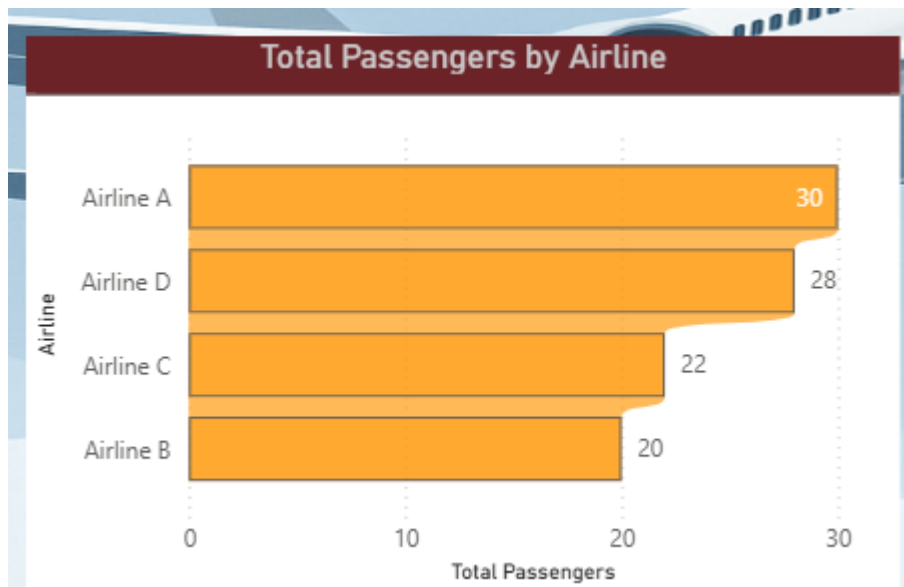


Figure 9: - Bar chart → Passengers Count by Airline

2. Ticket Booking Statuses: →

A donut chart was used to represent the distribution of different booking statuses (e.g., Confirmed, Cancelled). This gives a quick view of booking outcomes.

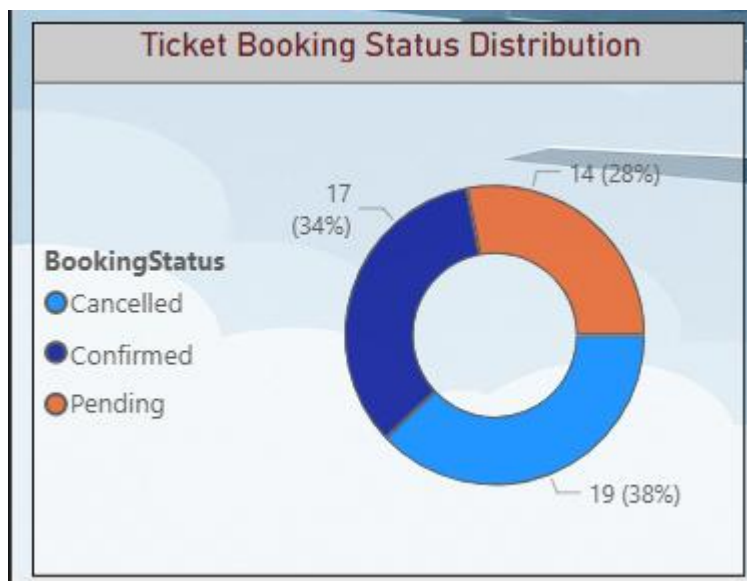


Figure 10:- Donut Chart → Ticket Booking status Distribution

3. Flights By Airline and Destination: →

A matrix was added to analyse how many flights each airline operates to various destinations.

- Rows: Destinations
- Columns: Airline
- Values: count of FlightID

Flights By Airline and Destination						
Destination	Airline A	Airline B	Airline C	Airline D	Total	
Chicago	✈️ 8	✈️ 5	✈️ 5	✈️ 15	33	
Houston	✈️ 14	✈️ 6	✈️ 14	✈️ 9	43	
Los Angeles	✈️ 7	✈️ 9	✈️ 10	✈️ 16	42	
New York	✈️ 9	✈️ 10	✈️ 13	✈️ 8	40	
Phoenix	✈️ 10	✈️ 11	✈️ 7	✈️ 14	42	
Total	48	41	49	62	200	

Figure 11:- Matrix → Flights by Airline and Destinations

4. Slicers Used: →

- Airline Slicer: - Allows filtering all visuals to show data only for the selected airlines.
- Destination Slicer: - Filters Visuals to display flights and tickets related to specific destinations.

