Table of Contents:

1. Project Overview :

1.1 Understanding the Dataset:

The dataset provides transactional data related to railway operations, capturing various aspects of railway journeys, ticket purchases, and delays. It is structured into multiple dimensions, representing different entities involved in railway transportation.

Key Components of the Dataset:

- **Fact_Transaction**: Contains the core transactional data, linking various dimensions together.
- **Dim_Railcard**: Information about railcards, including their ID and holders.
- **Dim_Arrival & Dim_Departure**: Details about arrival and departure stations.
- **Dim_Delay**: Captures reasons for delays and their associated IDs.
- **Dim_Purchase**: Information about the type of purchase made.
- **Dim_Ticket**: Specifies ticket class and type.
- **Dim_Payment**: Payment method used for the journey.
- **Dim_Calendar**: Includes temporal attributes such as year, quarter, month, and day of the week.

Each dimension helps in analyzing railway transactions, understanding travel patterns, and identifying potential delays or inefficiencies. The dataset will be cleaned and structured to extract meaningful insights for decision-making.

Dataset Link:
https://docs.google.com/spreadsheets/d/1cwXHTBtwJGXpDOsHBeROoJc88U8iX9Dck9PESoGZjes/edit?usp=sharing

1.2 Project Objectives:

The primary goal of this project is to analyze railway ticket sales, customer behavior, and operational efficiency to derive actionable insights that enhance business decision-making. The key objectives include:

- **Analyze Delays:** Identify key factors and patterns causing train delays and assess their impact on overall operations.
- **Revenue & Sales Trends:** Track and visualize ticket sales and revenue trends over time to understand performance fluctuations.
- **Refund Analysis:** Evaluate refund rates and understand their financial implications, linking these to delay data and customer dissatisfaction.
- **Operational Insights:** Determine the performance of different stations and routes to highlight areas needing operational improvements.
- **Enhanced Decision-Making:** Provide stakeholders with actionable insights through a well-structured, interactive dashboard.

1.3 Business Objectives and Key Metrics:

1. Sales and Revenue Analysis

- What is the total revenue generated from ticket sales?
- Which purchase type (Online vs. Station) generates more revenue?
- What is the average ticket price for each ticket class (Standard, etc.)?
- How does the payment method (Contactless, Credit Card) affect sales volume?

2. Customer Behavior Analysis

- Which ticket type (Advance, etc.) is most frequently purchased?
- Are customers more likely to purchase tickets online or at the station?
- When are the peak travel hours, and which are the busiest days of the week?
- What are the most common travel routes (origin-destination pairs)?

3. Operational Efficiency

- What percentage of journeys arrive early, on time, or late, and what are the common reasons for delays?
- What is the average delay time for delayed journeys?
- Which routes have the highest delay rates?

4. Refund and Customer Satisfaction

- How many refund requests are made, and what are the primary reasons?
- Are refund requests more common for specific ticket types or payment methods?
- How does journey delay impact refund requests and overall customer satisfaction?

5. Journey Planning and Optimization

- How does the ticket price vary based on the time of purchase and journey date?
- Which routes are the most congested, and which are underutilized?
- What are the most popular departure and arrival times for journeys

6. Railcard and Ticket Class Analysis

- How does the use of railcards (e.g., Adult) affect ticket prices and sales?
- Which ticket class (Standard, etc.) is most popular among customers?
- Are there differences in delay rates based on ticket class?

7. Seasonal and Time-Based Trends

- How do seasonal factors (holidays, weekends) impact journey delays?
- Are there trends in delays based on seasonality or journey date?

8. Geographical Analysis

- Which routes (departure and arrival stations) are the most profitable?
- Are there specific routes with higher delay rates?
- How does the distance between departure and arrival stations affect ticket prices?
- Which stations have the highest and lowest passenger traffic?

2.Data Cleaning

2.1 Staging Area

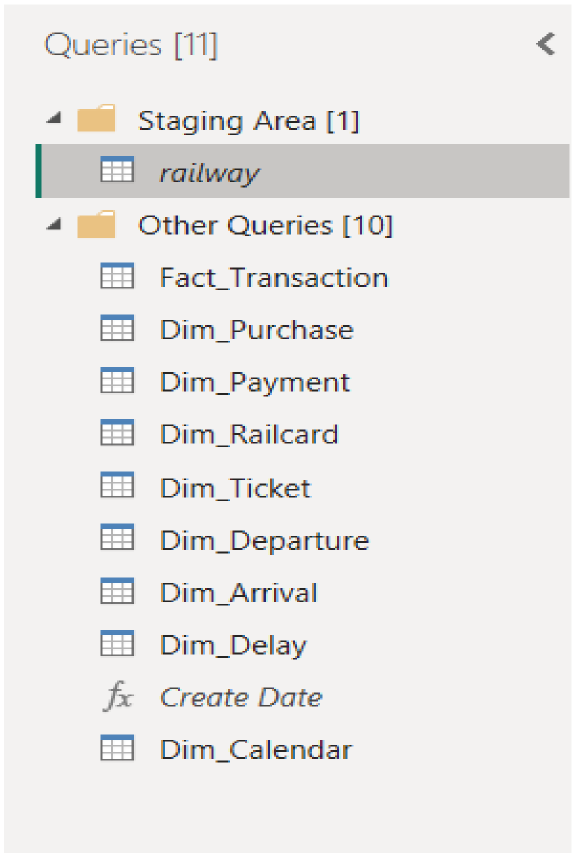
The **Staging Area** is a temporary layer where raw data is stored before transformation. It ensures **data integrity** before being processed into the final model.

Key Characteristics of the Staging Area:

- **Stores raw data without modifications** to preserve original records.
- **Acts as a validation layer** before data transformation.
- **Not loaded into the final data model** to optimize performance and storage.

2.2 Other Queries

- The **Other Queries** group contains the **Fact and Dimension tables**, which have been processed, cleaned, and optimized for reporting in Power BI.



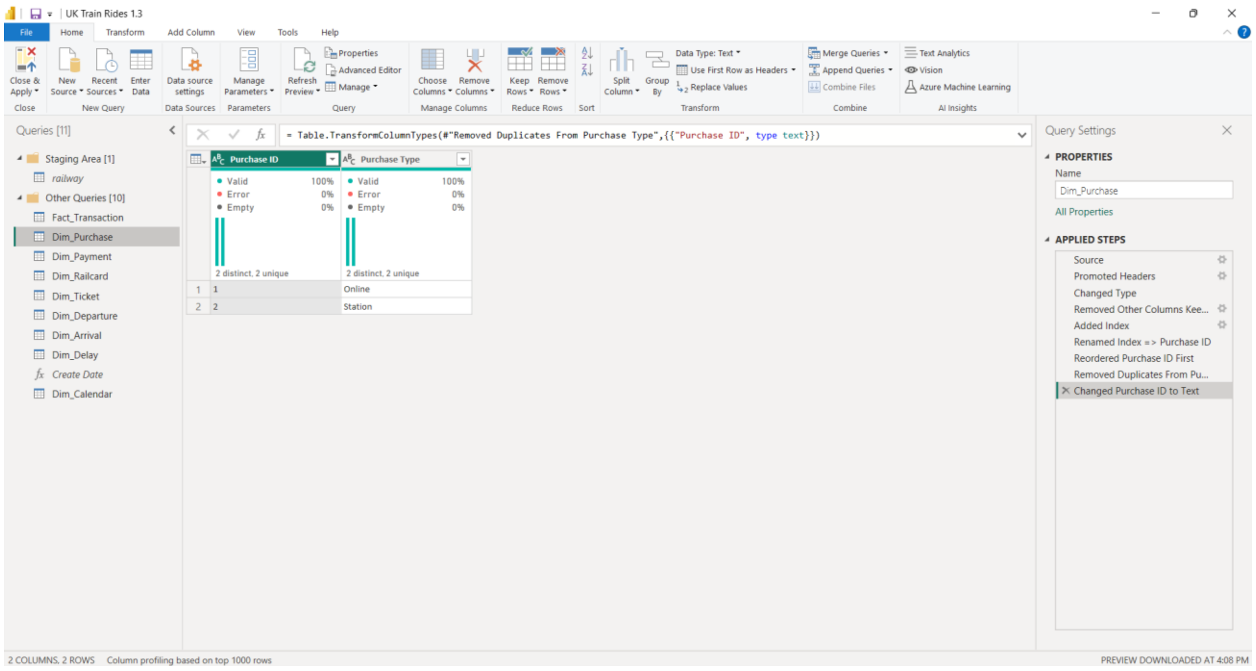
Queries [11]

- Staging Area [1]
  - *railway*
- Other Queries [10]
  - Fact_Transaction
  - Dim_Purchase
  - Dim_Payment
  - Dim_Railcard
  - Dim_Ticket
  - Dim_Departure
  - Dim_Arrival
  - Dim_Delay
  - *Create Date*
  - Dim_Calendar

Step 1: Dim_Purchase.

Objective: Move purchase details to a separate table and replace purchase-related information with Purchase ID.

1. **Extract Unique Purchases**
   - Select the Purchase Type.
   - Remove Duplicates to keep unique purchase records.
   - Rename the query as **Dim_Purchase**.
2. **Add an Index Column (Purchase ID)** => Start from 1
   - Rename the column to Purchase ID.
3. **Merge Purchase Data into Fact_Transactions**
   - Match **Purchase Type** in **Fact_Transaction** with **Dim_Purchase**.
   - Expand the merged table to keep only Purchase ID.
4. **Remove the Original Purchase Columns**
   - Delete the Purchase Type column from Fact_Transaction.



```
= Table.TransformColumnTypes(#"Removed Duplicates From Purchase Type",{{"Purchase ID", type text}})
```

	Purchase ID	Purchase Type
	Valid 100% / Error 0% / Empty 0%	Valid 100% / Error 0% / Empty 0%
	2 distinct, 2 unique	2 distinct, 2 unique
1	1	Online
2	2	Station

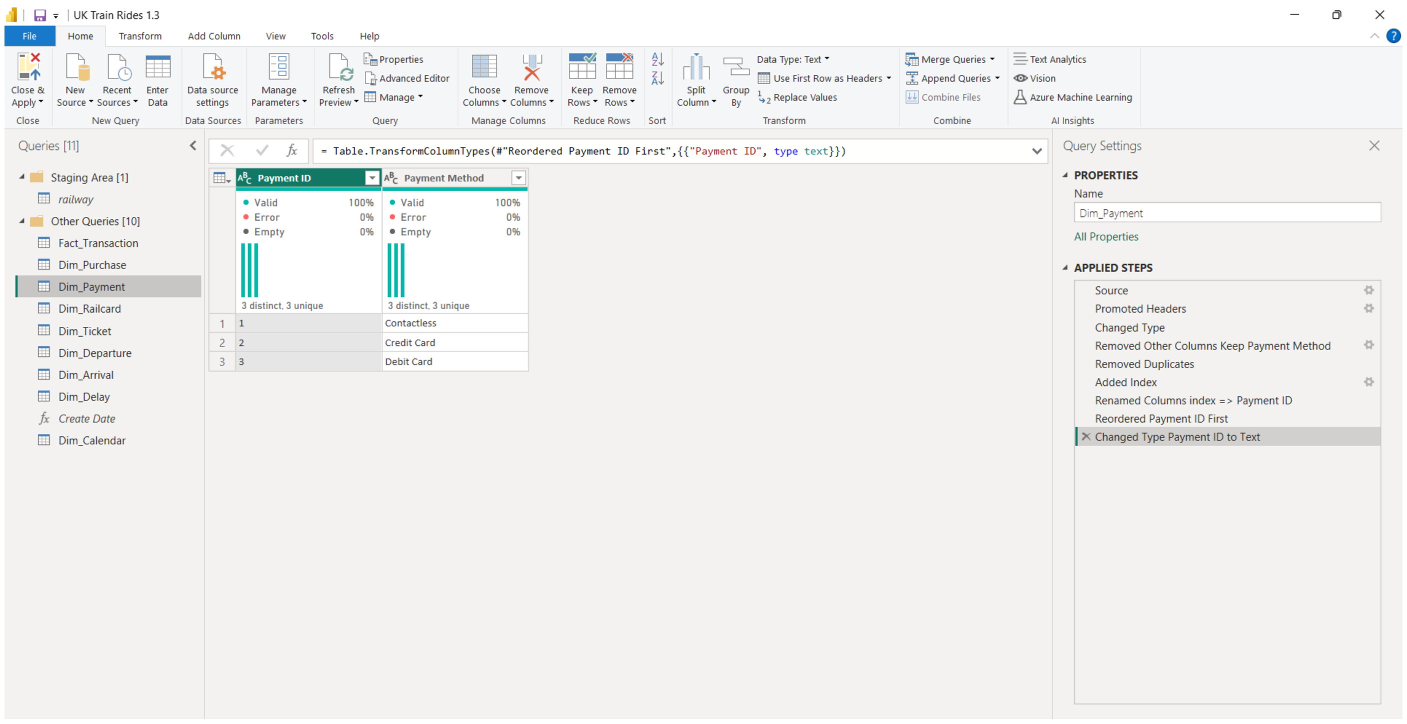
Applied steps:
- Source
- Promoted Headers
- Changed Type
- Removed Other Columns Kee...
- Added Index
- Renamed Index => Purchase ID
- Reordered Purchase ID First
- Removed Duplicates From Pu...
- Changed Purchase ID to Text

2 COLUMNS, 2 ROWS Column profiling based on top 1000 rows PREVIEW DOWNLOADED AT 4:08 PM

Step 2 : Create Dim_Payment

Objective: Move payment details to a separate table and replace payment method with Payment ID.

