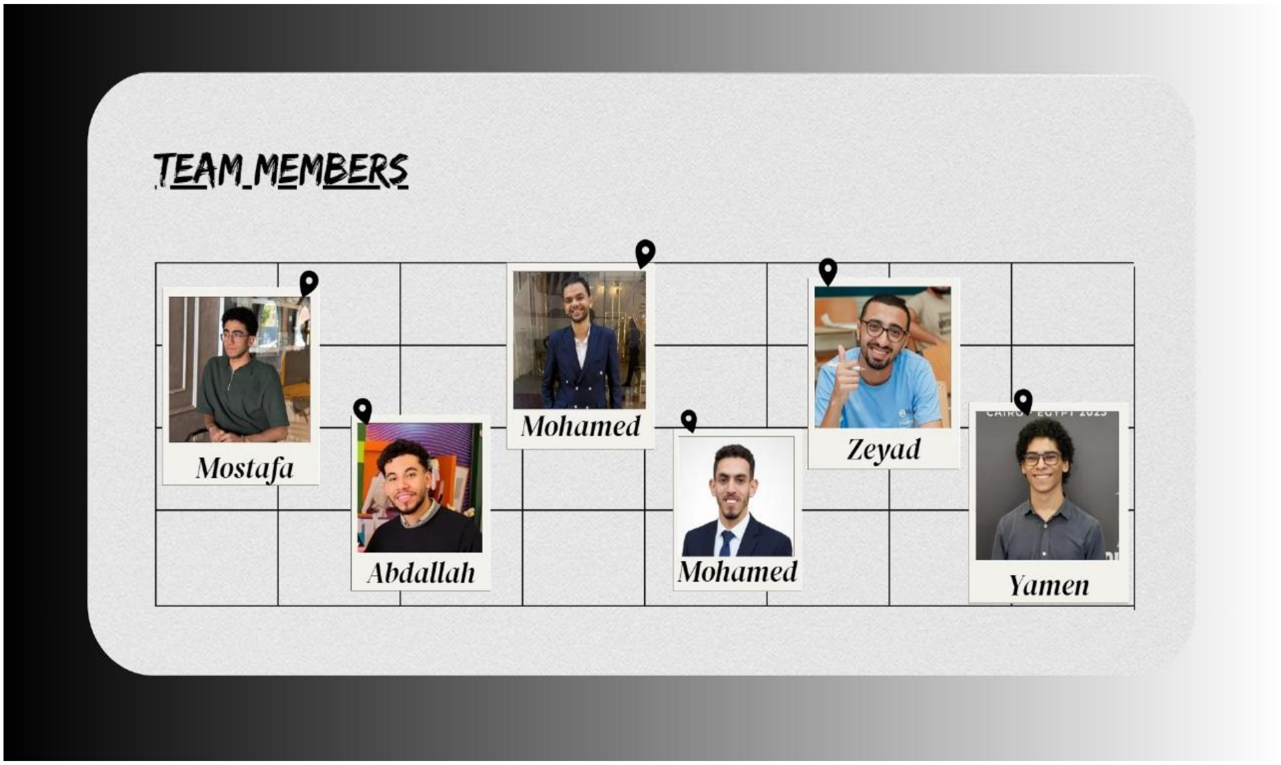
رواد مصر الرقمية

وزارة الاتصــــــالات
وتكنولوجيا المعلومات

UK Railway Data Analysis Report



TEAM MEMBERS

Mostafa

Abdallah

Mohamed

Mohamed

Zeyad

Yamen

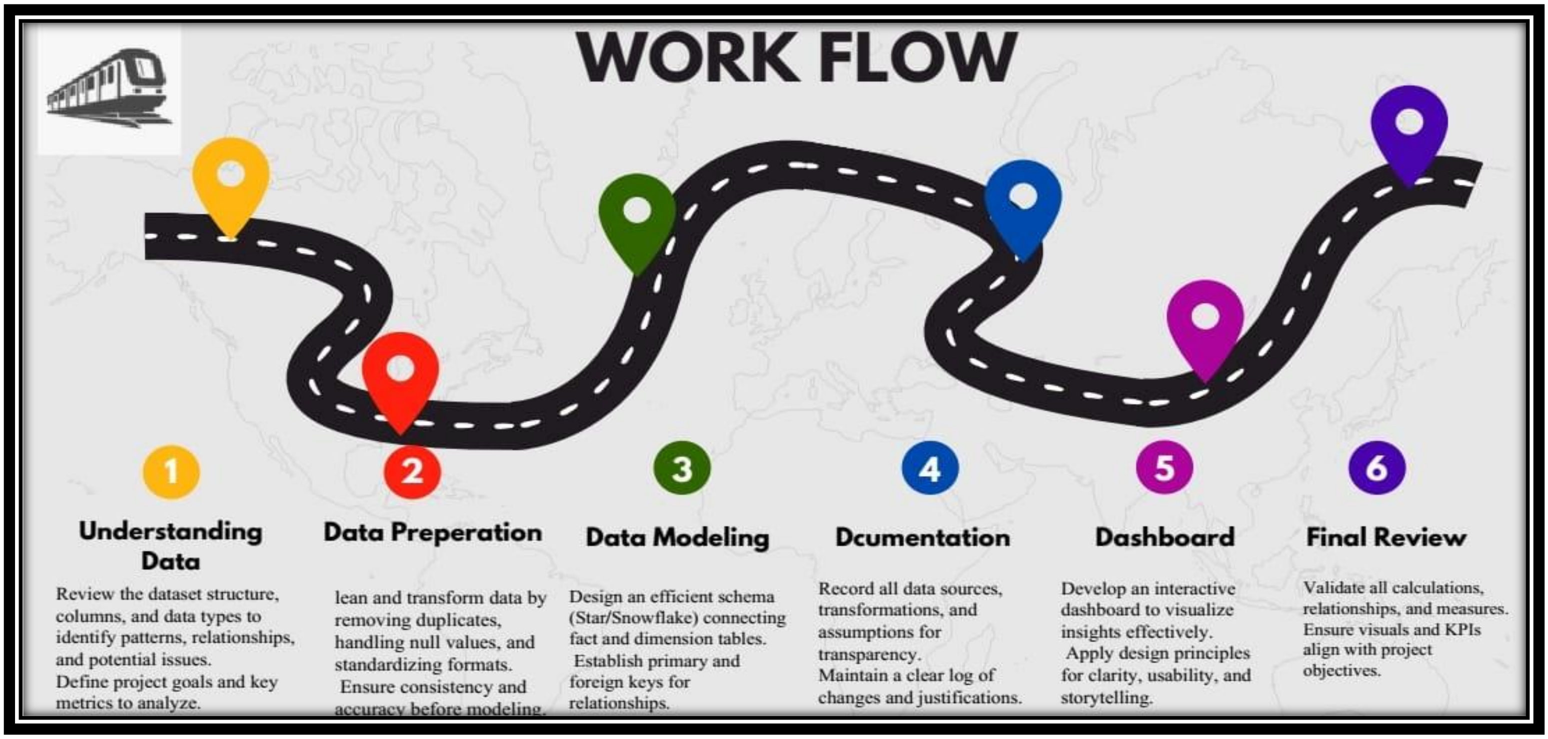
- ## Executive Summary

Delays don't just cost time; they cost money.This is a dashboard that links train delays to financial impact.

This project analyzes train delays and their financial impact on the railway network. With nearly 13% of journeys disrupted, delays not only reduce punctuality but also cause major revenue losses through refunds—particularly from premium tickets. These disruptions put customer trust and loyalty at risk, especially among frequent travelers. The dashboard connects delay patterns with financial outcomes, giving executives the insights needed to cut losses, improve reliability, and strengthen customer confidence.

Key outcomes include:

- Total revenue: **£741,921** from **31,653 tickets**.
- **Standard Class** accounts for 79.86% of sales; **First Class** 20.14%.
- **Advance Tickets** dominate with 41.69% of purchases.
- **Credit Cards** represent 62% of all payments.
- Peak travel occurs at **6:45 PM**, driven by evening commuters.
- Top routes are concentrated around London, reflecting high business travel demand.



1 Understanding Data

Review the dataset structure, columns, and data types to identify patterns, relationships, and potential issues.
Define project goals and key metrics to analyze.

2 Data Preperation

lean and transform data by removing duplicates, handling null values, and standardizing formats.
Ensure consistency and accuracy before modeling.

3 Data Modeling

Design an efficient schema (Star/Snowflake) connecting fact and dimension tables.
Establish primary and foreign keys for relationships.

4 Dcumentation

Record all data sources, transformations, and assumptions for transparency.
Maintain a clear log of changes and justifications.

5 Dashboard

Develop an interactive dashboard to visualize insights effectively.
Apply design principles for clarity, usability, and storytelling.

6 Final Review

Validate all calculations, relationships, and measures.
Ensure visuals and KPIs align with project objectives.

Stakeholders Analysis

1) **Operational Stakeholders:**

- **Operations Managers** → Need insights into delay causes, station/route bottlenecks, and time patterns to optimize scheduling.

- **Network & Infrastructure Teams** → Use delay reason data (signal failures, technical faults) to prioritize maintenance and investments.

2) **Financial Stakeholders:**

- **Finance Department / CFO** → Monitor revenue loss and refund costs, assess financial risk, and forecast the impact of service issues.

- **Revenue Management & Pricing Teams** → Evaluate which ticket types (e.g., First Class, Advance) are most impacted to adjust pricing and refund policies.

3) **Customer-Facing Stakeholders:**

- **Customer Experience & Service Teams** → Identify at-risk passenger segments (e.g., railcard holders, frequent travelers) to improve trust and loyalty.

- **Marketing & Retention Teams** → Use insights on high-impact routes and customers for targeted retention campaigns.

4) **Strategic Stakeholders:**

- **Executive Leadership / Board** → High-level view of financial impact vs. operational performance to guide strategic decisions.

- **Government / Regulators (if applicable)** → For compliance and transparency on service reliability.

- # Key Problems

1) Operational Problems:

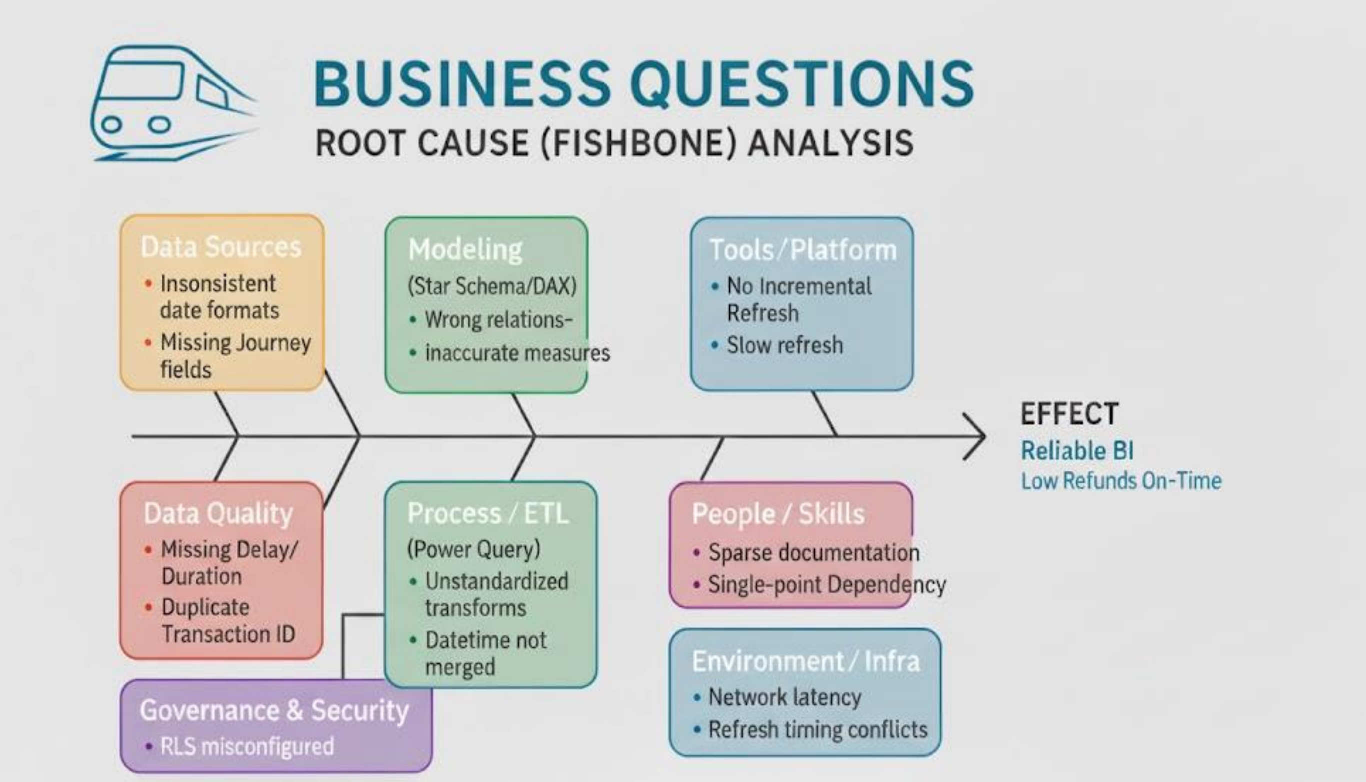
- **High % of delayed journeys:** reliability is low — frequent missed ETAs harm trust.
- **Recurring delay reasons (e.g., signal failures):** fixes are temporary; root causes aren't solved.
- **Certain routes are always delayed:** network bottlenecks or aging infrastructure concentrated on specific lines.
- **Net effect:** the core system/operations are underperforming and need structural fixes.

2) Financial Problems:

- **Significant revenue loss from refunds:** each refund directly reduces income.
- **Refunds concentrated in specific ticket types:** some fare classes (e.g., advance tickets) drive disproportionate refunds.
- **High revenue at risk in premium classes:** delays on high-price services multiply losses.
- **Delayed trains often have high occupancy:** delays affect many passengers, increasing financial impact.

3) Customer Problems:

- **Unhappy repeat customers:** frequent users (e.g., season-ticket holders) are at high risk of churn.
- **Refund process overload:** customer service resources are strained, worsening experience and reputation.
- **Advance ticket buyers penalized more:** customers who planned ahead feel unfairly treated → loss of trust.



