



# UK Train Rides

---

## Introduction to British Railways

The United Kingdom's railway network is among the oldest and most developed in the world. British Railways started with the groundbreaking Stockton and Darlington Railway in 1825, ushering in the age of rail transportation. Over time, the network expanded rapidly to become a vital national infrastructure for commuting, goods transport, and tourism.

Today, the UK's railways cover more than 10,000 miles of track and are managed by multiple private train operating companies under the oversight of Network Rail. Major hubs include London King's Cross, Birmingham New Street, and Manchester Piccadilly. The system combines historic charm with modern advancements like high-speed services (e.g., LNER, Avanti West Coast), playing a key role in economic development and regional connectivity.

Rail transport in Britain reflects a blend of tradition and innovation, maintaining its relevance in the era of sustainable mobility.

## Project Overview

This document provides a detailed overview of the "UK Train Rides Project." The goal was to analyze real-world railway transportation data extracted from **railway.csv**, cleanse and model it, and finally visualize insights using **Power BI**.

The motivation behind the project stems from the importance of efficient railway systems in supporting urban mobility, reducing congestion, and promoting green transportation. By extracting actionable insights, the project aids in better decision-making for stakeholders like transport companies, city planners, and travelers.

The work involved various stages: initial data exploration, data cleaning, advanced modeling, and comprehensive visualization through dashboards.

---

### Project Objectives

The project aimed to achieve the following objectives, each critical to understanding railway transportation patterns:

- 1. Identify Ticket Sales Trends:** Understand when and where ticket sales peak, whether seasonally, monthly, or weekly.
  - 2. Discover Popular Routes:** Recognize the most frequented routes based on the number of passengers and ticket revenue.
  - 3. Analyze Ticket Pricing:** Evaluate how ticket prices vary across distances, routes, and seasons.
  - 4. Understand Customer Behavior:** Identify preferences for ticket types (single vs return) and peak travel times.
  - 5. Build an Interactive Dashboard:** Create a dynamic and user-friendly Power BI dashboard offering key insights for non-technical users.
- 

### Data Source and Description

The dataset used, **railway.csv**, simulates data from UK train services. Key columns include:

### Data Dictionary



Column Name	Data Type	Description
Transaction ID	Text	Unique identifier for each transaction
Date of Purchase	Date	Date when ticket was purchased
Time of Purchase	Time	Time when ticket was purchased
Purchase Type	Text	Where ticket was purchased (Online/Station)
Payment Method	Text	Payment type (Credit Card, Debit Card, Contactless)
Railcard	Text	Discount card type (Adult, Senior, Disabled, None)
Ticket Class	Text	Service class (Standard/First Class)

<b>Ticket Type</b>	Text	Fare type (Advance, Off-Peak, Anytime)
<b>Price</b>	Currency	Ticket price in GBP
<b>Departure Station</b>	Text	Origin station
<b>Arrival Destination</b>	Text	Destination station
<b>Date of Journey</b>	Date	Scheduled travel date
<b>Departure Time</b>	Time	Scheduled departure time
<b>Arrival Time</b>	Time	Scheduled arrival time
<b>Actual Arrival Time</b>	Time	Real arrival time (blank if cancelled)
<b>Journey Status</b>	Text	On Time, Delayed, or Cancelled
<b>Reason for Delay</b>	Text	Explanation when delayed/cancelled
<b>Refund Request</b>	Text	Yes/No indicator for refund requests

Each data field provides a dimension for analysis, helping understand behaviors across geography, time, and economics.

#### **Additional Custom Tables Created:**

**Dim\_calender:** A table with all the dates, day by day, from the first purchase to the last story in your data.

 		1 Dim_calender = CALENDAR(MIN(railway[Date of Purchase]), MAX(railway[Date of Purchase]))				
Date	Month.No.	Month	Year	weekdays	weekDay	Weekday/Weekend
12/8/2023 12:00:00 AM	12	December	2023	5	Friday	Weekday
12/9/2023 12:00:00 AM	12	December	2023	6	Saturday	Weekend
12/10/2023 12:00:00 AM	12	December	2023	7	Sunday	Weekend
12/11/2023 12:00:00 AM	12	December	2023	1	Monday	Weekday
12/12/2023 12:00:00 AM	12	December	2023	2	Tuesday	Weekday
12/13/2023 12:00:00 AM	12	December	2023	3	Wednesday	Weekday
12/14/2023 12:00:00 AM	12	December	2023	4	Thursday	Weekday
12/15/2023 12:00:00 AM	12	December	2023	5	Friday	Weekday
12/16/2023 12:00:00 AM	12	December	2023	6	Saturday	Weekend
12/17/2023 12:00:00 AM	12	December	2023	7	Sunday	Weekend
12/18/2023 12:00:00 AM	12	December	2023	1	Monday	Weekday
12/19/2023 12:00:00 AM	12	December	2023	2	Tuesday	Weekday
12/20/2023 12:00:00 AM	12	December	2023	3	Wednesday	Weekday
12/21/2023 12:00:00 AM	12	December	2023	4	Thursday	Weekday
12/22/2023 12:00:00 AM	12	December	2023	5	Friday	Weekday
12/23/2023 12:00:00 AM	12	December	2023	6	Saturday	Weekend
12/24/2023 12:00:00 AM	12	December	2023	7	Sunday	Weekend
12/25/2023 12:00:00 AM	12	December	2023	1	Monday	Weekday
12/26/2023 12:00:00 AM	12	December	2023	2	Tuesday	Weekday
12/27/2023 12:00:00 AM	12	December	2023	3	Wednesday	Weekday
12/28/2023 12:00:00 AM	12	December	2023	4	Thursday	Weekday
12/29/2023 12:00:00 AM	12	December	2023	5	Friday	Weekday
12/30/2023 12:00:00 AM	12	December	2023	6	Saturday	Weekend
Dim_calender (145 rows)						

## Data Analysis Questions

The project addressed the following questions:

1. **Top 5 Stations by Passenger Volume:** Identify the most popular stations.
2. **Most Profitable Routes:** Which routes generated the most revenue?
3. **Monthly Ride Trends:** When do people travel the most?
4. **Ticket Type Preferences:** Are travelers preferring advanced or off-peak tickets?
5. **Route Pricing Analysis:** What is the average price by route?
6. **Travel Time vs Price:** Does a longer travel time always mean higher ticket price?

## General Data Understanding:

### 1.Data Scope:

- What is the time range covered by the dataset (earliest and latest Date of Journey)?
- How many unique transactions are recorded, and are there any duplicates?

## 2.Missing or Incomplete Data:

- Which columns have missing values, and how significant are these missing entries (e.g., Reason for Delay and Actual Arrival Time)?
- Are there any patterns in the missing data? For example, are certain routes or times more likely to lack delay reasons?

## 3.Data Distribution:

- How is the price distributed? What are the minimum, maximum, average, and median prices?
- Which Departure Station and Arrival Destination are the most frequent?

## 4.Understanding Journey Details:

### *Patterns in Timing:*

- What are the most common journey times and days of travel?
- Are there specific times or days when delays are more frequent?