1. **Extract Unique Payment Methods**
   - Remove Duplicates to keep unique payment records.
   - Rename the query as **Dim_Payment**.
2. **Add an Index Column (Payment ID)** => Start from 1
   - Rename the column to Payment ID.
3. **Merge Payment Data into Fact_Transactions**
   - Match Payment Method in Fact_Transaction with Dim_Payment.
   - Expand the merged table to keep only Payment ID.
4. **Remove the Original Payment Columns**
   - Delete Payment Method from Fact_Transaction.



```
= Table.TransformColumnTypes(#"Reordered Payment ID First",{{"Payment ID", type text}})
```

	Payment ID	Payment Method
	Valid 100% Error 0% Empty 0% 3 distinct, 3 unique	Valid 100% Error 0% Empty 0% 3 distinct, 3 unique
1	1	Contactless
2	2	Credit Card
3	3	Debit Card

Name: Dim_Payment

APPLIED STEPS
- Source
- Promoted Headers
- Changed Type
- Removed Other Columns Keep Payment Method
- Removed Duplicates
- Added Index
- Renamed Columns index => Payment ID
- Reordered Payment ID First
- Changed Type Payment ID to Text

Step 3: Create Dim_Railcard

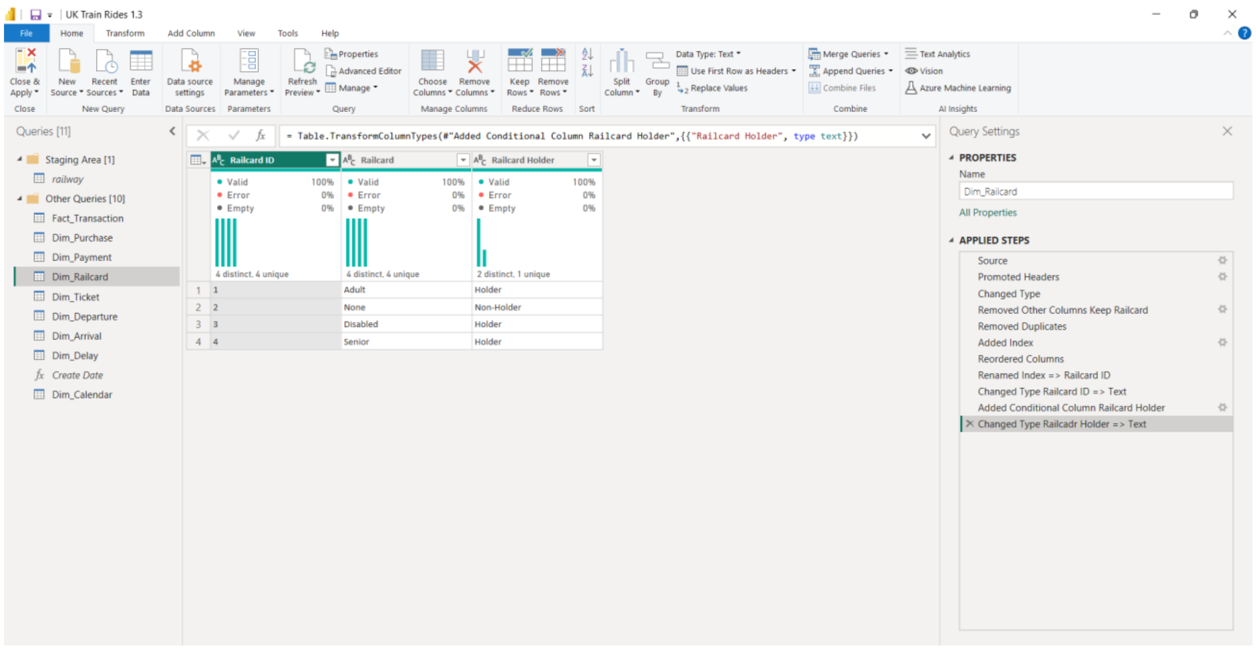
Objective: Move railcard discount details to a separate table and replace them with **Railcard ID**.

1. **Extract Unique Railcard Details**
   - Select **Railcard Type**.
   - Remove Duplicates to keep unique railcard records.
   - Rename the query as **Dim_Railcard**.
2. **Add an Index Column (Railcard ID)** => Start from 1
   - Rename the column to Railcard ID.
3. **Add a Conditional Column for Railcard Holder**

Set the condition:

   - If Railcard Type = "None", then "Non-Holder"
   - Else, "Holder"

4. **Merge Railcard Data into Fact_Transactions**
   - Match **Railcard Type** in **Fact_Transaction** with **Dim_Railcard**.
   - Expand the merged table to keep only **Railcard ID**.
5. **Remove the Original Railcard Columns**
   - Delete **Railcard Type** from **Fact_Transaction**.



```
= Table.TransformColumnTypes(#"Added Conditional Column Railcard Holder",{{"Railcard Holder", type text}})
```

	Railcard ID	Railcard	Railcard Holder
	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%
	4 distinct, 4 unique	4 distinct, 4 unique	2 distinct, 1 unique
1	1	Adult	Holder
2	2	None	Non-Holder
3	3	Disabled	Holder
4	4	Senior	Holder

Name: Dim_Railcard

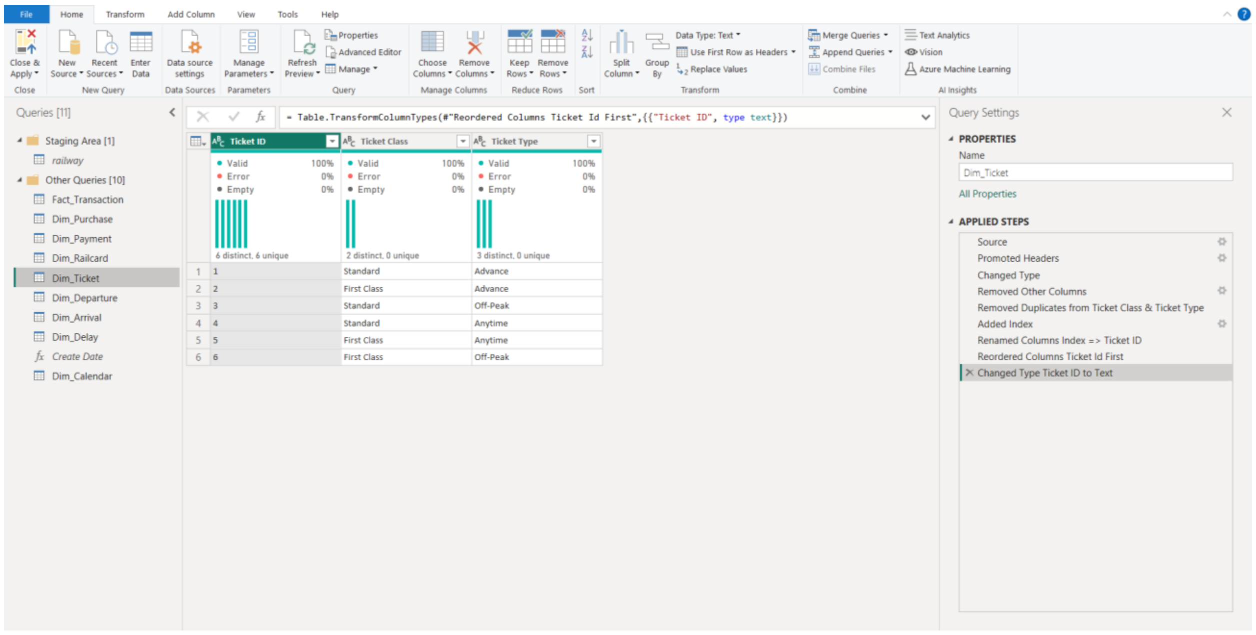
APPLIED STEPS

- Source
- Promoted Headers
- Changed Type
- Removed Other Columns Keep Railcard
- Removed Duplicates
- Added Index
- Reordered Columns
- Renamed Index => Railcard ID
- Changed Type Railcard ID => Text
- Added Conditional Column Railcard Holder
- Changed Type Railcadr Holder => Text

Step 4: Create Dim_Ticket

Objective: Move ticket details to a separate table and replace ticket type and class with **Ticket ID**.

1. **Extract Unique Ticket Details**
    - Select **Ticket Type** and **Ticket Class**.
    - Remove Duplicates to keep unique ticket records.
    - Rename the query as **Dim_Ticket**.
2. **Add an Index Column (Ticket ID)** => Start from 1
    - Rename the column to **Ticket ID**.
3. **Merge Ticket Data into Fact_Transactions**
    - Match Ticket Type & Class in Fact_Transaction with Dim_Ticket.
    - Expand the merged table to keep only **Ticket ID**.
4. **Remove the Original Ticket Columns**
    - Delete Ticket Type and Ticket Class from Fact_Transaction.



```
= Table.TransformColumnTypes(#"Reordered Columns Ticket Id First",{{"Ticket ID", type text}})
```

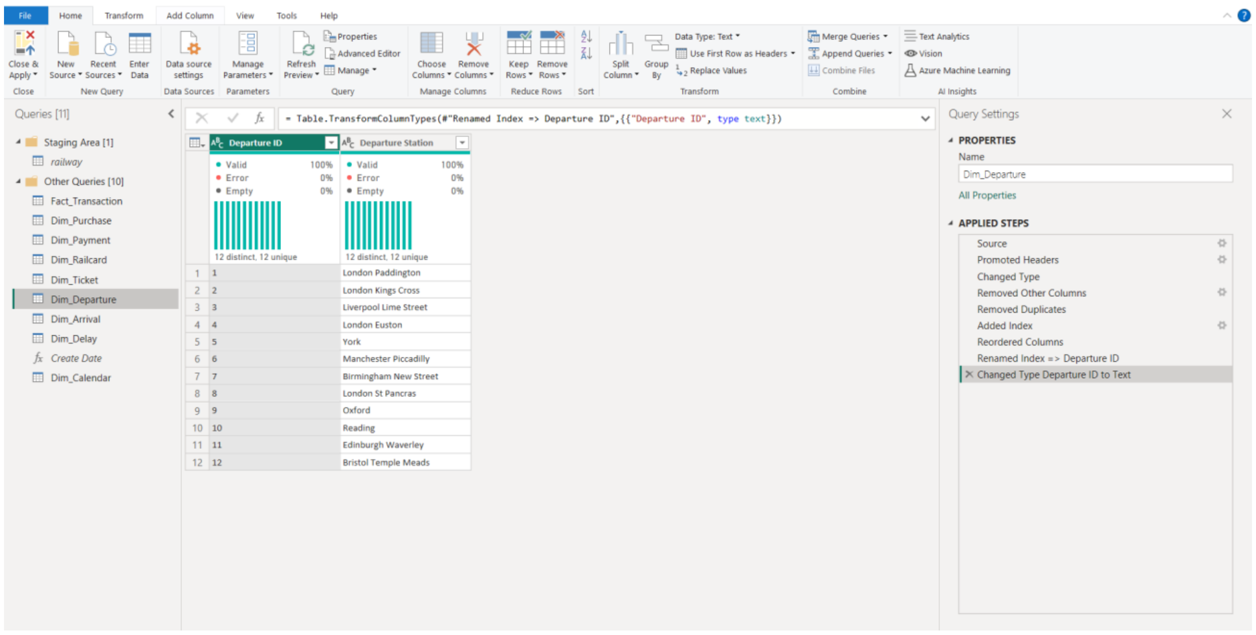
	Ticket ID	Ticket Class	Ticket Type
	Valid 100% Error 0% Empty 0% 6 distinct, 6 unique	Valid 100% Error 0% Empty 0% 2 distinct, 0 unique	Valid 100% Error 0% Empty 0% 3 distinct, 0 unique
1	1	Standard	Advance
2	2	First Class	Advance
3	3	Standard	Off-Peak
4	4	Standard	Anytime
5	5	First Class	Anytime
6	6	First Class	Off-Peak

Applied steps:
- Source
- Promoted Headers
- Changed Type
- Removed Other Columns
- Removed Duplicates from Ticket Class & Ticket Type
- Added Index
- Renamed Columns Index => Ticket ID
- Reordered Columns Ticket Id First
- Changed Type Ticket ID to Text

Step 5: Create Dim_Departure

Objective: Move departure station and time to a separate table and replace them with **Departure ID**.