- ## Purpose of the Dashboard / Report

Mean Purpose: provide a comprehensive view of train delays and their financial impact on the railway network.

Goals:

- **Quantify Financial Impact:** Translate delay data into clear and specific financial loss figures.

- **Link Causes to Outcomes:** Connect operational **`Reasons for Delay`** to the highest volumes of **`Refund Requests`**.

- **Expose Costly Bottlenecks:** Identify the most expensive routes, stations, and delay reasons for the business.

- **Enable Strategic Decisions:** Empower management to intelligently allocate resources to reduce losses and improve service.

- ## Business Questions

1) **Operational:**

   - What percentage of Total journeys , delayed , on time, Cancelled, and what are the most main reasons?
   - Which routes and stations have the most delays?
   - Do delays happen more often at certain times (days, months, or peak hours)?

2) **Financial:**

   - How much revenue is lost because of refunds?
   - How does how long a delay lasts affect the chance of a refund?
   - Which ticket types (e.g., Standard, First Class, Advance) are most affected?

3) **Customer:**

   - Which passenger groups (e.g., Railcard holders, frequent travelers) ask for the most refunds?
   - How do repeated delays impact customer loyalty on popular routes?

- # Dataset Description → Metadata

Dataset Summary:

- **Records:** 31,653
- **Fields:** 18
- **Coverage:** Ticket purchases and journey outcomes (late 2023 – early 2024)

Key Fields:

Field	Data Type	Description
Transaction ID	text	Unique identifier for an individual train ticket purchase
Date of Purchase	date	Date the ticket was purchased
Time of Purchase	time	Time the ticket was purchased
Purchase Type	text	Whether the ticket was purchased online or directly at a train station
Payment Method	text	Payment method used to purchase the ticket (Contactless, Credit Card, or Debit Card)
Railcard	text	Whether the passenger is a National Railcard holder (Adult, Senior, or Disabled) or not (None). Railcard holders get 1/3 off their ticket purchases.
Ticket Class	text	Seat class for the ticket (Standard or First)
Ticket Type	text	When you bought or can use the ticket. Advance tickets are 1/2 off and must be purchased at least a day prior to departure. Off-Peak tickets are 1/4 off and must be used outside of peak hours. Anytime = full.
Price	currency	Final cost of the ticket
Departure Station	text	Station to board the train
Arrival Destination	text	Station to exit the train
Date of Journey	date	Date the train departed
Departure Time	time	Time the train departed
Arrival Time	time	Time the train was scheduled to arrive at its destination (can be on the day after departure)
Actual Arrival Time	time	Time the train arrived at its destination (can be on the day after departure)
Journey Status	text	Whether the train was on time, delayed, or cancelled
Reason for Delay	text	Reason for the delay or cancellation
Refund Request (Yes/No)	boolean	Whether the passenger requested a refund after a delay or cancellation

- ## Is the dataset complete enough to answer the business questions?

The dataset is pretty good and sufficient for analyzing train delays and their financial impact.

Unique transaction IDs are there, making the ticket-level info reliable.

The data covers key stuff like ticket sales, delay duration, refund behavior, and customer attributes, so it's good for operational, financial, and customer insights.

One important limitation: We're missing trip or train-level identifiers. Without specific train IDs, we can't really track a specific train's performance over multiple days or fully dig into operational efficiency and recurring service issues.

So, while the dataset is a strong foundation for most business questions, adding train-level IDs in the future would unlock even deeper operational insights.

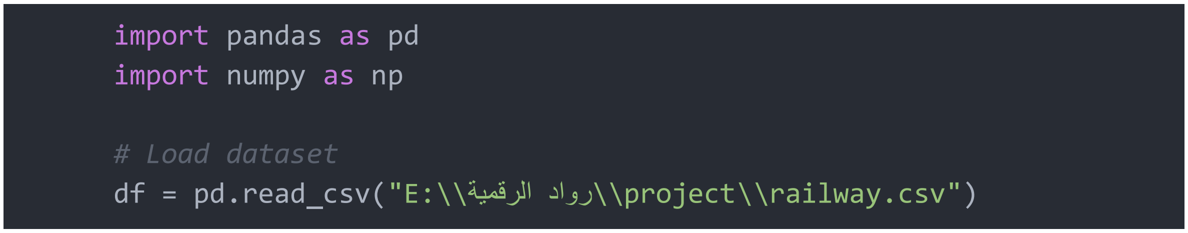
- # Data Preprocessing

1. ### Initial Data Inspection and Quality Checks

- ### Libraries and Data Loading

The essential Python libraries — `pandas`, `numpy`, `seaborn`, and `matplotlib` — were imported to enable data manipulation, statistical analysis, and visualization.

The dataset (`railway.csv`) was loaded into a Pandas DataFrame using:

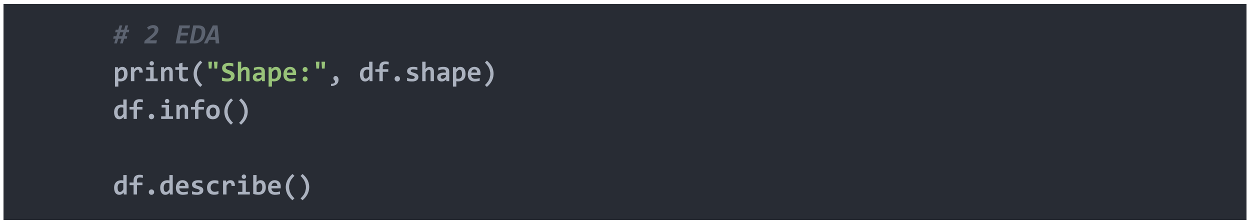


```
import pandas as pd
import numpy as np

# Load dataset
df = pd.read_csv("E:\\رواد الرقمية\\project\\railway.csv")
```

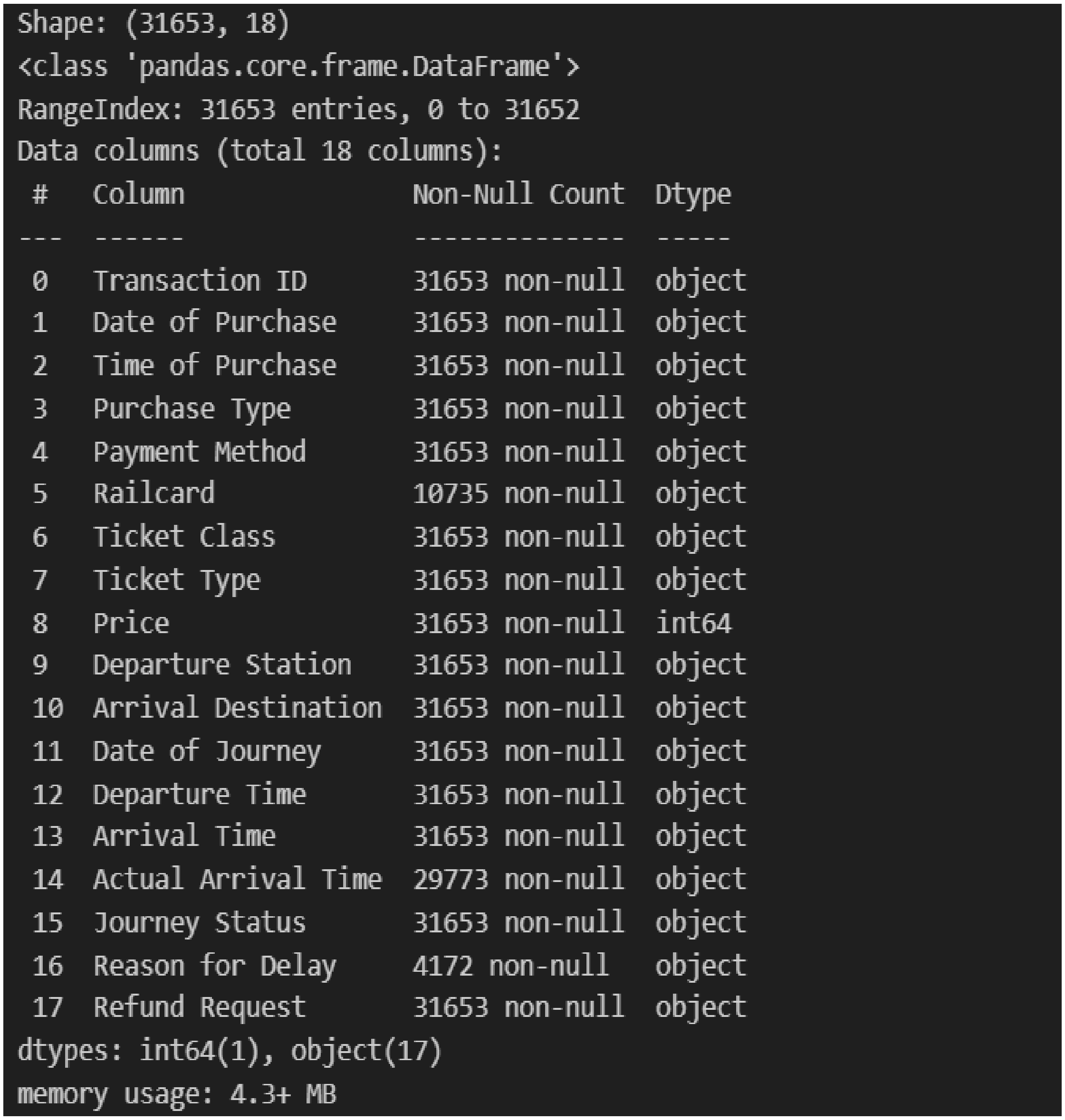
- ## EDA

Data Type Verification:

Each column’s data type was reviewed using df.info(). Time-related columns such as 'Departure Time', 'Arrival Time', and 'Actual Arrival Time' were later standardized into datetime format to enable accurate time-based calculations.



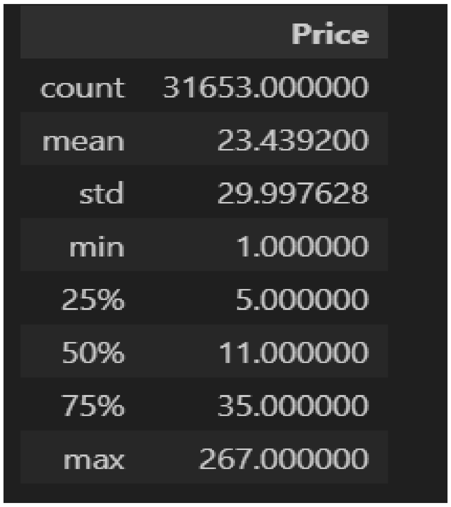
```
# 2 EDA
print("Shape:", df.shape)
df.info()

df.describe()
```



```
Shape: (31653, 18)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31653 entries, 0 to 31652
Data columns (total 18 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   Transaction ID       31653 non-null  object
 1   Date of Purchase     31653 non-null  object
 2   Time of Purchase     31653 non-null  object
 3   Purchase Type        31653 non-null  object
 4   Payment Method       31653 non-null  object
 5   Railcard             10735 non-null  object
 6   Ticket Class         31653 non-null  object
 7   Ticket Type          31653 non-null  object
 8   Price                31653 non-null  int64 
 9   Departure Station    31653 non-null  object
 10  Arrival Destination  31653 non-null  object
 11  Date of Journey      31653 non-null  object
 12  Departure Time       31653 non-null  object
 13  Arrival Time         31653 non-null  object
 14  Actual Arrival Time  29773 non-null  object
 15  Journey Status       31653 non-null  object
 16  Reason for Delay     4172 non-null   object
 17  Refund Request       31653 non-null  object
dtypes: int64(1), object(17)
memory usage: 4.3+ MB
```

- ## Descriptive Statistical Overview:

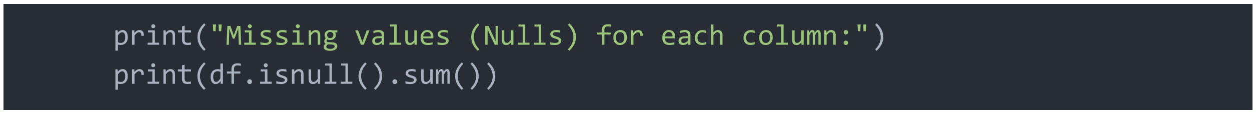
A statistical summary generated using df.describe() provided insights into the central tendency, dispersion, and range of numeric features like 'Price' and 'Planned Journey Duration'. This helped identify potential outliers and guided the next cleaning steps.



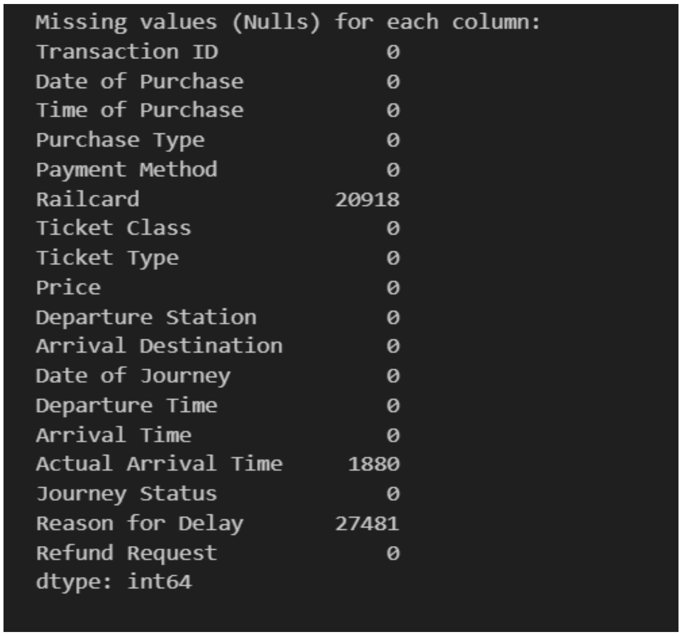
	Price
count	31653.000000
mean	23.439200
std	29.997628
min	1.000000
25%	5.000000
50%	11.000000
75%	35.000000
max	267.000000

- ## Checking and Removing Null Values

A completeness check was conducted using df.isnull().sum(), which verified that the dataset was entirely clean, with zero missing values across all columns. This confirmed that no imputation or record removal was required before further processing.



```
print("Missing values (Nulls) for each column:")
print(df.isnull().sum())
```



```
Missing values (Nulls) for each column:
Transaction ID              0
Date of Purchase            0
Time of Purchase            0
Purchase Type               0
Payment Method              0
Railcard                20918
Ticket Class                0
Ticket Type                 0
Price                       0
Departure Station           0
Arrival Destination         0
Date of Journey             0
Departure Time              0
Arrival Time                0
Actual Arrival Time      1880
Journey Status              0
Reason for Delay        27481
Refund Request              0
dtype: int64
```

2. Data Cleaning and Transformation

- **Removing Duplicates**

```
# Remove duplicate rows if any
df.drop_duplicates(inplace=True)
```

- **Standardizing Time Formats**

```
#3- Convert date columns

df["Date of Purchase"] = pd.to_datetime(df["Date of Purchase"],
errors="coerce")

df["Date of Journey"] = pd.to_datetime(df["Date of Journey"],
errors="coerce")
```

- **Convert Time Columns**

Time Conversion and Normalization:

All time-related columns were cleaned and converted into full datetime objects by combining 'Date of Journey' with each time column ('Departure Time', 'Arrival Time', and 'Actual Arrival Time').

