UK Train Rides

Introduction

This project involves analyzing railway network data using Power BI to understand operational performance, revenue, and customer behavior.

The report is based on data from journeys, stations, tickets, delays, and cancellations, and provides analytical dashboards to help make strategic decisions to improve service and increase revenue**.**

Description and analysis of the raw data file

The file contains data on railway ticket purchase transactions, and includes the following columns:

• Transaction ID: Unique identifier for each purchase

• Date of Purchase / Time of Purchase: Date and time of ticket purchase

• Purchase Type: Online or Station

• Payment Method: Credit Card, Contactless, Debit Card

• Railcard Discount Card Type: Adult, Senior, Disabled, None

• Ticket Class: Standard or First Class

• Ticket Type: Advance, Off-Peak, Anytime

• Price

• Departure Station / Arrival Destination

• Date of Journey / Departure Time / Arrival Time / Actual Arrival Time

• Journey Status: On Time, Delayed, Cancelled

• Reason for Delay: Signal Failure, Weather, Staffing

• Refund Request

Problems encountered with the data:

First: Reviewing and Addressing Raw Data Issues

The raw data presented several challenges that had to be addressed before any analysis or visualization could begin. The following is a comprehensive overview of the most prominent of these issues and how they were resolved:

---

1) Missing Values

Data analysis revealed deficiencies in some key fields, which may negatively impact the accuracy of the results. These deficiencies included:

• The absence of the "Reason for Delay" field for many flights, especially those that arrived on time.

• The absence of the "Actual Arrival Time" field for canceled flights.

•The "Railcard" column displaying the value "None" instead of a clear and understandable value.

Processing Actions

• Replace missing values in the "Reason for Delay" field with "No Delay" for journeys that did not experience delays.

• Replace the value "None" in the Railcard column with "No Railcard" for clarity.

2) Incorrect Data Formats

It was noted that some time and date fields are recorded in text format, which hinders calculations and time analysis.

Processing procedures:

• Convert date fields such as "Trip Date" and "Purchase Date" to the Date data type.

• Standardize the format of time fields (Purchase Time, Departure Time, Scheduled Arrival Time) in the HH:MM:SS format.

• Ensure that the ticket price is recorded as a numerical value ready for analytical use.

3) Excessive whitespace within text

Some text values or column names may contain unnecessary whitespace, which can lead to errors during classification or linking.

Processing procedures:

• Remove any unwanted whitespace from column names.

• Clean up text values within columns to ensure there are no spaces at the beginning or end.

4) Inconsistent Category Labels

Differences were discovered in the naming of some categories that express the same meaning, such as:

• Using "Staffing" instead of "Staff Shortage".

• Using "Weather Conditions" instead of "Weather".

• Different spellings of some values, such as "Signal failure" instead of "Signal Failure".

Processing procedures:

• Standardize all names to a single format, such as:

• Replace Staffing with Staff Shortage

• Replace Weather Conditions with Weather

• Standardize the spelling of Signal Failure across all records

5) Duplicate Records

The review revealed several duplicate rows that may affect the accuracy of comparisons and results.

Processing procedures:

• Check for duplicates based on matching flight data.

• Delete duplicate records, retaining only one copy of each instance.

Data Cleanup Summary

After performing all the previous cleanup steps, the data is:

• Free of unexplained missing values

• Uniform in formatting and formulas

• Free of extra spaces

• Consistent in category labels

• Free of duplication

Therefore, it became ready for use in building accurate and reliable analytical dashboards, paving the way for a more professional analysis phase in this project.

Python

1) Importing libraries:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import plotly.express as px

import seaborn as sns
```

The data management and handling libraries (pandas), arithmetic operations (numpy), graphing (matplotlib), (seaborn) and (ploty express) were imported in preparation for visual processing and analysis.

2) Reading the file:

```
df = pd.read_csv(r"C:\Users\CR_86\Desktop\Python Final Project\railway.csv")

df.head()
```

The raw data file in CSV format was loaded into a DataFrame named df.

3) Data cleanliness:

• Missing data:

df.isnull().sum()

```
[17]: ## Missing Values
      df.isnull().sum()

[17]: 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
```

Data analysis revealed deficiencies in some key fields, which could negatively impact the accuracy of the results. These deficiencies included:

•The absence of a "Reason for Delay" field for many flights, especially those that arrived on time.

•The absence of an "Actual Arrival Time" field for canceled flights.

•The "Railcard" field displays "None" instead of a clear and understandable value.

Processing procedures:

•Replace missing values in the "Reason for Delay" field with "No Delay" for journeys that did not experience delays.

•Replace the value "None" in the Railcard column with "No Railcard" for clarity.

df['Railcard'] = df['Railcard'].fillna('No Railcard')

df['Reason for Delay'] = df['Reason for Delay'].fillna('No Delay')

df['Actual Arrival Time'] = df['Actual Arrival Time'].fillna('Cancelled')

df.head()

```
## Filling Missing
df['Railcard'] = df['Railcard'].fillna('No Railcard')
df['Reason for Delay'] = df['Reason for Delay'].fillna('No Delay')
df['Actual Arrival Time'] = df['Actual Arrival Time'].fillna('Cancelled')
df.head()
```

The data was checked again to ensure that there were no missing values:

```
df.isnull().sum()
```

```
Transaction ID              0
Date of Purchase            0
Time of Purchase            0
Purchase Type               0
Payment Method              0
Railcard                    0
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         0
Journey Status              0
Reason for Delay            0
Refund Request              0
dtype: int64
```

•Review data types:

df.info()

