UK Train Rides

○ مقدمة

المشروع عبارة عن **تحليل بيانات شبكة السكك الحديدية** باستخدام Power BI بهدف فهم الأداء التشغيلي، الإيرادات، وسلوك العملاء.

يعتمد التقرير على بيانات الرحلات، المحطات، التذاكر، التأخيرات، والإلغاءات، ويقدم لوحات تحليلية تساعد في اتخاذ قرارات استراتيجية لتحسين الخدمة وزيادة الإيرادات.

وصف وتحليل ملف البيانات الخام

الملف يحتوي على بيانات معاملات شراء تذاكر السكك الحديدية، ويتضمن الأعمدة التالية:

- **Transaction ID** معرف فريد لكل عملية شراء
- **Date of Purchase / Time of Purchase** تاريخ ووقت شراء التذكرة
- **Purchase Type** طريقة الشراء Online أوStation
- **Payment Method** وسيلة الدفعDebit Card ،Contactless ، Credit Card
- **Railcard** نوع بطاقة التخفيضNone ،Disabled ،Senior ، Adult
- **Ticket Class** درجة التذكرة Standard أوFirst Class
- **Ticket Type** نوع التذكرةAnytime ،Off-Peak ، Advance
- **Price** سعر التذكرة.
- **Departure Station / Arrival Destination** محطة المغادرة والوصول
- **Date of Journey / Departure Time / Arrival Time / Actual Arrival Time** تفاصيل الرحلة
- **Journey Status** حالة الرحلةCancelled ،Delayed ، On Time
- **Reason for Delay** سبب التأخيرStaffing ،Weather ، Signal Failure
- **Refund Request** هل تم طلب استرداد المبلغ أم لا

○ المشكلات التي واجهت البيانات:

أولاً: مراجعة ومعالجة مشكلات البيانات الخام

شهدت البيانات الخام عدداً من التحديات التي كان لابد من التعامل معها قبل البدء في التحليل أو إعداد أي تمثيل بصري. فيما يلي عرض شامل لأبرز هذه المشكلات وكيف تم معالجتها:

1- القيم المفقودة (Missing Values)

أظهر فحص البيانات وجود نقص في بعض الحقول الأساسية، وهو ما قد ينعكس سلباً على دقة النتائج. تمثل هذا النقص في:

- غياب "سبب التأخير (Reason for Delay) "في الكثير من الرحلات، خصوصاً تلك التي وصلت في موعدها.
- غياب "وقت الوصول الفعلي (Actual Arrival Time) "في الرحلات التي تم إلغاؤها.
- ظهور قيمة "None" في عمود "بطاقة التخفيض (Railcard) "بدلاً من قيمة واضحة ومفهومة.

إجراءات المعالجة:

- تعويض القيم المفقودة في "سبب التأخير" بعبارة **"لا يوجد تأخير (No Delay) "**للرحلات التي لم تشهد تأخيراً.
- استبدال قيمة **"None"**في عمود بطاقة التخفيض بعبارة **"لا توجد بطاقة تخفيض (No Railcard) "** لتوضيح المعنى.

2- تنسيقات بيانات غير مناسبة(Incorrect Data Formats)

لوحظ أن بعض الحقول الزمنية والتاريخية مسجلة على شكل نصوص، مما يعيق إجراء العمليات الحسابية والتحليل الزمني.

إجراءات المعالجة:

- تحويل حقول التاريخ مثل **"تاريخ الرحلة "**و**"تاريخ الشراء"** إلى نوع بيانات.Date
- توحيد تنسيق الحقول الزمنية (وقت الشراء، وقت المغادرة، وقت الوصول المجدول) بصيغة **.HH:MM:SS**
- التأكد من أن **سعر التذكرة** مسجل كقيمة رقمية جاهزة للاستخدام التحليلي.

3- وجود مسافات زائدة داخل النصوص(Extra Whitespace)

قد تحتوي بعض القيم النصية أو أسماء الأعمدة على مسافات غير ضرورية، ما قد يؤدي إلى أخطاء أثناء التصنيف أو الربط.

إجراءات المعالجة:

- إزالة أي مسافات غير مرغوبة من أسماء الأعمدة.
- تنظيف القيم النصية داخل الأعمدة لضمان عدم وجود مسافات في البداية أو النهاية.

4- عدم توحيد المسميات النصية(Inconsistent Category Labels)

تم اكتشاف اختلافات في تسمية بعض الفئات التي تعبر عن نفس المعنى، مثل:

- استخدام "Staffing" بدلاً من."Staff Shortage"
- استخدام "Weather Conditions" بدلاً من."Weather"

- اختلاف طريقة كتابة بعض القيم مثل "Signal failure" بدلاً من."Signal Failure"

إجراءات المعالجة:

- توحيد جميع المسميات إلى صيغة واحدة، مثل:
  - استبدال **Staffing → Staff Shortage**
  - استبدال **Weather Conditions → Weather**
  - توحيد كتابة **Signal Failure**عبر جميع السجلات

5- السجلات المكررة(Duplicate Records)

كشفت المراجعة عن وجود عدد من الصفوف المكررة والتي قد تؤثر على دقة المقارنات والنتائج.

إجراءات المعالجة:

- التحقق من التكرارات بناءً على تطابق بيانات الرحلات.
- حذف السجلات المتكررة مع الاحتفاظ بنسخة واحدة فقط من كل حالة.