Trips that crossed midnight were handled by adding one day to arrival times when necessary.

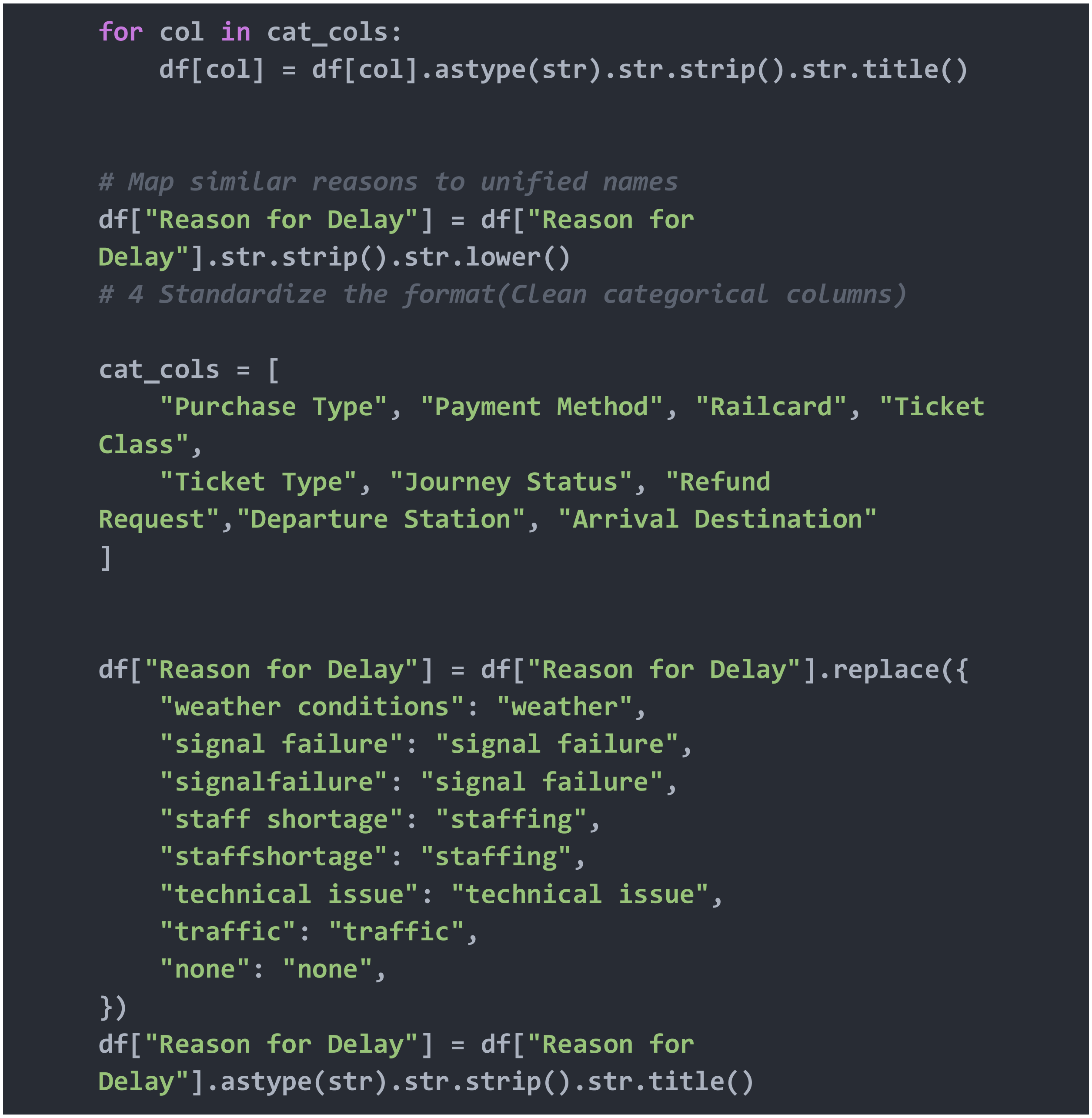
```
#4- Convert time columns

time_cols = ["Time of Purchase", "Departure Time", "Arrival Time",
"Actual Arrival Time"]

for col in time_cols:
    df[col] = pd.to_datetime(df[col], format="%H:%M:%S",
errors="coerce").dt.time
```

- **Validating and Cleaning Categorical Fields**

Reason for Delay Standardization:

Text inconsistencies in the 'Reason for Delay' column were cleaned by converting all text to lowercase, stripping whitespace, and mapping similar entries (e.g., 'Weather Conditions' → 'Weather', 'Staff Shortage' → 'Staffing').

This ensured cleaner visualizations and consistent grouping.



```
for col in cat_cols:
    df[col] = df[col].astype(str).str.strip().str.title()


# Map similar reasons to unified names
df["Reason for Delay"] = df["Reason for
Delay"].str.strip().str.lower()
# 4 Standardize the format(Clean categorical columns)

cat_cols = [
    "Purchase Type", "Payment Method", "Railcard", "Ticket
Class",
    "Ticket Type", "Journey Status", "Refund
Request","Departure Station", "Arrival Destination"
]


df["Reason for Delay"] = df["Reason for Delay"].replace({
    "weather conditions": "weather",
    "signal failure": "signal failure",
    "signalfailure": "signal failure",
    "staff shortage": "staffing",
    "staffshortage": "staffing",
    "technical issue": "technical issue",
    "traffic": "traffic",
    "none": "none",
})
df["Reason for Delay"] = df["Reason for
Delay"].astype(str).str.strip().str.title()
```

Fill missing values

```
# 5- Fill missing values

# Reason for Delay column fill with None

df["Reason for Delay"] = df["Reason for Delay"].fillna("None")

# for Railcard column fill with No Railcard
df['Railcard'] = df['Railcard'].replace('Nan', 'No Railcard').fillna('No
Railcard')

# we can not fill nulls in Actual Arrival Time because of the data type
that my lead to wrong calculation
#if we filled it with mean medin mode ....
#and we can not drop it because it represent canceled journey
```

2. Data Transformation

- **Combine Date + Time For time calculations**

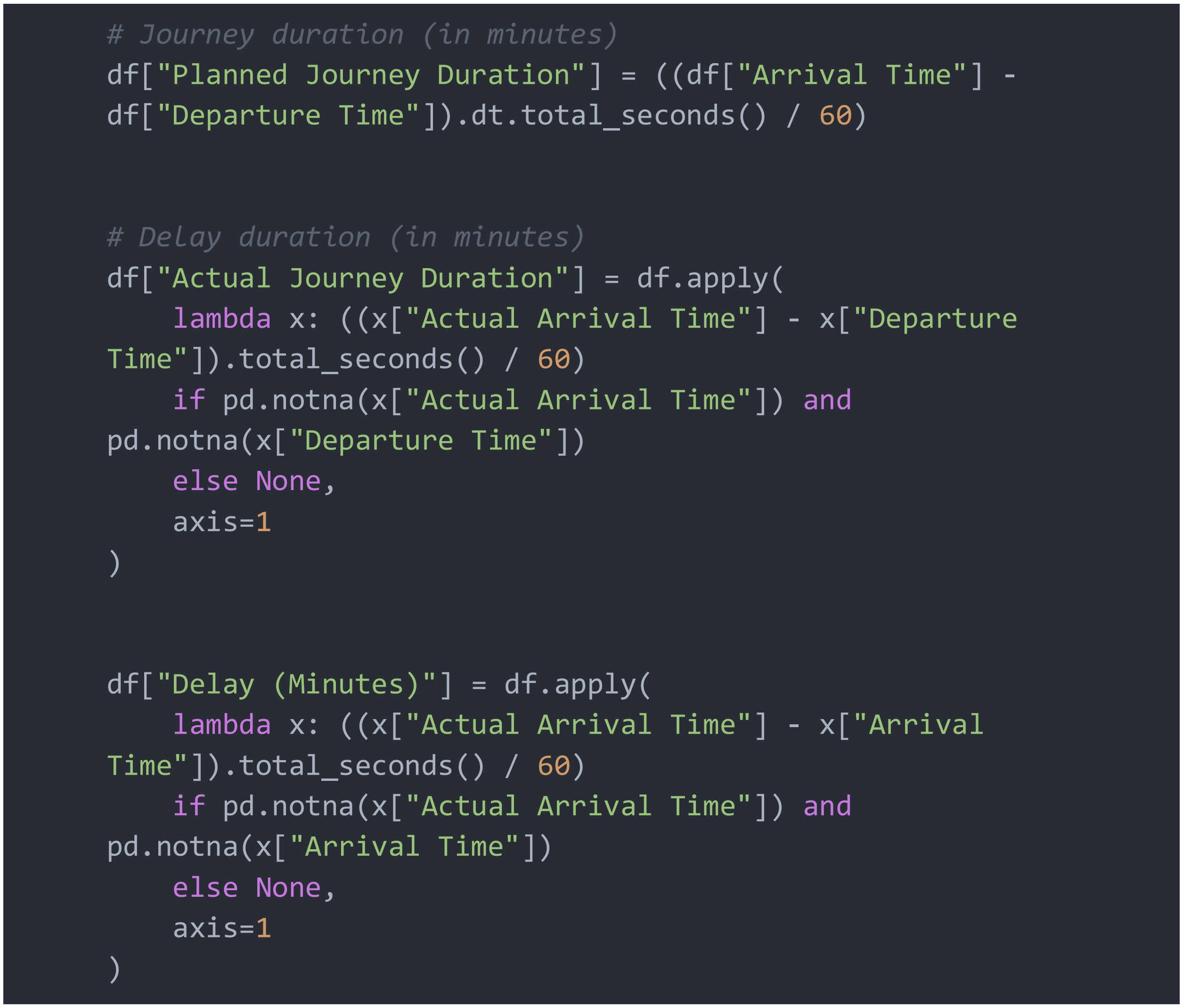
```
#  Safely combine Date + Time
def combine_date_time(row, time_col):
    """Combine Date of Journey and a time column into a full datetime"""
    if pd.notna(row['Date of Journey']) and pd.notna(row[time_col]):
        return pd.Timestamp.combine(row['Date of Journey'].date(), row[time_col])
    else:
        return pd.NaT

df['Departure Time'] = df.apply(lambda x: combine_date_time(x, 'Departure Time'),
axis=1)
df['Arrival Time'] = df.apply(lambda x: combine_date_time(x, 'Arrival Time'),
axis=1)
df['Actual Arrival Time'] = df.apply(lambda x: combine_date_time(x, 'Actual
Arrival Time'), axis=1)



# If arrival is before departure -->> add 1 day (crossed midnight)
df.loc[df['Arrival Time'] < df['Departure Time'], 'Arrival Time'] +=
pd.Timedelta(days=1)
df.loc[df['Actual Arrival Time'] < df['Departure Time'], 'Actual Arrival Time'] +=
pd.Timedelta(days=1)
```

- **Creating Delay Categories**

Time Conversion and Normalization:

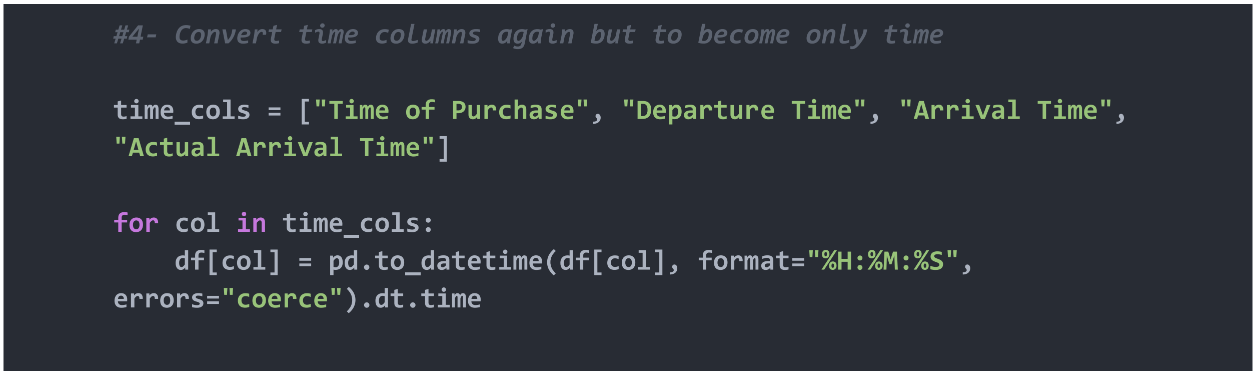
All time-related columns were cleaned and converted into full datetime objects by combining 'Date of Journey' with each time column ('Departure Time', 'Arrival Time', and 'Actual Arrival Time').

Trips that crossed midnight were handled by adding one day to arrival times when necessary.