```
## Data Overview
df.info()
```

```
<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                31653 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     31653 non-null  object
 15  Journey Status          31653 non-null  object
 16  Reason for Delay        31653 non-null  object
 17  Refund Request          31653 non-null  object
dtypes: int64(1), object(17)
memory usage: 4.3+ MB
```

The "Trip Date" and "Purchase Date" columns have been converted to a date format (Datetime) to facilitate time-based operations such as sorting and analysis over time.

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')
df.info()

```
# Change Type
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')
df.info()
```

•Unifying values:

df['Reason for Delay'] = df['Reason for Delay'].replace({
'Staffing': 'Staff Shortage',
'Weather Conditions': 'Weather',
'Signal failure': 'Signal Failure' })

```
## Modifying and Standardizing Column Labels
df['Reason for Delay'] = df['Reason for Delay'].replace({
    'Staffing': 'Staff Shortage',
    'Weather Conditions': 'Weather',
    'Signal failure': 'Signal Failure'
})
df
```

Multiple values that express the same meaning, such as replacing "Staffing" with "Staff Shortage" and "Weather Conditions" with "Weather", were standardized to ensure consistency in the analysis.

•Extra spaces:

Remove excess spaces from column names and text values.
str_cols = df.select_dtypes(include='object').columns
df[str_cols] = df[str_cols].apply(lambda x: x.str.strip())

```
## Removing Extra Whitespace from Column Names and Text Values
str_cols = df.select_dtypes(include='object').columns
df[str_cols] = df[str_cols].apply(lambda x: x.str.strip())
```

- **Reviewing the data after processing:**

```
df.isnull().sum()
df.info()
```

The columns were checked again to ensure they were free of missing values, and that all columns contained the correct data type.

- **Save a copy of the data:**

```
df.to_csv(r"C:\Users\CR_86\Desktop\Python Final Project\railway_cleaned.csv", index=False)
```

A new copy of the cleaned data was saved under the name railway_cleaned to work on in the next steps without needing to re-clean.

KPIS

```
## Total Revenue

Total_Revenue = df ['Price']. sum ()

print (f"Total Revenue: Total_Revenue:,.0f}")


## Average Ticket Price

Average_Ticket_price = round(df['Price']. mean(),1)

print (f"Average Ticket price: {Average_Ticket_price:.0f}")


## Total Users

Total_Transaction = df.shape[0]

print (f"Total Transaction: {Total_Transaction:,.0f}")


## Total Refund $ & Refund Count

refund_mask = df['Refund Request'] == 'Yes'

refund_count = refund_mask.sum()

Refund_Total= df.loc[refund_mask, 'Price']. sum ()

print(f"Total Refunded Revenue: Refund_Total:.0f}")

print (f"Qty Tickets Refunded: {refund_count:,.0f}")
```

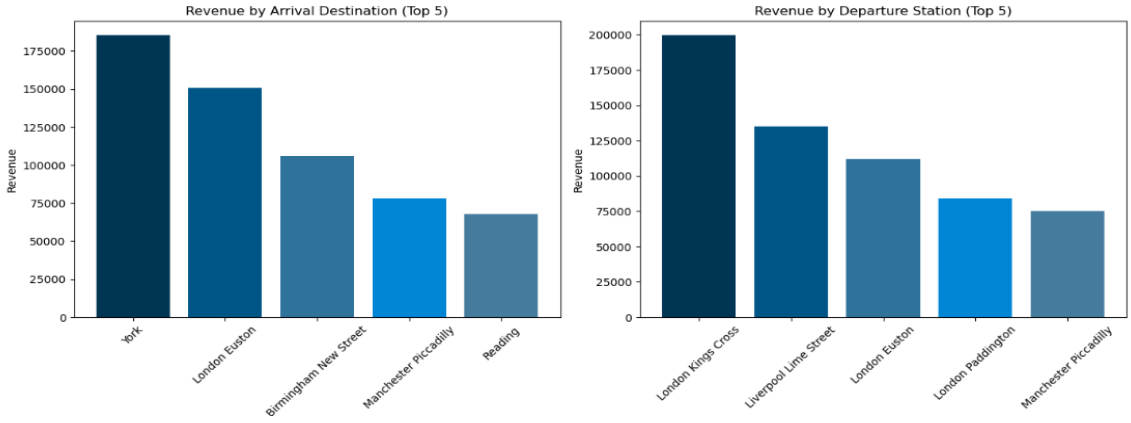
```
refund_percentage = (Refund_Total / Total_Revenue) * 100
print(f"Refund % of Total Revenue: {refund_percentage:.2f}%")

## Total Routes
df.insert(9,'Routes',df['Departure Station']+" ➔ "+ df['Arrival Destination'])
unique_routes = df['Routes'].nunique()
print(f"Number of Routes: {unique_routes:,}")

