

UK TRAIN RIDES

1- مقدمة المشروع

في عالمنا الحديث الذي تحركه البيانات، أصبحت المعلومات هي البوصلة التي توجه المؤسسات نحو اتخاذ قرارات أكثر ذكاءً وتحسين الأداء. يركز هذا المشروع على التحليل التشغيلي لشبكة سكك حديدية حقيقية، من خلال دراسة أكثر من واحد وثلاثون ألف معامله فعليه.

الهدف هو اكتشاف أنماط الحجز، تقييم أداء الرحلات، الكشف عن التحديات التشغيلية مثل التأخيرات وطلبات الاسترداد، وفهم سلوكيات الدفع والشراء باستخدام أدوات تحليل قوية مثل

SQL و Python و Power BI

مما ساهم في بناء رؤية شاملة ومتكلمة حول أداء شبكة السكك الحديدية وفرص تحسينها

خطوات العمل 2.

التحليل باستخدام SQL

تم تطوير استعلامات منظمة لاستكشاف المؤشرات الرئيسية

General Trip Analysis 1 التحليل العام للرحلات

- إجمالي عدد المعاملات 1.1 Total Transactions

The screenshot shows a SQL query results window. The query is:

```
-- إجمالي عدد المعاملات 1.1 - Total Transactions
SELECT COUNT(*) AS total_transactions FROM railway;
```

The results table has one row with the column 'total_transactions' containing the value '31653'. The window also includes tabs for 'Results' and 'Messages'.

total_transactions
1 31653

- إجمالي الإيرادات 1.2

The screenshot shows a SQL query results window. The query is:

```
-- 1.2 - إجمالي الإيرادات - Total Revenue
SELECT SUM(price) AS total_revenue FROM railway;
```

The results table has one row with the following data:

	total_revenue
1	741921

- Top Departure Stations 1.3

```
-- 1.3 - أكثر محطات المغادرة تكراراً
SELECT departure_station, COUNT(*) AS trips
FROM railway
GROUP BY departure_station
ORDER BY trips DESC;
```

121 %

	departure_station	trips
1	Manchester Piccadilly	5650
2	London Euston	4954
3	Liverpool Lime Street	4561
4	London Paddington	4500
5	London Kings Cross	4229
6	London St Pancras	3891
7	Birmingham New Street	2136
8	York	927
9	Reading	594
10	Oxford	144
11	Edinburgh Waverley	51
12	Bristol Temple Meads	16

Query executed successfully.

- Top Arrival Destinations 1.4

```
-- 1.4 - أكثر محطات الوصول تكراراً
SELECT arrival_destination, COUNT(*) AS trips
FROM railway
GROUP BY arrival_destination
ORDER BY trips DESC;
```

121 %

Results Messages

	arrival_destination	trips
1	Birmingham New Street	7742
2	Liverpool Lime Street	5022
3	York	4019
4	Manchester Piccadilly	3968
5	Reading	3920
6	London Euston	1567
7	London St Pancras	749
8	Oxford	623
9	London Paddington	351
10	Leicester	337
11	Sheffield	272
12	Durham	258
13	Leeds	255
14	Peterborough	242
15	Swindon	228
16	Tamworth	227
17	Nuneaton	219

Query executed successfully.

- أكثر الرحلات تكراراً بين المحطات 1.5

```
--1.5 - أكثر الرحلات تكراراً بين المحطات
select departure_station,arrival_destination,
count(*) as [Total transactions] from railway
group by departure_station,arrival_destination
order by count(*) desc
```

100 %

	departure_station	arrival_destination	Total transactions
1	Manchester Piccadilly	Liverpool Lime Street	4628
2	London Euston	Birmingham New Street	4209
3	London Kings Cross	York	3922
4	London Paddington	Reading	3873
5	London St Pancras	Birmingham New Street	3471
6	Liverpool Lime Street	Manchester Piccadilly	3002
7	Liverpool Lime Street	London Euston	1097
8	London Euston	Manchester Piccadilly	712
9	Birmingham New Street	London St Pancras	702
10	London Paddington	Oxford	485
11	Manchester Piccadilly	London Euston	345
12	London St Pancras	Leicester	337
13	York	Durham	258
14	York	Peterborough	242
15	Reading	Swindon	228
16	Birmingham New Street	Tamworth	227
17	Birmingham New Street	Manchester Piccadilly	224
18	Birmingham New Street	Nuneaton	219
19	York	Doncaster	211
20	Liverpool Lime Street	Crewe	193
21	Birmingham New Street	Stafford	190
22	Birmingham New Street	Liverpool Lime Street	175
23	Manchester Piccadilly	Sheffield	171

Query executed successfully.

REPAIRR-PC\SQLEXPRESS (16.0...)

- Unique Routes Count 1.6

The screenshot shows a SQL query window in Microsoft SQL Server Management Studio (SSMS). The query is designed to find the count of unique routes from a railway database. It uses a subquery to select distinct combinations of departure and arrival stations, and then counts the number of rows in that result set.

```
-- 1.6 -- عدد المسارات بدون تكرار
SELECT COUNT(*) AS unique_route_count
FROM (
    SELECT DISTINCT
        Departure_Station,
        Arrival_Destination
    FROM railway
) AS distinct_routes;
```

The results pane displays a single row with the value 65, indicating there are 65 unique routes in the database.

unique_route_count
65

At the bottom of the screen, a status bar shows the message "Query executed successfully." and other system information.

-- عدد الرحلات التي وصلت في الموعد والملغاة 1.7 Cancelled Trips

```
-- 1.7 - عدد الرحلات التي وصلت في الموعد والملغاة
SELECT
    SUM(CASE WHEN Journey_Status = 'On Time' THEN 1 ELSE 0 END)      AS on_time_trips,
    SUM(CASE WHEN Journey_Status = 'Cancelled' THEN 1 ELSE 0 END)     AS cancelled_trips
FROM railway;
```

121 %

Results Messages

	on_time_trips	cancelled_trips
1	27481	1880

Query executed successfully.

Abdalla (16.0 RTN)

- Average Ticket Price 1.8

The screenshot shows a SQL query being run in a database environment. The query is:

```
-- 1.8 - متوسط سعر التذكرة - Average Ticket Price
SELECT
    ROUND(AVG(Price), 2) AS avg_ticket_price
FROM railway;
```

The results pane displays a single row with the column name "avg_ticket_price" and the value "23". Below the results, a message bar indicates "Query executed successfully.".

1.9--الإيرادات حسب محطة الانطلاق

```
-- 1.9-- الإيرادات حسب محطة الانطلاق - Revenue by Departure Station
SELECT
    Departure_Station AS station,
    ROUND(SUM(Price), 2) AS total_revenue
FROM railway
GROUP BY Departure_Station
ORDER BY total_revenue DESC;
```

100 %

	station	total_revenue
1	London Kings Cross	199650
2	Liverpool Lime Street	135274
3	London Euston	112045
4	London Paddington	83842
5	Manchester Piccadilly	75314
6	London St Pancras	62957
7	Birmingham New Street	38116
8	York	19546
9	Reading	10127
10	Oxford	2859
11	Edinburgh Waverley	2093
12	Bristol Temple Meads	98

- Revenue by Arrival Station 1.10 -- الإيرادات حسب محطة الوصول

```
-- 1.10 -- الإيرادات حسب محطة الوصول - Revenue by Arrival Station
SELECT
    Arrival_Destination AS station,
    ROUND(SUM(Price), 2) AS total_revenue
FROM railway
GROUP BY Arrival_Destination
ORDER BY total_revenue DESC;
```

Results

	station	total_revenue
1	York	185403
2	London Euston	150753
3	Birmingham New Street	106050
4	Manchester Piccadilly	78279
5	Reading	67580
6	Liverpool Lime Street	33756
7	London St Pancras	23852
8	London Paddington	22265
9	Oxford	13768
10	Peterborough	10764
11	Leicester	7354
12	Edinburgh Waverley	6488
13	Edinburgh	4514
14	London Kings Cross	4072
15	Swindon	3548
16	Bristol Temple Meads	2859
17	Wolverhampton	2794
18	Leeds	2714
19	Nottingham	2208
20	Nuneaton	2184
21	Durham	1983
22	Tamworth	1726

| Timing & Delay Analysis

2. التحليل الزمني والتأخيرات

- Delayed Trips Count 2.1 -- عدد الرحلات المتأخرة

The screenshot shows a SQL query execution interface. The top bar displays the title "Timing & Delay Analysis" and the section "2. التحليل الزمني والتأخيرات". The main area contains the following SQL code:

```
-- 2.1 - عدد الرحلات المتأخرة
SELECT COUNT(*) AS delayed_trips
FROM railway
WHERE journey_status = 'Delayed';
```

The results pane shows a single row with the column name "delayed_trips" and the value "2292". A message at the bottom of the interface states "Query executed successfully." and includes the user "Abdalla (16.0 RTM)".