## 5.Delay Analysis:

- Which routes or stations experience the most delays?
- What are the most frequent reasons for delays?
- Are certain reasons associated with specific times or stations?

## 6.Ticket and Purchase Behavior:

### *Customer Preferences:*

- Which Purchase Type (online vs. station) is more popular?
- What is the distribution of Ticket Class and Ticket Type?
- Are some classes or types more prone to delays?

## 7.Railcard Usage:

- How many tickets were purchased with a railcard?
- Which type of railcard is most commonly used?
- How does railcard usage affect ticket prices and journey delays?

## 8.Revenue and Pricing Analysis:

### *Revenue Insights:*

- What is the total revenue generated from ticket sales?
- How does revenue vary by station, ticket type, or ticket class?

## 9.Price Comparisons:

- How do ticket prices vary by departure station, arrival destination, or journey distance?
- Are prices higher for tickets purchased at the station compared to online?

## 10. Customer Satisfaction:

### *Refund Trends:*

- How many refund requests were made? What percentage of total transactions does this represent?
- Is there a relationship between delayed journeys and refund requests?

## 11. Journey Status:

- What proportion of journeys are “On Time” versus “Delayed”?
- Are there specific stations, routes, or times where on-time performance is consistently better or worse?

## 12. Advanced Relationships:

### *Correlations and Dependencies:*

- Is there a correlation between ticket price and likelihood of delay?
- Do certain stations have consistently higher or lower prices for similar routes?

---

## Data Cleaning Process

### Data cleaning was critical for reliable analysis:

1. Null Value Handling: Records with missing station names, dates, or ticket prices were removed.
2. Standardization: Uniform formatting was applied to station names, ticket types, and date fields.
3. Outlier Removal: Outliers such as extremely high-ticket prices or implausible travel times were filtered.
4. Derived Fields: Revenue = Ticket Price x Passenger Count. Travel Day extracted from Travel Date.
5. Duplicate Elimination: Duplicate records were identified and removed to avoid inflating metrics.

---

Data cleaning was conducted using **Power Query** to ensure consistency and remove

any irrelevant information. Below are the detailed steps performed:

### Specific Cleansing Steps

#### Date/Time Handling

##### Issues Identified:

1. "Date of Purchase" includes time (00:00:00) when it shouldn't
2. "Time of Purchase" is separate but should be combined with date for analysis
3. "Date of Journey" includes irrelevant time component

##### Solutions:

- Power query Transform tab >>>>>>> data type Empty Cells

#### Empty Cells

##### Issues Identified:

Preserve meaningful nulls (like cancelled journeys)

B. "Actual Arrival Time" Column

Current State: Null when journey is cancelled

##### Solution:

Keep as null (indicates cancellation)

Replace meaningless nulls with defaults

A. "Reason for Delay" Column

Current State: Null when journey is "On Time"

##### Solution:

Replace with "No Delay"

Handling Redundancy in "Reason for Delay" Data

Common redundancy patterns found:

Multiple variations of the same reason (e.g., "Signal Failure" vs "Signal failure")

Similar reasons with different wording (Staffing vs Staff Shortage, Weather Conditions vs Weather)

#### Station Name Standardization

**Issues Identified:**

1. Potential inconsistencies in station naming (e.g., "London Paddington" vs "Paddington", Edinburgh vs Edinburgh Waverley, Durham vs Durham County))

**Solutions:**

From Power query

Replace Value

**Handling Missing Values****Issues Identified:**

1. Blank "Reason for Delay" when status is "On Time"
2. Blank "Actual Arrival Time" for cancelled journeys

**Solutions:**

Power query

Replace Value

**Final Output Structure**

The cleaned dataset should have these optimal fields:

1. Transaction ID
2. Purchase Date
3. Purchase Time
4. Purchase Date Time (combined)
5. Purchase Type
6. Payment Method
7. Railcard Type
8. Ticket Class
9. Ticket Type
10. Price
11. Price Valid (flag)
12. Departure Station (standardized)
13. Arrival Station (standardized)
14. Journey Date
15. Departure Time
16. Arrival Time
17. Scheduled Duration (min)



18. Actual Arrival Time
19. Journey Status
20. Reason for Delay (with "N/A" for blanks)
21. Refund Request

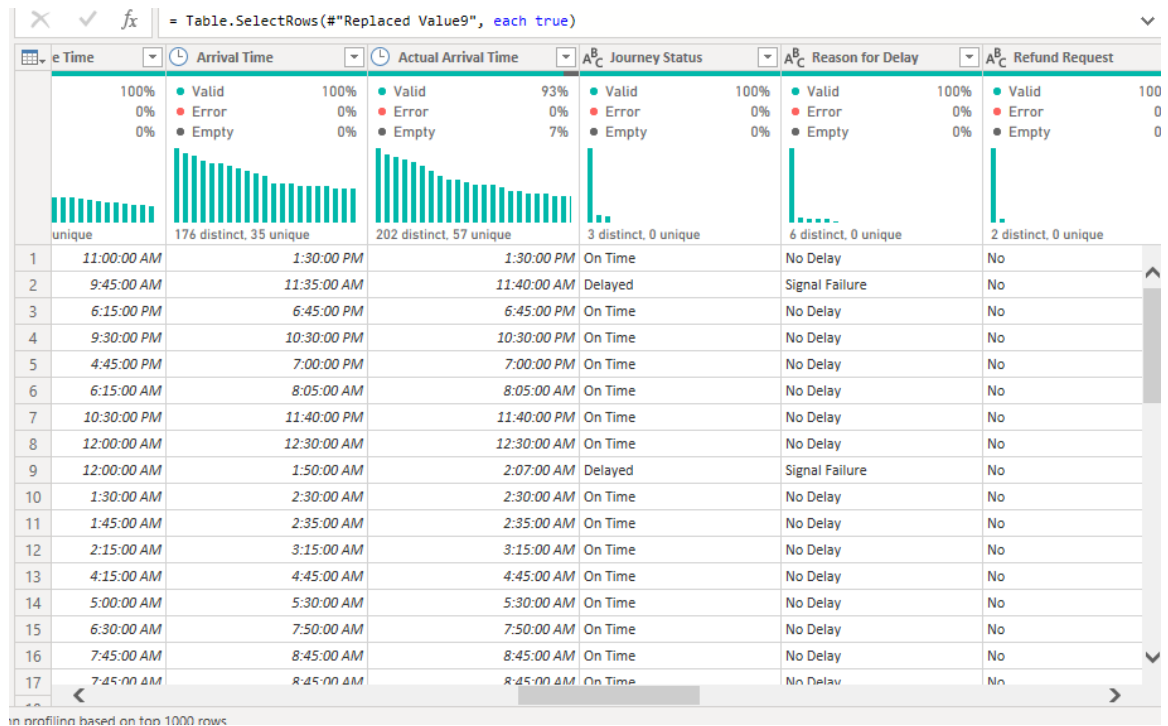
### Power Query Implementation Steps

- ❖ Open Power BI Desktop
- ❖ Get Data → Excel → Select your file
- ❖ In Power Query Editor:
- ❖ Apply all transformations above
- ❖ Set proper data types for each column
- ❖ Create calculated columns as needed
- ❖ Close & apply to load the cleaned data