## Cancelld & Delayed Trip
cancelled_count = (df['Journey Status'] == 'Cancelled').sum()
delayed_count = (df['Journey Status'] == 'Delayed').sum()
on_time_trips = (df['Journey Status'] == 'On Time').sum()
print(f"Cancelled Trips: {cancelled_count:,}")
print(f"Delayed Trips: {delayed_count:,}")
print(f"On Time Trips: {on_time_trips:,}")
on_time_percentage = (on_time_trips / Total_Transaction) * 100
print(f"On Time Percentage: {on_time_percentage:.2f}%")
```

Visualization

1) Revenue By Arrival & Departure Station

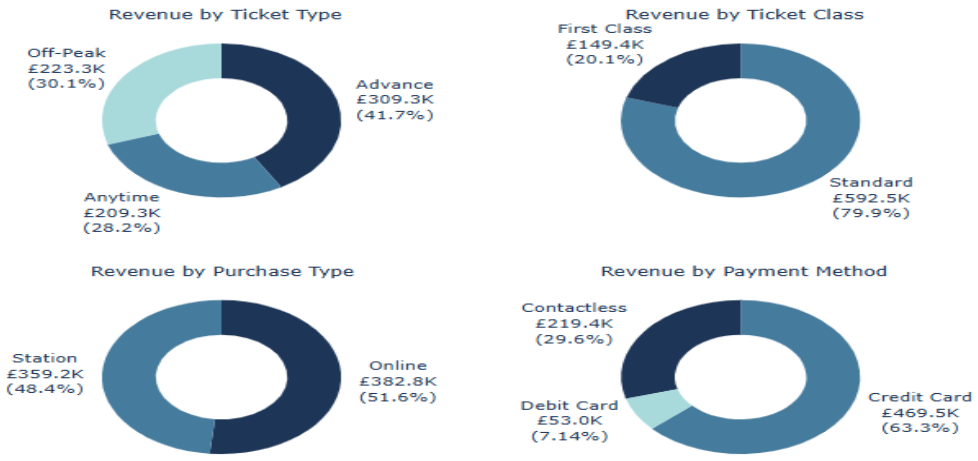
```
colors = ['#003554', '#005587', '#2F739B', '#0086D4', '#457b9d']
rev_arrival = df.groupby("Arrival Destination")["Price"].sum().sort_values(ascending=False).head(5)
plt.figure(figsize=(14,10))
plt.subplot(2,2,1)
plt.bar(rev_arrival.index, rev_arrival.values, color=colors[:5])
plt.xticks(rotation=45)
plt.title("Revenue by Arrival Destination (Top 5)")
plt.ylabel("Revenue")
```

```
rev_departure = df.groupby("Departure
Station")["Price"].sum().sort_values(ascending=False).head(5)

plt.subplot(2,2,2)

plt.bar(rev_departure.index, rev_departure.values, color=colors[:5])

plt.xticks(rotation=45)

plt.title("Revenue by Departure Station (Top 5)")

plt.ylabel("Revenue")

plt.tight_layout()
```



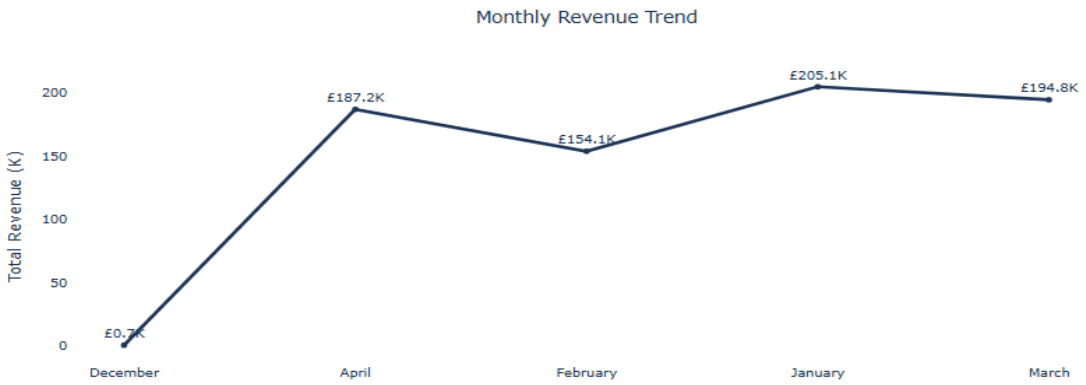

```
categories = {

  ' Revenue by Ticket Type': df.groupby('Ticket Type')['Price'].sum() / 1000,

  ' Revenue by Ticket Class': df.groupby('Ticket Class')['Price'].sum() / 1000,

   'Revenue by Purchase Type': df.groupby('Purchase Type')['Price'].sum() / 1000,

  ' Revenue by Payment Method': df.groupby('Payment Method')['Price'].sum() / 1000}

colors = ['#1d3557', '#457b9d', '#a8dadc']

fig = make_subplots(

  rows=2, cols=2,

  specs=[

    [{'type': 'domain'}, {'type': 'domain'}],

    [{'type': 'domain'}, {'type': 'domain'}]],

  subplot_titles=list(categories.keys()))

positions = [(1,1), (1,2), (2,1), (2,2)]