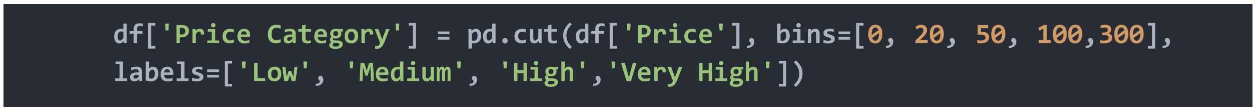
```
# Journey duration (in minutes)
df["Planned Journey Duration"] = ((df["Arrival Time"] -
df["Departure Time"]).dt.total_seconds() / 60)


# Delay duration (in minutes)
df["Actual Journey Duration"] = df.apply(
    lambda x: ((x["Actual Arrival Time"] - x["Departure
Time"]).total_seconds() / 60)
    if pd.notna(x["Actual Arrival Time"]) and
pd.notna(x["Departure Time"])
    else None,
    axis=1
)


df["Delay (Minutes)"] = df.apply(
    lambda x: ((x["Actual Arrival Time"] - x["Arrival
Time"]).total_seconds() / 60)
    if pd.notna(x["Actual Arrival Time"]) and
pd.notna(x["Arrival Time"])
    else None,
    axis=1
)
```

- **Convert time columns again but to become only time**



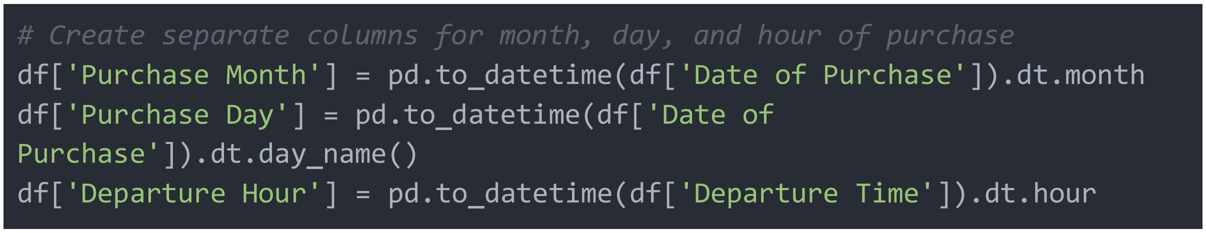
```
#4- Convert time columns again but to become only time

time_cols = ["Time of Purchase", "Departure Time", "Arrival Time",
"Actual Arrival Time"]

for col in time_cols:
    df[col] = pd.to_datetime(df[col], format="%H:%M:%S",
errors="coerce").dt.time
```

- **Creating Price Category Column**



```
df['Price Category'] = pd.cut(df['Price'], bins=[0, 20, 50, 100,300],
labels=['Low', 'Medium', 'High','Very High'])
```

- **Extracting Temporal Features for Analysis**



```
# Create separate columns for month, day, and hour of purchase
df['Purchase Month'] = pd.to_datetime(df['Date of Purchase']).dt.month
df['Purchase Day'] = pd.to_datetime(df['Date of
Purchase']).dt.day_name()
df['Departure Hour'] = pd.to_datetime(df['Departure Time']).dt.hour
```

3. Data Validation

- **Logical Time Validation**

```
# Ensure arrival time is after departure time
df = df[pd.to_datetime(df['Arrival Time']) > pd.to_datetime(df['Departure
Time'])]
b. Ticket Price Verification
# Check that First Class has higher average price
avg_prices = df.groupby('Ticket Class')['Price'].mean()
print(avg_prices)
```

- **Transaction ID Integrity Check**

```
# Ensure unique Transaction IDs
if not df['Transaction ID'].is_unique:
print('Duplicate Transaction IDs found!')
df.drop_duplicates(subset=['Transaction ID'], inplace=True)
else:
print('All Transaction IDs are unique.')
```

- **Data Type Conversion and Final Review**

```
# Convert numeric columns to correct data types
df['Price'] = df['Price'].astype(float)
df['Delay (Minutes)'] = df['Delay (Minutes)'].astype(float)

# Final info
print(df.info())
```

✅Result:

After executing these comprehensive preprocessing steps, the dataset was fully cleaned, transformed, and validated. It is now ready for detailed analysis and dashboard visualization in Power BI, ensuring consistency and high-quality insights.

- ## Database Design (Data Mapping)

The goal of this script is to transform a **flat CSV dataset** of railway transactions into a **star schema** — a dimensional model used in data warehousing.
A star schema organizes data into:

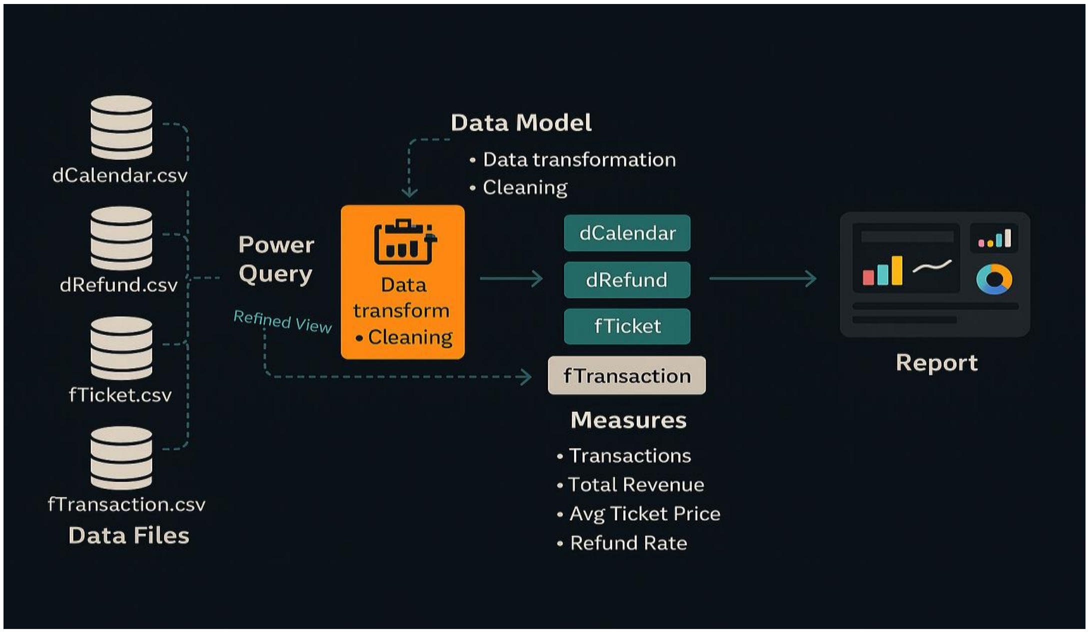
- **One central fact table** containing measurable events (e.g., transactions, sales, delays)

- **Several dimension tables** providing descriptive context (e.g., ticket details, stations, refund status, dates)

This structure makes it easier to perform analytical queries such as:

> “What is the average delay per ticket class?”
> “Which stations have the highest ticket prices on weekends?”
> “How many refunds were requested per payment method?”

Table	Type	Description
fTransaction	Fact	Stores the main measurable event (ticket purchase)
dJourney	Dimension	Describes journey details (departure, destination, timing)
dTicket	Dimension	Describes ticket type and class
dCalendar	Dimension	Describes the purchase date with time and calendar attributes
dRefund	Dimension	Describes the refund status for transactions



Steps:

- **Load the dataset**

```
import pandas as pd

# Load the dataset
df = pd.read_csv("E:\\رواد الرقمية\\project\\railway_cleaned_finalllll2.csv")
```

- **Create dJourney dimension table:**
  - This dimension describes each unique journey: its start, end, and timing details.
  - We drop duplicates because multiple transactions can share the same journey.

```
# Dimension: dJourney
# --------------------------
dJourney_cols = [
    "Departure Station", "Arrival Destination", "Date of Journey",
    "Departure Time", "Arrival Time", "Actual Arrival Time",
    "Planned Journey Duration","Journey Status", "Reason for Delay"
]
dJourney = df[dJourney_cols].drop_duplicates().reset_index(drop=True)
dJourney.insert(0, "Journey_ID", range(1, len(dJourney) + 1))

path_for_save="E:\\رواد الرقمية\\project\\dJourney.csv"

dJourney.to_csv(path_for_save, index=False)
```

- **Create dTicket dimension table:**
    - Describes the properties of a ticket (class and type).
    - The 'Railcard' is excluded here since it belongs to the transaction (changes per purchase).

```
dTicket_cols = ["Ticket Class", "Ticket Type"]
dTicket = df[dTicket_cols].drop_duplicates().reset_index(drop=True)
dTicket.insert(0, "Ticket_ID", range(1, len(dTicket) + 1))

path_for_save="E:\\رواد الرقمية\\project\\dTicket.csv"

dTicket.to_csv(path_for_save, index=False)
```

- **Create dCalendar dimension table**
    - This table provides calendar-related information based on the purchase date and time.
    - Additional fields (year, month, day, weekend) are derived for time-based analysis.

```
dCalendar_cols = ["Date of Purchase"]
dCalendar = df[dCalendar_cols].drop_duplicates().reset_index(drop=True)

# Derive date features
dCalendar["Year"] = pd.to_datetime(dCalendar["Date of Purchase"]).dt.year
dCalendar["Month"] = pd.to_datetime(dCalendar["Date of Purchase"]).dt.month
dCalendar["Month_Name"] = pd.to_datetime(dCalendar["Date of 
Purchase"]).dt.month_name()
dCalendar["Day_Name"] = pd.to_datetime(dCalendar["Date of 
Purchase"]).dt.day_name()
dCalendar["Weekend"] = dCalendar["Day_Name"].isin(["Saturday", "Sunday"])
dCalendar.insert(0, "Calendar_ID", range(1, len(dCalendar) + 1))

path_for_save="E:\\رواد الرقمية\\project\\dCalendar.csv"

dCalendar.to_csv(path_for_save, index=False)
```

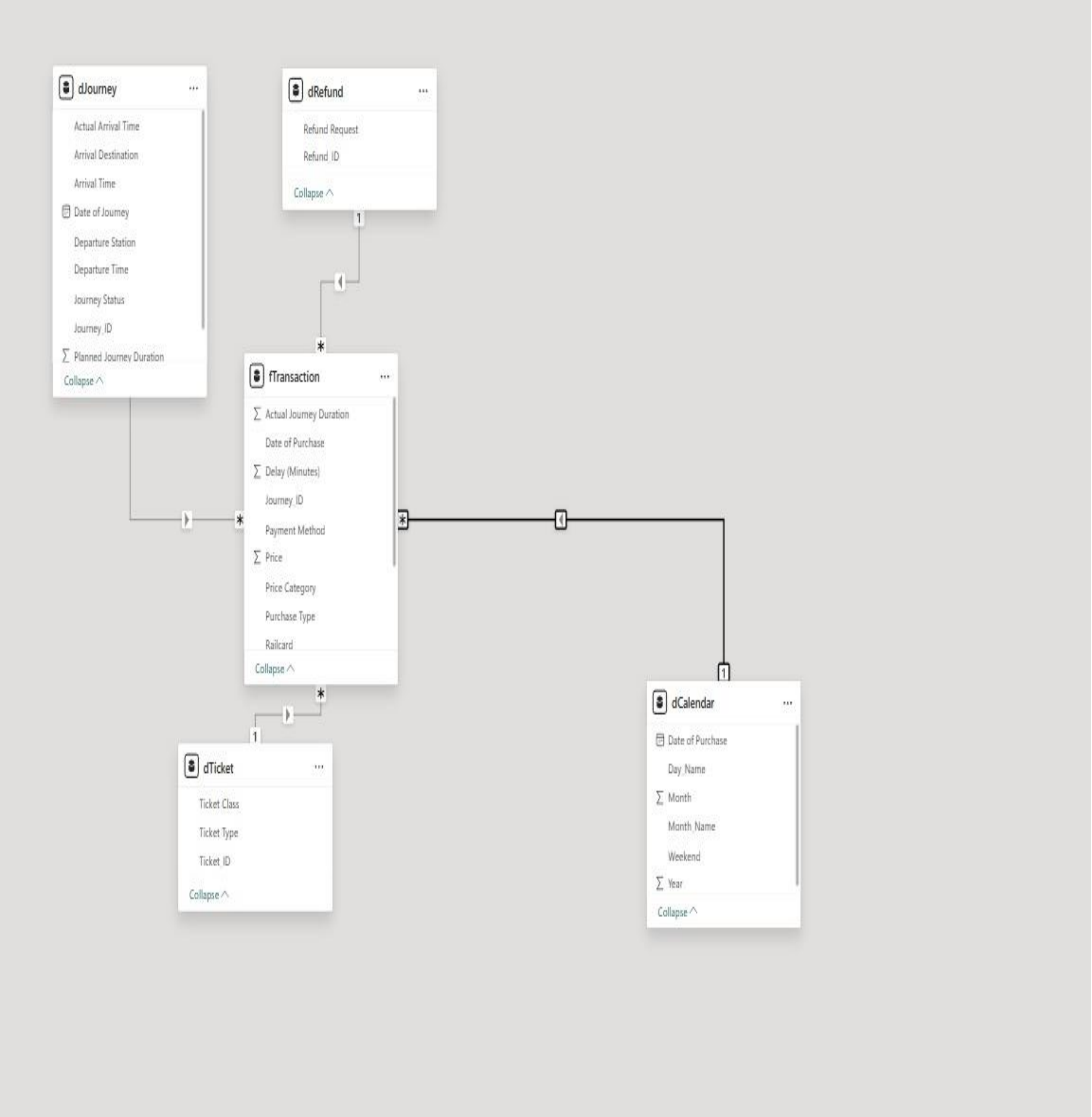
- **Create dRefund dimension table:**
    - Contains refund status information (e.g., Yes/No).
    - If additional refund details exist in the future, they can be added here.