1. **Extract Unique Departure Stations**
   - Select **Departure Station** .
   - Remove Duplicates to keep unique departure records.
   - Rename the query as **Dim_Departure**.
2. **Add an Index Column (Departure ID)** => Start from 1
   - Rename the column to **Departure ID**.
3. **Merge Departure Data into Fact_Transactions**
   - Match **Departure Station** in **Fact_Transaction** with **Dim_Departure**.
   - Expand the merged table to keep only **Departure ID**.
4. **Remove the Original Departure Columns**
   - Delete **Departure Station** from **Fact_Transaction**.



```
= Table.TransformColumnTypes(#"Renamed Index => Departure ID",{{"Departure ID", type text}})
```

	Departure ID	Departure Station
	Valid 100%, Error 0%, Empty 0%, 12 distinct, 12 unique	Valid 100%, Error 0%, Empty 0%, 12 distinct, 12 unique
1	1	London Paddington
2	2	London Kings Cross
3	3	Liverpool Lime Street
4	4	London Euston
5	5	York
6	6	Manchester Piccadilly
7	7	Birmingham New Street
8	8	London St Pancras
9	9	Oxford
10	10	Reading
11	11	Edinburgh Waverley
12	12	Bristol Temple Meads

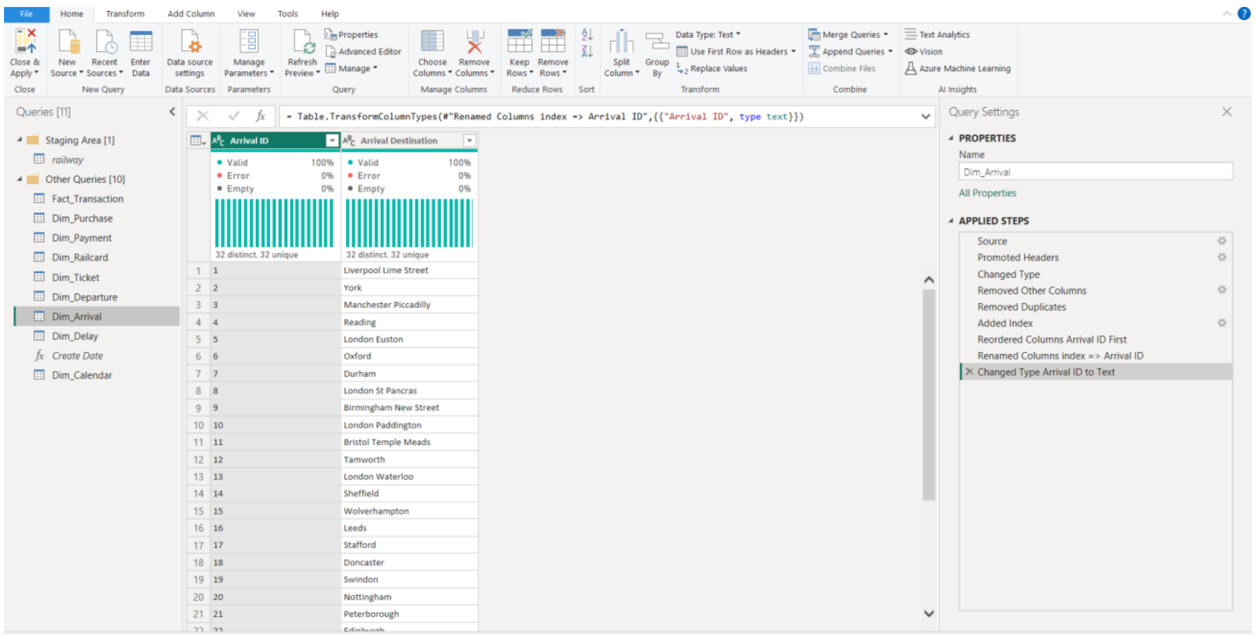
Name: Dim_Departure

Applied steps:
- Source
- Promoted Headers
- Changed Type
- Removed Other Columns
- Removed Duplicates
- Added Index
- Reordered Columns
- Renamed Index => Departure ID
- Changed Type Departure ID to Text

Step 6: Create Dim_Arrival

Objective: Move arrival station and time to a separate table and replace them with **Arrival ID**.

1. **Extract Unique Arrival Stations**
   - Select **Arrival Station**.
   - Remove Duplicates to keep unique arrival records.
   - Rename the query as **Dim_Arrival**.
2. **Add an Index Column (Arrival ID)** => Start from 1
   - Rename the column to **Arrival ID**.
3. **Merge Arrival Data into Fact_Transactions**
   - Match **Arrival Station** in **Fact_Transaction** with **Dim_Arrival**.
   - Expand the merged table to keep only **Arrival ID**.
4. **Remove the Original Arrival Columns**
   - Delete **Arrival Station** from **Fact_Transaction**.



Queries [11]
- Staging Area [1]
  - railway
- Other Queries [10]
  - Fact_Transaction
  - Dim_Purchase
  - Dim_Payment
  - Dim_Railcard
  - Dim_Ticket
  - Dim_Departure
  - Dim_Arrival
  - Dim_Delay
  - Create Date
  - Dim_Calendar

```
= Table.TransformColumnTypes(#"Renamed Columns index => Arrival ID",{{"Arrival ID", type text}})
```

	Arrival ID	Arrival Destination
	Valid 100% · Error 0% · Empty 0% · 32 distinct, 32 unique	Valid 100% · Error 0% · Empty 0% · 32 distinct, 32 unique
1	1	Liverpool Lime Street
2	2	York
3	3	Manchester Piccadilly
4	4	Reading
5	5	London Euston
6	6	Oxford
7	7	Durham
8	8	London St Pancras
9	9	Birmingham New Street
10	10	London Paddington
11	11	Bristol Temple Meads
12	12	Tamworth
13	13	London Waterloo
14	14	Sheffield
15	15	Wolverhampton
16	16	Leeds
17	17	Stafford
18	18	Doncaster
19	19	Swindon
20	20	Nottingham
21	21	Peterborough

Query Settings

PROPERTIES

Name: Dim_Arrival

All Properties

APPLIED STEPS
- Source
- Promoted Headers
- Changed Type
- Removed Other Columns
- Removed Duplicates
- Added Index
- Reordered Columns Arrival ID First
- Renamed Columns index => Arrival ID
- Changed Type Arrival ID to Text

Step 7: Create Dim_Delay

Objective: Move train delay details to a separate table and replace them with **Delay ID**.

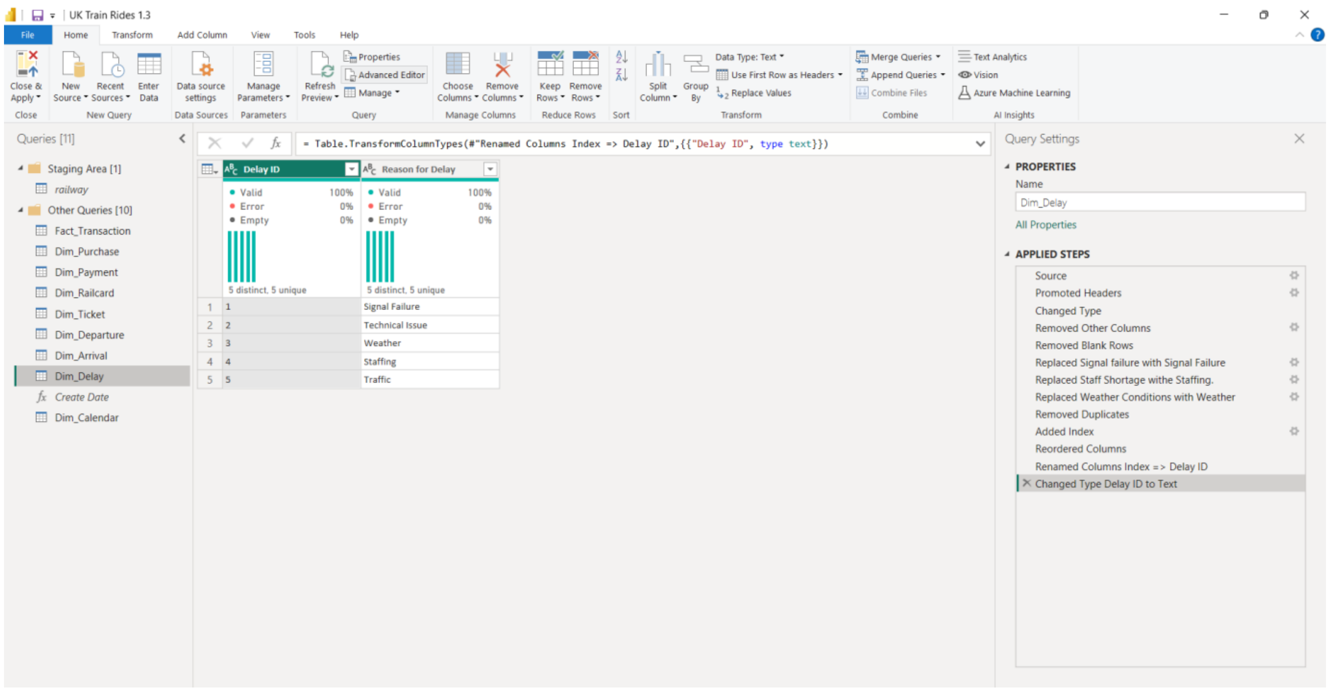
1. **Extract Unique Delay Records**
   a. Select **Reason for Delay**.
   b. Remove Duplicates to keep unique delay records.
   c. Rename the query as **Dim_Delay**.
2. **Replace Inconsistent Values**

   Go to **Transform → Replace Values**.

   Apply the following replacements:
   a. "Signal failure" → "Signal Failure"
   b. "Staff Shortage" → "Staffing"
   c. "Weather Conditions" → "Weather"
3. **Verify the Changes**

   Scroll through the column to ensure that the replacements have been correctly applied.
4. **Add an Index Column (Delay ID)** => Start from 1
   a. Rename the column to **Delay ID**.
5. **Merge Delay Data into Fact_Transactions**
   a. Match **Reason for Delay** in **Fact_Transaction** with **Dim_Delay**.
   b. Expand the merged table to keep only **Delay ID**.
6. **Remove the Original Delay Columns**
   a. Delete **Reason for Delay** from **Fact_Transaction**.