for pos, data in zip(positions, categories.values()):
```

```
    fig.add_trace(go.Pie(
        labels=data.index,
        values=data.values,
        hole=0.55,
        marker=dict(colors=colors),
        texttemplate='%{label}<br>£%{value:.1f}K<br>(%{percent})',
        textposition='outside'),
        row=pos[0], col=pos[1] )
fig.update_layout(
    font=dict(size=15, color='#1d3557'),
    height=700, width=1100,
    paper_bgcolor='white',
    showlegend=False,
    title=dict(
        x=0.5,
        y=0.95 ),
    margin=dict(t=100) )
for i, annotation in enumerate(fig['layout']['annotations']):
    annotation['y'] += 0.05
fig.show()
```



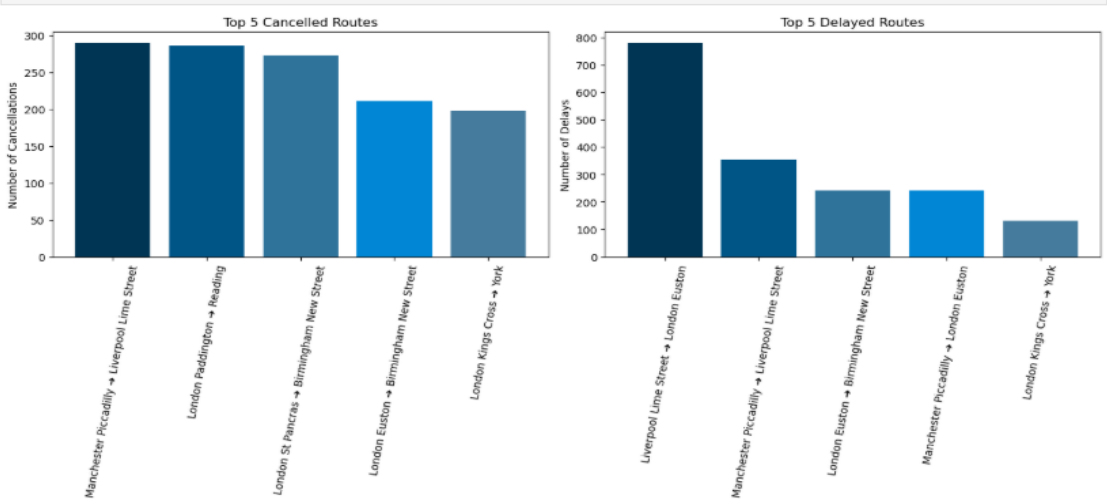
3) Monthly Revenue Trend

```
df['Purchase_Year'] = df['Date of Purchase'].dt.year
df['Purchase Month Name'] = df['Date of Purchase'].dt.month_name()
monthly_rev = df.groupby(['Purchase_Year', 'Purchase Month Name'], as_index=False)['Price'].sum()
monthly_rev['Price'] = monthly_rev['Price'] / 1000
monthly_rev = monthly_rev.sort_values('Purchase_Year')
fig = px.line(
   monthly_rev,
   x='Purchase Month Name',
   y='Price',
   markers= True,
   text='Price',
   title=' Monthly Revenue Trend')
fig.update_traces(
   line_color='#1d3557',
   line_width=3.5,
   texttemplate='£%{text:.1f}K',
   textposition='top center')
fig.update_layout(
      title_x=0.5,
   yaxis_title='Total Revenue (K)',
   xaxis_title=None,
   font=dict(size=13, color='#1d3557'),
   plot_bgcolor='white',
   paper_bgcolor='white',
   height=500,
   width=1150)
fig.show()
```



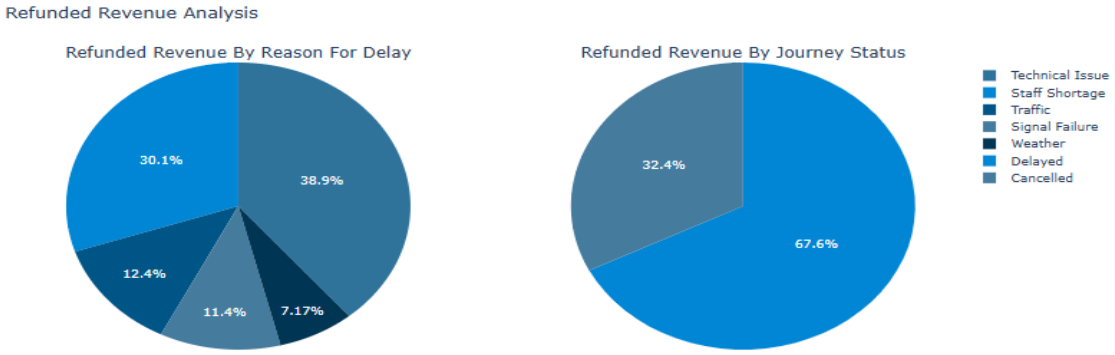
4) Top 5 Cancelled & Delayed Routes

```
cancelled_routes = df[df['Journey Status']=='Cancelled'].groupby('Routes')['Journey Status'].count().sort_values(ascending=False).head(5)

delayed_routes = df[df['Journey Status']=='Delayed'].groupby('Routes')['Journey Status'].count().sort_values(ascending=False).head(5)

colors = ['#003554', '#005587', '#2F739B', '#0086D4', '#457b9d']

plt.figure(figsize=(14,10))

# Top 5 Cancelled Routes

plt.subplot(2,2,1)

plt.bar(cancelled_routes.index, cancelled_routes.values, color=colors[:5])

plt.xticks(rotation=80)

plt.title("Top 5 Cancelled Routes")

plt.ylabel("Number of Cancellations")