Table: TransformColumnTypes(("#Promoted Headers",{"Transaction ID", type text}, {"Date of Purchase", type date}, {"Time of Purchase", type time}, {"Date of Journey", type date}, {"Departure Time", type time}, {"Arrival Time", type time}, {"Actual Arrival Time", type time}, {"Journey Status", type text}, {"Reason for Delay", type text}))

	Date of Journey	Departure Time	Arrival Time	Actual Arrival Time	Journey Status	Reason for Delay
1	1/1/2024	11:00:00 AM	1:30:00 PM	1:30:00 PM	On Time	
2	1/1/2024	9:45:00 AM	11:35:00 AM	11:40:00 AM	Delayed	Signal Failure
3	1/2/2024	6:15:00 PM	6:45:00 PM	6:45:00 PM	On Time	
4	1/1/2024	9:30:00 PM	10:30:00 PM	10:30:00 PM	On Time	
5	1/1/2024	4:45:00 PM	7:00:00 PM	7:00:00 PM	On Time	
6	1/1/2024	6:15:00 AM	8:05:00 AM	8:05:00 AM	On Time	
7	1/1/2024	10:30:00 PM	11:40:00 PM	11:40:00 PM	On Time	
8	1/1/2024	12:00:00 AM	12:30:00 AM	12:30:00 AM	On Time	
9	1/1/2024	12:00:00 AM	1:50:00 AM	2:07:00 AM	Delayed	Signal Failure
10	1/1/2024	1:30:00 AM	2:30:00 AM	2:30:00 AM	On Time	
11	1/1/2024	1:45:00 AM	2:35:00 AM	2:35:00 AM	On Time	
12	1/1/2024	2:15:00 AM	3:15:00 AM	3:15:00 AM	On Time	
13	1/1/2024	4:15:00 AM	4:45:00 AM	4:45:00 AM	On Time	
14	1/1/2024	5:00:00 AM	5:30:00 AM	5:30:00 AM	On Time	
15	1/1/2024	6:30:00 AM	7:50:00 AM	7:50:00 AM	On Time	
16	1/1/2024	7:45:00 AM	8:45:00 AM	8:45:00 AM	On Time	
17	1/1/2024	7:45:00 AM	8:45:00 AM	8:45:00 AM	On Time	

1-data before cleaning



## 2-data after cleaning

This cleaning ensured consistency and improved model performance.

## Data Modeling

A **Star Schema** was designed:

### Fact Table:

- Train Rides (Passenger data, Ticket Price, Distance, etc.)

### Dimension Tables:

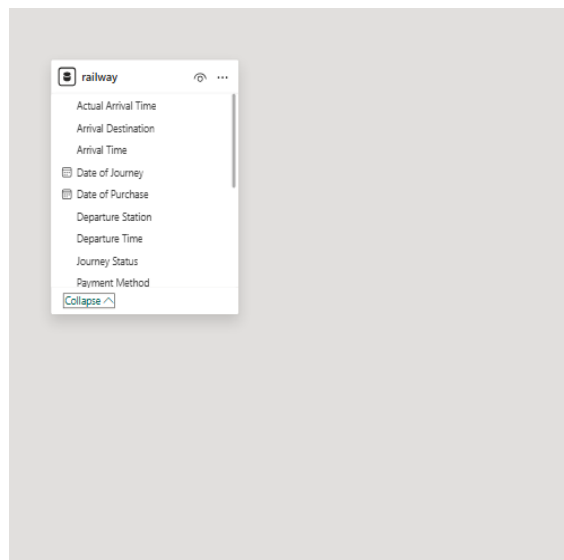
- Stations Table (Origin and Destination).
- Date Table (Generated calendar based on Travel Date).
- Ticket Type Table (Single or Return).

### Relationships:

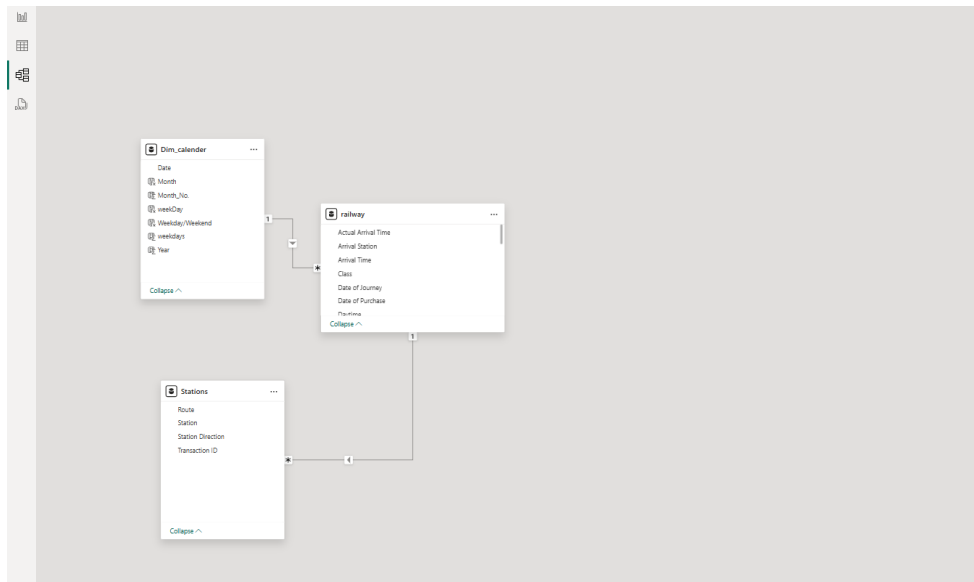
- One-to-many relationships connecting dimensions to the fact table.

### **Advantages of Star Schema:**

- Simpler, faster querying.
- Clear structure for slicing and dicing data.
- Enhanced Power BI performance.



**3-Fact Table railway before modeling**



#### 4-final data model

---

# Analysis Methodology

## Dax Measures

### 1. Cancelled Journeys

- **Description:** The number of journeys that were cancelled.
- **Purpose:** Helps monitor service reliability and detect operational issues causing frequent cancellations.
- **Calculation (DAX):**

**Cancelled Journeys = CALCULATE([Total Transactions], railway[Journey Status] = "Cancelled")**

---

### 2. Count of Refund Requests

- **Description:** The number of refund requests submitted by customers.
- **Purpose:** Measures customer dissatisfaction and potential revenue losses.
- **Calculation (DAX):**

**Count of Refund Requests = CALCULATE(COUNT(railway[Refund Request]), railway[Refund Request] = "Yes")**

---

### 3. Count of Routes

- **Description:** The total number of unique train routes.
- **Purpose:** Helps assess operational complexity and network size.
- **Calculation (DAX):**

**Count of Routes = DISTINCTCOUNT(railway[Route])**

---

### 4. Count of Stations

- **Description:** The total number of unique stations (departure and arrival).
- **Purpose:** Supports station-level service analysis.
- **Calculation (DAX):**

Count of Stations = DISTINCTCOUNT('AllStations'[Station])

---

#### 5. Delayed Journeys

- **Description:** The number of journeys that were delayed.
- **Purpose:** Tracks punctuality and identifies service inefficiencies.
- **Calculation (DAX):**