ملخص ما بعد تنظيف البيانات

بعد تنفيذ جميع خطوات التنظيف السابقة، أصبحت البيانات:

- خالية من القيم المفقودة غير المبررة
- موحدة في التنسيق والصيغ
- خالية من المسافات الزائدة
- متسقة في تسميات الفئات
- خالية من التكرار

وبالتالي أصبحت جاهزة للاستخدام في بناء لوحات مؤشرات تحليلية دقيقة وموثوقة، مما مهد لمرحلة تحليل أكثر احترافية في هذا المشروع.

Python

1) استيراد المكتبات:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
```

تم استيراد مكتبات إدارة والتعامل مع البيانات (pandas) ، العمليات الحسابية (numpy) ، والرسم البياني (matplotlib) و(seaborn) و (ploty express) استعداداً للمعالجة والتحليل البصري.

2) قراءة الملف:

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

تم تحميل ملف البيانات الخام بصيغة CSV إلى إطار بيانات (DataFrame) يسمي df.

3) نظافة البيانات:

- **البيانات المفقودة:**

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
```

أظهر فحص البيانات وجود نقص في بعض الحقول الأساسية، وهو ما قد ينعكس سلباً على دقة النتائج. تمثل هذا النقص في:

- غياب "سبب التأخير (Reason for Delay) "في الكثير من الرحلات، خصوصاً تلك التي وصلت في موعدها.
- غياب "وقت الوصول الفعلي (Actual Arrival Time) "في الرحلات التي تم إلغاؤها.
- ظهور قيمة "None" في عمود "بطاقة التخفيض (Railcard) "بدلاً من قيمة واضحة ومفهومة.

إجراءات المعالجة:

- تعويض القيم المفقودة في "سبب التأخير" بعبارة **"لا يوجد تأخير (No Delay) "**للرحلات التي لم تشهد تأخيراً.
- استبدال قيمة **"None"**في عمود بطاقة التخفيض بعبارة **"لا توجد بطاقة تخفيض (No Railcard) "**لتوضيح المعنى.

```
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()
```

تم فحص البيانات مجددا للتأكد من انه لا توجد قيم مفقودة:

```
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
```

- **مراجعة أنواع البيانات:**

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
```

تم تحويل أعمدة "تاريخ الرحلة" و"تاريخ الشراء" إلى صيغة تاريخية(Datetime) ، لتسهيل العمليات الزمنية مثل الفرز والتحليل عبر الزمن.

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()
```

- **توحيد القيم:**

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
```

تم توحيد القيم المتعددة التي تعبر عن نفس المعنى، مثل استبدال "Staffing" بـ"Staff Shortage"، و "Weather Conditions"بـ"Weather"، لضمان الاتساق في التحليل.

- **المسافات الزائدة:**

إزالة المسافات الزائدة من أسماء الأعمدة والقيم النصية

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())
```

- **مراجعة البيانات بعد المعالجة:**

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

تم التحقق مجدداً من خلو الأعمدة من القيم المفقودة، ومن أن جميع الأعمدة تحمل النوع الصحيح للبيانات.

- **حفظ نسخة من البيانات:**

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

تم حفظ نسخة جديدة من البيانات المنظفة تحت اسم railway_cleaned للعمل عليها في الخطوات التالية دون الحاجة إلى إعادة التنظيف.

KPIS

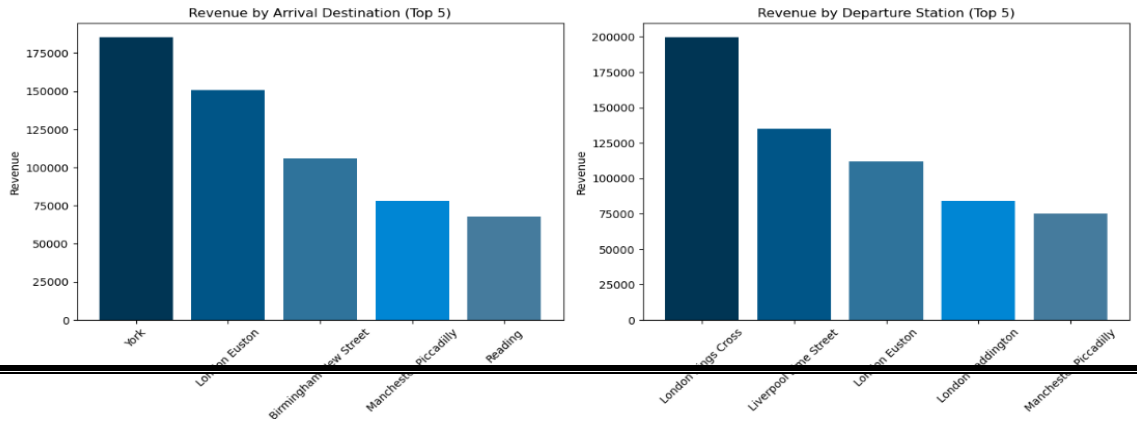
```
## Total Revenue

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

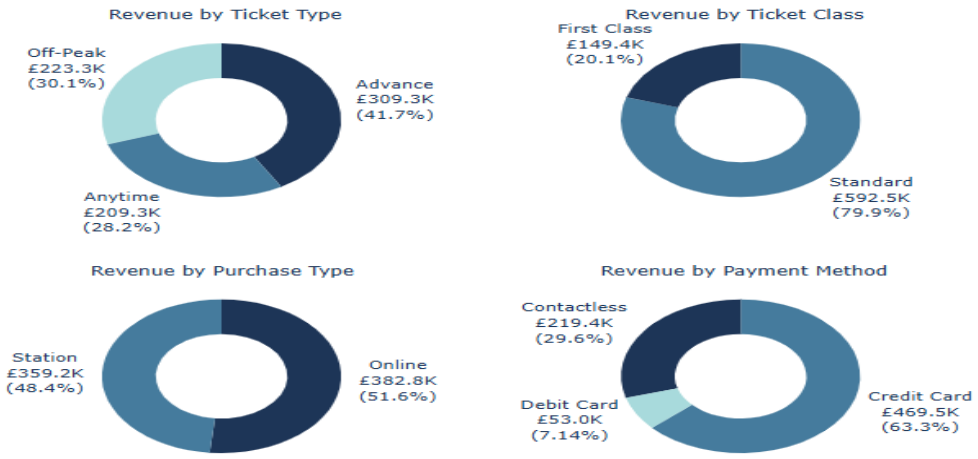
```
colors = ['#003554', '#005587', '#2F739B', '#0086D4', '#457b9d']
rev_arrival = df.groupby("Arrival Destination")["Price"].sum().sort_values(ascending=False).head(5)
plt.figure(figsize=(14,10))
plt.subplot(2,2,1)
plt.bar(rev_arrival.index, rev_arrival.values, color=colors[:5])
plt.xticks(rotation=45)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