# Top 5 Delayed Routes

plt.subplot(2,2,2)

plt.bar(delayed_routes.index, delayed_routes.values, color=colors[:5])

plt.xticks(rotation=80)

plt.title("Top 5 Delayed Routes")

plt.ylabel("Number of Delays")

plt.tight_layout()

plt.show()
```



4) Refunded Revenue by Reason for Delay & Journey Status

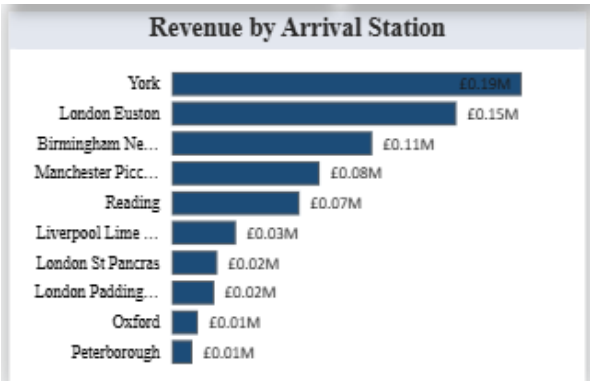
```
import plotly.graph_objects as go

from plotly.subplots import make_subplots

colors = ['#457b9d', '#0086D4', '#2F739B', '#005587', '#003554','#BBE8F2', '#94D7F2', '#5FB6D9']

refund_df = df[df['Refund Request'] == 'Yes']

revenue_by_delay = refund_df.groupby('Reason for Delay')['Price'].sum().reset_index()

revenue_by_status = refund_df.groupby('Journey Status')['Price'].sum().reset_index()

fig = make_subplots(rows=1, cols=2, specs=[[{'type':'domain'}, {'type':'domain'}]],

                    subplot_titles=['Refunded Revenue By Reason For Delay', 'Refunded Revenue By Journey Status'])

fig.add_trace(go.Pie(labels=revenue_by_delay['Reason for Delay'],

                     values=revenue_by_delay['Price'],

                     name="Reason For Delay",

                     marker_colors=colors),

              row=1, col=1)

fig.add_trace(go.Pie(labels=revenue_by_status['Journey Status'],

                     values=revenue_by_status['Price'],

                     name="Journey Status",

                     marker_colors=colors),

              row=1, col=2)

fig.update_layout(title_text="Refunded Revenue Analysis", height=500, width=1000)