- Average Delay (Minutes) 2.2

```
-- متوسط مدة التأخير بالدقائق 2.2 - Average Delay (Minutes)
SELECT
    AVG(DATEDIFF(MINUTE, arrival_time, actual_arrival_time)) AS avg_delay_minutes
FROM railway
WHERE journey_status = 'Delayed' AND actual_arrival_time IS NOT NULL;
```

121 %

avg_delay_minutes
42

Results Messages

Query executed successfully.

- Reasons for Delay 2.3 -- أسباب التأخير

```
-- 2.3 - أسباب التأخير
SELECT reason_for_delay, COUNT(*) AS delay_count
FROM railway
WHERE journey_status = 'Delayed'
GROUP BY reason_for_delay
ORDER BY delay_count DESC;
```

121 %

	reason_for_delay	delay_count
1	Weather	758
2	Technical Issue	472
3	Signal Failure	451
4	Staff Shortage	183
5	Staffing	172
6	Weather Conditions	169
7	Traffic	87

Query executed successfully.

- Top 5 Delaying Departure Stations -- 2.4

```
-- أكثر 5 محطات مغادرة تسبب تأخير -- 2.4 - Top 5 Delaying Departure Stations
SELECT TOP 5
    Departure_Station,
    COUNT(*) AS delayed_trips
FROM railway
WHERE Journey_Status = 'Delayed'
GROUP BY Departure_Station
ORDER BY delayed_trips DESC;
```

121 %

	Departure_Station	delayed_trips
1	Liverpool Lime Street	900
2	Manchester Piccadilly	672
3	London Euston	259
4	Birmingham New Street	140
5	London Kings Cross	131

Query executed successfully.

2.5 -- معدل الرحلات المتأخرة (%) - Delay Rate (%)

```
-- 2.5 - Delay Rate (%)  
SELECT  
    ROUND(  
        100.0 * COUNT(CASE WHEN Journey_Status = 'Delayed' THEN 1 END) / COUNT(*),  
        2  
    ) AS delay_rate_percentage  
FROM railway;
```

The screenshot shows a SQL query execution interface. The query calculates the delay rate percentage from a 'railway' table. The results are displayed in a table with one row, showing a value of 7.24000000000 for the 'delay_rate_percentage' column. The interface includes tabs for 'Results' and 'Messages', and a status bar at the bottom indicating the query was executed successfully.

delay_rate_percentage
1 7.24000000000

Query executed successfully. | Abdalla (16.0 RTM) | ABDAl

- Most Delayed Routes 2.6--أكثر المسارات تأخيرًا

```
-- 2.6 - أكثر المسارات تأخيرًا - Most Delayed Routes
SELECT
    Departure_Station,
    Arrival_Destination,
    COUNT(*) AS total_trips,
    COUNT(CASE WHEN Journey_Status = 'Delayed' THEN 1 END) AS delayed_trips,
    ROUND(100.0 * COUNT(CASE WHEN Journey_Status = 'Delayed' THEN 1 END) / COUNT(*), 2) AS delay_percentage
FROM railway
GROUP BY Departure_Station, Arrival_Destination
ORDER BY
    delay_percentage DESC;
```

121 % 4 Results Messages

	Departure_Station	Arrival_Destination	total_trips	delayed_trips	delay_percentage
1	London Euston	York	17	17	100.00000000000000
2	York	Wakefield	15	15	100.00000000000000
3	Edinburgh Waverley	London Kings Cross	51	51	100.00000000000000
4	Liverpool Lime Street	London Euston	1097	780	71.10000000000000
5	Manchester Piccadilly	London Euston	345	240	69.57000000000000
6	Liverpool Lime Street	London Paddington	27	13	48.15000000000000
7	Manchester Piccadilly	Leeds	142	64	45.07000000000000
8	Birmingham New Street	Manchester Piccadilly	224	96	42.86000000000000
9	Birmingham New Street	London Euston	125	44	35.20000000000000
10	York	Doncaster	211	27	12.80000000000000
11	Oxford	Bristol Temple Meads	144	15	10.42000000000000
12	Manchester Piccadilly	Nottingham	158	14	8.86000000000000
13	Manchester Piccadilly	Liverpool Lime Street	4628	354	7.65000000000000
14	York	Durham	258	16	6.20000000000000
15	London Euston	Birmingham New Street	4209	242	5.75000000000000
16	Liverpool Lime Street	Manchester Piccadilly	3002	107	3.56000000000000
17	London Kings Cross	York	3922	131	3.34000000000000

Query executed successfully. Abdalla (16.0 RTM) | ABDALLA\Options (63) | uk_t

2.7-----تحليل شامل لمحطات الانطلاق مقارنة بالمسارات حسب الأداء والتأخير (Departure vs Route Analysis)

-----2.7 (Departure vs Route Analysis) تحليل شامل لمحطات الانطلاق مقارنة بالمسارات حسب الأداء والتأخير

```
SELECT
    Departure_Station,
    COUNT(*) AS total_trips,
    COUNT(CASE WHEN Journey_Status = 'Delayed' THEN 1 END) AS delayed_trips,
    ROUND(100.0 * COUNT(CASE WHEN Journey_Status = 'Delayed' THEN 1 END) / COUNT(*), 2) AS delay_percentage
FROM railway
GROUP BY Departure_Station
ORDER BY total_trips DESC.
```

121 %

Departure_Station	total_trips	delayed_trips	delay_percentage
Manchester Piccadilly	5650	Click to select the whole column	
London Euston	4954	299	5.2300000000000005
Liverpool Lime Street	4561	900	19.730000000000002
London Paddington	4500	66	1.4700000000000001
London Kings Cross	4229	131	3.1000000000000003
London St Pancras	3891	0	0.0000000000000001
Birmingham New Street	2136	140	6.5500000000000005
York	927	58	6.2600000000000005
Reading	594	0	0.0000000000000001
Oxford	144	15	10.420000000000001
Edinburgh Waverley	51	51	100.00000000000001
Bristol Temple Meads	16	0	0.0000000000000001

Query executed successfully. | Abdalla (16.0 RTM) | ABDALLA\Options (6)

Ln 119 Col 1 Ch 1 INS

-----2.8-----تحليل شامل لمحطات الوصول مقارنة بالمسارات حسب الأداء والتأخير(Arrival vs Route Analysis)

-2.8 (Arrival vs Route Analysis)

```
-----2.8 ( Arrival vs Route Analysis )
SELECT
    Arrival_Destination,
    COUNT(*) AS total_trips,
    COUNT(CASE WHEN Journey_Status = 'Delayed' THEN 1 END) AS delayed_trips,
    ROUND(100.0 * COUNT(CASE WHEN Journey_Status = 'Delayed' THEN 1 END) / COUNT(*), 2) AS delay_percentage
FROM railway
GROUP BY Arrival_Destination
ORDER BY delay_percentage DESC;
```

121 % ▶

Results Messages

	Arrival_Destination	total_trips	delayed_trips	delay_percentage
1	Wakefield	15	15	100.0000000000000
2	London Euston	1567	1064	67.9000000000000
3	London Kings Cross	84	51	60.7100000000000
4	Leeds	255	64	25.1000000000000
5	Doncaster	211	27	12.8000000000000
6	Bristol Temple Meads	144	15	10.4200000000000
7	Nottingham	158	14	8.8600000000000
8	Liverpool Lime Street	5022	354	7.0500000000000
9	Durham	258	16	6.2000000000000
10	Manchester Piccadilly	3968	203	5.1200000000000
11	London Paddington	351	13	3.7000000000000
12	York	4019	148	3.6800000000000
13	Birmingham New Street	7742	242	3.1300000000000
14	Reading	3920	66	1.6800000000000
15	Coventry	65	0	0.0000000000000
16	Leicester	337	0	0.0000000000000
17	Warrington	15	0	0.0000000000000

Query executed successfully.