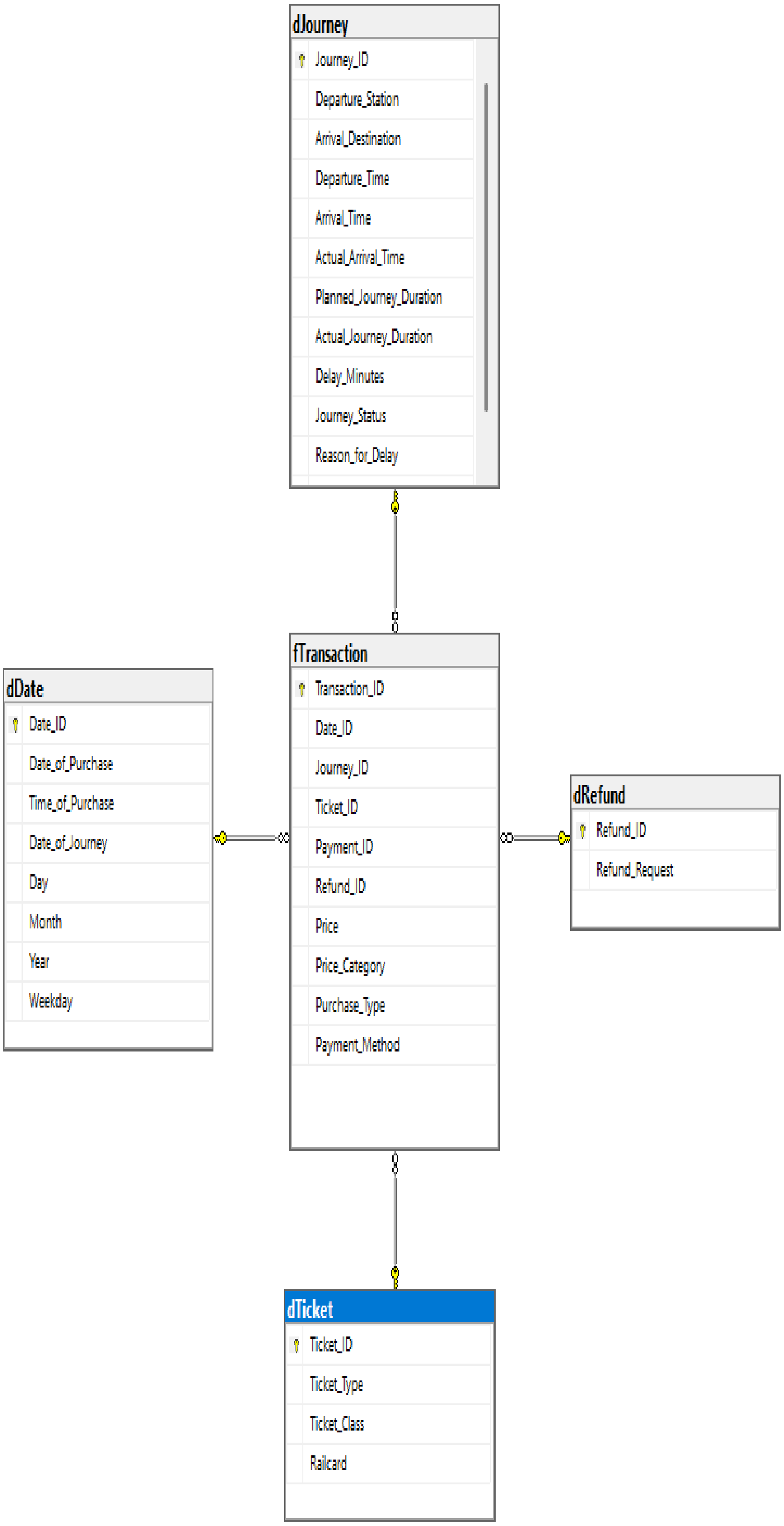
```
# Dimension: dRefund
# ---------------------------
dRefund_cols = ["Refund Request"]
dRefund = df[dRefund_cols].drop_duplicates().reset_index(drop=True)
dRefund.insert(0, "Refund_ID", range(1, len(dRefund) + 1))

path_for_save="E:\\رواد الرقمية\\project\\dRefund.csv"

dRefund.to_csv(path_for_save, index=False)
```

- **Create fTransaction fact table:**
    - This is the core of the star schema.
    - Each row represents one ticket purchase and includes:
        - Foreign keys linking to dimension tables
        - Quantitative measures (price, durations, delay)
        - Transaction attributes (railcard, payment method, etc.)

```
# Fact: fTransaction
# ---------------------------
# Map foreign keys
fTransaction = df.copy()

# Merge with dimension IDs
fTransaction = fTransaction.merge(dJourney, how="left", on=dJourney_cols)
fTransaction = fTransaction.merge(dTicket, how="left", on=dTicket_cols)
fTransaction = fTransaction.merge(dRefund, how="left", on=dRefund_cols)

# Select relevant fact columns
fTransaction = fTransaction[[
    "Transaction ID", "Journey_ID", "Ticket_ID", "Refund_ID", "Date of Purchase", "Time of Purchase",
    "Purchase Type", "Payment Method", "Railcard", "Price", "Price Category",
     "Actual Journey Duration", "Delay (Minutes)"
]]

path_for_save="E:\\رواد الرقمية\\project\\fTransaction.csv"

fTransaction.to_csv(path_for_save, index=False)
```

The Result:


dJourney
Actual Arrival Time
Arrival Destination
Arrival Time
Date of Journey
Departure Station
Departure Time
Journey Status
Journey_ID
Planned Journey Duration
Collapse
dRefund
Refund Request
Refund_ID
Collapse
1
*
fTransaction
Actual Journey Duration
Date of Purchase
Delay (Minutes)
Journey_ID
Payment Method
Price
Price Category
Purchase Type
Railcard
Collapse
*
*
*
1
dTicket
Ticket Class
Ticket Type
Ticket_ID
Collapse
1
dCalendar
Date of Purchase
Day_Name
Month
Month_Name
Weekend
Year
Collapse


dJourney
Journey_ID
Departure_Station
Arrival_Destination
Departure_Time
Arrival_Time
Actual_Arrival_Time
Planned_Journey_Duration
Actual_Journey_Duration
Delay_Minutes
Journey_Status
Reason_for_Delay
fTransaction
Transaction_ID
Date_ID
Journey_ID
Ticket_ID
Payment_ID
Refund_ID
Price
Price_Category
Purchase_Type
Payment_Method
dDate
Date_ID
Date_of_Purchase
Time_of_Purchase
Date_of_Journey
Day
Month
Year
Weekday
dRefund
Refund_ID
Refund_Request
dTicket
Ticket_ID
Ticket_Type
Ticket_Class
Railcard

- ## Dashboard Storyboard: Railway Performance Analysis

Title & Overview

- **Title**: Railway Performance Dashboard

- **Purpose**: To provide a comprehensive, interactive view of UK railway operations, linking journey efficiency (punctuality, delays, and cancellations) with financial metrics (revenue, refunds, and pricing strategies). It analyzes trends in customer behavior, route popularity, and peak times to support data-driven decisions for improving service reliability and profitability.

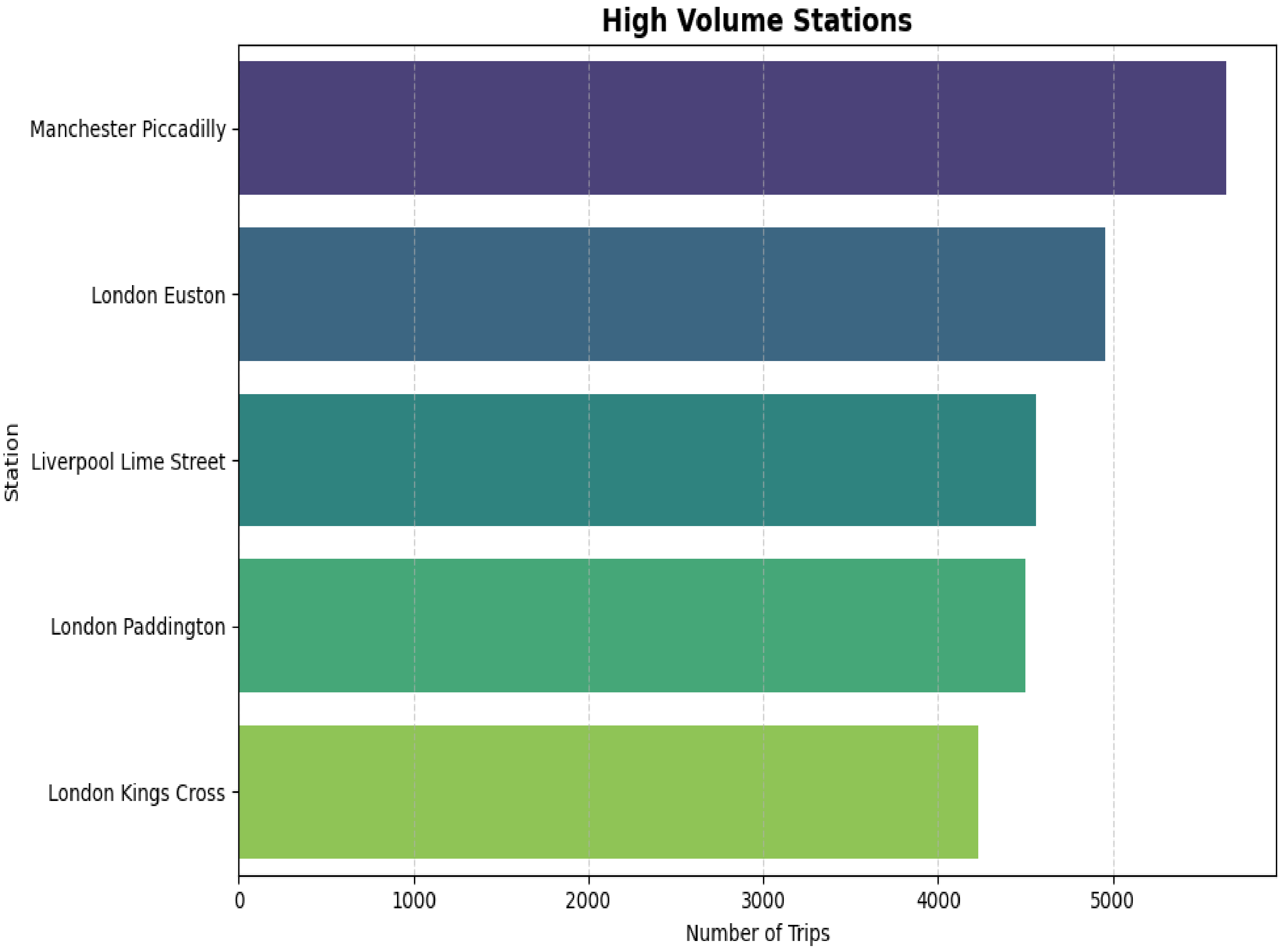
- This dashboard enables executives, analysts, and operations teams to monitor key performance indicators (KPIs), identify bottlenecks (e.g., delay causes), and forecast potential improvements, using data from transactions, journeys, tickets, calendar, and refunds.

Section: KPI Summary Cards

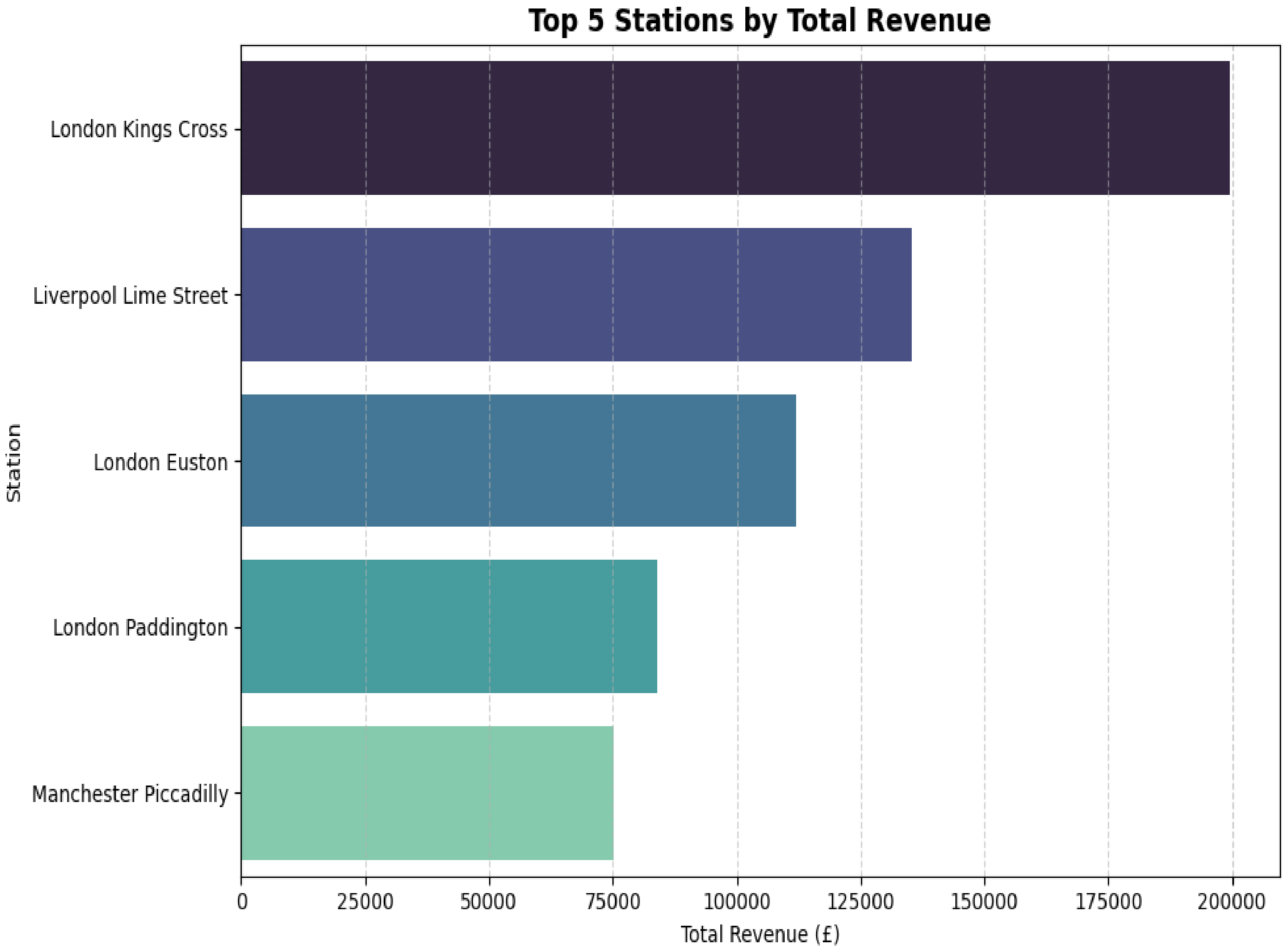
- **Total Passengers/Trips**: 31.7K
- **Total Revenue**: £741.9K
- **Total Refund Amount**: £38.7K
- **Average Ticket Price**: £23.4
- **Number of Routes**: 65
- **Cancelled Trips**: 0.8K (5.3%)
- **Delayed Trips**: 2.3K (7.2%)
- **Average Delay (Delayed Trips Only)**: 16.7 minutes
- **Refund Request Count (Yes)**: 559 (5.9%)
- **Delay Rate %**: 7.2%
- **Refund Rate %**: 5.2%
- **Peak Hour**: 08:00
- → These cards offer a quick snapshot of operational health, financial performance, and customer impact, with gauges for targets (e.g., delay rate <15%).