fig.show()
```



Power BI dashboard

I created an interactive **Power BI dashboard** consisting of three pages to provide a comprehensive analysis of train journey data, covering performance, customer behavior, and revenue insights.

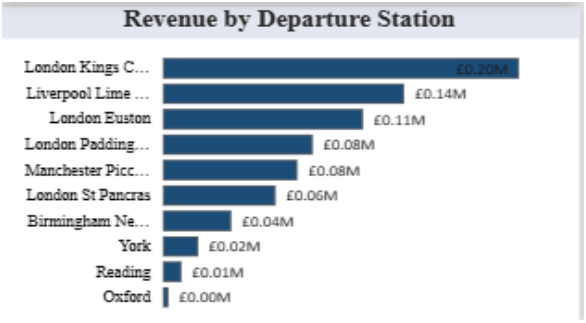
Revenue



Revenue by Arrival Station
Top revenue by arrival station:

1. York – £0.19M
2. London Euston – £0.15M
3. Birmingham New Street – £0.11M
4. Manchester Piccadilly – £0.08M

- York is an important tourist and historical station → many trips from London.
- London Euston is the northern gateway and a major hub for fast trains.
- Birmingham and Manchester are business and daily travel cities.

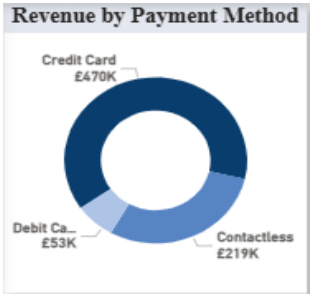


Revenue by Departure Station
Top revenue by departure station:

1. London Kings Cross – £0.20M
2. Liverpool Lime Street – £0.14M
3. London Euston – £0.11M

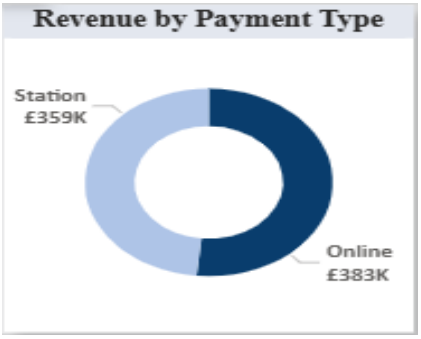
- London is a main travel hub between cities.
- Kings Cross specifically connects London to the North (Edinburgh, York).
- Liverpool and Manchester have significant business/university traffic.

Revenue by Payment Method



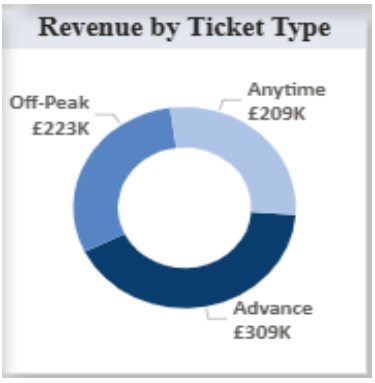
- Credit Card = £470K
- Contactless = £219K
- Debit Card = £53K
- British travelers mainly use credit cards for travel because they offer reward points.
- Contactless is very common within London stations and its use is growing.
- Debit cards are less used as they don't provide benefits.

Revenue by Payment Type (Station vs Online)



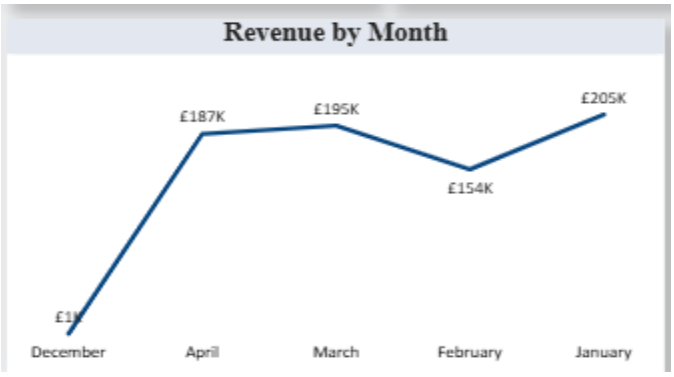
- Station = £359K
- Online = £383K
- Many rely on mobile or website ticket purchases.
- However, a significant portion still buys at stations due to:
  - Last-minute trips
  - Changing plans
  - Lack of trust in apps

Revenue by Ticket Type



- Anytime = £209K
- Off-Peak = £223K
- Advance = £233K
- Advance is the highest because it's cheaper, so many buy in advance.
- Off-Peak comes next as people avoid peak prices.
- Anytime is also high because it allows travel at any time, so the price is higher.

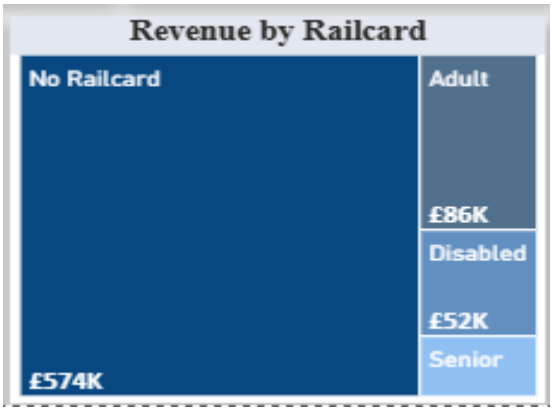
Revenue by Month



- December = £118K
- April = £187K
- March = £195K
- February = £154K
- January = £205K
- January usually sees high travel (return to school/work).

- March and April include Easter holidays → tourism travel increases.
-

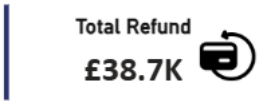
Revenue by Railcard (Treemap)



- No Railcard = £574K
- Adult Railcard = £86K
- Disabled = £52K
- Senior = £29K
- Most revenue comes from "No Railcard" because:
    1. Their number is much larger.
    2. They pay full price without discounts.
- Adult, Senior, and Disabled use discounts, so revenue from them is lower.

Delay & Refund

(KPIs)



Total Refund: £38.7K