3. تحليل الدفع والاسترداد Payment & Refund Analysis

- تحليل طرق الدفع 3.1 Payment Method Analysis

The screenshot shows a SQL query being run in a database environment. The query is:

```
-- تحليل طرق الدفع 3.1 - Payment Method Analysis
SELECT payment_method, COUNT(*) AS transactions, SUM(price) AS revenue
FROM railway
GROUP BY payment_method
ORDER BY revenue DESC;
```

The results are displayed in a table:

	payment_method	transactions	revenue
1	Credit Card	19136	469511
2	Contactless	10834	219444
3	Debit Card	1683	52966

At the bottom of the interface, a message indicates the query was executed successfully.

- Purchase Method Analysis 3.2

Group 2 - UK Train...ALLA\Options (63)) X ----- 3.2 - تحليل طرق الشراء

```
SELECT
    Purchase_Type,
    COUNT(*) AS total_transactions,
    SUM(Price) AS total_revenue,
    COUNT(CASE WHEN Journey_Status = 'Delayed' THEN 1 END) AS delayed_trips,
    ROUND(
        100.0 * COUNT(CASE WHEN Journey_Status = 'Delayed' THEN 1 END) / COUNT(*),
        2
    ) AS delay_rate_percentage
FROM railway
GROUP BY Purchase_Type
ORDER BY total_transactions DESC;
```

121 % Results Messages

	Purchase_Type	total_transactions	total_revenue	delayed_trips	delay_rate_percentage
1	Online	18521	382754	207	1.120000000000
2	Station	13132	359167	2085	15.880000000000

Query executed successfully. Abdalla (16.0)

- Refund Request Count 3.3 -- عدد طلبات الاسترداد

```
-- 3.3 - عدد طلبات الاسترداد - Refund Request Count
SELECT refund_request, COUNT(*) AS requests
FROM railway
GROUP BY refund_request;
```

121 % < Results Messages

	refund_request	requests
1	0	30535
2	1	1118

Query executed successfully.

(%) - معدل طلبات الاسترداد (%) - Refund Rate (%) 3.4

```
-- 3.4 معدل طلبات الاسترداد (%) - Refund Rate (%)  
SELECT  
    ROUND(  
        100.0 * SUM(CASE WHEN Refund_Request = 1 THEN 1 ELSE 0 END) / COUNT(*),  
        2  
    ) AS refund_rate_percentage  
FROM railway;
```

121 % ▾

	refund_rate_percentage
1	3.53000000000

Query executed successfully. Abdalla

- Refund Count & Value 3.5 -- عدد طلبات الاسترداد وقيمتها

```
-- 3.5 - عدد طلبات الاسترداد وقيمتها
SELECT
    SUM(CASE WHEN Refund_Request = 1 THEN 1 ELSE 0 END)      AS refund_count,
    SUM(CASE WHEN Refund_Request = 1 THEN Price ELSE 0 END)   AS total_refund_value
FROM railway;
```

121 % ▾

Results Messages

	refund_count	total_refund_value
1	1118	38702

Query executed successfully. A

- Refunds - طلبات الاسترداد ونسبة الاسترداد حسب محطة الانطلاق & Rate by Departure Station

```
-- 3.6 طلبات الاسترداد ونسبة الاسترداد حسب محطة الانطلاق - Refunds & Rate by Departure Station
SELECT
    Departure_Station AS station,
    COUNT(*) AS total_transactions,
    SUM(CASE WHEN Refund_Request = 1 THEN 1 ELSE 0 END) AS refund_count,
    ROUND(
        100.0 * SUM(CASE WHEN Refund_Request = 1 THEN 1 ELSE 0 END) / COUNT(*),
        2
    ) AS refund_rate_percentage
FROM railway
GROUP BY Departure_Station
ORDER BY refund_rate_percentage DESC;
```

121 %

	station	total_transactions	refund_count	refund_rate_percentage
1	Edinburgh Waverley	51	51	100.00000000000000
2	Oxford	144	19	13.19000000000000
3	Birmingham New Street	2136	153	7.16000000000000
4	Liverpool Lime Street	4561	305	6.69000000000000
5	Manchester Piccadilly	5650	195	3.45000000000000
6	Reading	594	17	2.86000000000000
7	London Paddington	4500	101	2.24000000000000
8	London Euston	4954	104	2.10000000000000
9	London Kings Cross	4229	86	2.03000000000000
10	London St Pancras	3891	71	1.82000000000000
11	York	927	16	1.73000000000000
12	Bristol Temple Meads	16	0	0.00000000000000

Query executed successfully.

Abdalla (16.0 RTM) ABC

- طلبات الاسترداد ونسبة الاسترداد حسب محطة الوصول & Rate by Arrival Station

```
-- طلبات الاسترداد ونسبة الاسترداد حسب محطة الوصول 3.7 - Refunds & Rate by Arrival Station
SELECT
    Arrival_Destination AS station,
    COUNT(*) AS total_transactions,
    SUM(CASE WHEN Refund_Request = 1 THEN 1 ELSE 0 END) AS refund_count,
    ROUND(
        100.0 * SUM(CASE WHEN Refund_Request = 1 THEN 1 ELSE 0 END) / COUNT(*),
        2
    ) AS refund_rate_percentage
FROM railway
GROUP BY Arrival_Destination
ORDER BY refund_rate_percentage DESC;
```

121 % 4 Results Messages

	station	total_transactions	refund_count	refund_rate_percentage
1	London Kings Cross	84	51	60.71000000000000
2	London Euston	1567	283	18.06000000000000
3	Bristol Temple Meads	144	19	13.19000000000000
4	Didcot	48	5	10.42000000000000
5	London Paddington	351	23	6.55000000000000
6	Manchester Piccadilly	3968	239	6.02000000000000
7	Coventry	65	2	3.08000000000000
8	London Waterloo	68	2	2.94000000000000
9	Swindon	228	6	2.63000000000000
10	Nottingham	158	4	2.53000000000000
11	Reading	3920	92	2.35000000000000
12	Durham	258	6	2.33000000000000
13	Nuneaton	219	5	2.28000000000000
14	Peterborough	242	5	2.07000000000000
15	York	4019	82	2.04000000000000
16	Birmingham New Street	7742	150	1.94000000000000
17	Oxford	623	12	1.93000000000000

Query executed successfully. Group 2 - UK Train.sql - Abdalla.uk_train (ABDALLA) Abdalla (16.0 RTM) | ABDALLA\O

- Purchase & Payment معاً Method Combined Analysis

```
-- 3.8 - تحليل طرق الشراء والدفع معاً
-- Purchase & Payment Method Combined Analysis

SELECT
    Purchase_Type,
    Payment_Method,
    COUNT(*) AS total_transactions,
    SUM(Price) AS total_revenue,
    COUNT(CASE WHEN Journey_Status = 'Delayed' THEN 1 END) AS delayed_trips,
    ROUND(
        100.0 * COUNT(CASE WHEN Journey_Status = 'Delayed' THEN 1 END) / COUNT(*),
        2
    ) AS delay_rate_percentage
FROM railway
GROUP BY Purchase_Type, Payment_Method
ORDER BY total_transactions DESC;
```

Results

	Purchase_Type	Payment_Method	total_transactions	total_revenue	delayed_trips	delay_rate_percentage
1	Online	Credit Card	11487	243399	143	1.240000000000
2	Station	Credit Card	7649	226112	1182	15.450000000000
3	Online	Contactless	6390	118940	32	0.500000000000
4	Station	Contactless	4444	100504	354	7.970000000000
5	Station	Debit Card	1039	32551	549	52.840000000000
6	Online	Debit Card	644	20415	32	4.970000000000

Query executed successfully.