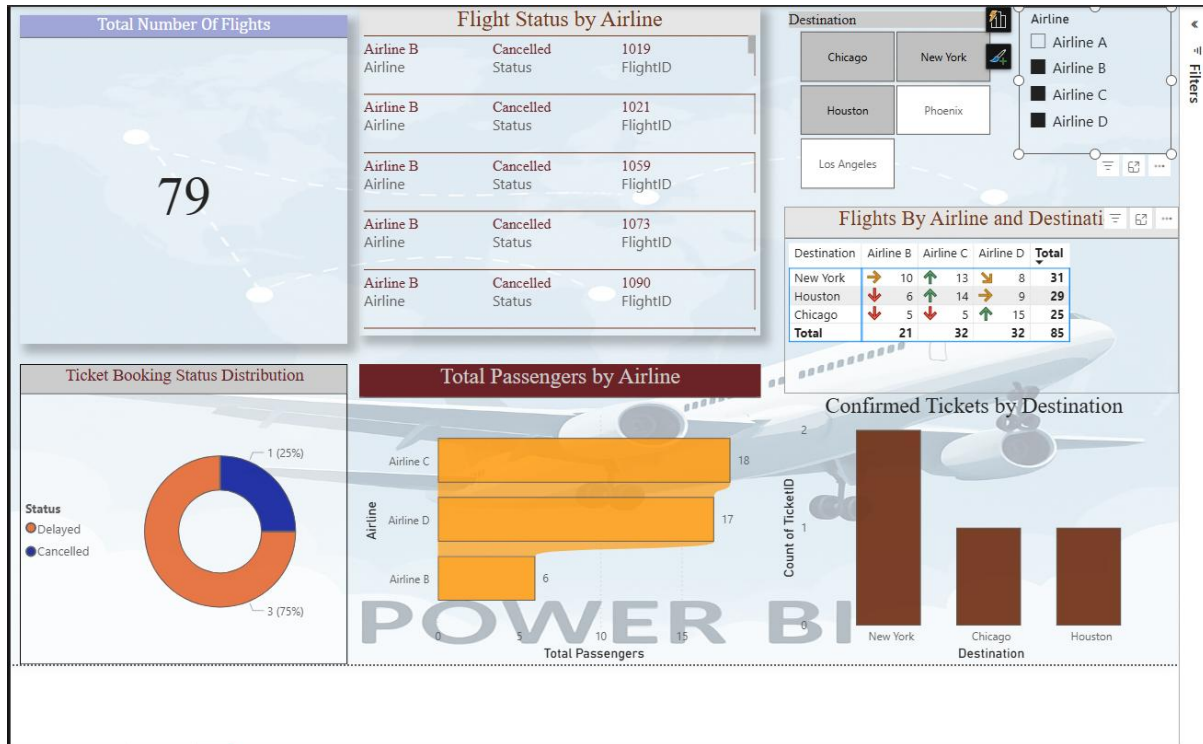


Figure 12: → Interactive Slicers Filtering ALL Dashboard Visuals

5. Confirmed Tickets by Destination: →

To analyse demand across different travel routes, I created a clustered column chart that display the number of confirmed tickets for each destination.

This visual helps identify: -

- Which destinations are most popular
- Where the highest booking volume occurs
- Pattern in customer travel behavior



Figure 13: → Confirmed Tickets by Destinations

Final Dashboard and Power BI Service

The final version of the Power BI dashboard was published to the Power BI Service for online access and sharing. Key deployment steps included:

1. Publishsing:-

The .pbix file was uploaded to Power BI Services via My Workspace.