Total amount refunded to customers due to delays or cancellations.

On Time: 27.5K (86.8%)
Number of journeys that arrived on time and their percentage of the total, indicating operational efficiency.

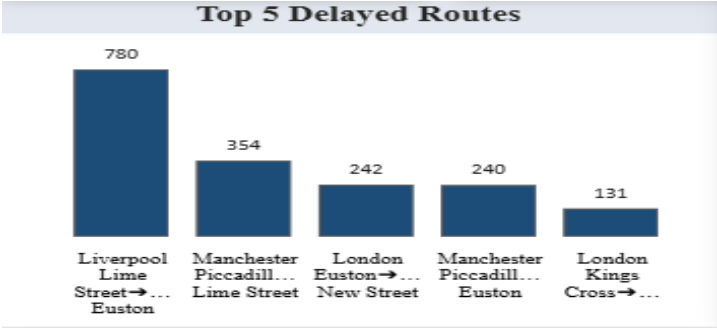
Cancelled: 1,880 (5.94%)
Number and percentage of cancelled journeys, reflecting service disruptions.

Delayed: 2,292 (7.2%)
Number and percentage of delayed journeys, showing punctuality performance.

Top 5 Delayed Routes

Bar chart showing the five routes with the highest delays:

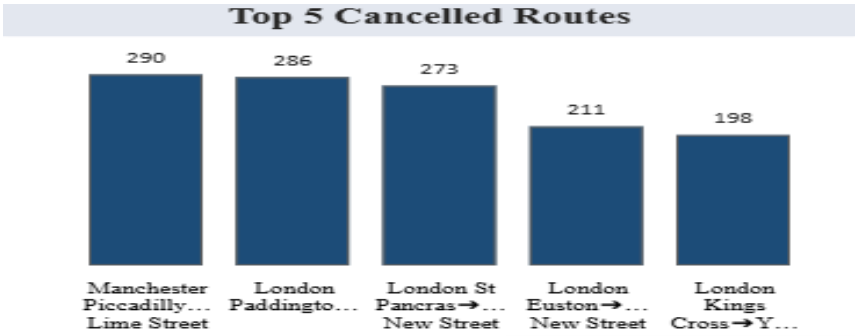
- Liverpool Lime Street → Euston ranks first (780 delays).
- Followed by Manchester and London routes. Insight: These are high-traffic routes, which increases the likelihood of delays.



Top 5 Cancelled Routes

Routes most affected by cancellations:

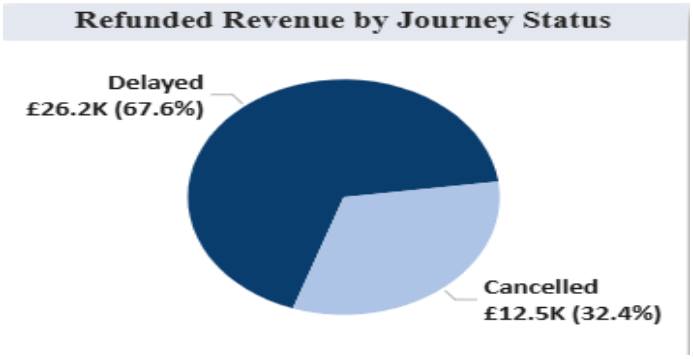
- Manchester Piccadilly → Lime Street leads the list.
- London Paddington → Reading also appears prominently. Insight: Cancellations are concentrated on major routes, likely due to operational or capacity issues.



Refunded Revenue by Journey Status

Pie chart showing refund distribution by journey status:

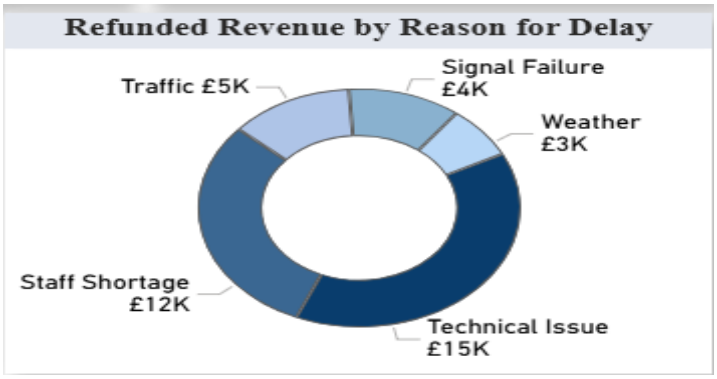
- Delayed: £26.2K (67.6%)
- Cancelled: £12.5K (32.4%) Insight: Delays have a greater financial impact than cancellations.



Refunded Revenue by Reason for Delay

Pie chart showing refund distribution by delay reason:

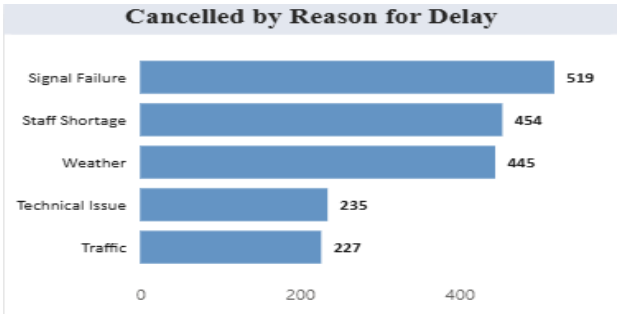
- Staff Shortage: £12K
- Technical Issue: £15K
- Signal Failure: £4K
- Weather: £3K Insight: Technical issues and staff shortages are the biggest contributors to revenue loss.



Cancelled by Reason for Delay

Horizontal bar chart showing reasons for cancellations:

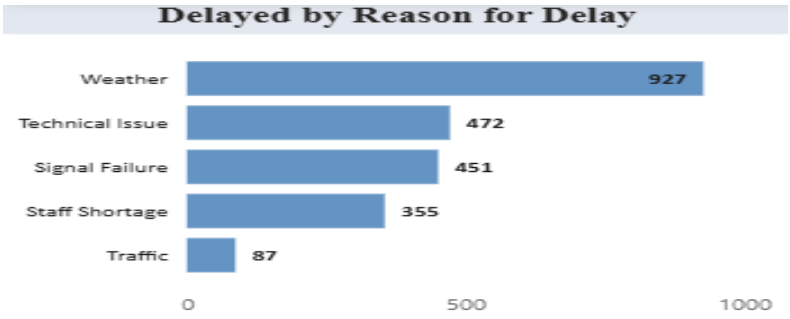
- Signal Failure and Staff Shortage dominate. Insight: Infrastructure and workforce challenges are key drivers of cancellations.



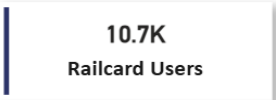
Delayed by Reason for Delay

Horizontal bar chart showing reasons for delays:

- Weather tops the list (927 cases).
- Followed by Technical Issues and Signal Failures. Insight: External factors like weather significantly affect punctuality.



Customer & Purchase Behavior



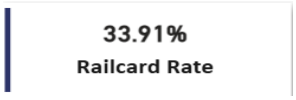
1) Railcard Users — 10.7K

Shows the number of customers who used a Railcard during the period covered by the data.

- This means roughly one-third of users use a Railcard.
- This makes sense because Railcards offer a 33% discount on most train tickets, so many passengers prefer buying a Railcard to benefit from the discount.

---

2) Railcard Rate — 33.91%

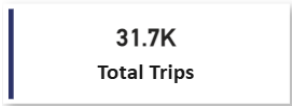


Percentage of Railcard users out of all travelers.

- About one-third of passengers use Railcards, while the rest pay full price.