4. تحليل التذاكر والبطاقات Ticket & Railcard Analysis

4.1 -- استخدام Railcards - Railcard Usage

```
-- 4.1 استخدام Railcards - Railcard Usage
SELECT
    Railcard,
    COUNT(*) AS usage_count,
    SUM(price) AS total_paid
FROM railway
GROUP BY Railcard
ORDER BY usage_count DESC;
```

The screenshot shows a database query interface with the following details:

- Query Editor:** The top half displays the SQL query for analyzing railcard usage.
- Results Tab:** The bottom half shows the results of the query execution. The results are presented in a table with four columns: Railcard, usage_count, and total_paid. The table has 4 rows, indexed from 1 to 4.
- Message Bar:** A yellow message bar at the bottom indicates that the query was executed successfully.

	Railcard	usage_count	total_paid
1	None	20918	573697
2	Adult	4846	86330
3	Disabled	3089	52278
4	Senior	2800	29616

Query executed successfully.

- Railcard Delay Analysis 4.2

```
-- تحليل الريلكارد من حيث عدد التأخيرات 4.2
SELECT
    Railcard,
    COUNT(*) AS total_users,
    COUNT(CASE WHEN Journey_Status = 'Delayed' THEN 1 END) AS delayed_users,
    ROUND(100.0 * COUNT(CASE WHEN Journey_Status = 'Delayed' THEN 1 END) / COUNT(*), 2) AS delay_rate
FROM railway
GROUP BY Railcard
ORDER BY delay_rate DESC;
```

121 % ▶

Railcard	total_users	delayed_users	delay_rate
1 Adult	4846	665	13.72000000000000
2 None	20918	1410	6.74000000000000
3 Disabled	3089	136	4.40000000000000
4 Senior	2800	81	2.89000000000000

Query executed successfully. | Abdalla (16.0 RTM) | ABDALLA

- Revenue by Ticket Class

```
-- 4.3 -- الإيرادات حسب فئة التذكرة - Revenue by Ticket Class
SELECT
    Ticket_Class,
    COUNT(*) AS trips,
    SUM(price) AS revenue
FROM railway
GROUP BY Ticket_Class
ORDER BY revenue DESC;
```

121 %

	Results	Messages	
	Ticket_Class	trips	revenue
1	Standard	28595	592522
2	First Class	3058	149399

✓ Query executed successfully.

- Trips by Ticket Type 4.4

```
-- 4.4 - عدد الرحلات حسب نوع التذكرة - Trips by Ticket Type
SELECT
    Ticket_Type,
    COUNT(*) AS trip_count,
    SUM(price) AS total_revenue
FROM railway
GROUP BY Ticket_Type
ORDER BY total_revenue DESC;
```

121 %

	Ticket_Type	trip_count	total_revenue
1	Advance	17561	309274
2	Off-Peak	8752	223338
3	Anytime	5340	209309

Query executed successfully.

- Delay by Ticket Type

```
-- 4.5 -- تحليل التأخير حسب نوع التذكرة - Delay by Ticket Type
SELECT
    Ticket_Type,
    COUNT(*) AS total_trips,
    COUNT(CASE WHEN Journey_Status = 'Delayed' THEN 1 END) AS delayed_trips,
    ROUND(100.0 * COUNT(CASE WHEN Journey_Status = 'Delayed' THEN 1 END) / COUNT(*), 2) AS delay_percentage
FROM railway
GROUP BY Ticket_Type
ORDER BY delay_percentage DESC;
```

121 %

	Ticket_Type	total_trips	delayed_trips	delay_percentage
1	Anytime	5340	486	9.100000000000
2	Advance	17561	1211	6.900000000000
3	Off-Peak	8752	595	6.800000000000

Query executed successfully.

Abdalla (16.0 RTM) | ABDALLA\Options (

- Avg Price by Ticket Type 4.6

```
-- 4.6 - متوسط السعر حسب نوع التذكرة
SELECT
    Ticket_Type,
    ROUND(AVG(Price), 2) AS avg_price
FROM railway
GROUP BY Ticket_Type
ORDER BY avg_price DESC;
```

The screenshot shows a SQL query execution interface. At the top, there is a status bar with '121 %' and a back arrow. Below it is a toolbar with 'Results' and 'Messages' tabs, with 'Results' selected. The main area displays a table with three rows of data. At the bottom, a green success message is shown.

	Ticket_Type	avg_price
1	Anytime	39
2	Off-Peak	25
3	Advance	17

Query executed successfully.

| Advanced Date/Time التحليل الزمني المتقدم 5.

Analysis

- Trips by Day of Week الرحلات حسب أيام الأسبوع 5.1

The screenshot shows a SQL query execution interface. The query is:

```
-- الرحلات حسب أيام الأسبوع 5.1 - Trips by Day of Week
SELECT
    DATENAME(WEEKDAY, Date_of_Journey) AS journey_day,
    COUNT(*) AS trips
FROM railway
GROUP BY DATENAME(WEEKDAY, Date_of_Journey)
ORDER BY trips DESC;
```

The results table is:

journey_day	trips
Wednesday	4692
Tuesday	4607
Thursday	4580
Sunday	4580
Monday	4436
Saturday	4407
Friday	4351

At the bottom, a message says "Query executed successfully." and the status bar shows "Abdalla (16.0 RTM) ABDALL".

- Peak Purchase Hours 5.2

```
-- ساعت الذروة في الشراء - Peak Purchase Hours
SELECT DATEPART(HOUR, CAST(Time_of_Purchase AS TIME)) AS purchase_hour,
       COUNT(*) AS purchases
FROM railway
GROUP BY DATEPART(HOUR, CAST(Time_of_Purchase AS TIME))
ORDER BY purchases DESC;
```

121 %

	purchase_hour	purchases
1	17	2740
2	20	2239
3	9	2070
4	7	2046
5	8	2008
6	6	1910
7	14	1869
8	5	1566
9	15	1468
10	18	1425
11	10	1187
12	19	1160
13	3	1107
14	16	1056
15	1	1032
16	12	1025
17	0	925

Query executed successfully.

Abdalla (16.0 RTM) | ABDALL

- Monthly Seasonality(شهری) 5.3

```
-- 5.3 - تحلیل موسمی (شهری) - Monthly Seasonality
SELECT
    MONTH(Date_of_Journey) AS journey_month,
    COUNT(*) AS trips,
    SUM(price) AS revenue
FROM railway
GROUP BY MONTH(Date_of_Journey)
ORDER BY journey_month;
```

121 %

	journey_month	trips	revenue
1	1	8111	199618
2	2	7644	159374
3	3	8117	195147
4	4	7781	187782

Query executed successfully.

- Yearly Analysis 5.4 -- تحليل سنوي

```
-- 5.4 - Yearly Analysis
SELECT
    YEAR(Date_of_Journey) AS journey_year,
    COUNT(*) AS trips
FROM railway
GROUP BY YEAR(Date_of_Journey)
ORDER BY journey_year;
```

121 %

	journey_year	trips
1	2024	31653

Query executed successfully.