Section: Station Performance :

- **Title**: High Volume Stations
  - **Visual**: bar chart (Top 5 stations).
  - **Description:**Top stations by trips
  - → Highlights concentration of traffic and income, aiding decisions on resource allocation like staffing or infrastructure upgrades.



- **Title:** Top Revenue-Generating Stations
  - **Visual:** Bar chart (Top 5 stations).
  - **Description:**Top stations by total revenue
  - → This visualization highlights where the **highest ticket sales and earnings** are concentrated, supporting **strategic financial planning** and **investment prioritization** across stations.

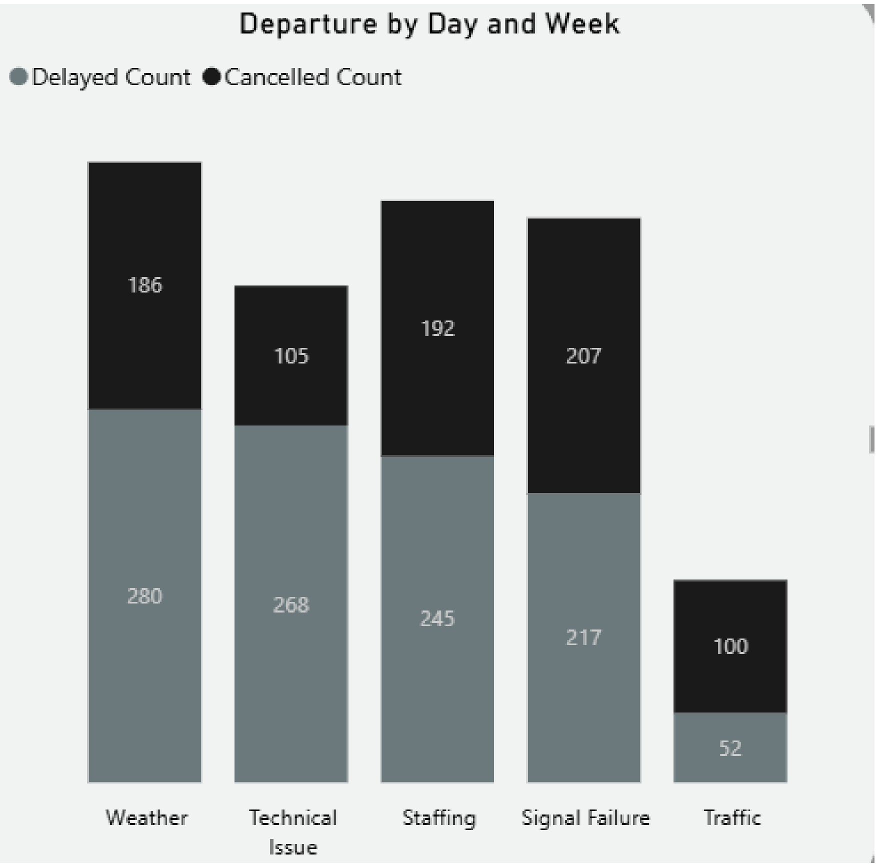


- **Interpretation:**

  This chart identifies the top-performing stations by trip volume and revenue.
  Stations like **Liverpool Lime Street**, **Manchester Piccadilly**, and **London Euston** handle the highest passenger traffic and contribute significantly to overall income.
  Insights like these help guide **resource allocation**, **staffing optimization**, and **infrastructure upgrades**.

Section: Delay Analysis

- **Title:** Departure Performance by Delay Cause
  - **Visual:** Stacked bar chart (Top 5 causes).
  - **Description:**
  - This visualization compares **delayed** and **cancelled departures** across major causes such as *Weather*, *Technical Issues*, *Staffing*, *Signal Failures*, and *Traffic*.
  - → Provides root cause analysis for delays and cancellations, essential for maintenance planning and reducing operational disruptions.

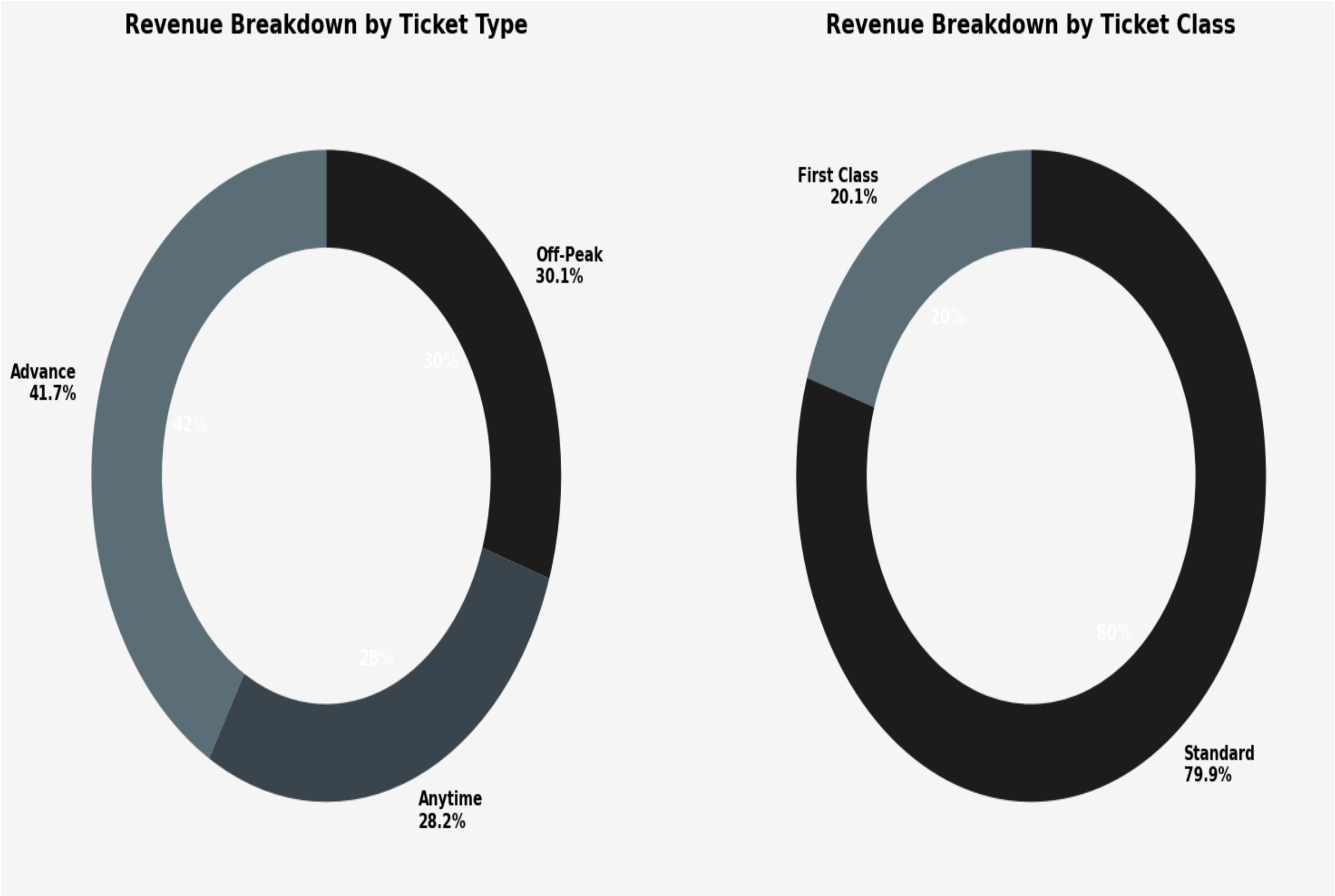


- **Interpretation:**

  The chart reveals that **Signal Failures** and **Staffing Problems** account for the highest number of total disruptions, with **207** and **192** cancellations respectively, while **Weather-related issues** led to the greatest number of overall delays (**280 delayed journeys**).
  This highlights key operational weaknesses where preventive maintenance, staffing optimization

Section: Revenue Breakdown

- **Title**: Revenue Distribution by Ticket Category
  - **Visual**: Two Donut Charts (Ticket Type & Ticket Class)
  - **Description**: This visualization compares revenue contribution across ticket categories, highlighting the dominance of Advance tickets and Standard class in overall revenue generation.
  - → Illustrates contribution to total revenue, helping optimize pricing strategies and identify high-value segments like First Class.

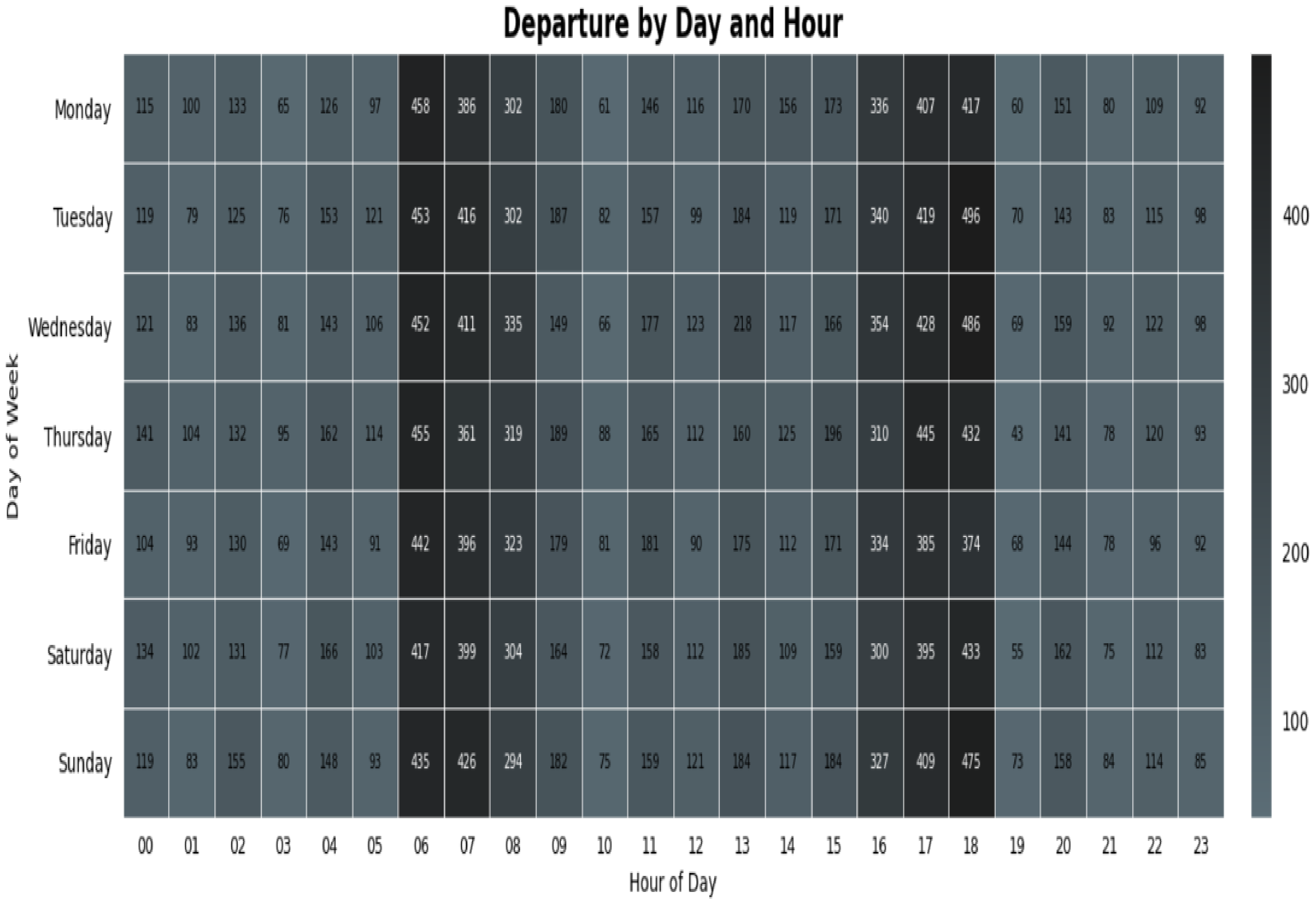


- **Interpretation**:

  This chart reveals the revenue distribution across ticket categories, highlighting key revenue drivers and premium segments. Advance tickets and Standard class dominate overall revenue generation, while First Class represents a significant premium segment despite lower volume. These insights help guide pricing strategy optimization, marketing focus, and resource allocation toward high-value ticket categories.

Section: Peak Hours

- **Title**: Count of Departure Time
- **Visual**: heatmap matrix for day vs. hour.
- Peak departures: 08:00 (highest trips), with breakdowns by AM/PM (e.g., 07:00–09:00 AM: 45% of daily trips).
- → Identifies congestion periods correlated with delays, supporting scheduling adjustments.



- **Interpretation**

  The analysis shows that 08:00 is the peak departure hour, with a strong concentration of trips between 07:00 and 09:00. These early-morning hours account for a significant share of total daily departures, indicating the highest operational pressure. This pattern highlights the critical window where scheduling efficiency, staffing, and platform availability have the greatest impact on reducing congestion and delays.