Delayed Journeys = CALCULATE([Total Transactions], railway[Journey Status] = "Delayed")

---

#### 6. Incomes

- **Description:** The total sold ticket price.
- **Purpose:** Measures financial performance and supports revenue analysis.
- **Calculation (DAX):**

Incomes = SUM(railway[Price])

---

#### 7. On Time Journeys

- **Description:** The number of journeys that arrived and departed on time.
- **Purpose:** Measures service quality and customer satisfaction.
- **Calculation (DAX):**

On Time Journeys = CALCULATE([Total Transactions], railway[Journey Status] = "On Time")

---

#### 8. Refunded Income

- **Description:** The total amount of revenue refunded to customers.
- **Purpose:** Tracks financial loss due to service issues and supports refund policy evaluation.
- **Calculation (DAX):**

Refunded Income = CALCULATE([Incomes], FILTER(railway, [Total Refund Request]))

---

### 9. Total Affected Journeys

- **Description:** The sum of journeys that were either cancelled or delayed.
- **Purpose:** Measures the impact of disruptions on customer experience.
- **Calculation (DAX):**

**Total Affected Journeys = 'All Measures'[Cancelled Journeys] + 'All Measures'[Delayed Journeys]**

---

### 10. Total Refund Requests

- **Description:** The number of transactions that involved a refund request.
- **Purpose:** Helps assess service failures and customer service efficiency.
- **Calculation (DAX):**

**Total Refund Requests = CALCULATE([Total Transactions], railway[Refund Request] = "Yes")**

---

### 11. Total Transactions

- **Description:** The total number of journey transactions.
- **Purpose:** Serves as the baseline for calculating other KPIs.
- **Calculation (DAX):**

**Total Transactions = COUNT(railway[Transaction ID])**

---

### 12. All Stations

- **Description:** A unified table combining all unique departure and arrival stations.
- **Purpose:** Enables complete station-level analysis without duplicates.
- **Calculation (DAX):**

**All Stations =**

**UNION(**

```
SELECTCOLUMNS('railway', "Station", 'railway'[Departure Station]),  
SELECTCOLUMNS('railway', "Station", 'railway'[Arrival Station])  
)
```

---

## DAX Functions Used and Their Purpose

DAX Function	Purpose
<b>CALCULATE()</b>	Changes the filter context to evaluate an expression under specific conditions (like "Cancelled" journeys).
<b>COUNT()</b>	Counts the number of non-blank entries in a column (e.g., Refund Requests).
<b>DISTINCTCOUNT()</b>	Counts only the distinct (unique) values in a column (e.g., Routes, Stations).
<b>SUM()</b>	Adds all numeric values together (e.g., ticket Prices).
<b>FILTER()</b>	Creates a temporary table filtered by a specific condition (e.g., Refund Requests only).
<b>UNION()</b>	Merges two tables together into one (e.g., Departure and Arrival Stations).
<b>SELECTCOLUMNS()</b>	Creates a new table with selected columns, allowing renaming as needed.



# M Query

## 1- Journey Duration

### Code Breakdown:

```
if [Journey Status] = "Cancelled" then  
    null  
  
else if [Arrival Time] < [Departure Time] then  
    #duration(1, 0, 0, 0) + ([Arrival Time] - [Departure Time])  
  
else  
    [Arrival Time] - [Departure Time]
```

### Key Points:

- The #duration(1, 0, 0, 0) part is used to add one day (24 hours) to the time difference when the arrival time is earlier than the departure time, ensuring that the calculation accounts for possible date changes (e.g., crossing midnight).
- If the journey is cancelled, no duration is calculated (i.e., the result is null).
- If the arrival time is after the departure time, the duration is calculated normally as the difference between the two times.

This code ensures accurate journey duration calculations, even in edge cases like cancelled journeys or when the journey spans across different days.

---

### Purpose:

This code is used to calculate the duration between the departure and arrival times of a journey, while handling specific cases such as the journey being cancelled or the arrival time being earlier than the departure time.

---

### Explanation:

#### 1. If Journey Status is "Cancelled":

- If the [Journey Status] is "Cancelled," the result will be null. This means that no duration will be calculated for cancelled journeys.

## 2. Arrival Time is Before Departure Time:

- If the [Arrival Time] is earlier than the [Departure Time], it indicates that the journey might have crossed over midnight or there might be data issues. In this case, the duration is calculated by adding one full day (#duration(1, 0, 0, 0)) to the difference between [Arrival Time] and [Departure Time]. This ensures the result reflects the time difference correctly, accounting for the possibility of the arrival being the next day.

## 3. Standard Duration Calculation:

- If neither of the above conditions is met (i.e., the journey is not cancelled, and the arrival time is after the departure time), the duration is simply the difference between the [Arrival Time] and [Departure Time].

---

## 2- Journey Duration Category

### Code Breakdown:

```
if [Journey Status] = "Cancelled" then
```

```
    null
```

```
else if [Journey Duration] > #duration(0, 0, 0, 0) and [Journey Duration] <= #duration(0, 1, 0, 0) then
```

```
    "Short (<1h)"
```

```
else if [Journey Duration] > #duration(0, 1, 0, 0) and [Journey Duration] <= #duration(0, 2, 0, 0) then
```

```
    "Standard (1-2h)"
```

```
else if [Journey Duration] > #duration(0, 2, 0, 0) and [Journey Duration] <= #duration(0, 3, 0, 0) then
```

```
    "Extended (2-3h)"
```

```
else
```

```
    "Long-Distance (>3h)"
```

This code provides a clear categorization of journey durations, making it easier to analyze journey length data in terms of time intervals.

---

## Purpose:

This code is used to categorize the journey duration into different categories based on the value in the [Journey Duration] field. It checks the journey status and assigns a specific label based on the time range of the journey duration.

---

## Explanation:

### 1. If Journey Status is "Cancelled":

- If the [Journey Status] is "Cancelled," the output is set to null, meaning no further categorization is applied for cancelled journeys.

### 2. Journey Duration Categories:

- **Short (<1h):**  
If the journey duration is greater than 0 hours and less than or equal to 1 hour, the journey is categorized as "Short (<1h)."
  - **Standard (1-2h):**  
If the journey duration is greater than 1 hour and less than or equal to 2 hours, the journey is categorized as "Standard (1-2h)."
  - **Extended (2-3h):**  
If the journey duration is greater than 2 hours and less than or equal to 3 hours, the journey is categorized as "Extended (2-3h)."
  - **Long-Distance (>3h):**  
If none of the above conditions are met (i.e., the journey duration is greater than 3 hours), the journey is categorized as "Long-Distance (>3h)."
- 

## 3- Price Category

### Code

```
if [Price] >= 1 and [Price] <= 20 then "Very Cheap (1-20)"  
else if [Price] > 20 and [Price] <= 40 then "Low (21-40)"  
else if [Price] > 40 and [Price] <= 70 then "Moderate (41-70)"  
else if [Price] > 70 and [Price] <= 100 then "High (71-100)"  
else if [Price] > 100 and [Price] <= 200 then "Expensive (101-200)"
```

else "Extremely Expensive (>200)"

## Description

This M Query classifies ticket prices into **different pricing categories** based on the amount paid.