2) Revenue By Ticket Type & Ticket Class & Payment Method & Purchase Type

```
from plotly.subplots import make_subplots
import plotly.graph_objects as go
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',
```

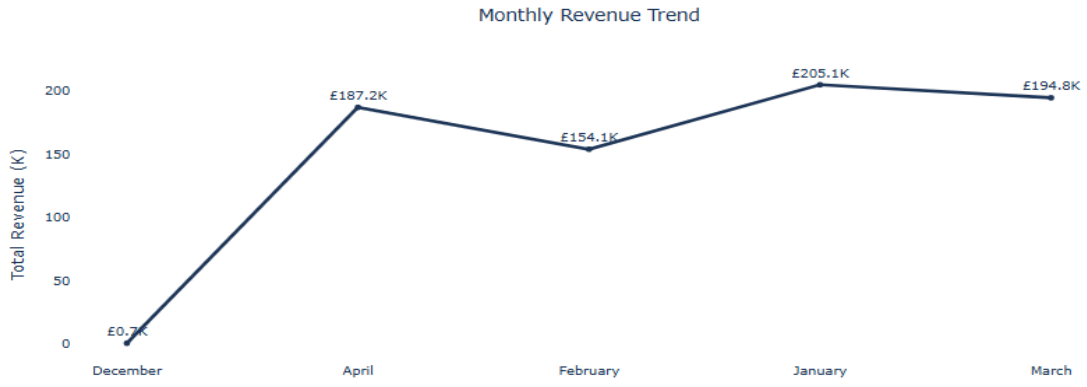
```
    showlegend=False,
    title=dict(
        x=0.5,
        y=0.95 ),
    margin=dict(t=100) )
for i, annotation in enumerate(fig['layout']['annotations']):
    annotation['y'] += 0.05
fig.show()
```



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',
```

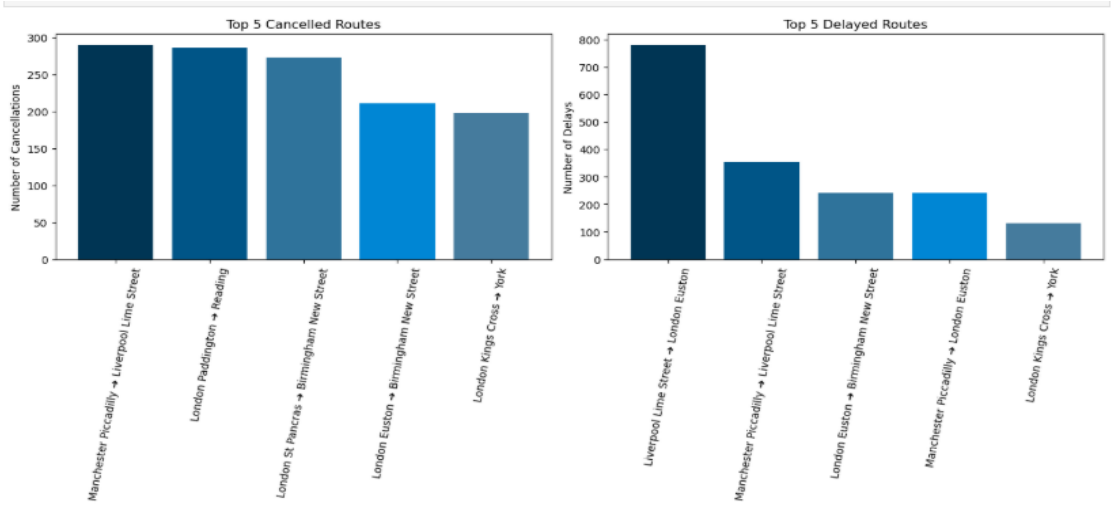
```
    title=' Monthly Revenue Trend')
fig.update_traces(
    line_color='#1d3557',
    line_width=3.5,
    texttemplate='£%{text:.1f}K',
    textposition='top center')
fig.update_layout(
        title_x=0.5,
    yaxis_title='Total Revenue (K)',
    xaxis_title=None,
    font=dict(size=13, color='#1d3557'),
    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']
```

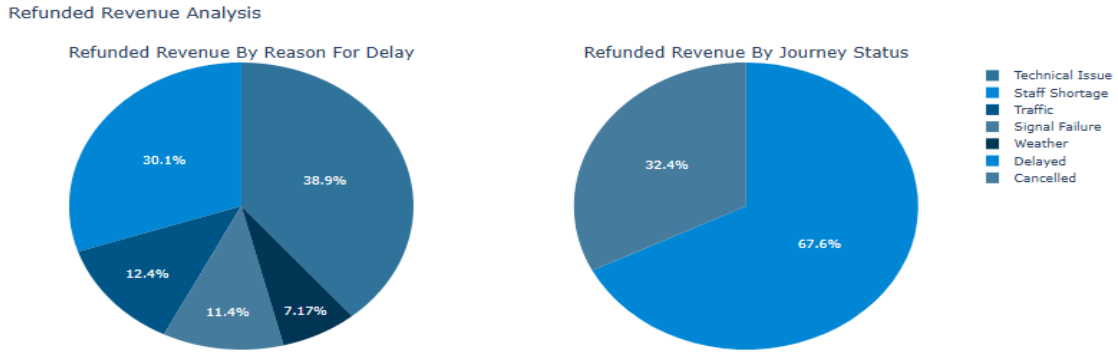
```
plt.figure(figsize=(14,10))