Section: Top Stations & Popular Routes

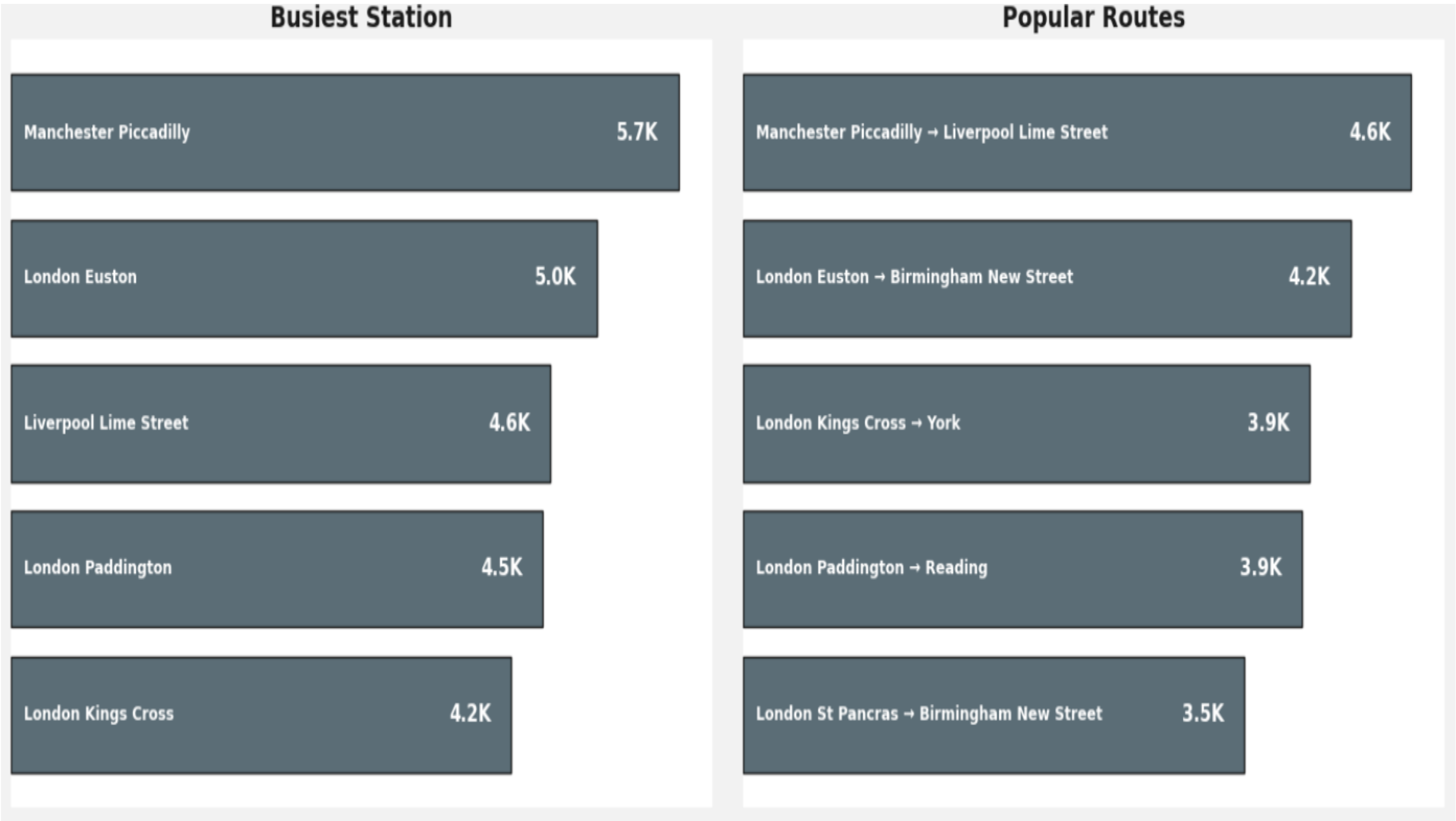
Title: Busiest Stations and Most Popular Routes

Visual: Dual Bar Charts (Top 5 Stations & Top 5 Routes)

The visual presents two side-by-side bar charts:

- **Busiest Station:** Top five stations ranked by total departures.

- **Popular Routes:** Highest-volume origin–destination routes based on trip counts.
  Both charts use the unified dashboard color theme (#5B6D76 and #1C1C1C), with clear value labels to highlight relative differences in traffic intensity.



- **Interpretation**

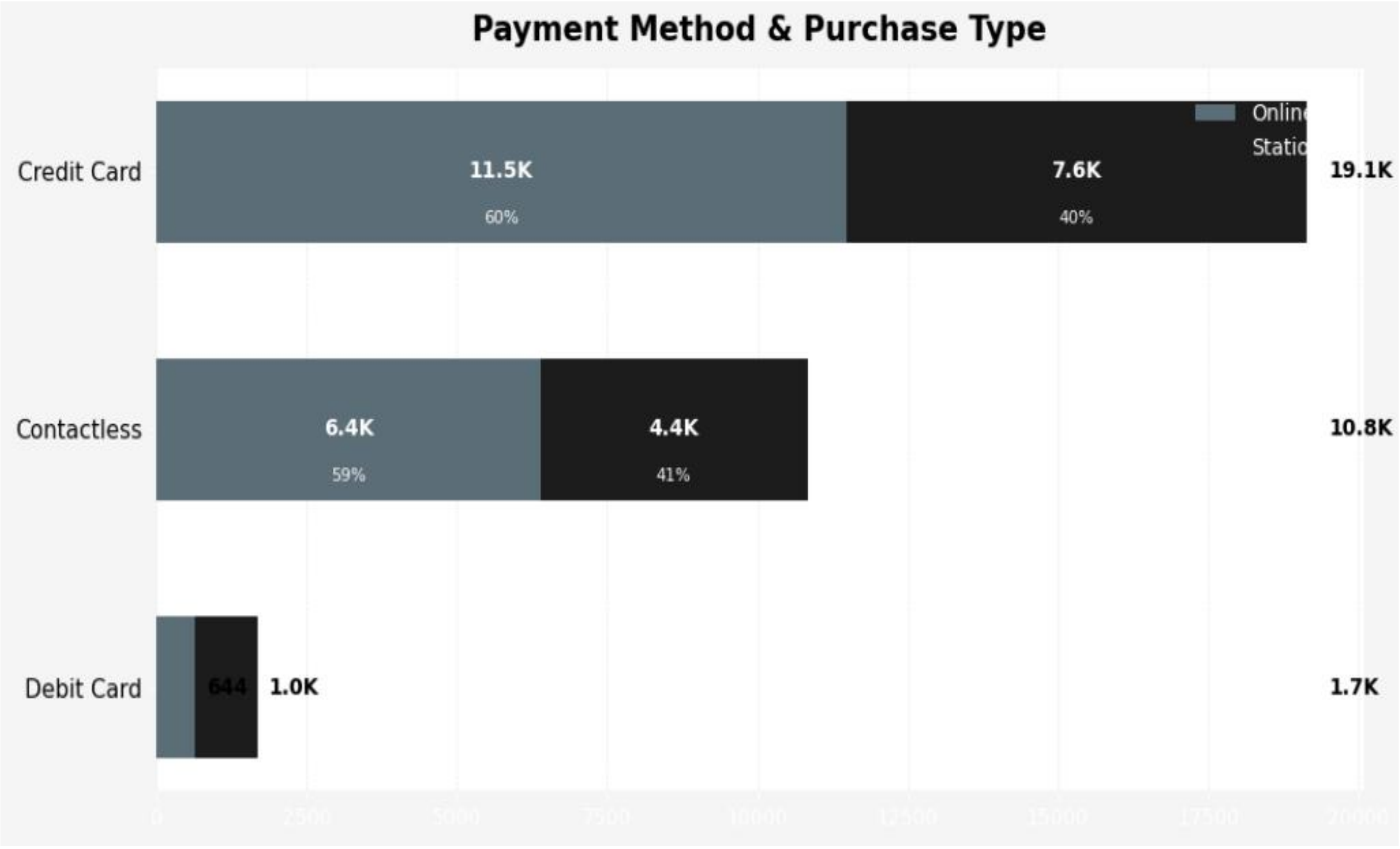
  The results show that Manchester Piccadilly, London Euston, and Liverpool Lime Street dominate as the busiest stations, acting as key operational hubs within the network. Similarly, the most popular routes—such as Manchester → Liverpool and London Euston → Birmingham—account for a significant share of overall trip volume.

  These concentration patterns signal where passenger demand is strongest, helping prioritize capacity improvements, staffing decisions, and targeted service enhancements. Focusing operational resources on these high-impact stations and routes can significantly improve flow efficiency and customer experience.

Section: Payment Method Performance

- **Visual: Stacked Horizontal Bar Chart (Online vs. Station Purchases)**

This visualization displays the total number of transactions per payment method, broken down into **Online** and **Station** purchases.
Each payment method is represented by a segmented horizontal bar, with transaction counts placed inside each segment for clear and quick interpretation.



- **Interpretation**

  The chart highlights **how customers prefer to pay** across different purchase channels:

  - Certain payment methods are predominantly used **in stations**, while others show stronger adoption in **online channels**.
  - The size of each bar segment reflects real demand, making it easy to identify:
    - The most frequently used payment methods
    - The balance between digital and station-based purchases
    - Opportunities to optimize underperforming payment options

  Overall, this visualization helps decision-makers **strengthen high-performing payment methods**, improve the customer payment experience, and refine channel strategies to enhance revenue and operational efficiency.

Storyline Summary (Narrative Flow)

- Start with KPIs → Quick overview of trips, revenue, delays, and refunds.
- Move to Financial Impact → Refund amounts and rates linked to delays.
- Dive into Operations → Busiest stations, routes, and delay causes.
- Analyze Revenue Mix → Ticket types/classes driving profits and Railcard usage.
- Explore Customer Behavior → Purchase channels, payments, and refund correlations.
- Close with Time Analysis → Peak hours and forecasts for future trends.

Insights & Actions

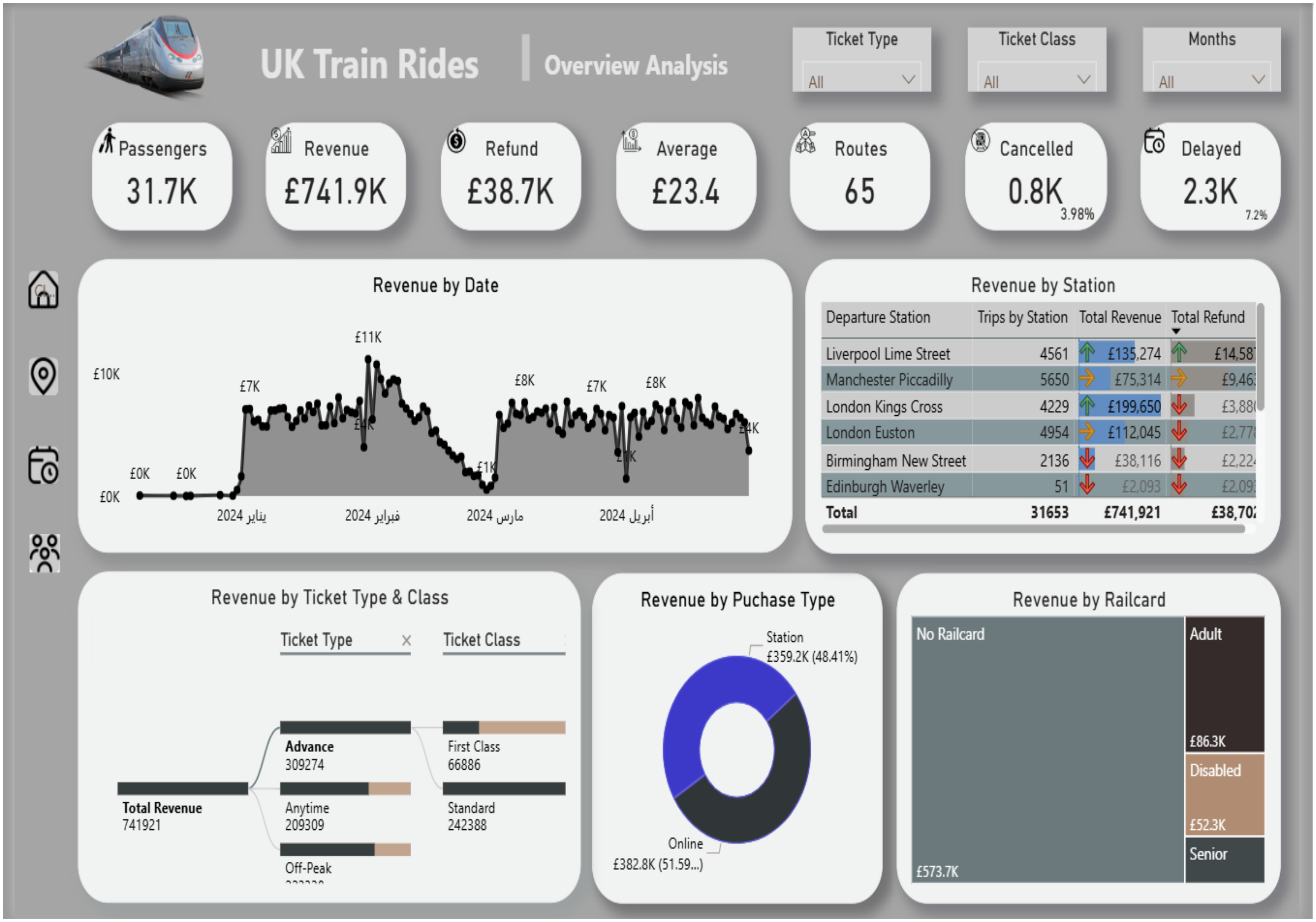
- **Operational Insight**: Signal Failure causes 38% of delays; recommend infrastructure upgrades in high-traffic routes like Liverpool-Manchester. Peak hours (08:00–09:00) show 20% higher delays—suggest adding trains or staff during these times.
- **Financial Insight**: Refunds total £38.7K (5.2% of revenue), mostly from delayed trips >15 minutes; implement automatic compensation thresholds to reduce losses. First Class contributes 27% of revenue but has higher refund rates—focus on premium service reliability for ROI.
- **Customer Insight**: 60% of purchases are online, with Credit Card dominance; promote digital discounts to boost adoption. Railcard usage at 30%—target seniors/disabled for growth. Delays >15 min lead to 16.9% refunds; enhance communication for better satisfaction.

Overall Impact

- This dashboard transforms raw data into actionable intelligence— from spotting operational inefficiencies (e.g., delay hotspots) to financial optimizations (e.g., refund mitigation)— empowering stakeholders to enhance punctuality, boost revenue, and improve customer loyalty in the UK railway network.