- Busiest Travel Days (Top 5) 5.5

```
-- 5.5 أكثر أيام ازدحامًا - Busiest Travel Days (Top 5)
SELECT TOP 5
    Date_of_Journey,
    COUNT(*) AS trip_count
FROM railway
GROUP BY Date_of_Journey
ORDER BY trip_count DESC;
```

121 %

	Date_of_Journey	trip_count
1	2024-03-09	313
2	2024-04-19	304
3	2024-03-25	304
4	2024-04-09	302
5	2024-01-18	301

Query executed successfully.

- Revenue by Month & Class

```
-- الإيرادات حسب الشهر والفئة 5.6 - Revenue by Month & Class
SELECT
    FORMAT(Date_of_Journey, 'yyyy-MM') AS month,
    Ticket_Class,
    SUM(Price) AS revenue
FROM railway
GROUP BY FORMAT(Date_of_Journey, 'yyyy-MM'), Ticket_Class
ORDER BY month;
```

121 % ▶

	month	Ticket_Class	revenue
1	2024-01	First Class	40976
2	2024-01	Standard	158642
3	2024-02	Standard	129023
4	2024-02	First Class	30351
5	2024-03	Standard	154376
6	2024-03	First Class	40771
7	2024-04	First Class	37301
8	2024-04	Standard	150481

Query executed successfully.

- Ticket Purchase Day Analysis 5.7 -- تحليل يوم شراء التذكرة

```
-- 5.7 - تحليل يوم شراء التذكرة - Ticket Purchase Day Analysis
SELECT
    DATENAME(WEEKDAY, Date_of_Purchase) AS purchase_day,
    COUNT(*) AS purchases
FROM railway
GROUP BY DATENAME(WEEKDAY, Date_of_Purchase)
ORDER BY purchases DESC;
```

121 %

Results Messages

	purchase_day	purchases
1	Sunday	4676
2	Friday	4627
3	Wednesday	4602
4	Saturday	4477
5	Monday	4455
6	Tuesday	4454
7	Thursday	4362

Query executed successfully.

٦. تحليل وقت الحجز والتأخير | Booking Lead Time & Delay

6.1--فرق الوقت بين الحجز والرحلة - Booking Lead Time

```
-- ----- ٦. تحليل وقت الحجز والتأخير | Booking Lead Time & Delay
-- ٦.١-- فرق الوقت بين الحجز والرحلة - Booking Lead Time
SELECT
    DATEDIFF(DAY, Date_of_Purchase, Date_of_Journey) AS days_in_advance,
    COUNT(*) AS bookings
FROM railway
GROUP BY DATEDIFF(DAY, Date_of_Purchase, Date_of_Journey)
ORDER BY days_in_advance;
```

21 %

days_in_advance	bookings
0	14092
1	13744
2	367
3	284
4	254
5	221
6	212
7	253
8	219
9	182
10	193
11	190
12	160
13	149
14	163
15	148
16	135

Query executed successfully.  Abdalla (16.0 RTM) ABDALLA\Options (63) uk_

6.2--تحليل وقت الحجز مقابل التأخير - Booking Time vs Delay

```
-- 6.2-- تحليل وقت الحجز مقابل التأخير - Booking Time vs Delay
SELECT
    DATEDIFF(DAY, Date_of_Purchase, Date_of_Journey) AS days_in_advance,
    COUNT(*) AS total_trips,
    COUNT(CASE WHEN Journey_Status = 'Delayed' THEN 1 END) AS delayed_trips
FROM railway
GROUP BY DATEDIFF(DAY, Date_of_Purchase, Date_of_Journey)
ORDER BY days_in_advance,
```

121 %

Results Messages

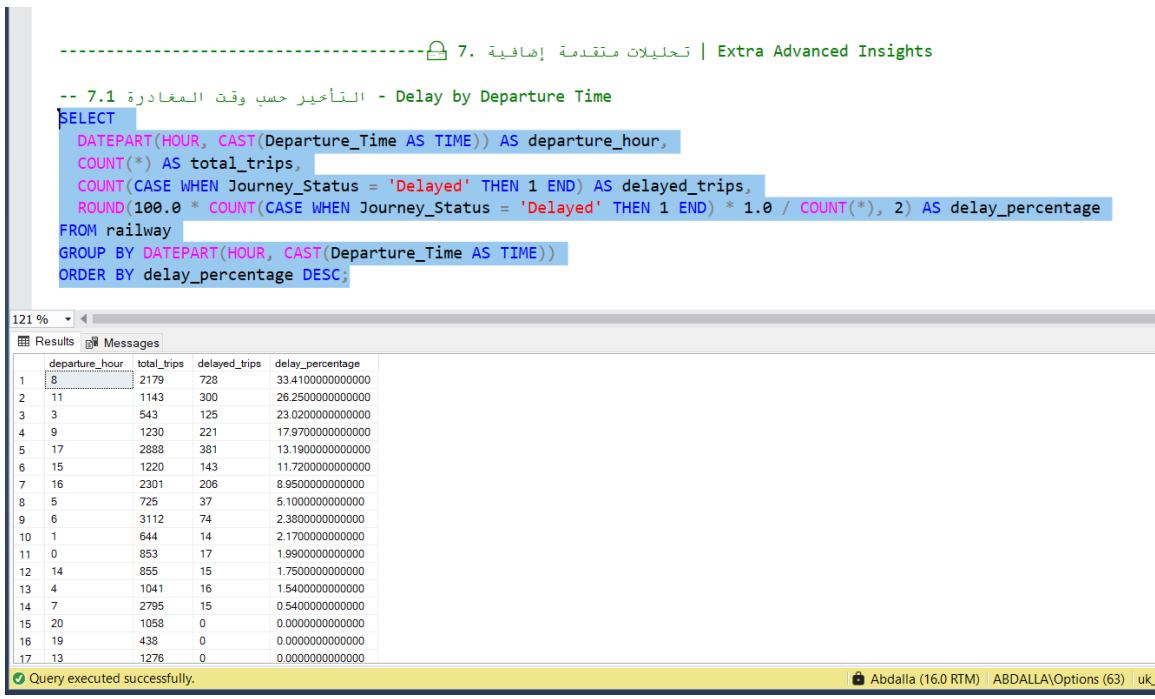
days_in_advance	total_trips	delayed_trips
0	14092	1081
1	13744	939
2	367	23
3	264	18
4	254	16
5	221	14
6	212	14
7	253	17
8	219	16
9	182	18
10	193	11
11	190	18
12	160	11
13	149	10
14	163	8
15	148	9
16	135	11

Query executed successfully.

Abdalla (16.0 RTM) ABDALLA\Options (63) uk_train 00:00:00 29 rows

تحليلات متقدمة إضافية Extra Advanced Insights | 7.

- Delay by Departure Time 7.1



The screenshot shows a SQL query being executed in a database environment. The query is designed to analyze travel delays based on departure times. It uses the DATEPART function to extract the hour from the departure time, counts the total trips, and then filters those trips where the journey status is 'Delayed'. Finally, it calculates the percentage of delayed trips for each hour and orders the results by this percentage in descending order.

```
-- 7.1 - التأخير حسب وقت المغادرة
SELECT
    DATEPART(HOUR, CAST(Departure_Time AS TIME)) AS departure_hour,
    COUNT(*) AS total_trips,
    COUNT(CASE WHEN Journey_Status = 'Delayed' THEN 1 END) AS delayed_trips,
    ROUND(100.0 * COUNT(CASE WHEN Journey_Status = 'Delayed' THEN 1 END) * 1.0 / COUNT(*), 2) AS delay_percentage
FROM railway
GROUP BY DATEPART(HOUR, CAST(Departure_Time AS TIME))
ORDER BY delay_percentage DESC;
```

departure_hour	total_trips	delayed_trips	delay_percentage
8	2179	728	33.41000000000000
11	1143	300	26.25000000000000
3	543	125	23.02000000000000
9	1230	221	17.97000000000000
17	2888	381	13.19000000000000
15	1220	143	11.72000000000000
7	2301	206	8.95000000000000
5	725	37	5.10000000000000
6	3112	74	2.38000000000000
10	644	14	2.17000000000000
11	853	17	1.99000000000000
12	855	15	1.75000000000000
13	1041	16	1.54000000000000
14	2795	15	0.54000000000000
15	1058	0	0.00000000000000
19	438	0	0.00000000000000
13	1276	0	0.00000000000000

Query executed successfully. Abdalla (16.0 RTM) ABDALLA\Options (63) uk.