# 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()
```



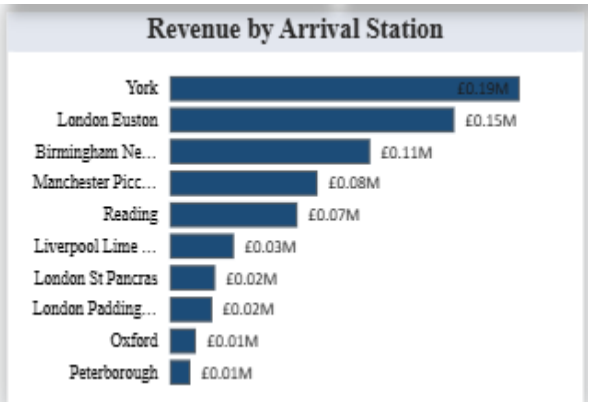
4) Refunded Revenue by Reason for Delay & Journey Status

```
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()
```



Revenue

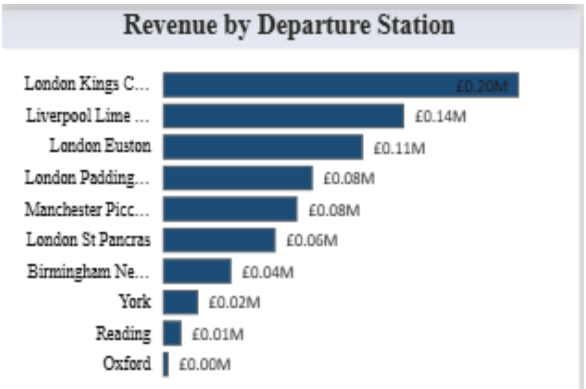


1) Revenue by Arrival Station :

أعلى الإيرادات على مستوى محطة الوصول جاءت من :

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

- Yorkمحطة سياحية وتاريخية مهمة ← رحلات كثيرة من لندن.
- London Eustonهي بوابة الشمال ومركز رئيسي للقطارات السريعة.
- Birminghamو Manchester مدن عمل وسفر يومي.

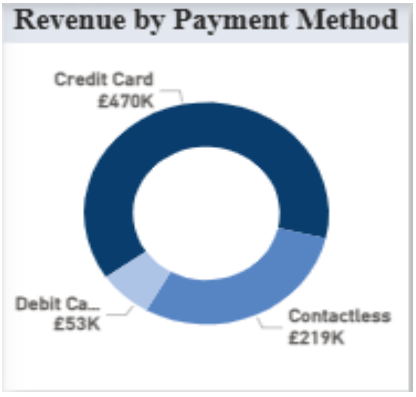


2) Revenue by Departure Station :

أعلى الإيرادات من محطات الانطلاق:

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

- Londonمحاور رئيسية للسفر بين المدن.
- Kings Crossبالذات تربط لندن بالشمال(Edinburgh ، York).
- Liverpoolو Manchester لديهم حركة عمل/جامعة كبيرة.



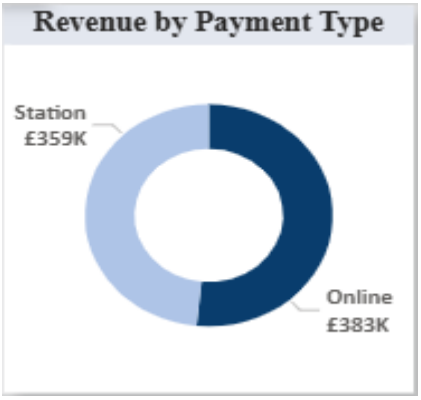
3) Revenue by Payment Method :

- Credit Card = £470K
- Contactless = £219K
- Debit Card = £53K
- البريطانيون يعتمدون بشكل أساسي على **Credit Cards**في السفر لأنها تقدم نقاط مكافآت.
- Contactlessشائع جداً داخل محطات لندن واستخدامه ينمو.

- Debitيستخدم أقل لأنه لا يقدم.Benefits

---

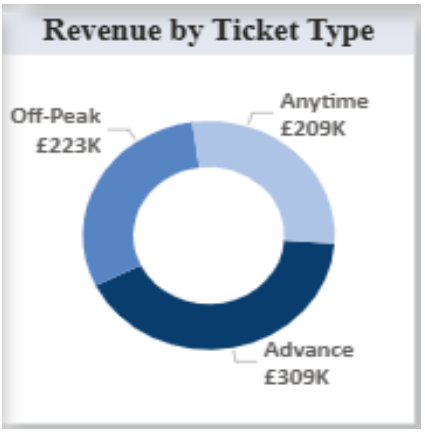
4) Revenue by Payment Type (Station vs Online)



- Station = £359K
- Online = £383K
- عدد كبير يعتمد على شراء التذاكر من الموبايل أو المواقع الإلكترونية.
- ولكن مازال جزء كبير يشتري من المحطات بسبب:
  - رحلات اللحظة الأخيرة
  - تغيير الخطط
  - عدم ثقة البعض في التطبيقات

---

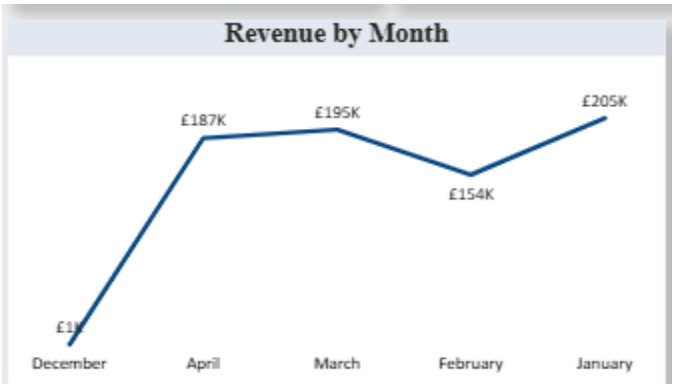
5) Revenue by Ticket Type :



- Anytime = £209K
- Off-Peak = £223K
- Advance = £233K
- Advanceهو الأعلى لأن سعره أرخص فيشتريه عدد كبير مسبقًا.
- Off-Peakيأتي بعده لأن الناس تتجنب أسعار الذروة.
- Anytimeمرتفع أيضاً لأنه يسمح بالسفر في أي وقت، لذا سعره عالي.

---

6) Revenue by Month :

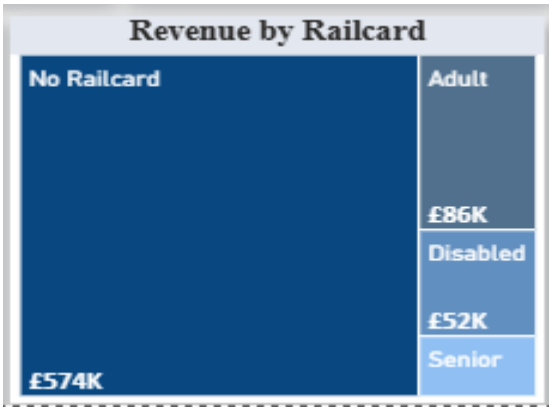