- This aligns with reality in the UK, where around 7 million people hold Railcards, but not all travelers are eligible (students, seniors, families).

---

3) Total Trips — 31.7K

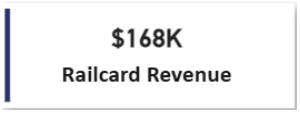


Total number of trips made by users.

- Indicates high usage, suggesting the data represents a busy travel period such as work months or holiday seasons.

---

4) Railcard Revenue — $168K



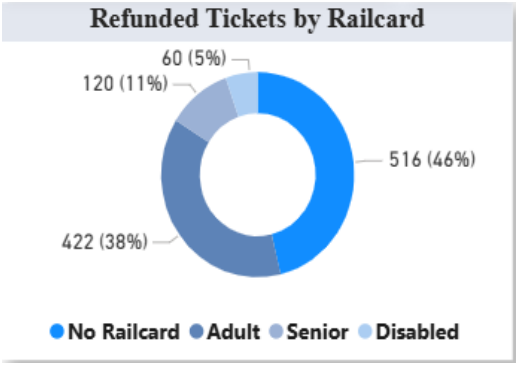
Total revenue from tickets purchased with Railcard discounts.

- Although Railcard holders get a discount, frequent travel results in high overall revenue.
- Railcard users often travel more than non-users because the cost is lower.

---

5) Refunded Tickets by Railcard (Donut Chart)

Number of refunded tickets by Railcard type:



- No Railcard
- Adult
- Senior
- Disabled

Insights:

- “No Railcard” has the highest refund requests.

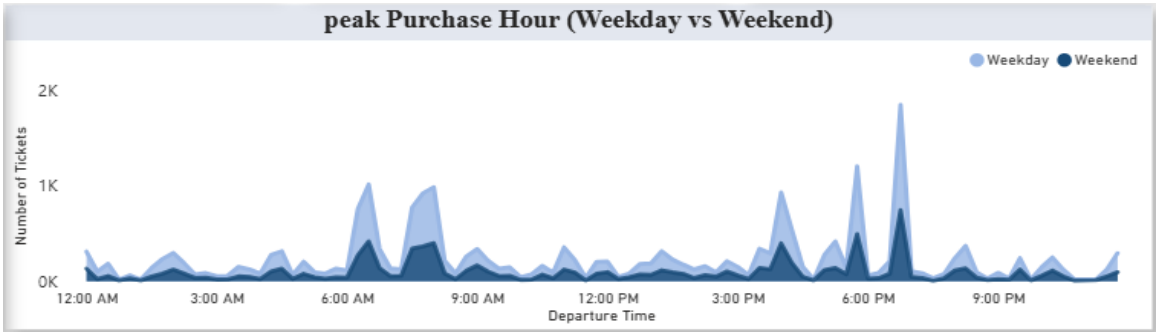
Reasons:

1. This group is the largest overall, so naturally, refunds are higher.

2. Railcard users usually buy cheaper tickets → lower financial loss → fewer refund requests.
3. Some Railcard ticket types are linked to “non-refundable” fares.

---

6) Peak Purchase Hour (Weekday vs Weekend

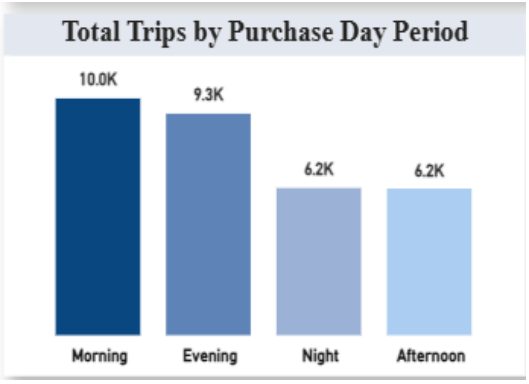


Ticket purchase timing during the day, comparing weekdays and weekends:

- Weekdays:
  - Peak between 6–9 AM → commuters (employees/students)
  - Second peak 5–7 PM → return trips from work
- Weekends:
  - Purchases spread more evenly throughout the day, as people travel for leisure rather than work.

---

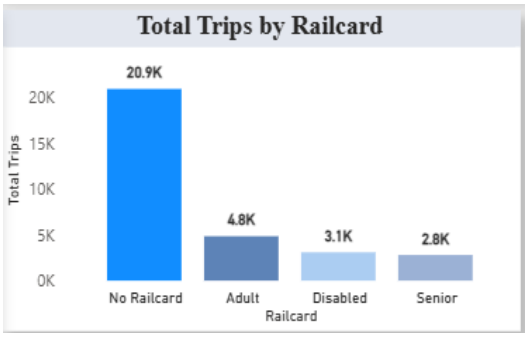
7) Total Trips by Purchase Day Period



Morning – Evening – Night – Afternoon

- Highest number of trips purchased in:
  1. Morning
  2. Evening
- Lowest in Night
- Morning and evening represent train rush hours.
- Night is lower because trains reduce after 10 PM.
- Afternoon is moderate due to leisure travel.

8) Total Trips by Railcard



Distribution of trips by Railcard type:

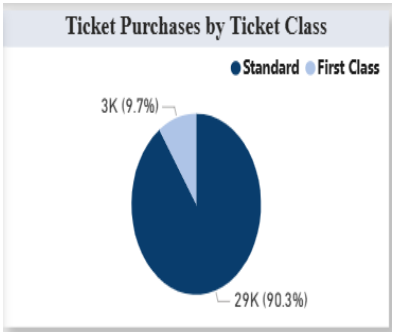
- Highest segment: No Railcard — 20.9K trips
- Then Adult
- Then Disabled
- Then Senior

Reasons why most travelers don't use Railcards:

1. Many passengers are employees with monthly or annual subscriptions that don't require Railcards.
2. Railcards have age and eligibility conditions — not available to everyone.
3. Some travelers don't travel frequently enough to justify buying a Railcard.

---

9) Ticket Purchases by Ticket Class



Standard tickets = 90%
First Class tickets = 10%
Insights:

- In the UK, most journeys are in Standard class because:
  - Cheaper
  - More seats available
  - First Class is usually for long-distance or business travel
- A 10% share for First Class is very realistic for UK train data.