Dim_Delay



Queries [11]

- Staging Area [1]
  - *railway*
- Other Queries [10]
  - Fact_Transaction
  - Dim_Purchase
  - Dim_Payment
  - Dim_Railcard
  - Dim_Ticket
  - Dim_Departure
  - Dim_Arrival
  - Dim_Delay
  - *Create Date*
  - Dim_Calendar

```
= Table.TransformColumnTypes(#"Renamed Columns Index => Delay ID",{{"Delay ID", type text}})
```

	Delay ID	Reason for Delay
	Valid 100% · Error 0% · Empty 0% · 5 distinct, 5 unique	Valid 100% · Error 0% · Empty 0% · 5 distinct, 5 unique
1	1	Signal Failure
2	2	Technical Issue
3	3	Weather
4	4	Staffing
5	5	Traffic

Query Settings

PROPERTIES

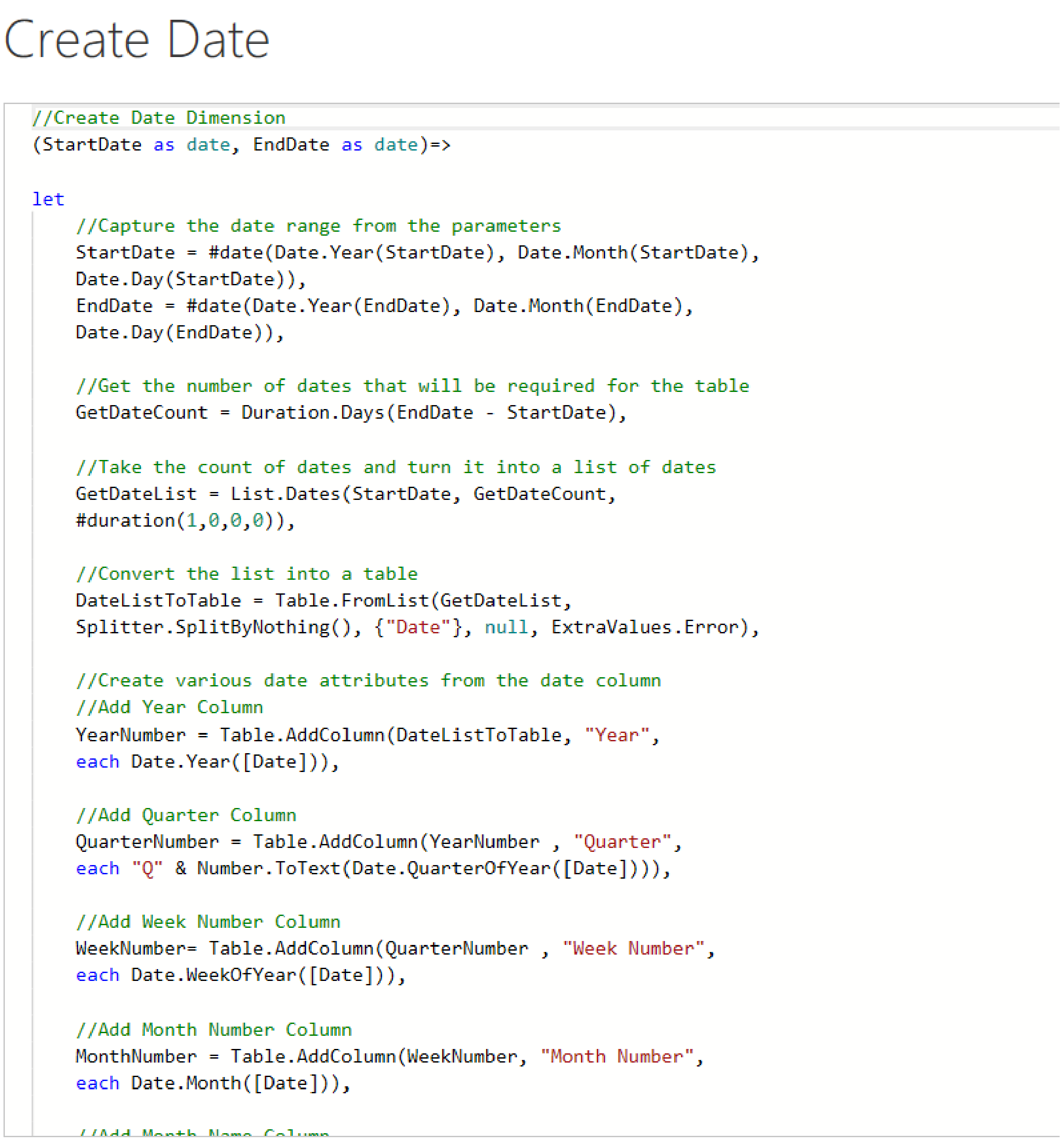
Name: Dim_Delay

All Properties

APPLIED STEPS

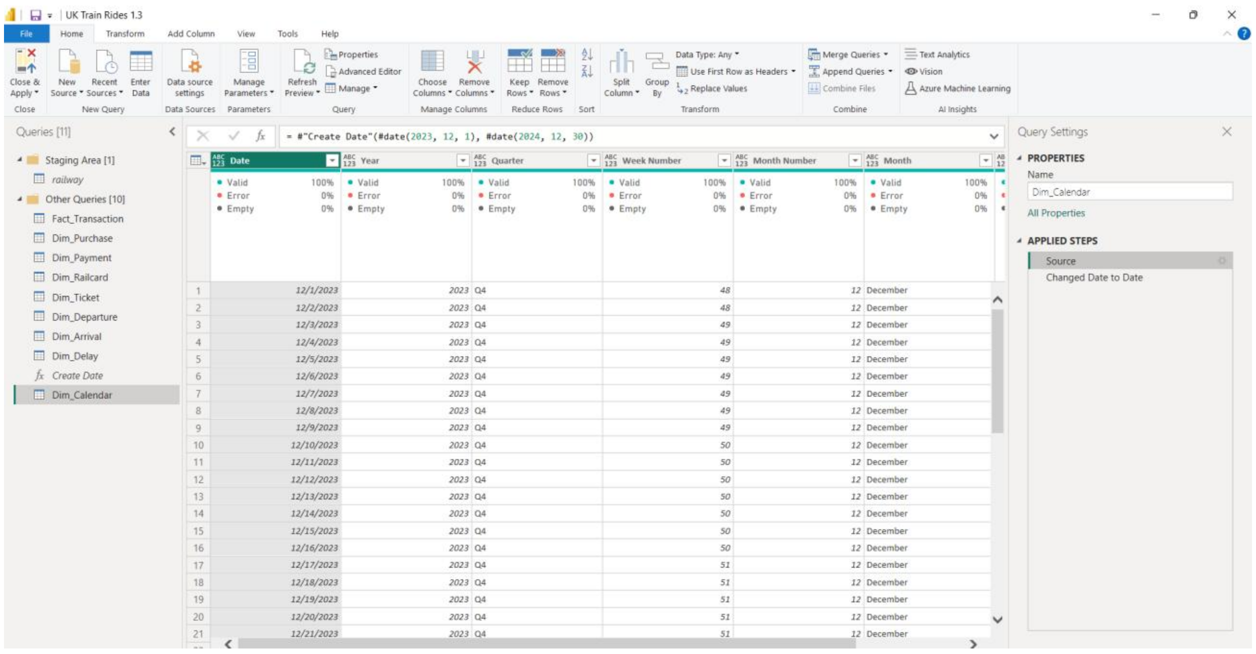
- Source
- Promoted Headers
- Changed Type
- Removed Other Columns
- Removed Blank Rows
- Replaced Signal failure with Signal Failure
- Replaced Staff Shortage withe Staffing.
- Replaced Weather Conditions with Weather
- Removed Duplicates
- Added Index
- Reordered Columns
- Renamed Columns Index => Delay ID
- Changed Type Delay ID to Text

Step 8: Create Dim_Calendar



Create Date

```
//Create Date Dimension
(StartDate as date, EndDate as date)=>

let
    //Capture the date range from the parameters
    StartDate = #date(Date.Year(StartDate), Date.Month(StartDate),
    Date.Day(StartDate)),
    EndDate = #date(Date.Year(EndDate), Date.Month(EndDate),
    Date.Day(EndDate)),

    //Get the number of dates that will be required for the table
    GetDateCount = Duration.Days(EndDate - StartDate),

    //Take the count of dates and turn it into a list of dates
    GetDateList = List.Dates(StartDate, GetDateCount,
    #duration(1,0,0,0)),

    //Convert the list into a table
    DateListToTable = Table.FromList(GetDateList,
    Splitter.SplitByNothing(), {"Date"}, null, ExtraValues.Error),

    //Create various date attributes from the date column
    //Add Year Column
    YearNumber = Table.AddColumn(DateListToTable, "Year",
    each Date.Year([Date])),

    //Add Quarter Column
    QuarterNumber = Table.AddColumn(YearNumber , "Quarter",
    each "Q" & Number.ToText(Date.QuarterOfYear([Date]))),

    //Add Week Number Column
    WeekNumber= Table.AddColumn(QuarterNumber , "Week Number",
    each Date.WeekOfYear([Date])),

    //Add Month Number Column
    MonthNumber = Table.AddColumn(WeekNumber, "Month Number",
    each Date.Month([Date])),

    //Add Month Name Column
    //Add Month Number Column
    MonthNumber = Table.AddColumn(WeekNumber, "Month Number",
    each Date.Month([Date])),

    //Add Month Name Column
    MonthName = Table.AddColumn(MonthNumber , "Month",
    each Date.ToText([Date],"MMMM")),

    //Add Day of Week Column
    DayOfWeek = Table.AddColumn(MonthName , "Day of Week",
    each Date.ToText([Date],"dddd"))

in
    DayOfWeek
```



```
= #"Create Date"(#date(2023, 12, 1), #date(2024, 12, 30))
```

	Date	Year	Quarter	Week Number	Month Number	Month
	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%
1	12/1/2023	2023	Q4	48	12	December
2	12/2/2023	2023	Q4	48	12	December
3	12/3/2023	2023	Q4	49	12	December
4	12/4/2023	2023	Q4	49	12	December
5	12/5/2023	2023	Q4	49	12	December
6	12/6/2023	2023	Q4	49	12	December
7	12/7/2023	2023	Q4	49	12	December
8	12/8/2023	2023	Q4	49	12	December
9	12/9/2023	2023	Q4	49	12	December
10	12/10/2023	2023	Q4	50	12	December
11	12/11/2023	2023	Q4	50	12	December
12	12/12/2023	2023	Q4	50	12	December
13	12/13/2023	2023	Q4	50	12	December
14	12/14/2023	2023	Q4	50	12	December
15	12/15/2023	2023	Q4	50	12	December
16	12/16/2023	2023	Q4	50	12	December
17	12/17/2023	2023	Q4	51	12	December
18	12/18/2023	2023	Q4	51	12	December
19	12/19/2023	2023	Q4	51	12	December
20	12/20/2023	2023	Q4	51	12	December
21	12/21/2023	2023	Q4	51	12	December

9. Fact_Transaction

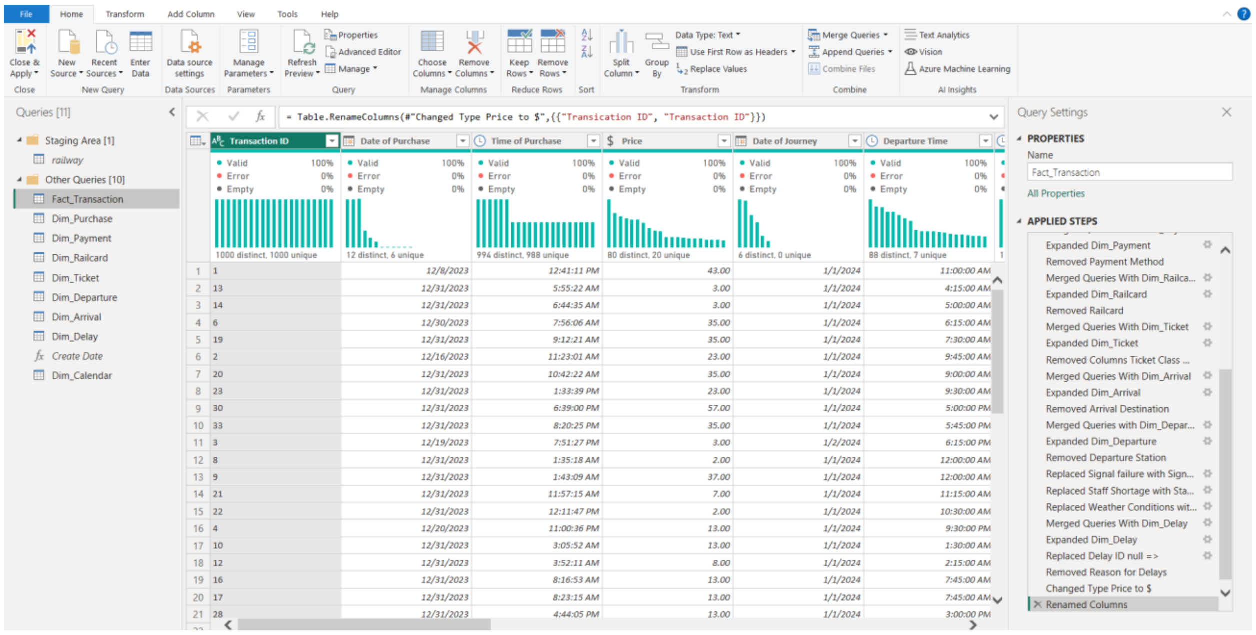
Objective:

The **Fact_Transaction** table captures transactional details related to train journeys, including ticket purchases, journey times, delays, and payments. It links to various dimension tables to ensure a well-structured and optimized data model. Ensure data integrity and replace NULL values in Actual Arrival Time for canceled journeys.