- Morning vs Evening Trips 7.2

```
Group 2 - UK Train...ALLA(Options (63)) X
-- مقارنة الفترات الزمنية للرحلات 7.2 - Morning vs Evening Trips
SELECT
CASE
    WHEN DATEPART(HOUR, Departure_Time) BETWEEN 6 AND 11 THEN 'Morning'
    WHEN DATEPART(HOUR, Departure_Time) BETWEEN 12 AND 17 THEN 'Afternoon'
    WHEN DATEPART(HOUR, Departure_Time) BETWEEN 18 AND 23 THEN 'Evening'
    ELSE 'Night'
END AS time_slot,
COUNT(*) AS trips,
COUNT(CASE WHEN Journey_Status = 'Delayed' THEN 1 END) AS delayed_trips
FROM railway
GROUP BY
CASE
    WHEN DATEPART(HOUR, Departure_Time) BETWEEN 6 AND 11 THEN 'Morning'
    WHEN DATEPART(HOUR, Departure_Time) BETWEEN 12 AND 17 THEN 'Afternoon'
    WHEN DATEPART(HOUR, Departure_Time) BETWEEN 18 AND 23 THEN 'Evening'
    ELSE 'Night'
END
ORDER BY time_slot
```

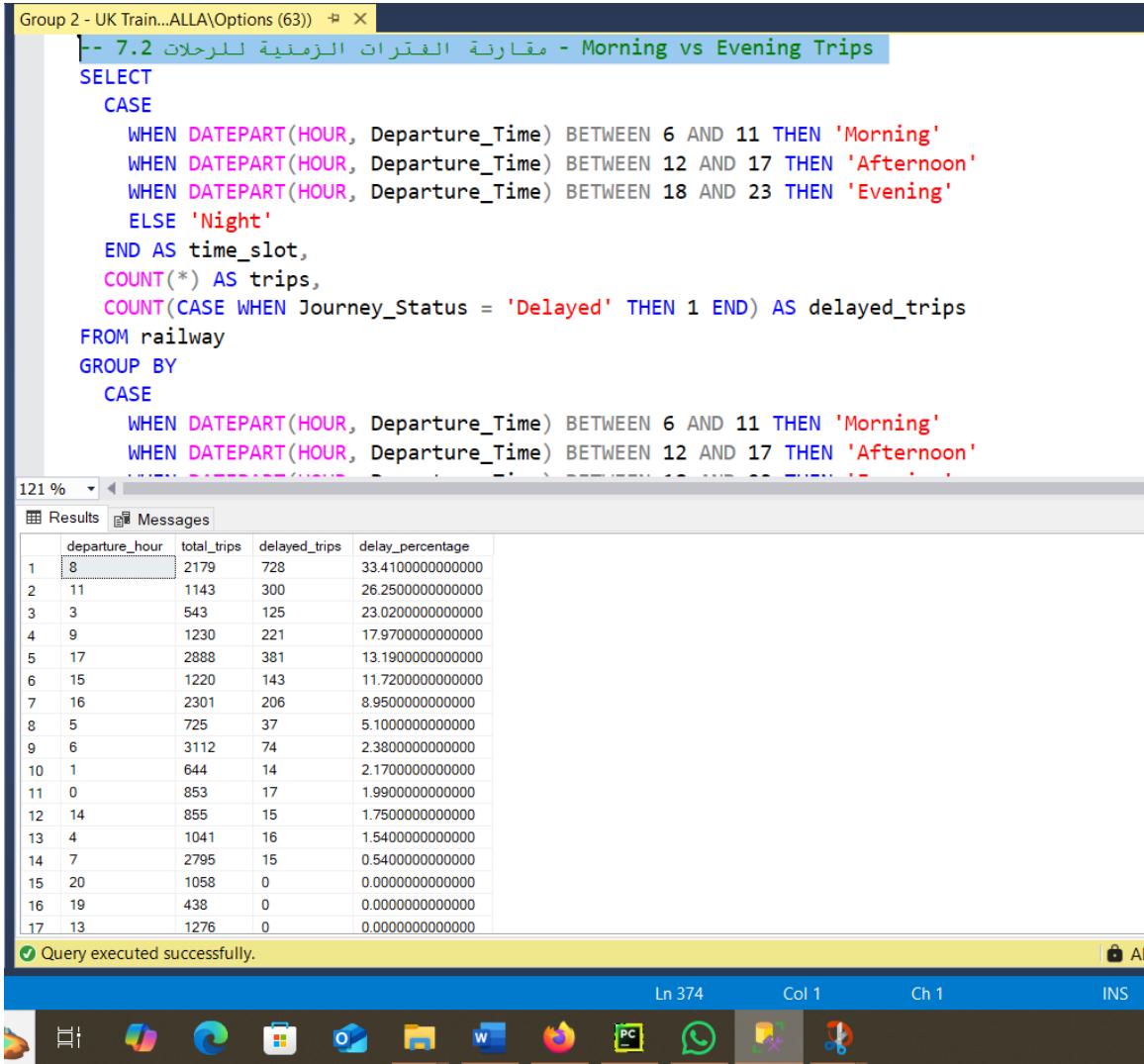
121 %

Results Messages

	departure_hour	total_trips	delayed_trips	delay_percentage
1	8	2179	728	33.41000000000000
2	11	1143	300	26.25000000000000
3	3	543	125	23.02000000000000
4	9	1230	221	17.97000000000000
5	17	2888	381	13.19000000000000
6	15	1220	143	11.72000000000000
7	16	2301	206	8.95000000000000
8	5	725	37	5.10000000000000
9	6	3112	74	2.38000000000000
10	1	644	14	2.17000000000000
11	0	853	17	1.99000000000000
12	14	855	15	1.75000000000000
13	4	1041	16	1.54000000000000
14	7	2795	15	0.54000000000000
15	20	1058	0	0.00000000000000
16	19	438	0	0.00000000000000
17	13	1276	0	0.00000000000000

Query executed successfully.

Ln 374 Col 1 Ch 1 INS



```
Group 2 - UK Train...ALLA\Options (63) ✎ X
    WHEN DATEPART(HOUR, Departure_Time) BETWEEN 18 AND 23 THEN 'Evening'
    ELSE 'Night'
END AS time_slot,
COUNT(*) AS trips,
COUNT(CASE WHEN Journey_Status = 'Delayed' THEN 1 END) AS delayed_trips
FROM railway
GROUP BY
CASE
    WHEN DATEPART(HOUR, Departure_Time) BETWEEN 6 AND 11 THEN 'Morning'
    WHEN DATEPART(HOUR, Departure_Time) BETWEEN 12 AND 17 THEN 'Afternoon'
    WHEN DATEPART(HOUR, Departure_Time) BETWEEN 18 AND 23 THEN 'Evening'
    ELSE 'Night'
END
ORDER BY trips DESC;
```

121 % ▾

Results Messages

	departure_hour	total_trips	delayed_trips	delay_percentage
1	8	2179	728	33.41000000000000
2	11	1143	300	26.25000000000000
3	3	543	125	23.02000000000000
4	9	1230	221	17.97000000000000
5	17	2888	381	13.19000000000000
6	15	1220	143	11.72000000000000
7	16	2301	206	8.95000000000000
8	5	725	37	5.10000000000000
9	6	3112	74	2.38000000000000
10	1	644	14	2.17000000000000
11	0	853	17	1.99000000000000
12	14	855	15	1.75000000000000
13	4	1041	16	1.54000000000000
14	7	2795	15	0.54000000000000
15	20	1058	0	0.00000000000000
16	19	438	0	0.00000000000000
17	13	1276	0	0.00000000000000

 Query executed successfully.