UI/UX Design:

the dashboard was done using power pi



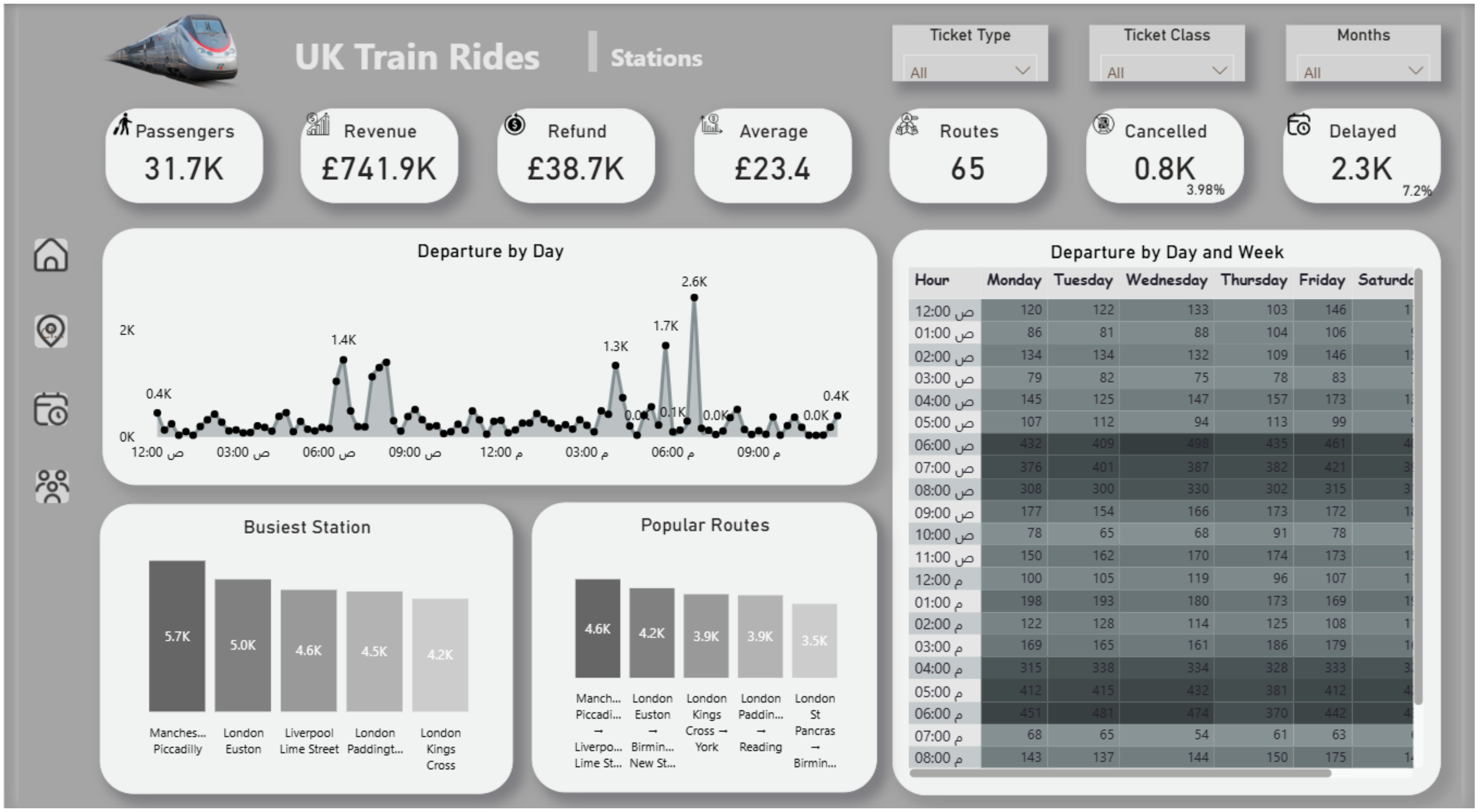
UK Train Rides | Overview Analysis

Ticket Type: All
Ticket Class: All
Months: All

Passengers 31.7K
Revenue £741.9K
Refund £38.7K
Average £23.4
Routes 65
Cancelled 0.8K 3.98%
Delayed 2.3K 7.2%

Revenue by Station

Departure Station	Trips by Station	Total Revenue	Total Refund
Liverpool Lime Street	4561	↑ £135,274	↑ £14,58
Manchester Piccadilly	5650	→ £75,314	→ £9,46
London Kings Cross	4229	↑ £199,650	↓ £3,88
London Euston	4954	→ £112,045	↓ £2,77
Birmingham New Street	2136	↓ £38,116	↓ £2,22
Edinburgh Waverley	51	↓ £2,093	↓ £2,09
Total	**31653**	**£741,921**	**£38,70**



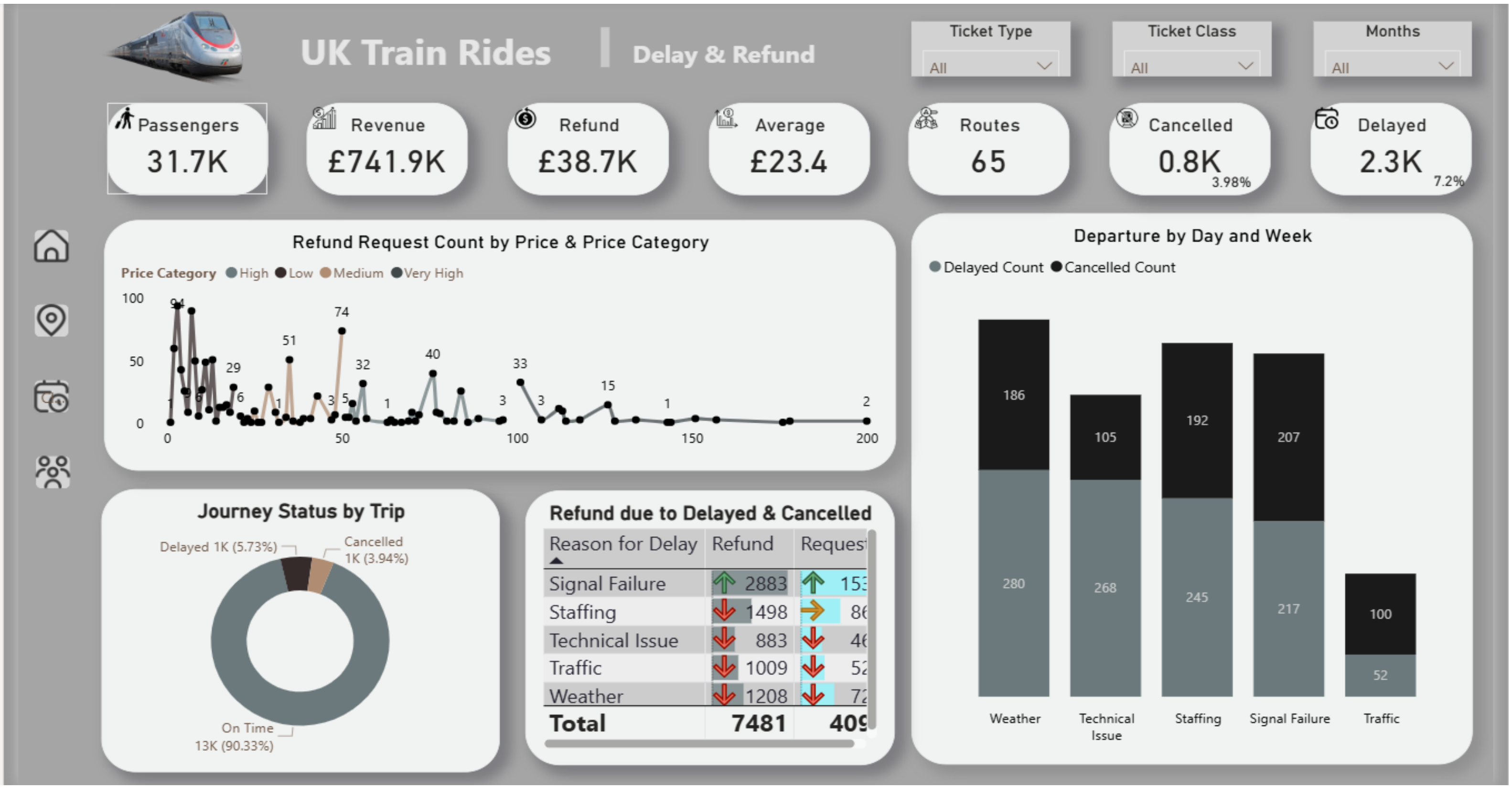
UK Train Rides | Stations

Ticket Type: All
Ticket Class: All
Months: All

Passengers	Revenue	Refund	Average	Routes	Cancelled	Delayed
31.7K	£741.9K	£38.7K	£23.4	65	0.8K 3.98%	2.3K 7.2%

Departure by Day and Week

Hour	Monday	Tuesday	Wednesday	Thursday	Friday	Saturda
12:00 ص	120	122	133	103	146	1
01:00 ص	86	81	88	104	106	
02:00 ص	134	134	132	109	146	1
03:00 ص	79	82	75	78	83	
04:00 ص	145	125	147	157	173	1
05:00 ص	107	112	94	113	99	
06:00 ص	432	409	498	435	461	4
07:00 ص	376	401	387	382	421	3
08:00 ص	308	300	330	302	315	3
09:00 ص	177	154	166	173	172	1
10:00 ص	78	65	68	91	78	
11:00 ص	150	162	170	174	173	1
12:00 م	100	105	119	96	107	1
01:00 م	198	193	180	173	169	1
02:00 م	122	128	114	125	108	1
03:00 م	169	165	161	186	179	1
04:00 م	315	338	334	328	333	3
05:00 م	412	415	432	381	412	4
06:00 م	451	481	474	370	442	4
07:00 م	68	65	54	61	63	
08:00 م	143	137	144	150	175	1



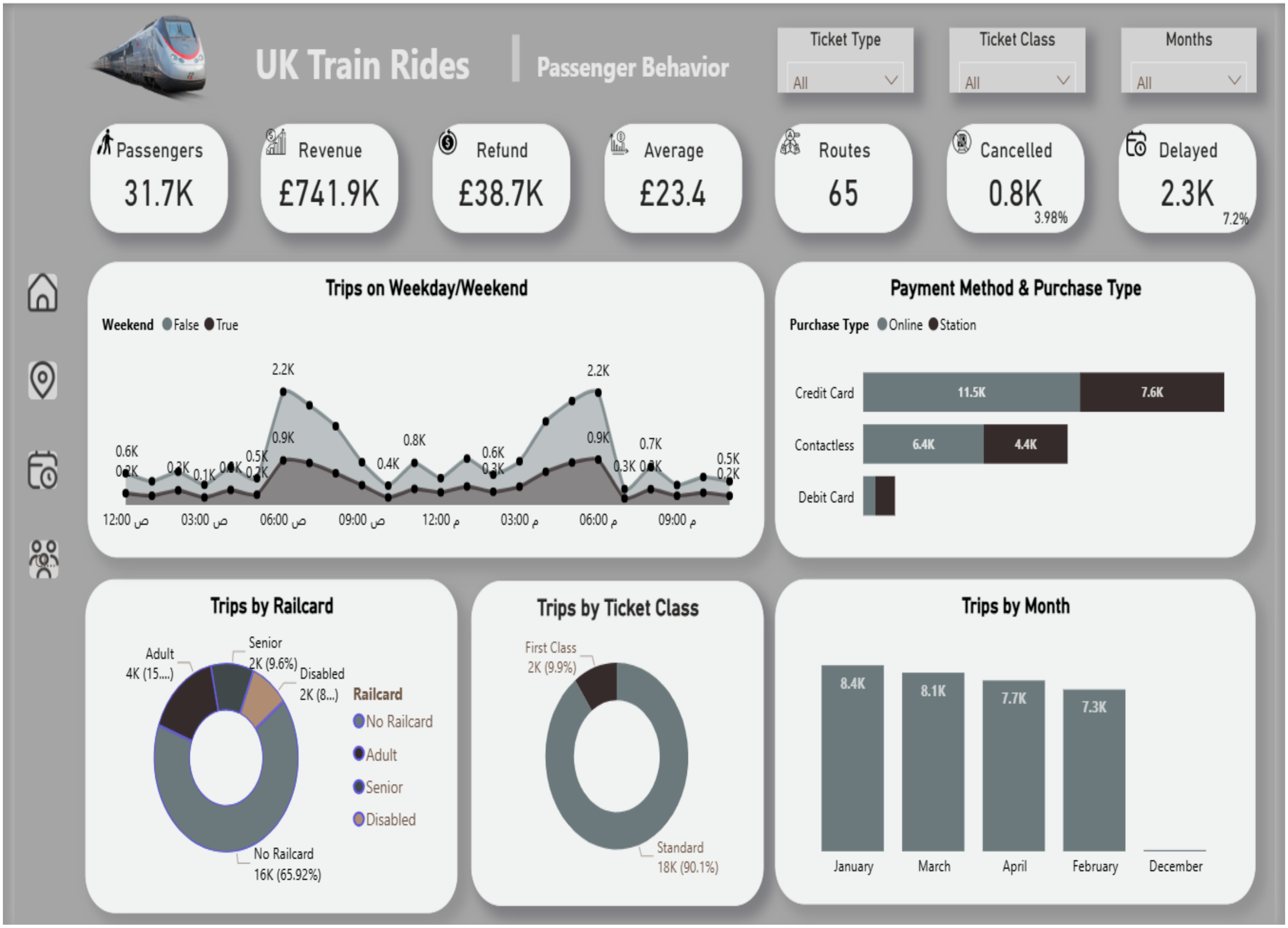
UK Train Rides | Delay & Refund

Ticket Type: All
Ticket Class: All
Months: All

Passengers	Revenue	Refund	Average	Routes	Cancelled	Delayed
31.7K	£741.9K	£38.7K	£23.4	65	0.8K 3.98%	2.3K 7.2%

Refund due to Delayed & Cancelled

Reason for Delay ▲	Refund	Request
Signal Failure	↑ 2883	↑ 153
Staffing	↓ 1498	→ 86
Technical Issue	↓ 883	↓ 46
Traffic	↓ 1009	↓ 52
Weather	↓ 1208	↓ 72
Total	**7481**	**409**



UK Train Rides | Passenger Behavior

Ticket Type: All

Ticket Class: All

Months: All

Passengers	Revenue	Refund	Average	Routes	Cancelled	Delayed
31.7K	£741.9K	£38.7K	£23.4	65	0.8K 3.98%	2.3K 7.2%