Ensure data integrity and replace NULL values in Actual Arrival Time for canceled journeys.

1. Handling NULL Values in Actual Arrival Time

- Left NULL values as they are to represent canceled journeys, ensuring accurate data



```
= Table.RenameColumns(#"Changed Type Price to $",{{"Transication ID", "Transaction ID"}})
```

	Transaction ID	Date of Purchase	Time of Purchase	Price	Date of Journey	Departure Time
	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%
	1000 distinct, 1000 unique	12 distinct, 6 unique	994 distinct, 988 unique	80 distinct, 20 unique	6 distinct, 0 unique	88 distinct, 7 unique
1	1	12/8/2023	12:41:11 PM	43.00	1/1/2024	11:00:00 AM
2	13	12/31/2023	5:55:22 AM	3.00	1/1/2024	4:15:00 AM
3	14	12/31/2023	6:44:35 AM	3.00	1/1/2024	5:00:00 AM
4	6	12/30/2023	7:56:06 AM	35.00	1/1/2024	6:15:00 AM
5	19	12/31/2023	9:12:21 AM	35.00	1/1/2024	7:30:00 AM
6	2	12/16/2023	11:23:01 AM	23.00	1/1/2024	9:45:00 AM
7	20	12/31/2023	10:42:22 AM	35.00	1/1/2024	9:00:00 AM
8	23	12/31/2023	1:33:39 PM	23.00	1/1/2024	9:30:00 AM
9	30	12/31/2023	6:39:00 PM	57.00	1/1/2024	5:00:00 PM
10	33	12/31/2023	8:20:25 PM	35.00	1/1/2024	5:45:00 PM
11	3	12/19/2023	7:51:27 PM	3.00	1/2/2024	6:15:00 PM
12	8	12/31/2023	1:35:18 AM	2.00	1/1/2024	12:00:00 AM
13	9	12/31/2023	1:43:09 AM	37.00	1/1/2024	12:00:00 AM
14	21	12/31/2023	11:57:15 AM	7.00	1/1/2024	11:15:00 AM
15	22	12/31/2023	12:11:47 PM	2.00	1/1/2024	10:30:00 AM
16	4	12/20/2023	11:00:36 PM	13.00	1/1/2024	9:30:00 PM
17	10	12/31/2023	3:05:52 AM	13.00	1/1/2024	1:30:00 AM
18	12	12/31/2023	3:52:11 AM	8.00	1/1/2024	2:15:00 AM
19	16	12/31/2023	8:16:53 AM	13.00	1/1/2024	7:45:00 AM
20	17	12/31/2023	8:23:15 AM	13.00	1/1/2024	7:45:00 AM
21	28	12/31/2023	4:44:05 PM	13.00	1/1/2024	3:00:00 PM

Query Settings

PROPERTIES

Name: Fact_Transaction

All Properties

APPLIED STEPS

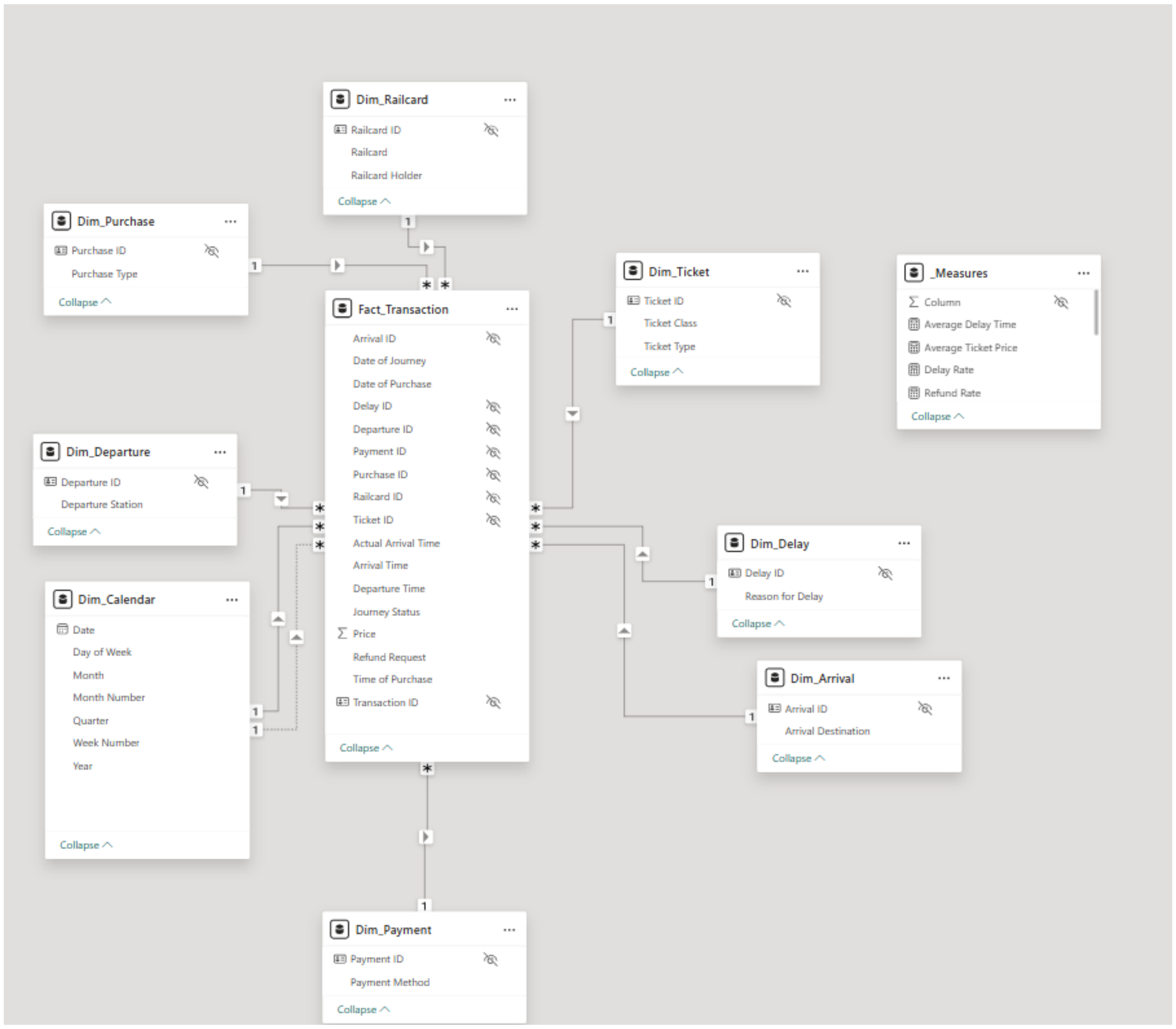
- Expanded Dim_Payment
- Removed Payment Method
- Merged Queries With Dim_Railca...
- Expanded Dim_Railcard
- Removed Railcard
- Merged Queries With Dim_Ticket
- Expanded Dim_Ticket
- Removed Columns Ticket Class ...
- Merged Queries With Dim_Arrival
- Expanded Dim_Arrival
- Removed Arrival Destination
- Merged Queries with Dim_Depar...
- Expanded Dim_Departure
- Removed Departure Station
- Replaced Signal failure with Sign...
- Replaced Staff Shortage with Sta...
- Replaced Weather Conditions wit...
- Merged Queries With Dim_Delay
- Expanded Dim_Delay
- Replaced Delay ID null =>
- Removed Reason for Delays
- Changed Type Price to $
- Renamed Columns

3.Data Modeling

3.1 Schema Design and Relationships

The data model follows a star schema design, where the Fact_Transaction table is at the center, linking to multiple dimension tables. This schema optimizes query performance and simplifies data analysis by organizing information into facts (measurable business processes) and dimensions (descriptive attributes).

The relationships between the tables are based on primary keys (PKs) and foreign keys (FKs), ensuring referential integrity.



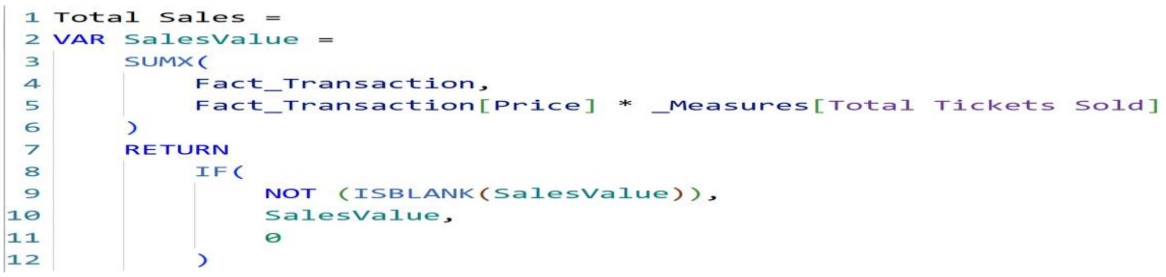
4.DAX & Measures

(1) Total Sales

Purpose:

This measure calculates the total sales revenue by multiplying the price of tickets by the total number of tickets sold. It ensures that if there are no sales, the result is 0 instead of blank.

DAX Formul:



```
Total Sales =
VAR SalesValue =
    SUMX(
        Fact_Transaction,
        Fact_Transaction[Price] * _Measures[Total Tickets Sold]
    )
    RETURN
        IF(
            NOT (ISBLANK(SalesValue)),
            SalesValue,
            0
        )
```

Explaination:

VAR SalesValue = SUMX(Fact_Transaction, Fact_Transaction[Price] * [Total Tickets Sold])

- Iterates over the Fact_Transaction table, calculating price × total tickets sold for each row and summing the results.

IF(NOT(ISBLANK(SalesValue)), SalesValue, 0)

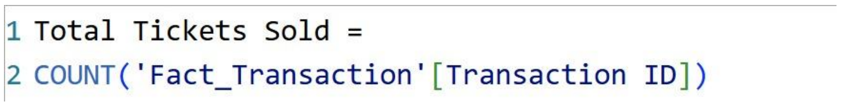
- Ensures that if there are no transactions, the measure returns 0 instead of a blank value.

(2) Total Tickets Sold

Purpose:

This measure calculates the total number of tickets sold, representing the total transactions in the system.

DAX Formul:



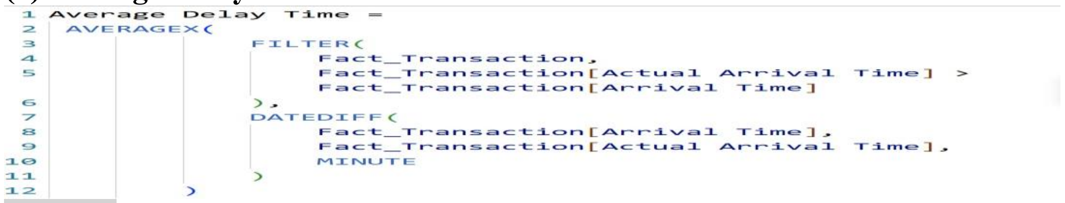
```
Total Tickets Sold =
COUNT('Fact_Transaction'[Transaction ID])
```

Explaination:

COUNT(Fact_Transaction[Transaction ID])

- Counts the total number of unique transactions, which correspond to tickets sold.