- December = £118K
- April = £187K
- March = £195K
- February = £154K

- January = £205K
- يناير عادة فيه سفر كبير (العودة للدراسة/العمل).
- مارس وأبريل فيهما عطلات → Easter السفر السياحي يزيد.

---

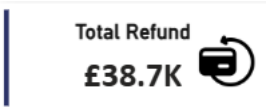
7) Revenue by Railcard (Treemap) :



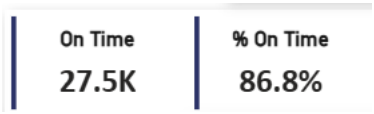
- No Railcard = £574K
- Adult Railcard = £86K
- Disabled = £52K
- Senior = £29K
- معظم الإيرادات تأتي من "No Railcard" لأن:
  1. عددهم أكبر بكثير.
  2. يدفعون سعر كامل بدون خصم.
- Adultو Senior و Disabled يستخدمون خصومات، وبالتالي الإيرادات منهم أقل.

Delay & Refund

المؤشرات الأساسية(KPIs)



1) **Total Refund: £38.7K**
إجمالي المبالغ المستردة للعملاء بسبب التأخيرات أو الإلغاءات .



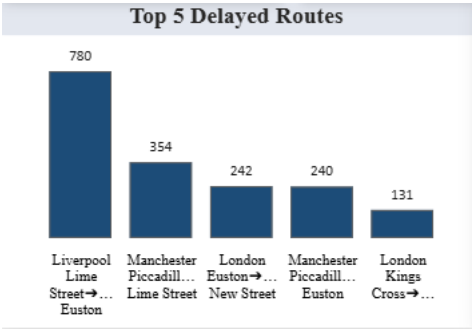
2) **On Time: 27.5K (86.8%)**
عدد الرحلات التي وصلت في الوقت المحدد ونسبتها من الإجمالي، مؤشر على كفاءة التشغيل.

3) **Cancelled: 1,880 (5.94%)**
عدد الرحلات الملغاة ونسبتها، مؤشر على مستوى الانقطاعات.

4) **Delayed: 2,292 (7.2%)**
عدد الرحلات المتأخرة ونسبتها، مؤشر على جودة الخدمة.

---

5) Top 5 Delayed Routes

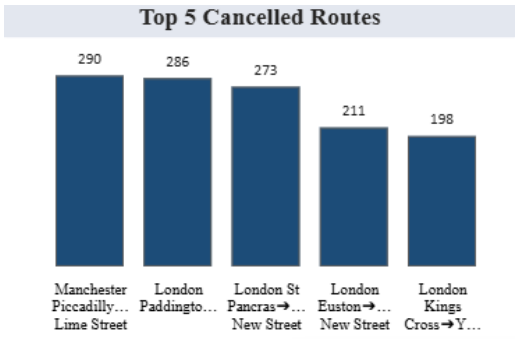


رسم بياني عمودي يوضح أكثر 5 خطوط تعرضت للتأخير:

- **Liverpool Lime Street → Euston** في المركز الأول (780 تأخير).
- يليه خطوط بين Manchester و London.
- **الاستنتاج:** هذه الخطوط عالية الحركة، مما يزيد احتمالية التأخير.

---

6) Top 5 Cancelled Routes

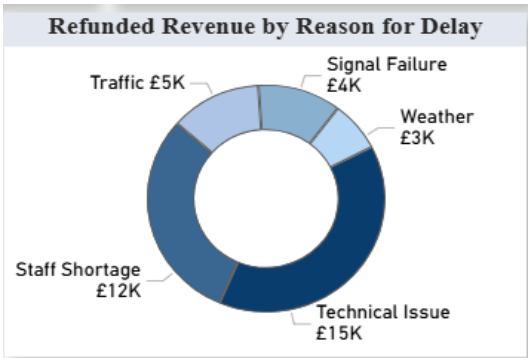


أكثر 5 خطوط تعرضت للإلغاء :

- Manchester Piccadilly → Lime Street في الصدارة.
- London Paddington → Reading أيضًا ضمن القائمة.
- **الاستنتاج:** الإلغاءات تتركز في خطوط رئيسية، ربما بسبب مشاكل تشغيلية أو ازدحام.

---

7) Refunded Revenue by Journey Status

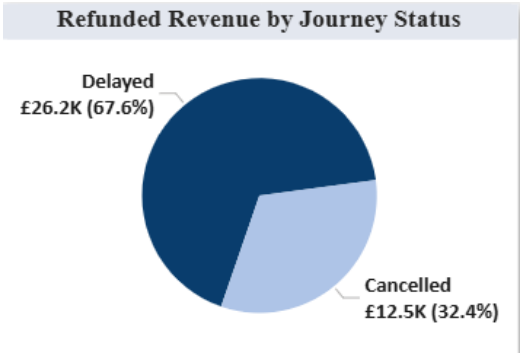


مخطط دائري يوضح توزيع المبالغ المستردة حسب حالة الرحلة:

- **Delayed:** £26.2K (67.6%)
- **Cancelled:** £12.5K (32.4%)
- **الاستنتاج:** التأخيرات تؤثر ماليًا أكثر من الإلغاءات.

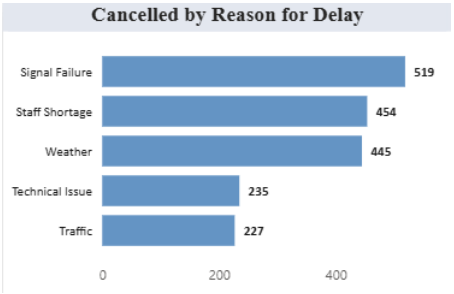