---

## Purpose

- To group journeys by price range.
  - Useful for price analysis, understanding customer spending behavior, and building price-segmented reports.
  - Helps easily visualize "cheap vs expensive" ticket purchases.
- 

## Explanation

- It checks the value of [Price] and places it into a category depending on its range.
  - It uses a simple **if-else if-else** structure.
  - The output is a **text label** that describes the price category.
- 

## Price Segments Created

Price Range	Label	Meaning
1–20	Very Cheap (1–20)	Very low-cost tickets
21–40	Low (21–40)	Affordable tickets
41–70	Moderate (41–70)	Mid-range tickets
71–100	High (71–100)	High-priced tickets
101–200	Expensive (101–200)	Very expensive tickets
More than 200	Extremely Expensive (>200)	Premium/very expensive tickets

#### 4- Column: Purchase-Journey Period

- **Description:**  
Calculates the time difference (duration) between the purchase date and the journey date.
- **Purpose:**  
Used to segment tickets based on how early or late they were purchased before the trip.
- **Calculation (M Query):**

Purchase-Journey Period = [Date of Journey] - [Date of Purchase]

---

#### 5-Purchase-Journey Period

##### Code

```
if [#"Purchase-Journey Period"] >= #duration(1, 0, 0, 0) and [#"Purchase-Journey Period"]  
<= #duration(3, 0, 0, 0) then "Last-Minute (<1-3d)"  
  
else if [#"Purchase-Journey Period"] >= #duration(3, 0, 0, 0) and [#"Purchase-Journey  
Period"] <= #duration(7, 0, 0, 0) then "Final Week (3-7d)"  
  
else if [#"Purchase-Journey Period"] > #duration(7, 0, 0, 0) and [#"Purchase-Journey  
Period"] <= #duration(14, 0, 0, 0) then "Short-Term (7-14d)"  
  
else if [#"Purchase-Journey Period"] > #duration(14, 0, 0, 0) and [#"Purchase-Journey  
Period"] <= #duration(21, 0, 0, 0) then "Mid-Term (14-21d)"  
  
else if [#"Purchase-Journey Period"] > #duration(21, 0, 0, 0) then "Long-Term (>21d)"  
  
else "Same Day"
```

---

##### Description

This M Query code classifies the **time difference** between the **purchase date** and the **journey date** into different booking periods.

It categorizes journeys based on how far in advance a customer purchased the ticket relative to their travel date.

---

## Purpose

- To segment customer purchasing behavior into meaningful time periods.
  - Useful for marketing, pricing strategy, and operational planning (e.g., identifying last-minute buyers vs. early planners).
  - Enables visualizing and analyzing sales patterns by booking window.
- 

## Explanation

- [Purchase-Journey Period] is expected to be a **Duration** type column, representing the number of days (and optionally time) between the **purchase** and the **journey**.
  - The function uses a series of **if-else** statements and compares the duration against fixed day ranges using #duration(days, hours, minutes, seconds).
  - The output is a **categorical label** based on the duration range.
- 

## Segments Created

Period Category	Condition (Days)	Meaning
Same Day	Less than 1 day	Ticket purchased on the journey day
Last-Minute (<1-3d)	Between 1 and 3 days	Very close to journey
Final Week (3-7d)	Between 3 and 7 days	Purchased within final week
Short-Term (7-14d)	Between 7 and 14 days	1–2 weeks before journey
Mid-Term (14-21d)	Between 14 and 21 days	2–3 weeks before journey
Long-Term (>21d)	More than 21 days	Booked well in advance

---

## 6-Daytime Category

### Code

```
if [Departure Time] >= #time(6,0,0) and [Departure Time] < #time(12,0,0) then  
    "Morning"
```

else if [Departure Time] >= #time(12,0,0) and [Departure Time] < #time(18,0,0) then

"Afternoon"

else if [Departure Time] >= #time(18,0,0) and [Departure Time] < #time(24,0,0) then

"Evening"

else "Mid-Night"

### Key Points:

- The code splits the day into four distinct periods: "Morning", "Afternoon", "Evening", and "Mid-Night", based on the value of the [Departure Time].
  - The #time(h, m, s) function is used to define specific time intervals for comparison.
  - This categorization helps in understanding the distribution of journeys over different times of the day, useful for time-based analysis or reporting.
- 

### Purpose:

This code categorizes the departure time of a journey into different time periods: Morning, Afternoon, Evening, or Mid-Night. It is based on the departure time in the [Departure Time] field.

---

### Explanation:

#### 1. Morning (6:00 AM to 12:00 PM):

- If the [Departure Time] is greater than or equal to 6:00 AM and less than 12:00 PM, the journey is categorized as "Morning."

#### 2. Afternoon (12:00 PM to 6:00 PM):

- If the [Departure Time] is greater than or equal to 12:00 PM and less than 6:00 PM, the journey is categorized as "Afternoon."

#### 3. Evening (6:00 PM to 12:00 AM):

- If the [Departure Time] is greater than or equal to 6:00 PM and less than 12:00 AM, the journey is categorized as "Evening."

#### 4. Mid-Night (12:00 AM to 6:00 AM):

- If the [Departure Time] falls outside the above conditions (i.e., between 12:00 AM and 6:00 AM), it is categorized as "Mid-Night."

## 7- Delay Duration

### Code

```
if [Journey Status] = "Cancelled" then  
    null  
  
else if [Actual Arrival Time] < [Arrival Time] then  
    #duration(1, 0, 0, 0) + ([Actual Arrival Time] - [Arrival Time])  
else  
    [Actual Arrival Time] - [Arrival Time]
```

### Key Points:

- **Handling "Cancelled" Journeys:** If the journey is cancelled, the result will be null, meaning no delay is calculated.
- **Crossing Midnight or Data Issues:** If the actual arrival time is earlier than the scheduled arrival time (perhaps due to crossing over midnight or data issues), the code adds 1 full day to the delay to account for the next day.
- **Normal Delay Calculation:** If the actual arrival time is later than or equal to the scheduled arrival time, the delay is simply the difference between the two times.

This code is useful for accurately calculating the delay in journeys, even when there are edge cases like early arrivals or cancelled journeys. It ensures that the delay duration is calculated appropriately in all scenarios.

---

### Purpose:

This code calculates the delay duration of a journey based on the actual arrival time compared to the scheduled arrival time. It handles special cases like cancelled journeys or situations where the actual arrival time is earlier than the scheduled arrival time (such as crossing over midnight).

---

### Explanation:

1. **If Journey Status is "Cancelled":**



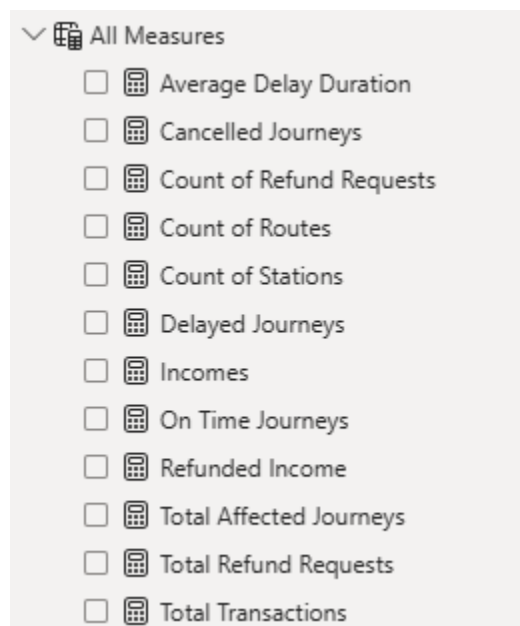
If the [Journey Status] is "Cancelled," the result is set to null. This means no delay duration will be calculated for cancelled journeys.