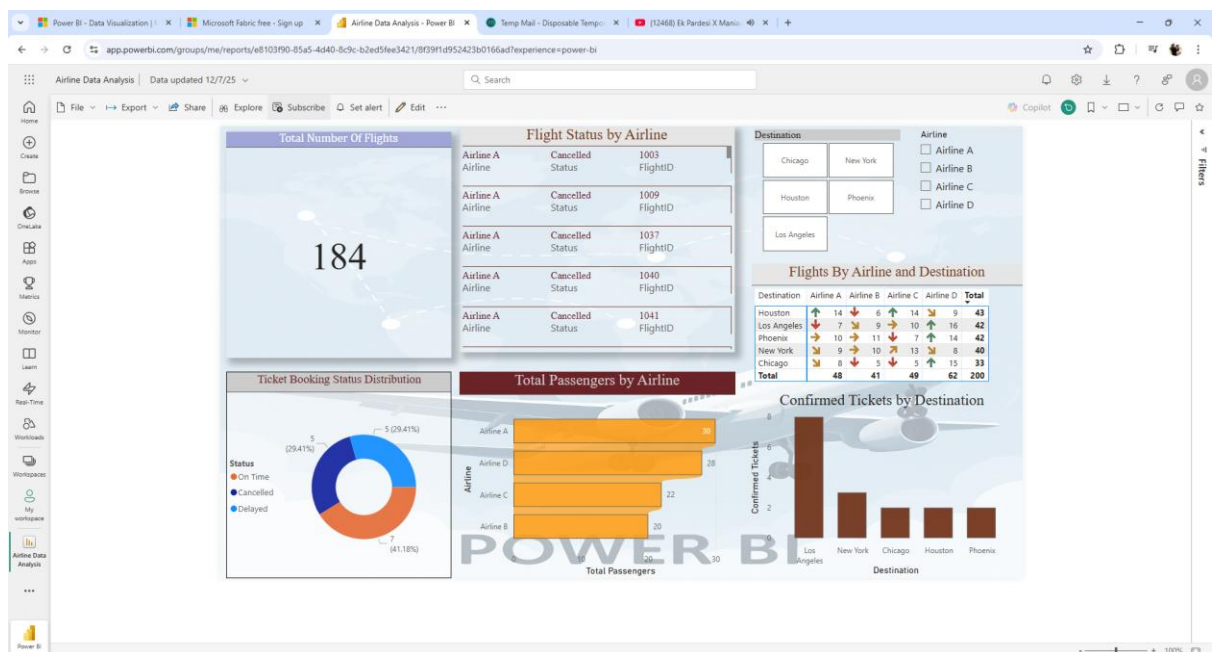


Figure 14: → Published Dashboard on Power BI Service

2. Row-Level Security (RLS):→

To Restrict data visibility for specific users, a Row-Level Security (RLS) role named AirlineARole was created.

This role limits access to records where the Airline column equals "Airline A" using the filter expression.