- Avg Revenue Last 3 Months 7.4

```
-- 7.4 -- متوسط الإيرادات الـ آخر 3 شهور - Avg Revenue Last 3 Months
SELECT
    AVG(monthly_revenue) AS forecast_next_month_revenue
FROM (
    SELECT
        FORMAT(Date_of_Journey, 'yyyy-MM') AS month,
        SUM(Price) AS monthly_revenue
    FROM railway
    GROUP BY FORMAT(Date_of_Journey, 'yyyy-MM')
    ORDER BY month DESC
    OFFSET 0 ROWS FETCH NEXT 3 ROWS ONLY
) AS last_3_months;
```

121 %

	forecast_next_month_revenue
1	180767

Query executed successfully.

Python

أولاً: المشكلات التي وُجدت في البيانات الخام

واجهت البيانات الخام عدداً من المشكلات الجوهرية التي استلزمت معالجتها قبل البدء في أي تحليل أو عرض رسومي. فيما يلي عرض تفصيلي لهذه المشكلات مع توضيح طرق التعامل معها:

١- القيم المفقودة(Missing Values)

كانت بعض الأعمدة تحتوي على قيم مفقودة، مما قد يؤدي إلى تحليلات غير دقيقة. تمثلت هذه القيم المفقودة في:

- غيب بيانات "سبب التأخير" (Reason for Delay) "في العديد من السجلات، خاصةً للرحلات التي وصلت في موعدها.
- غيب بيانات "وقت الوصول الفعلي" (Actual Arrival Time) "للرحلات التي تم إلغاؤها.
- غياب أو ظهور قيمة "None" في عمود "بطاقة التخفيض" (Railcard).

طريقة المعالجة:

- تم ملء القيم المفقودة في عمود "سبب التأخير" بقيمة توضيحية "لا يوجد تأخير" (No Delay) "في حالة الرحلات التي لم تتأخر.
- تم ملء القيم المفقودة في عمود "وقت الوصول الفعلي" بكلمة "ملغاة" (Cancelled) "للرحلات التي تم إلغاؤها.
- تم استبدال قيمة "None" في عمود "بطاقة التخفيض" بعبارة "لا توجد بطاقة تخفيض" (No Railcard) "لتعزيز وضوح البيانات.

٢- تنسيقات البيانات غير الصحيحة(Incorrect Data Formats)

لوحظ أن بعض الأعمدة الزمنية والتاريخية كانت مخزنة كنصوص عادية(Strings)، مما يصعب إجراء الحسابات أو التحليلات الزمنية.

طريقة المعالجة:

- تم تحويل أعمدة "تاريخ الرحلة" (Date of Journey) و"تاريخ الشراء" (Date of Purchase) إلى نوع بيانات التاريخ (Datetime).
- تم التأكد من أن الأوقات (مثل "وقت الشراء"، و"وقت المغادرة"، و"وقت الوصول المجدول") موحدة التتنسيق بصيغة (ساعة: دقيقة: ثانية) (HH:MM:SS).
- تم التأكد من أن عمود "سعر التذكرة" (Ticket Price) مخزن كبيانات رقمية مناسبة للتحليل الإحصائي.

٣ - وجود مسافات زائدة في النصوص (Extra Whitespace)

كان هناك احتمال لوجود مسافات زائدة في أسماء الأعمدة أو القيم النصية، مما قد يؤدي إلى أخطاء أثناء التصنيف أو التجميع.

طريقة المعالجة:

- تم إزالة جميع المسافات الزائدة من أسماء الأعمدة باستخدام دوال تنقية النصوص.
- تم تنظيف جميع القيم النصية داخل الأعمدة، بحيث لا تحتوي على مسافات غير مرغوبية في البداية أو النهاية.

٤ - عدم اتساق تسمية الفئات (Inconsistent Category Labels)

للحظ أن بعض الفئات النصية التي تمثل نفس المعنى كانت مكتوبة بأشكال مختلفة، مثل:

- ظهور "Staff Shortage" و "Staffing" للإشارة إلى نفس المشكلة (نقص الموظفين).
- ظهور "Weather Conditions" و "Weather" للإشارة إلى نفس الظاهرة الجوية.
- وجود اختلاف في حالة الأحرف مثل "Signal failure" بدلاً من "Signal Failure".

طريقة المعالجة:

- تم توحيد جميع هذه القيم إلى صيغة واحدة واضحة:
 - توحيد "Staffing" إلى "Staff Shortage".
 - توحيد "Weather" إلى "Weather Conditions".
 - تصحيح حالة الأحرف لتصبح "Signal Failure" بشكل موحد عبر جميع السجلات.

٥ - التكرارات في السجلات (Duplicate Records)

عند مراجعة البيانات الخام، تم رصد وجود بعض السجلات المكررة التي تحمل نفس تفاصيل الرحلة مع اختلاف طفيف أو بدون اختلاف، مما قد يؤدي إلى تحليلات مضللة.

طريقة المعالجة:

- تم فحص البيانات واكتشاف التكرارات بناءً على تطابق الأعمدة.
 - تم حذف السجلات المكررة، والإبقاء فقط على سجل واحد لكل رحلة لضمان دقة النتائج.
-

بعد تنفيذ عمليات التنظيف السابقة، أصبحت البيانات خالية من القيم المفقودة، موحدة الصيغ، خالية من المسافات الزائدة، خالية من التكرارات، ومنظمة بشكل يسمح بإجراء تحليلات دقيقة وموثوقة. هذه المعالجة الدقيقة مهدت الطريق لإنشاء داشبوردات تحليلية دقيقة وعالية الجودة لاحقاً في المشروع.

ثانياً: شرح كود التنظيف والتحليل

```
استيراد المكتبات #
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

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

```
تحميل البيانات #
df = pd.read_csv(r"D:\\railway 2.csv")
```

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

```
نظافة البيانات #
print(df.head())
print(df.tail())
print(df.describe())
print(df.info())
print(df.isnull().sum())
print(df.dtypes)
print(df.duplicated().sum())
```

استعراض أول وآخر خمس صفوف، وإظهار ملخص إحصائي، ومراجعة أنواع البيانات والقيم المفقودة والتكرارات بهدف تقييم جودة البيانات قبل بدء التنظيف.

```
# إزالة المسافات الزائدة من أسماء الأعمدة والقيم النصية
df.columns = df.columns.str.strip()
df = df.applymap(lambda x: x.strip() if isinstance(x, str)
else x)
```

تمت إزالة أي مسافات إضافية في أسماء الأعمدة وفي جميع القيم النصية الموجودة داخل الجدول، لضمان تنسيق موحد ودقيق.

```
# تنظيف النصوص بشكل أدق للأعمدة النصية فقط
str_cols = df.select_dtypes(include=['object']).columns
df[str_cols] = df[str_cols].apply(lambda x: x.str.strip())
```

تم تحديد الأعمدة النصية فقط، ثم تنظيفها من أي مسافات زائدة بشكل خاص دون التأثير على الأعمدة العددية.