## 2. Actual Arrival Time is Before Scheduled Arrival Time:

If the [Actual Arrival Time] is earlier than the [Arrival Time] (which could happen due to data issues, time zone differences, or early arrivals), the duration is calculated by adding 1 full day (#duration(1, 0, 0, 0)) to the difference between the actual and scheduled arrival times. This ensures the result reflects the time difference correctly, especially when the arrival time is on the next day (i.e., crossing over midnight).

## 3. Standard Delay Duration Calculation:

If neither of the above conditions is met (i.e., the journey is not cancelled, and the actual arrival time is later than or equal to the scheduled arrival time), the delay duration is simply the difference between the [Actual Arrival Time] and the [Arrival Time].



### 5-All Measures

## Dashboard Creation and Design

The Power BI dashboard features four main pages:

### 1. Overview:

- KPIs: Total Revenue, Passenger Count, Avg Ticket Price.
- Map showing passengers by station.

### 2. Operational Performance

- Total affected journeys .
- Analysis of peak hours and days.

### 3. Ticket Sales Analysis:

- Single vs Return ticket distribution.
- Sales trends by month.

### 4. Purchase Preference Analysis:

- Top profitable routes.
- Distance vs Ticket Price scatter plot.

### 5. Temporal Customer Behavior:

- Analysis customer behavior and how they deal with train delays.

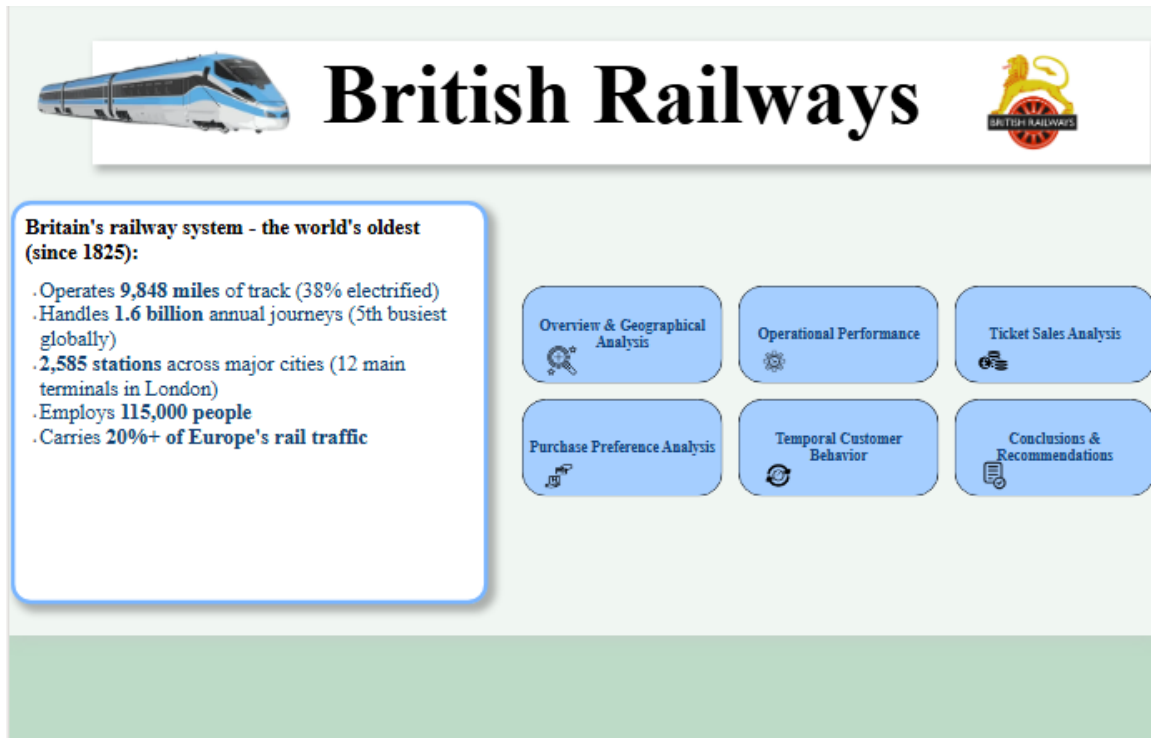
**A consistent blue-white color theme was applied for professional aesthetics.**

## Visuals:

1. Bar Chart
2. Stalkes Column Chart
3. Stalked Bar Chart

4. Column Chart
  5. Pie Chart
  6. Donut Chart
  7. Map
  8. Bowtie Chart by MAQ
  9. Line Chart
  10. Funnel Chart
  11. Ribbon Chart
  12. Table Chart
  13. Button Slicer
  14. Dropdown Slicer
  15. New Card
  16. Scroller by Fredrik Hedenstrom
-

## Dashboard 1: Introduction:



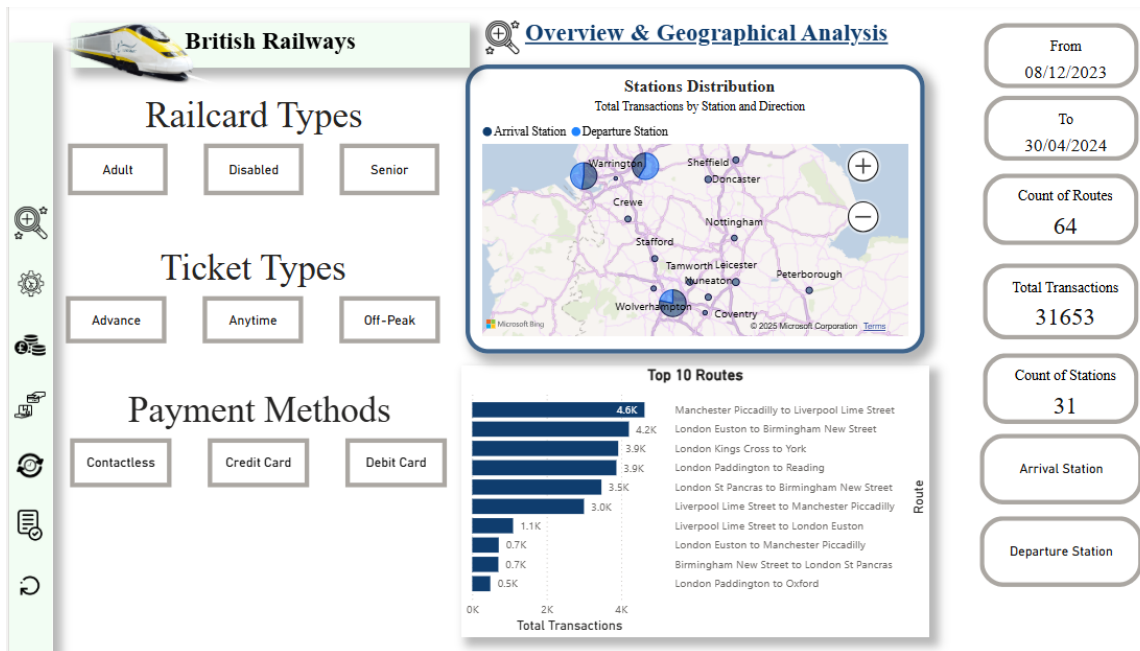
## Dashboard 2:

Overview & Geographical Analysis

Ticket Types, Payment Methods, Railcards

Total transactions by stations (arrival & departure) >>>>>(Map)

Most used routes (Bar Chart)



### **Dashboard 3: Operational Performance**

Total affected journeys by hour & reason for delay (Line Chart)

Total transactions % by journey status (Pie Chart)

Top 10 routes in affected transactions (Bar Chart)

Total delayed & cancelled journeys by reason for delay (Column Chart)

Top 10 routes in average delay duration by reason for delay (Bar Chart)



## Dashboard 4: Ticket Sales Analysis

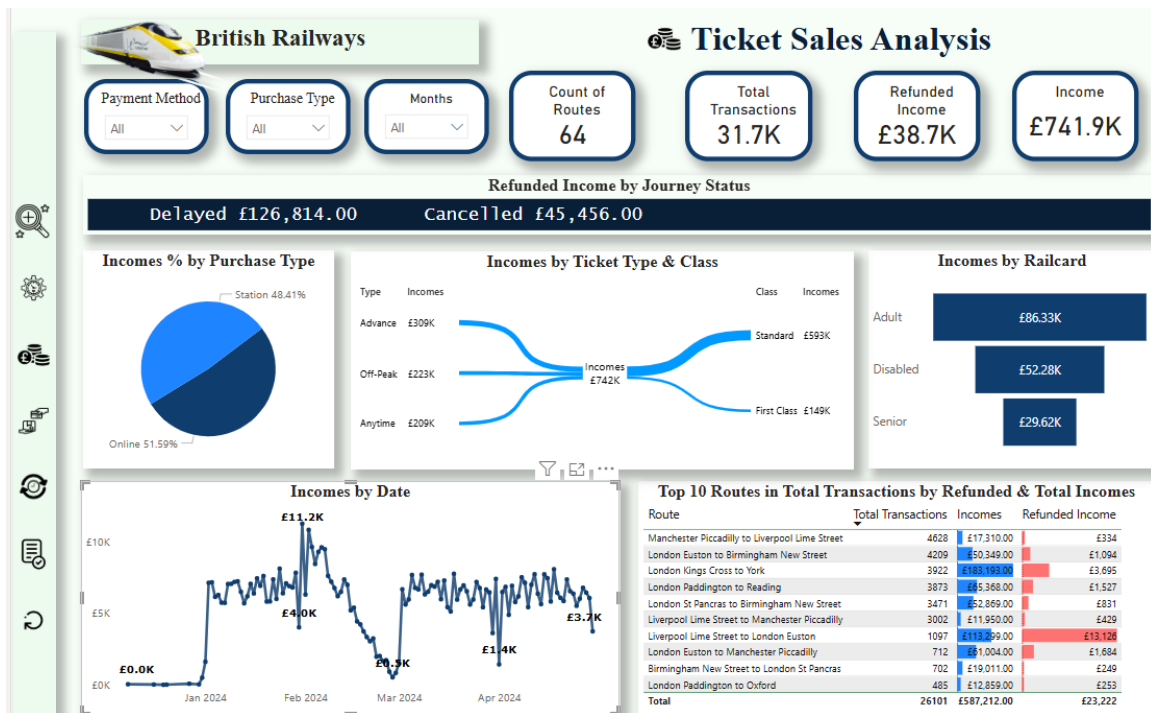
Incomes % by purchase type (Pie Chart)

Incomes by Date (Line Chart)

Incomes by railcard (Funnel Chart)

Incomes by ticket type & class (Bowtie Chart by MAQ)

Top routes in total transaction by total & refunded incomes (Table)



## Dashboard 5: Purchase Preference Analysis

Total transactions by railcard (Pie Chart)

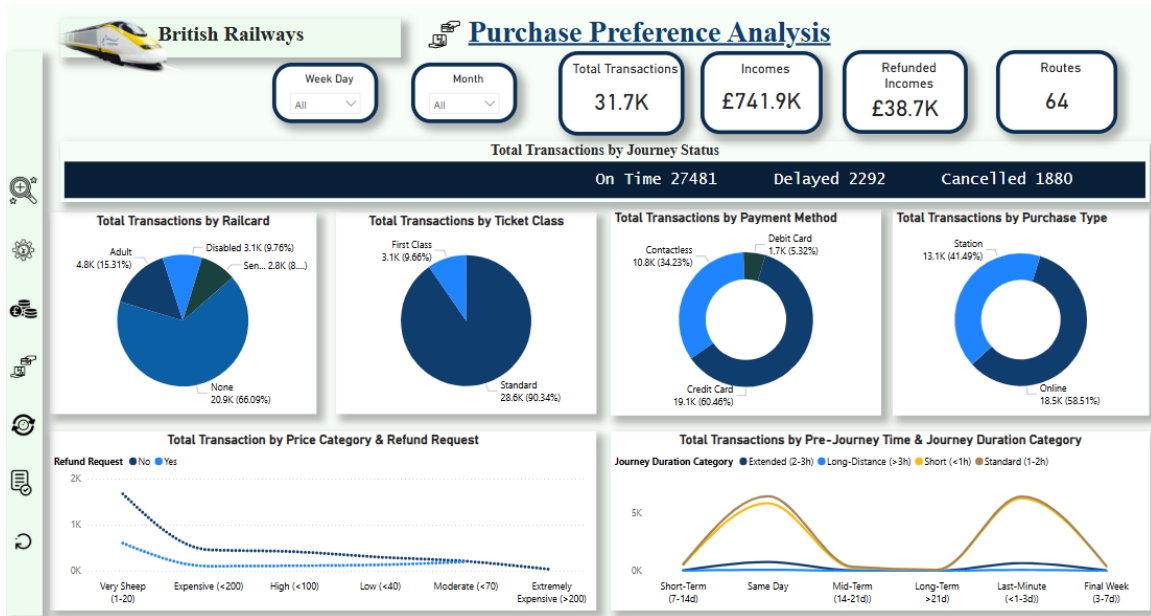
Total transactions by ticket class (Pie Chart)

Total transactions by payment method (Donut Chart)

Total transactions by purchase type (Donut Chart)

Total transactions by price category & refund request (Line Chart)

Total transactions by pre-journey duration category & duration category (Line Chart)