---

8) Refunded Revenue by Reason for Delay



مخطط دائري يوضح الأسباب وراء الاسترداد المالي :

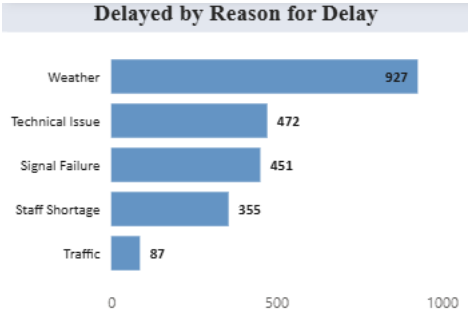
- **Staff Shortage:** £12K
- **Technical Issue:** £15K
- **Signal Failure:** £4K
- **Weather:** £3K
- **الاستنتاج:** المشاكل التقنية ونقص الموظفين هما أكبر أسباب الخسائر.



9) Cancelled by Reason for Delay

رسم بياني أفقي يوضح الأسباب وراء الإلغاءات:

- Signal Failureو Staff Shortageفي الصدارة .
- **الاستنتاج :**البنية التحتية والموارد البشرية هما التحديان الرئيسيان.



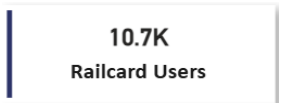
10) Delayed by Reason for Delay

رسم بياني أفقي يوضح الأسباب وراء التأخيرات :

- الطقس في المركز الأول (927 حالة).
- يليه مشاكل تقنية وفشل الإشارة.
- **الاستنتاج :**العوامل الخارجية (الطقس) لها تأثير كبير على الالتزام بالمواعيد.

Customer & Purchase Behavior

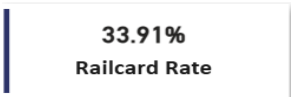
1) Railcard Users — 10.7K :



يعرض عدد العملاء الذين يستخدمو Railcard في الفترة التي تغطيها البيانات

- يعني أن ثلث المستخدمين تقريبًا يستخدمون Railcard .
- هذا منطقي لأن Railcard يعطي **خصم 33 %**على معظم تذاكر القطار، وبالتالي الكثير من الركاب يفضلون شراء بطاقة Railcard للاستفادة من التخفيض.

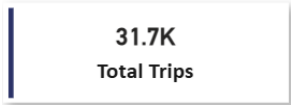
2) Railcard Rate — 33.91% :



نسبة مستخدمي Railcard من إجمالي المسافرين.

- حوالي **ثلث الركاب** يستخدمونRailcard ، والباقي يدفعون كامل السعر.
- هذا يتماشى مع الواقع في UK حيث حوالي 7 مليون شخص يحملونRailcard ، لكن ليس كل المسافرين مؤهلين (الطلاب/كبار السن/العائلات).

3) Total Trips — 31.7K :

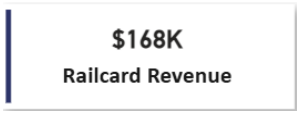


إجمالي عدد الرحلات التي قام بها المستخدمون.

- حجم استخدام كبير، يشير إلى أن العينة تمثل فترة سفر نشطة مثل شهور العمل أو مواسم الأعياد.

---

4) Railcard Revenue — $168K :

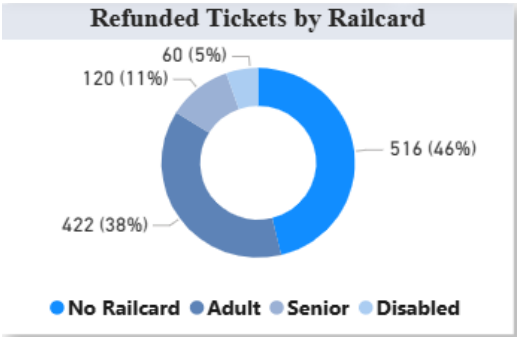


إجمالي الإيرادات من التذاكر التي استخدم أصحاب Railcard فيها خصماً.

- رغم أن حاملي Railcard يحصلون على تخفيض، إلا أن السفر المتكرر يجعل الإيرادات الإجمالية عالية.