(3) Average Delay Time



```
Average Delay Time =
AVERAGEX(
            FILTER(
                Fact_Transaction,
                Fact_Transaction[Actual Arrival Time] >
                Fact_Transaction[Arrival Time]
            ),
            DATEDIFF(
                Fact_Transaction[Arrival Time],
                Fact_Transaction[Actual Arrival Time],
                MINUTE
            )
        )
```

Purpose:

This measure calculates the average delay time for train journeys where the actual arrival time was later than the scheduled arrival time.

DAX Formul:

Explaination:

FILTER(Fact_Transaction, Fact_Transaction[Actual Arrival Time] > Fact_Transaction[Arrival Time])

- Filters transactions where the actual arrival time is greater than the scheduled arrival time (i.e., delayed trains).

DATEDIFF(Fact_Transaction[Arrival Time], Fact_Transaction[Actual Arrival Time], MINUTE)

- Calculates the delay in minutes for each transaction.

AVERAGEX(...)

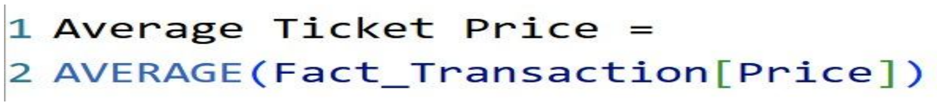
- Computes the average delay time across all delayed transactions.

(4) Average Ticket Price

Purpose:

This measure calculates the average price of all ticket transactions.

DAX Formula:



```
Average Ticket Price =
AVERAGE(Fact_Transaction[Price])
```

Explaination:

AVERAGE(Fact_Transaction[Price])

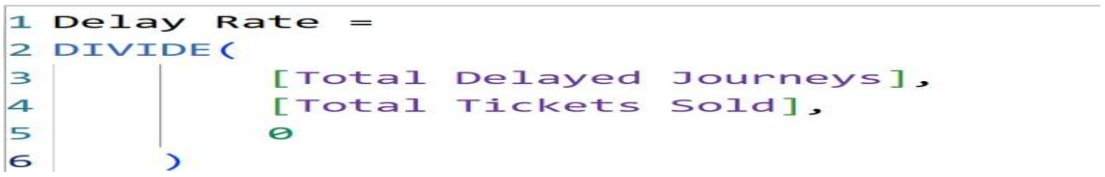
- Computes the mean ticket price across all transactions in the Fact_Transaction table.

(5) Delay Rate

Purpose:

This measure calculates the percentage of delayed journeys compared to the total tickets sold.

DAX Formula:



```
Delay Rate =
DIVIDE(
            [Total Delayed Journeys],
            [Total Tickets Sold],
            0
        )
```

Explaination:

DIVIDE([Total Delayed Journeys], [Total Tickets Sold], 0)

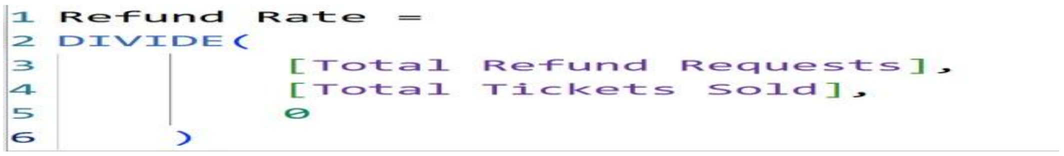
- Computes the delay rate as a ratio of delayed journeys to total ticket sales.
- The third argument (0) ensures the function returns 0 instead of an error if the denominator is zero.

(6) Refund Rate

Purpose:

This measure calculates the percentage of refund requests compared to the total tickets sold.

DAX Formula:



```
Refund Rate =
DIVIDE(
            [Total Refund Requests],
            [Total Tickets Sold],
            0
        )
```

Explaination:

DIVIDE([Total Refund Requests], [Total Tickets Sold], 0)

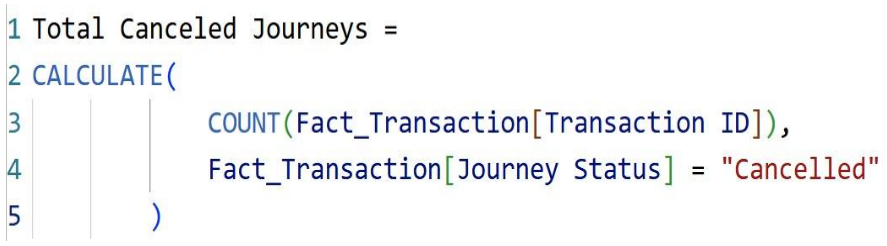
- Computes the refund rate as a ratio of refund requests to total ticket sales.
- The third argument (0) ensures the function returns 0 instead of an error if the denominator is zero.

(7) Total Canceled Journeys

Purpose:

This measure calculates the total number of journeys that were canceled.

DAX Formula:



```
Total Canceled Journeys =
CALCULATE(
            COUNT(Fact_Transaction[Transaction ID]),
            Fact_Transaction[Journey Status] = "Cancelled"
        )
```

Explaination:

COUNT(Fact_Transaction[Transaction ID])

- Counts the total number of transactions (i.e., ticket purchases).

CALCULATE(..., Fact_Transaction[Journey Status] = "Cancelled")

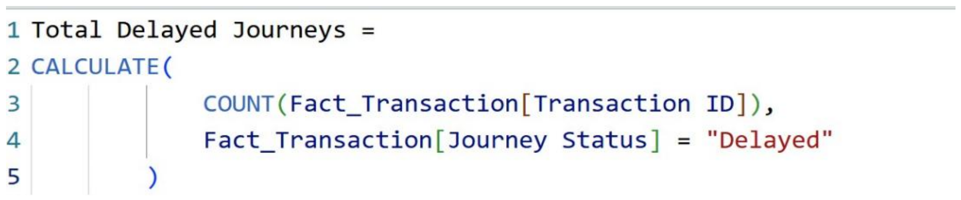
- Filters the transactions to only include those where the Journey Status is "Cancelled".

(8) Total Delayed Journeys

Purpose:

This measure calculates the total number of journeys that were delayed.

DAX Formula:



```
Total Delayed Journeys =
CALCULATE(
            COUNT(Fact_Transaction[Transaction ID]),
            Fact_Transaction[Journey Status] = "Delayed"
        )
```

Explaination:

COUNT(Fact_Transaction[Transaction ID])

- Counts the total number of ticket transactions.

CALCULATE(..., Fact_Transaction[Journey Status] = "Delayed")

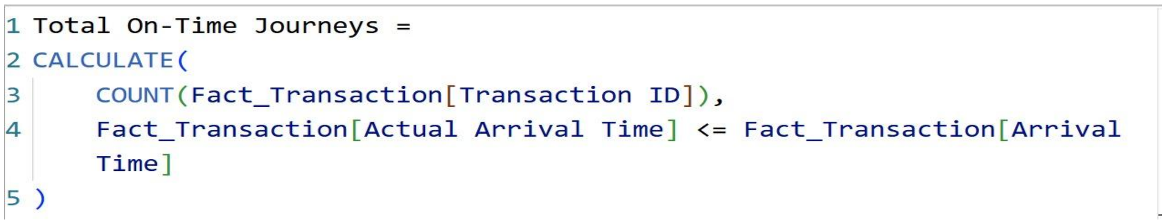
- Filters the dataset to include only transactions where Journey Status is "Delayed".

(9) Total On-Time Journeys

Purpose:

This measure calculates the total number of journeys that arrived on time or earlier than the scheduled arrival time.

DAX Formula:



```
Total On-Time Journeys =
CALCULATE(
    COUNT(Fact_Transaction[Transaction ID]),
    Fact_Transaction[Actual Arrival Time] <= Fact_Transaction[Arrival Time]
)
```

Explaination:

COUNT(Fact_Transaction[Transaction ID])

- Counts the total number of ticket transactions.

CALCULATE(..., Fact_Transaction[Actual Arrival Time] <= Fact_Transaction[Arrival Time])

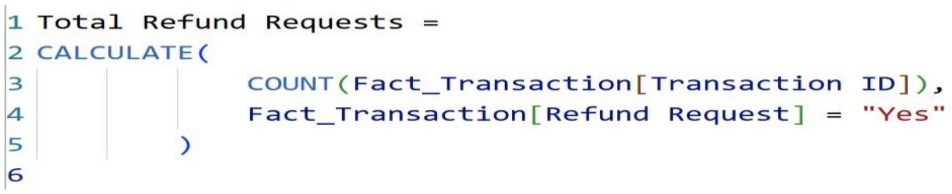
- Filters the dataset to count only transactions where the Actual Arrival Time is on time or earlier than the scheduled Arrival Time.

(10) Total Refund Requests

Purpose:

This measure calculates the total number of refund requests made by passengers.

DAX Formula:



```
Total Refund Requests =
CALCULATE(
                COUNT(Fact_Transaction[Transaction ID]),
                Fact_Transaction[Refund Request] = "Yes"
          )

```

Explaination:

COUNT(Fact_Transaction[Transaction ID])

- Counts the total number of ticket transactions.

CALCULATE(..., Fact_Transaction[Refund Request] = "Yes")

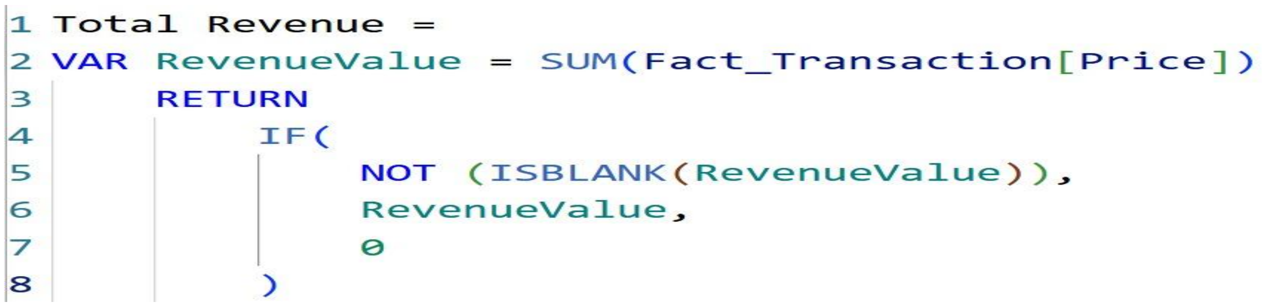
- Filters the dataset to count only transactions where a refund request was made.

(11) Total Revenue :

Purpose:

This measure adds up all the prices in your sales data to get the total revenue

DAX Formula:



```
Total Revenue =
VAR RevenueValue = SUM(Fact_Transaction[Price])
    RETURN
        IF(
            NOT (ISBLANK(RevenueValue)),
            RevenueValue,
            0
        )
```

Explaination:

VAR RevenueValue = SUM(Fact_Transaction[Price])

- This creates a variable RevenueValue that sums up all values in the Price column.

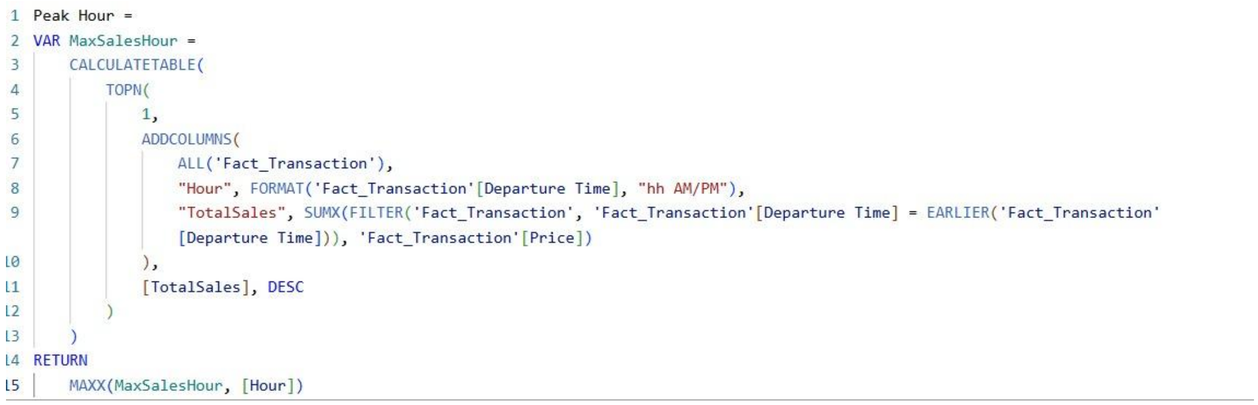
The IF checks if RevenueValue is not blank. If true, it returns the sum; if false, it returns 0 .