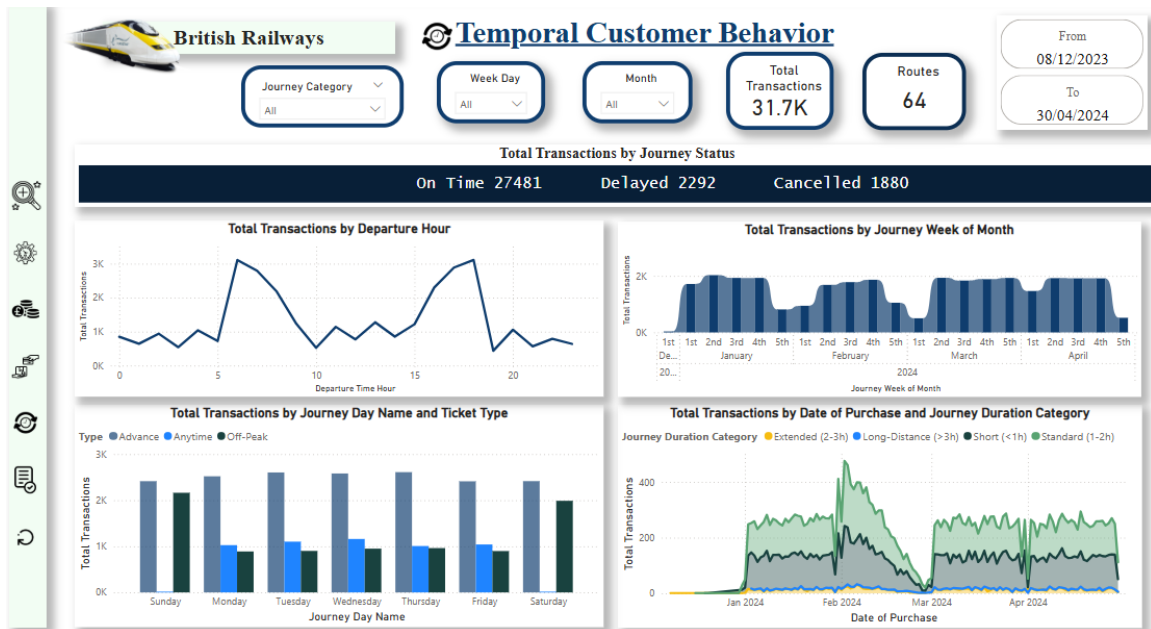
## Dashboard 6: Temporal Customer Behavior

Total transactions by journey week of month (Ribbon Chart)

Total transactions by departure hour (Line Chart)

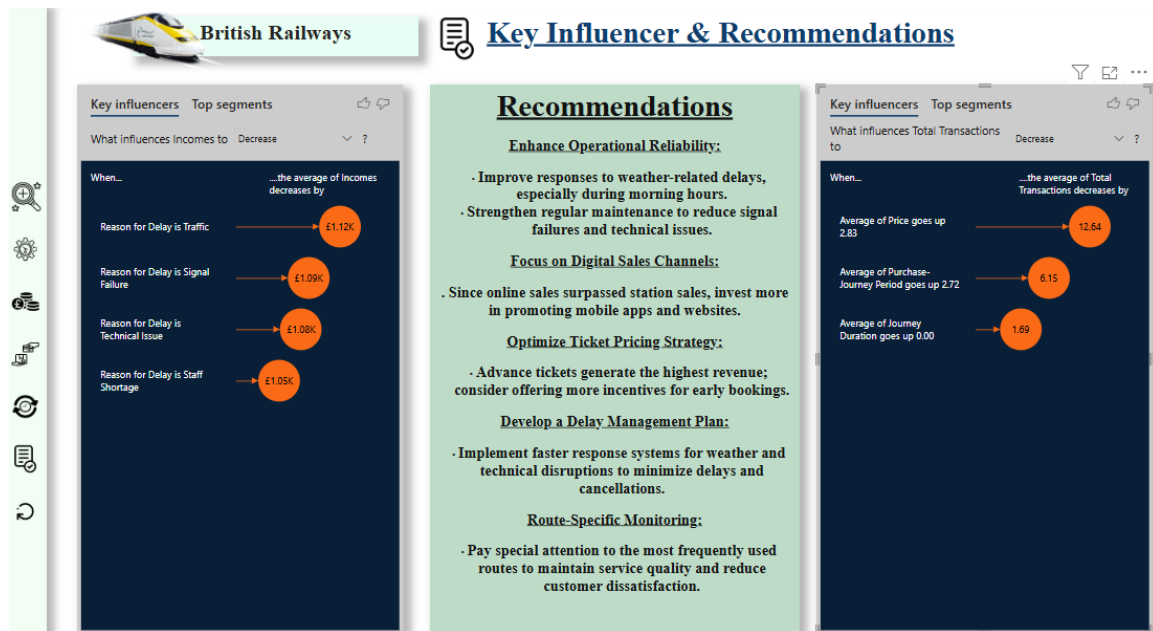
Total transactions by purchase date & journey duration category (Line Chart)

Total transactions by journey day name & ticket type (Column Chart)





## Dashboard 7: Key Influencer & Recommendations



### Challenges Faced

- 1. Inconsistent Station Names:** Some stations had slight variations needing normalization.
- 2. Missing Critical Fields:** Several entries lacked travel times or prices.
- 3. High Cardinality:** Many unique station pairs complicated modeling.
- 4. Performance:** Large data volume needed performance optimization (aggregations and model simplification).

Solutions included Power Query transformations, DAX measures, and optimized relationship management.

## Key Findings

- **Passenger Hotspots:** Major stations like London and Birmingham dominated traffic.
- **Revenue Concentration:** High revenue on intercity routes like London to Manchester.
- **Seasonality:** Summer months showed a clear spike in rides.
- **Ticket Type Behavior:** Return tickets dominated long-distance travel.
- **Distance vs Pricing:** A moderate positive correlation observed.

These findings can assist transport authorities in resource allocation and scheduling.

---

## Conclusion and Recommendations

- Target marketing efforts toward major stations.
- Offer dynamic pricing during peak months.
- Enhance return ticket offers for weekends.
- Expand infrastructure based on top-demand routes.
- Continue monitoring seasonal variations for better scheduling.

This project demonstrates the power of business intelligence tools like Power BI in transforming transportation data into actionable insights.

---

## References:

- Dataset: Simulated UK Railway Ticketing Data.
- Tool Used: Microsoft Power BI.
- Methods: Power Query, DAX, Star Schema Design.  
["ORR Table 1220 – Passenger Journeys"](#). Office of Rail and Road. 12 June 2024. Retrieved 12 January 2025
- ["Rail infrastructure, assets and environmental – April 2023 to March 2024"](#) (PDF). [Office of Rail and Road](#). 10 October 2024. Retrieved 12 January 2024

- ["Nine out of ten trains arrive on time during January"](#) (Press release). Network Rail. 18 February 2010. Archived from [the original](#) on 29 September 2011
- ["The Economic Contribution of UK Rail 2018"](#) (PDF). p. 13. Archived from [the original](#) (PDF) on 19 October 2018. Retrieved 18 October 2018.

**Prepared by:**

Omnia Osama Hussein  
Mohamed Ali Ibrahim  
Mohamed Ahmed AbdElkhalek  
Mohamed Ayman Mohamed  
DEPI-CLS-POWER BI -G8

---

**Thank you!**