- أصحاب Railcard غالبًا يسافرون **مرات أكثر** من غيرهم لأن السعر أقل.

---

5) Refunded Tickets by Railcard (Donut Chart) :



عدد التذاكر التي تم استردادها (Refunded) حسب نوع Railcard :

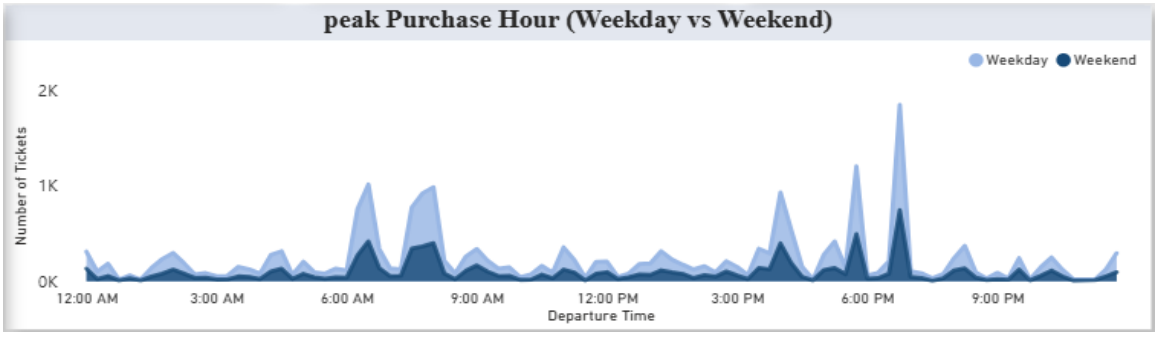
- No Railcard
- Adult
- Senior
- Disabled

- "No Railcard"هو الأكثر في طلبات الاسترداد.
- الأسباب:

1. **هذه الشريحة هي الأكبر من الأساس**، لذا طبيعي تكون استرداداتها أعلى.
2. مستخدمو Railcard غالبًا يشترون تذاكر أرخص ← أقل خسارة ← أقل طلبات رد أموال.
3. بعض أنواع Railcard مرتبطة بتذاكر "غير قابلة للاسترداد."

---

6) Peak Purchase Hour (Weekday vs Weekend) :



توقيت شراء التذاكر خلال اليوم، مقارنة بين:

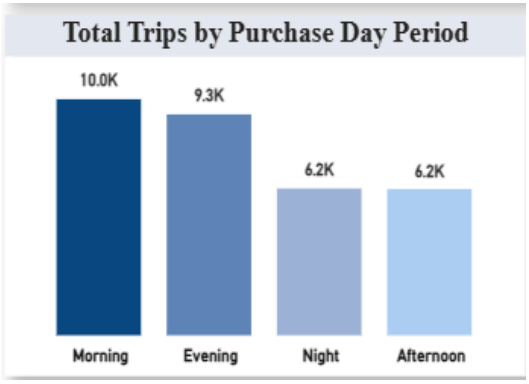
- أيام الأسبوع
- نهاية الأسبوع.

- في أيام الأسبوع:
  - الذروة بين **6–9صباحاً** commuters ← موظفين/طلبة
  - ذروة ثانية **5–7مساءً** ← رحلة العودة من العمل
- في نهاية الأسبوع:
  - شراء التذاكر يكون منتشر بشكل أكبر خلال اليوم، لأن الناس تسافر للترفيه، ليس للعمل.

---

7) Total Trips by Purchase Day Period :

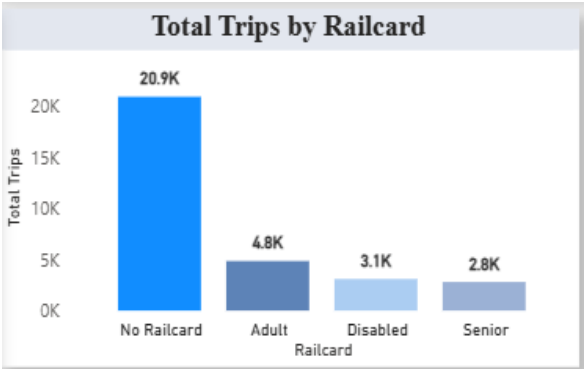
Afternoon - Night – Evening – Morning



- أعلى عدد رحلات يتم شراؤها في:
  1. Morning
  2. Evening
- وأقلها في Night
- الصباح والمساء يمثلان ساعات الذروة للقطارات.
- فترة Night أقل لأن القطارات تقل بعد الساعة 10 مساءً.
- Afternoonمتوسط بسبب الرحلات الترفيهية.

---

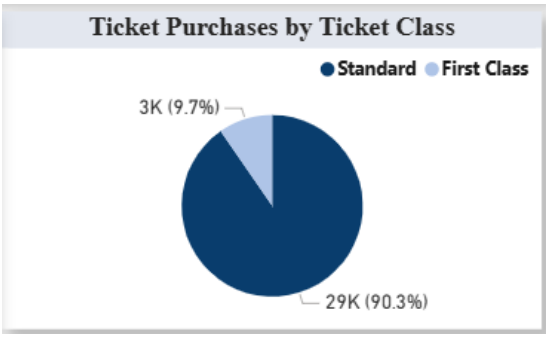
8) Total Trips by Railcard :



توزيع عدد الرحلات حسب نوع Railcard.

- أعلى شريحة **No Railcard — 20.9K رحلة**
- ثم Adult
- ثم Disabled
- ثم Senior
- أغلب المسافرين لا يستخدمون Railcard لأن:
  1. جزء كبير من الركاب هم "موظفون" لديهم اشتراكات شهرية أو سنوية لا تحتاج Railcard.
  2. Railcard له فئات عمرية وشروط — ليس متاح للجميع.
  3. بعض المسافرين لا يسافرون بما يكفي لشراء Railcard.

Ticket Purchases by Ticket Class (9



تذاكر Standard = 90%

تذاكر First Class = 10%

- في UK أغلب الرحلات تتم في Standard لأن:
  - أرخص
  - توفر مقاعد كثيرة