12 Peak Hour

Purpose:

find the hour of the day (in 12-hour AM/PM format) during which the highest total ticket sales occurred

DAX Formula:

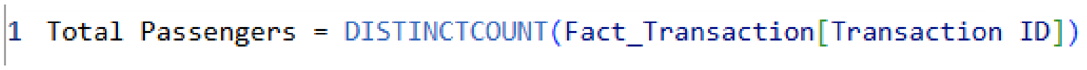


```
Peak Hour =
VAR MaxSalesHour =
    CALCULATETABLE(
        TOPN(
            1,
            ADDCOLUMNS(
                ALL('Fact_Transaction'),
                "Hour", FORMAT('Fact_Transaction'[Departure Time], "hh AM/PM"),
                "TotalSales", SUMX(FILTER('Fact_Transaction', 'Fact_Transaction'[Departure Time] = EARLIER('Fact_Transaction'
                [Departure Time])), 'Fact_Transaction'[Price])
            ),
            [TotalSales], DESC
        )
    )
RETURN
    MAXX(MaxSalesHour, [Hour])
```

13 Total Passengers

Purpose:

To count the number of unique passengers

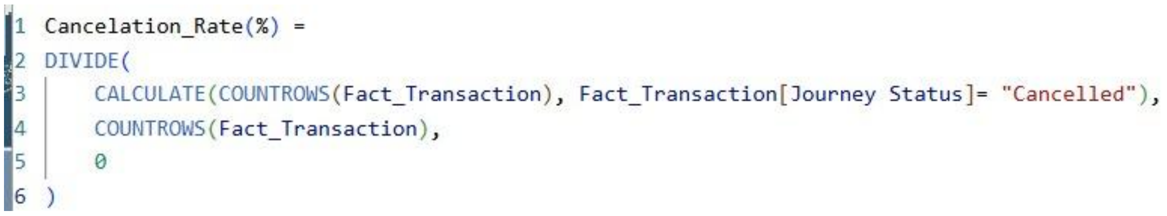


```
Total Passengers = DISTINCTCOUNT(Fact_Transaction[Transaction ID])
```

14 Cancelation_Rate(%)

Purpose: To calculate the percentage of cancelled journeys out of all transactions**.**

DAX Formula:



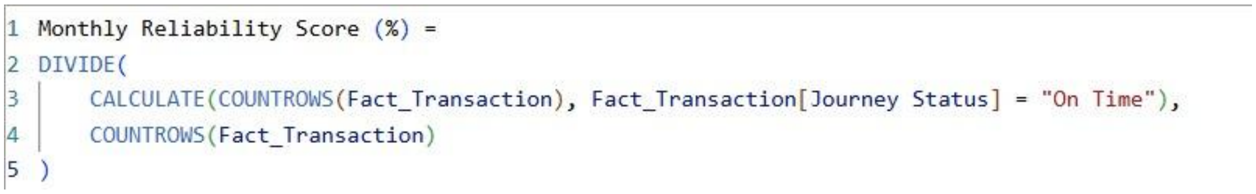
```
Cancelation_Rate(%) =
DIVIDE(
    CALCULATE(COUNTROWS(Fact_Transaction), Fact_Transaction[Journey Status]= "Cancelled"),
    COUNTROWS(Fact_Transaction),
    0
)
```

15 Monthly Reliability Score (%)

Purpose:

To measure the reliability of train journeys by calculating the percentage of journeys that were “On Time” out of all journeys within a month.

DAX Formula:



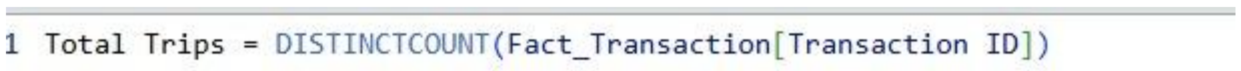
```
Monthly Reliability Score (%) =
DIVIDE(
    CALCULATE(COUNTROWS(Fact_Transaction), Fact_Transaction[Journey Status] = "On Time"),
    COUNTROWS(Fact_Transaction)
)
```

16. Total Trips

Purpose:

To count the total number of unique trips or bookings

DAX Formula:



```
Total Trips = DISTINCTCOUNT(Fact_Transaction[Transaction ID])
```

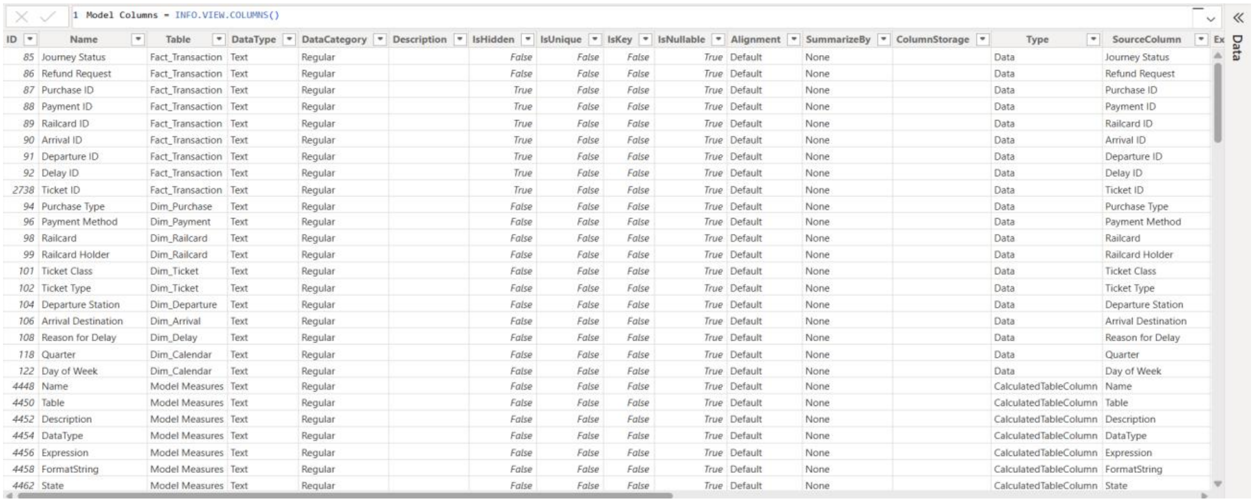
5.Data dectionary :

This is a document that explains what data is available, where it's stored (tables/columns), how it's defined (data types/descriptions), how tables connect (relationships), and how calculations work (measures).

(1) Model Columns (Data Fields)

Purpose:

Helps users understand what each column means and how it should be used in analysis.



```
1 Model Columns = INFO.VIEW.COLUMNS()
```

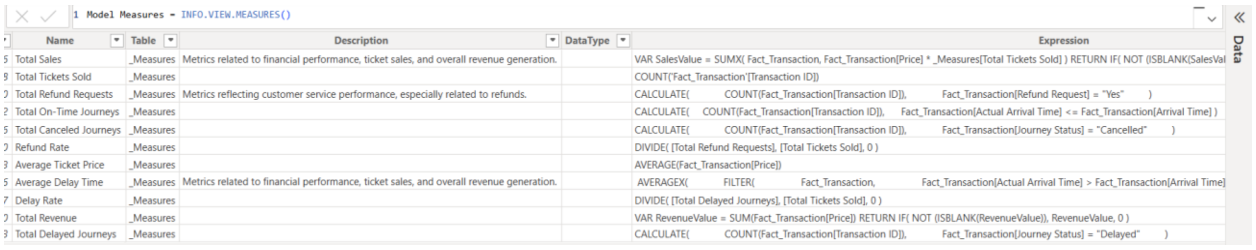
ID	Name	Table	DataType	DataCategory	Description	IsHidden	IsUnique	IsKey	IsNullable	Alignment	SummarizeBy	ColumnStorage	Type	SourceColumn	Ex
85	Journey Status	Fact_Transaction	Text	Regular		False	False	False	True	Default	None		Data	Journey Status	
86	Refund Request	Fact_Transaction	Text	Regular		False	False	False	True	Default	None		Data	Refund Request	
87	Purchase ID	Fact_Transaction	Text	Regular		True	False	False	True	Default	None		Data	Purchase ID	
88	Payment ID	Fact_Transaction	Text	Regular		True	False	False	True	Default	None		Data	Payment ID	
89	Railcard ID	Fact_Transaction	Text	Regular		True	False	False	True	Default	None		Data	Railcard ID	
90	Arrival ID	Fact_Transaction	Text	Regular		True	False	False	True	Default	None		Data	Arrival ID	
91	Departure ID	Fact_Transaction	Text	Regular		True	False	False	True	Default	None		Data	Departure ID	
92	Delay ID	Fact_Transaction	Text	Regular		True	False	False	True	Default	None		Data	Delay ID	
2738	Ticket ID	Fact_Transaction	Text	Regular		True	False	False	True	Default	None		Data	Ticket ID	
94	Purchase Type	Dim_Purchase	Text	Regular		False	False	False	True	Default	None		Data	Purchase Type	
96	Payment Method	Dim_Payment	Text	Regular		False	False	False	True	Default	None		Data	Payment Method	
98	Railcard	Dim_Railcard	Text	Regular		False	False	False	True	Default	None		Data	Railcard	
99	Railcard Holder	Dim_Railcard	Text	Regular		False	False	False	True	Default	None		Data	Railcard Holder	
101	Ticket Class	Dim_Ticket	Text	Regular		False	False	False	True	Default	None		Data	Ticket Class	
102	Ticket Type	Dim_Ticket	Text	Regular		False	False	False	True	Default	None		Data	Ticket Type	
104	Departure Station	Dim_Departure	Text	Regular		False	False	False	True	Default	None		Data	Departure Station	
106	Arrival Destination	Dim_Arrival	Text	Regular		False	False	False	True	Default	None		Data	Arrival Destination	
108	Reason for Delay	Dim_Delay	Text	Regular		False	False	False	True	Default	None		Data	Reason for Delay	
118	Quarter	Dim_Calendar	Text	Regular		False	False	False	True	Default	None		Data	Quarter	
122	Day of Week	Dim_Calendar	Text	Regular		False	False	False	True	Default	None		Data	Day of Week	
4448	Name	Model Measures	Text	Regular		False	False	False	True	Default	None		CalculatedTableColumn	Name	
4450	Table	Model Measures	Text	Regular		False	False	False	True	Default	None		CalculatedTableColumn	Table	
4452	Description	Model Measures	Text	Regular		False	False	False	True	Default	None		CalculatedTableColumn	Description	
4454	DataType	Model Measures	Text	Regular		False	False	False	True	Default	None		CalculatedTableColumn	DataType	
4456	Expression	Model Measures	Text	Regular		False	False	False	True	Default	None		CalculatedTableColumn	Expression	
4458	FormatString	Model Measures	Text	Regular		False	False	False	True	Default	None		CalculatedTableColumn	FormatString	
4462	State	Model Measures	Text	Regular		False	False	False	True	Default	None		CalculatedTableColumn	State	

This is a listing of all columns (fields) in your dataset, including:

- Table Name (`Fact_Transaction`, `Dim_Payment`…etc) Which table the column belongs to.
- Column Name (`Journey Status`, `Payment Method`…etc) The name of the field.
- Data Type (`Text`…etc) The kind of data stored (text, numbers, dates).
- Description: (where available) What the column represents (`Ticket Class` = Economy/Business…etc).
- Key Properties :Whether a column is a unique identifier (`IsKey`), hidden (`IsHidden`), or nullable (`IsNullable`).