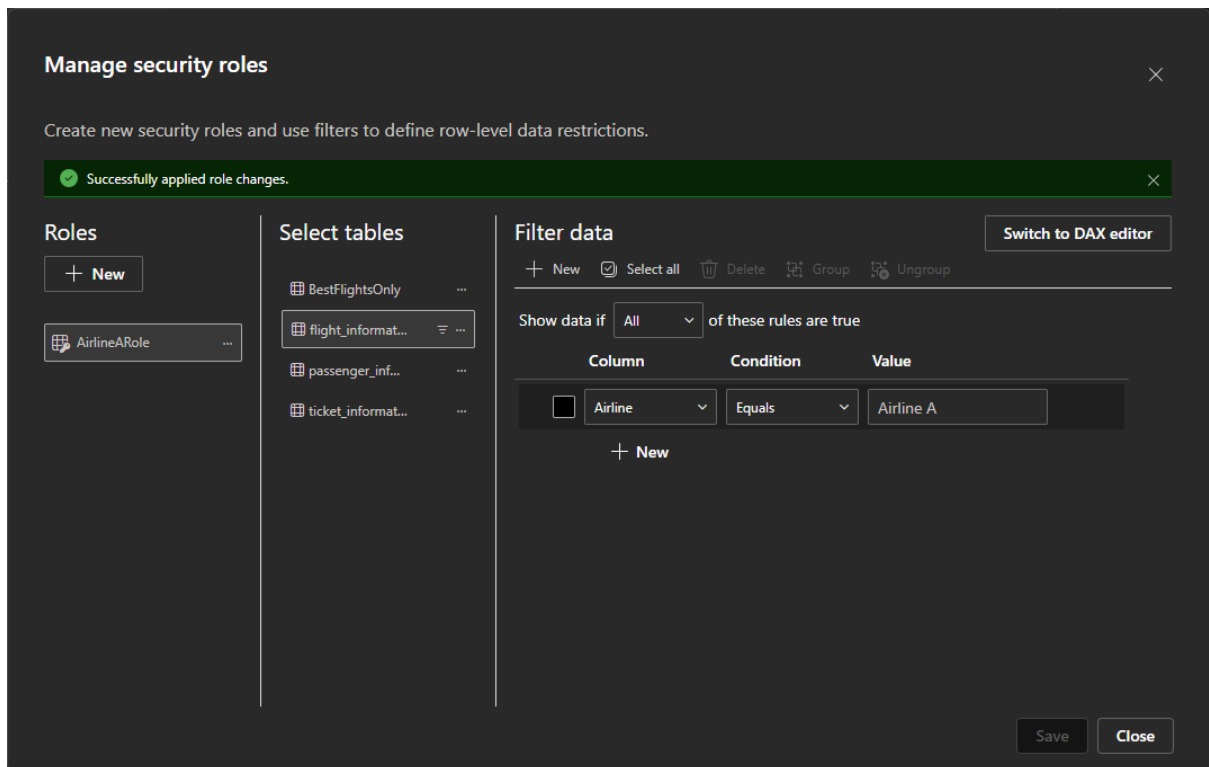


Figure 15: → RLS Role Setup in Power BI desktop for Airline A

3. Testing RLS: →

After creating the RLS role ArilineARole, the “View as” feature in Power BI desktop was used to simulate the view for users assigned to this role.

This allowed testing of the restriction Airline = “Airline A” to confirm that only data related to Airline A is Visible when the role is applied.

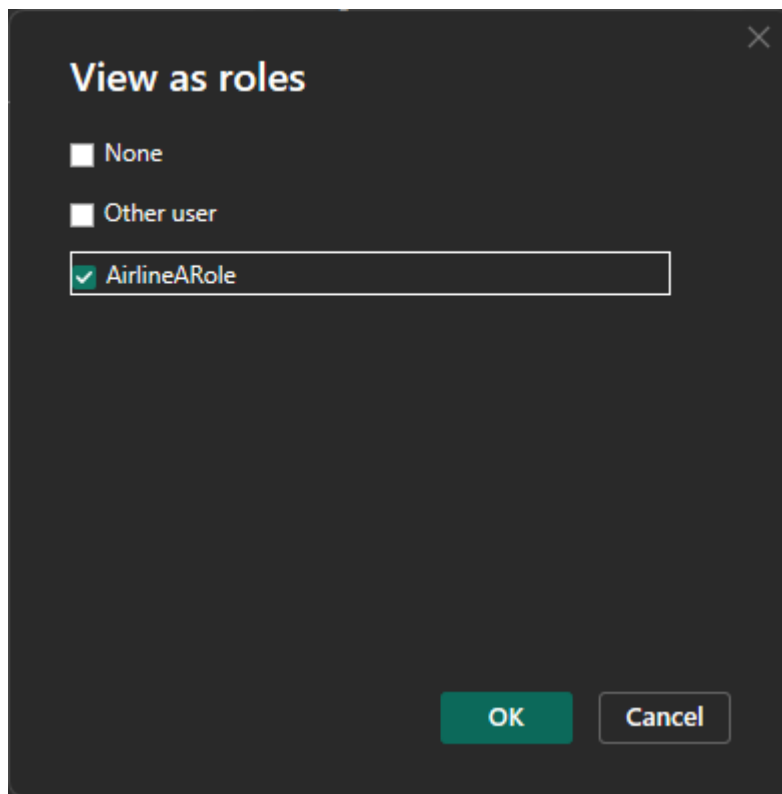


Figure 16:- Testing RLS using “View as role” feature in Power BI Desktop



Figure 17: → Dashboard View with AirlineARole Applied (RLS Active)

4. Scheduled Refresh setup (5 PM): →

To automate data updates, a daily scheduled refresh was configured in Power BI Service using the default gateway. The refresh is set for 5:00 PM daily, ensuring that the published report remains current without manual updates. This feature was enabled through the dataset settings after publishing the .pbix file to My Workspace.

The figure consists of two screenshots of the Power BI Service interface, specifically the 'Settings for Airline Data Analysis' page under the 'Semantic models' tab.

Top Screenshot: This view shows the 'Semantic models' tab. A yellow warning box states: 'Last refresh failed: 7/12/2025, 12:16:16 PM. Scheduled refresh has been disabled. [Show details](#)'. Below this, it indicates the 'Next refresh: 7/13/2025, 10:30:00 PM' and provides a 'Refresh history' link. There is also a section for 'Semantic model description' with a text area and 'Apply'/'Discard' buttons.

Bottom Screenshot: This view shows the 'Parameters' section, specifically the 'Refresh' settings. The 'Time zone' is set to '(UTC) Coordinated Universal Time'. The 'Refresh' toggle is turned 'On'. The 'Refresh frequency' is set to 'Daily'. The 'Time' is configured as '5:00 PM'. There is a link to 'Add another time'. Under 'Send refresh failure notifications to', the 'Semantic model owner' checkbox is checked, and there is a field for 'These contacts' with an 'Enter email addresses' placeholder. 'Apply' and 'Discard' buttons are at the bottom.

Figure 18: → Scheduled Daily Refresh Enabled at 5 PM via Default Gateway