  - First Class عادة للرحلات الطويلة أو رجال الأعمال
- نسبة First Class 10 % نسبة واقعية جداً في بيانات القطارات البريطانية.

القراءة العامة للداشبورد

- الأداء العام جيد نسبيًا (86.8% في الموعد)، لكن التأخيرات والإلغاءات لها أثر مالي كبير.
- الأسباب الرئيسية للتأخير والإلغاء: الطقس، مشاكل تقنية، نقص الموظفين، وفشل الإشارة.
- الخطوط الأكثر تأثرًا هي بين المدن الكبرى، مما يشير إلى الحاجة لتحسين البنية التحتية في هذه المسارات.

توصيات عملية لتحسين الأداء

1. **تقليل التأخيرات**

- **الطقس (Weather)** هو السبب الأكبر للتأخير (927 حالة):
  - الاستثمار في أنظمة تنبؤ الطقس وتخطيط جداول مرنة.
  - تجهيز القطارات والبنية التحتية لمواجهة الظروف الجوية القاسية.
- **المشاكل التقنية:(Technical Issue)**
  - تعزيز الصيانة الوقائية للقطارات والإشارات.
  - إنشاء فرق دعم فني سريع للتعامل مع الأعطال أثناء التشغيل.
- **فشل الإشارة:(Signal Failure)**
  - تحديث أنظمة الإشارات القديمة.
  - تطبيق أنظمة مراقبة ذكية للكشف المبكر عن الأعطال.

---

2. **تقليل الإلغاءات**

- **نقص الموظفين:(Staff Shortage)**
  - تحسين سياسات التوظيف وجدولة الورديات.
  - تدريب موظفين متعددين المهام لتغطية النقص المفاجئ.

- **الإلغاءات بسبب الأعطال**:
  - وضع خطط بديلة (قطارات احتياطية أو تحويل المسارات).
  - تحسين التواصل مع الركاب لتقليل الأثر السلبي على رضا العملاء.

---

3. تحسين تجربة العملاء

- **زيادة الشفافية**:
  - إرسال إشعارات فورية عن التأخيرات والإلغاءات عبر التطبيقات.
  - توفير خيارات استرداد سهلة وسريعة.
- **برامج تعويض مرنة**:
  - تقديم خصومات أو نقاط مكافأة للعملاء المتأثرين بالتأخيرات المتكررة.

---

4. تحسين الإيرادات وتقليل الخسائر

- **تحليل المسارات عالية التأخير والإلغاء** :
  - إعادة جدولة الرحلات في أوقات أقل ازدحامًا.
  - تحسين البنية التحتية في الخطوط الأكثر تأثرًا) مثل Liverpool – Euston و Manchester – Lime Street).

---

5. استخدام التحليلات التنبؤية

- تطبيق نماذج **تنبؤ بالتأخيرات** بناءً على البيانات التاريخية (الطقس، الأعطال، أوقات الذروة).
- استخدام الذكاء الاصطناعي لتحديد المسارات الأكثر عرضة للمشاكل قبل حدوثها.