(2) Model Measures (Calculated Metrics):

Purpose: Ensures consistent calculations across reports and clarifies how metrics are derived**.**



```
Model Measures = INFO.VIEW.MEASURES()
```

	Name	Table	Description	DataType	Expression
5	Total Sales	_Measures	Metrics related to financial performance, ticket sales, and overall revenue generation.		VAR SalesValue = SUMX(Fact_Transaction, Fact_Transaction[Price] * _Measures[Total Tickets Sold]) RETURN IF(NOT (ISBLANK(SalesVal
3	Total Tickets Sold	_Measures			COUNT('Fact_Transaction'[Transaction ID])
0	Total Refund Requests	_Measures	Metrics reflecting customer service performance, especially related to refunds.		CALCULATE(COUNT(Fact_Transaction[Transaction ID]), Fact_Transaction[Refund Request] = "Yes")
2	Total On-Time Journeys	_Measures			CALCULATE(COUNT(Fact_Transaction[Transaction ID]), Fact_Transaction[Actual Arrival Time] <= Fact_Transaction[Arrival Time])
5	Total Canceled Journeys	_Measures			CALCULATE(COUNT(Fact_Transaction[Transaction ID]), Fact_Transaction[Journey Status] = "Cancelled")
0	Refund Rate	_Measures			DIVIDE([Total Refund Requests], [Total Tickets Sold], 0)
3	Average Ticket Price	_Measures			AVERAGE(Fact_Transaction[Price])
5	Average Delay Time	_Measures	Metrics related to financial performance, ticket sales, and overall revenue generation.		AVERAGEX(FILTER(Fact_Transaction, Fact_Transaction[Actual Arrival Time] > Fact_Transaction[Arrival Time]
7	Delay Rate	_Measures			DIVIDE([Total Delayed Journeys], [Total Tickets Sold], 0)
0	Total Revenue	_Measures			VAR RevenueValue = SUM(Fact_Transaction[Price]) RETURN IF(NOT (ISBLANK(RevenueValue)), RevenueValue, 0)
3	Total Delayed Journeys	_Measures			CALCULATE(COUNT(Fact_Transaction[Transaction ID]), Fact_Transaction[Journey Status] = "Delayed")

This section defines key business calculations (KPIs) used in reports as:

- Total Revenue = Sum of all ticket prices.

- Refund Rate = Percentage of tickets refunded.

- Delay Rate = Percentage of delayed journeys.

- Average Ticket Price = Mean price per ticket.

Each measure includes:

- Name

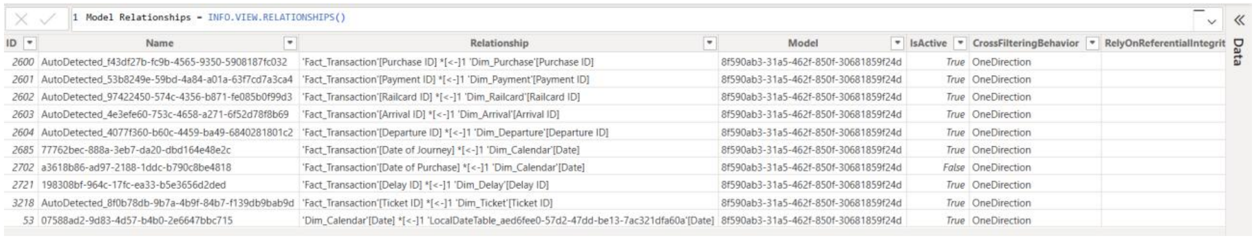
- Formula

- Description

(3) Model Relationships (Data Links):

Purpose:

Shows how data flows between tables, ensuring accurate filtering and analysis.



```
Model Relationships = INFO.VIEW.RELATIONSHIPS()
```

ID	Name	Relationship	Model	IsActive	CrossFilteringBehavior	RelyOnReferentialIntegrit
2600	AutoDetected_f43df27b-fc9b-4565-9350-5908187fc032	'Fact_Transaction'[Purchase ID] *[<-]1 'Dim_Purchase'[Purchase ID]	8f590ab3-31a5-462f-850f-30681859f24d	True	OneDirection	
2601	AutoDetected_53b8249e-59bd-4a84-a01a-63f7cd7a3ca4	'Fact_Transaction'[Payment ID] *[<-]1 'Dim_Payment'[Payment ID]	8f590ab3-31a5-462f-850f-30681859f24d	True	OneDirection	
2602	AutoDetected_97422450-574c-4356-b871-fe085b0f99d3	'Fact_Transaction'[Railcard ID] *[<-]1 'Dim_Railcard'[Railcard ID]	8f590ab3-31a5-462f-850f-30681859f24d	True	OneDirection	
2603	AutoDetected_4e3efe60-753c-4658-a271-6f52d78f8b69	'Fact_Transaction'[Arrival ID] *[<-]1 'Dim_Arrival'[Arrival ID]	8f590ab3-31a5-462f-850f-30681859f24d	True	OneDirection	
2604	AutoDetected_4077f360-b60c-4459-ba49-6840281801c2	'Fact_Transaction'[Departure ID] *[<-]1 'Dim_Departure'[Departure ID]	8f590ab3-31a5-462f-850f-30681859f24d	True	OneDirection	
2685	77762bec-888a-3eb7-da20-dbd164e48e2c	'Fact_Transaction'[Date of Journey] *[<-]1 'Dim_Calendar'[Date]	8f590ab3-31a5-462f-850f-30681859f24d	True	OneDirection	
2702	a3618b86-ad97-2188-1ddc-b790c8be4818	'Fact_Transaction'[Date of Purchase] *[<-]1 'Dim_Calendar'[Date]	8f590ab3-31a5-462f-850f-30681859f24d	False	OneDirection	
2721	198308bf-964c-17fc-ea33-b5e3656d2ded	'Fact_Transaction'[Delay ID] *[<-]1 'Dim_Delay'[Delay ID]	8f590ab3-31a5-462f-850f-30681859f24d	True	OneDirection	
3218	AutoDetected_8f0b78db-9b7a-4b9f-84b7-f139db9bab9d	'Fact_Transaction'[Ticket ID] *[<-]1 'Dim_Ticket'[Ticket ID]	8f590ab3-31a5-462f-850f-30681859f24d	True	OneDirection	
53	07588ad2-9d83-4d57-b4b0-2e6647bbc715	'Dim_Calendar'[Date] *[<-]1 'LocalDateTable_aed6fee0-57d2-47dd-be13-7ac321dfa60a'[Date]	8f590ab3-31a5-462f-850f-30681859f24d	True	OneDirection	

This documents how tables are connected in the database, such as:

- `Fact_Transaction[Payment ID]` → `Dim_Payment[Payment ID]`

- `Fact_Transaction[Ticket ID]` → `Dim_Ticket[Ticket ID]`

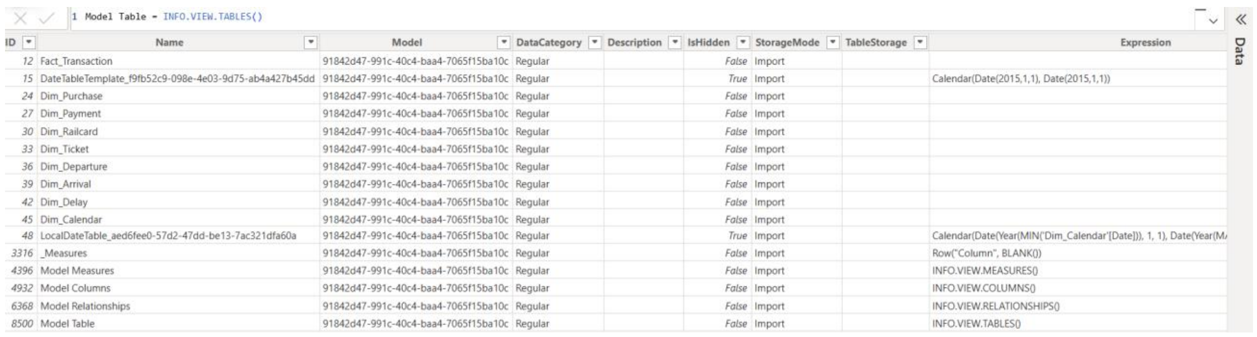
Key details:

- Source & Target Tables– Which columns link together.
- Cross-Filtering Behavior– How filters propagate

(4) Model Table :

Purpose:

- Clarifies which tables hold raw data vs. descriptive attributes.
- Helps users locate data for analysis (e.g., revenue in Fact_Transaction, ticket types in Dim_Ticket).



```
1 Model Table = INFO.VIEW.TABLES()
```

ID	Name	Model	DataCategory	Description	IsHidden	StorageMode	TableStorage	Expression
12	Fact_Transaction	91842d47-991c-40c4-baa4-7065f15ba10c	Regular		False	Import		
15	DateTableTemplate_f9fb52c9-098e-4e03-9d75-ab4a427b45dd	91842d47-991c-40c4-baa4-7065f15ba10c	Regular		True	Import		Calendar(Date(2015,1,1), Date(2015,1,1))
24	Dim_Purchase	91842d47-991c-40c4-baa4-7065f15ba10c	Regular		False	Import		
27	Dim_Payment	91842d47-991c-40c4-baa4-7065f15ba10c	Regular		False	Import		
30	Dim_Railcard	91842d47-991c-40c4-baa4-7065f15ba10c	Regular		False	Import		
33	Dim_Ticket	91842d47-991c-40c4-baa4-7065f15ba10c	Regular		False	Import		
36	Dim_Departure	91842d47-991c-40c4-baa4-7065f15ba10c	Regular		False	Import		
39	Dim_Arrival	91842d47-991c-40c4-baa4-7065f15ba10c	Regular		False	Import		
42	Dim_Delay	91842d47-991c-40c4-baa4-7065f15ba10c	Regular		False	Import		
45	Dim_Calendar	91842d47-991c-40c4-baa4-7065f15ba10c	Regular		False	Import		
48	LocalDateTable_aed6fee0-57d2-47dd-be13-7ac321dfa60a	91842d47-991c-40c4-baa4-7065f15ba10c	Regular		True	Import		Calendar(Date(Year(MIN('Dim_Calendar'[Date])), 1, 1), Date(Year(M
3316	_Measures	91842d47-991c-40c4-baa4-7065f15ba10c	Regular		False	Import		Row("Column", BLANK())
4396	Model Measures	91842d47-991c-40c4-baa4-7065f15ba10c	Regular		False	Import		INFO.VIEW.MEASURES()
4932	Model Columns	91842d47-991c-40c4-baa4-7065f15ba10c	Regular		False	Import		INFO.VIEW.COLUMNS()
6368	Model Relationships	91842d47-991c-40c4-baa4-7065f15ba10c	Regular		False	Import		INFO.VIEW.RELATIONSHIPS()
8500	Model Table	91842d47-991c-40c4-baa4-7065f15ba10c	Regular		False	Import		INFO.VIEW.TABLES()

6.Recommendations for Stakeholders:

Based on data insights, we propose the following recommendations to enhance operational efficiency and customer satisfaction:

Investigate Sales Drop Post-May

The complete halt in sales from June to December requires immediate investigation. This could be due to:

- Service disruptions
- Operational or seasonal pauses

Review High Refund Activity

Focus on improving service quality or refund policies at Liverpool Lime Street and Edinburgh Waverley to reduce refund rates.

Optimize Debit Card Transactions

Given their high refund rate, enhance clarity of policies or improve reliability for debit card payments, especially for off-peak tickets.

Enhance Online Sales Strategy

Strengthen marketing around online channels and credit card payments, which have shown strong performance.

Monitor Ticket Types

Assess the reasons behind higher refund rates and request volumes in off-peak and advance tickets to mitigate loss.

7.Conclusion

This project provided a comprehensive analysis of train operations, focusing on performance, revenue, and customer satisfaction. Through data cleaning, modeling, and advanced DAX calculations, we identified key insights such as delay patterns, cancellation rates, and revenue trends.

By leveraging Power BI's analytical capabilities, stakeholders can make data-driven decisions to optimize train schedules, improve service reliability, and enhance passenger experiences. The recommended strategies, including operational improvements, dynamic pricing, and customer engagement initiatives, will contribute to a more efficient and profitable railway system.

Going forward, continuous data monitoring and predictive analytics can further enhance decision-making, ensuring sustained growth and improvement in train services.