```
# تحويل التواریخ إلى النوع التاریخي
df['Date of Journey'] = pd.to_datetime(df['Date of Journey'],
errors='coerce')
df['Date of Purchase'] = pd.to_datetime(df['Date of Purchase'],
errors='coerce')
```

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

```
# معالجة القيم المفقودة
df.loc[(df['Journey Status'] == 'On Time') & (df['Reason for Delay'].isna()), 'Reason for Delay'] = 'No Delay'
df.loc[(df['Journey Status'] == 'Delayed') & (df['Reason for Delay'].isna()), 'Reason for Delay'] = 'Unknown'
df['Actual Arrival Time'] = df['Actual Arrival Time'].fillna('Cancelled')
df['Railcard'] = df['Railcard'].fillna('No Railcard')
```

تم معالجة القيم المفقودة:

- تعيين "No Delay" للرحلات في موعدها بدون سبب تأخير ،
 - تعيين "Unknown" للرحلات المتأخرة بدون سبب ،
 - إدخال "Cancelled" كوقت وصول فعلي للرحلات الملغاة،
 - إدخال "No Railcard" للركاب الذين لم يستخدمو بطاقة تخفيض.
-

تعديل بعض القيم الموحدة #

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

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

مراجعة النظافة بعد المعالجة #

```
print(df.isnull().sum())  
print(df.dtypes)
```

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

حفظ نسخة من الداتا بعد التنظيف #

```
df.to_csv(r"D:\\railway_2_cleaned.csv", index=False)
```

تم حفظ نسخة جديدة من البيانات المنظفة تحت اسم railway_2_cleaned.csv للعمل عليها في الخطوات التالية دون الحاجة إلى إعادة التنظيف.

تهيئة الرسوم #

```
sns.set_style("whitegrid")  
plt.style.use('default')
```

تم تحديد نمط الرسوم البيانية ليكون بسيطاً ومنظماً بخلفية بيضاء مع شبكة خفيفة، مما يسهل قراءة البيانات المرئية.

```
# ألوان متدرجة مخصصة للرسوم
custom_palette = ['#004c6d', '#2176b7', '#57a0d3',
'#a9c9eb', '#dceefb']
```

تم إعداد لوحة ألوان مخصصة تستخدم في جميع الرسوم البيانية لخلق تناسق لوني مميز وجذاب.

```
# Dashboard 1
fig1, axs1 = plt.subplots(2, 2, figsize=(18, 12))
axs1 = axs1.flatten()
```

تم إنشاء لوحة رسم بياني مكونة من أربعة مخططات فرعية (2×2) بمساحة واسعة مناسبة لعرض الرسوم بشكل واضح.

```
# رسم التوزيعات المختلفة (Dashboard 1)
# توزيع حالة الرحلات - 1-
sns.countplot(x='Journey Status', data=df,
palette=custom_palette, ax=axs1[0])
```

تم إنشاء رسم عمودي يظهر توزيع حالات الرحلات بين الوصول في الموعد، التأخير، والإلغاء.

```
# 2- أسباب التأخير
delay_reasons = df[df['Reason for Delay'] != 'No
Delay']['Reason for Delay'].value_counts()
sns.barplot(x=delay_reasons.values, y=delay_reasons.index,
palette=custom_palette, ax=axs1[1])
```

تم إنشاء رسم عمودي يُظهر أكثر الأسباب شيوعاً لحدوث التأخير في الرحلات.

```
# 3- أكثر 5 محطات مغادرة
top_departure = df['Departure
Station'].value_counts().head(5)
sns.barplot(x=top_departure.values, y=top_departure.index,
palette=custom_palette, ax=axs1[2])
```

تم رسم أكثر خمس محطات مغادرة نشطة وفقاً لعدد الرحلات التي تطلق منها.

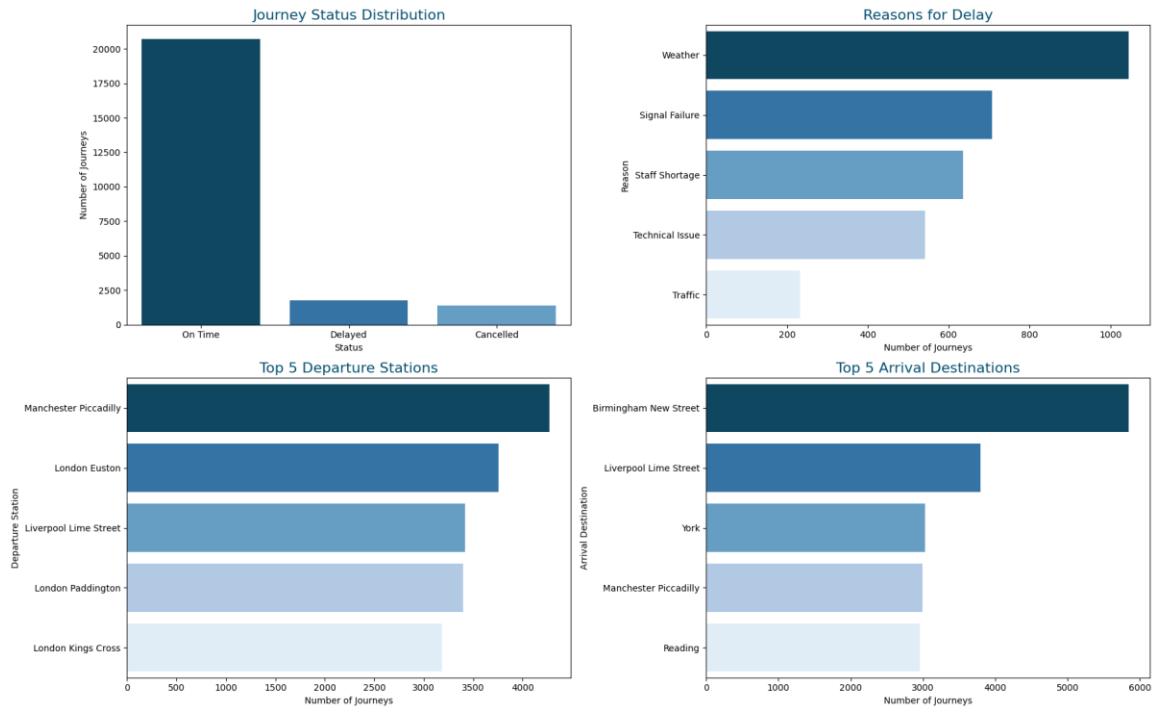
```
# 4- أكثر 5 وجهات وصول
top_arrival = df['Arrival Destination'].value_counts().head(5)
sns.barplot(x=top_arrival.values, y=top_arrival.index,
palette=custom_palette, ax=axs1[3])
```

تم رسم أكثر خمس وجهات وصول شعبية للرحلات.

```
# إعداد عنوان الداشبورد الأول
plt.suptitle('Railway Dashboard - Part 1', fontsize=20,
color='#004c6d', y=1.02)
plt.tight_layout(rect=[0, 0, 1, 0.96])
fig1.savefig('railway_dashboard_part1.png', dpi=300,
bbox_inches='tight')
plt.show()
```

تم إضافة عنوان موحد للداشبورد الأول، وضبط المسافات بين المخططات، ثم حفظ الداشبورد كصورة PNG عالية الجودة.

Railway Dashboard - Part 1



```
# Dashboard 2
fig2, axs2 = plt.subplots(2, 2, figsize=(18, 12))
axs2 = axs2.flatten()
```

تم إنشاء لوحة ثانية تحتوي على أربعة مخططات فرعية لعرض تحليلات إضافية.

```
# 1- توزيع وسائل الدفع
payment_methods = df['Payment Method'].value_counts()
axs2[0].pie(payment_methods.values,
            labels=payment_methods.index, autopct='%.1f%%',
            startangle=140,
            colors=custom_palette,
            textprops={'color':'black'})
```

تم رسم مخطط دائري يوضح نسب استخدام كل وسيلة دفع من وسائل الدفع المختلفة.

```
# 2- توزيع استخدام بطاقات التخفيض
railcard_usage = df['Railcard'].value_counts()
sns.barplot(x=railcard_usage.values,
y=railcard_usage.index, palette=custom_palette, ax=axs2[1])
```

تم رسم توزيع استخدام أنواع بطاقات التخفيض المختلفة.

```
# 3- التذاكر المباعة عبر الزمن
sales_over_time = df['Date of
Purchase'].value_counts().sort_index()
axs2[2].plot(sales_over_time.index, sales_over_time.values,
marker='o', color='#00BFFF')
```

تم رسم مخطط خطي يوضح كيف تغير عدد التذاكر المباعة مع مرور الوقت.

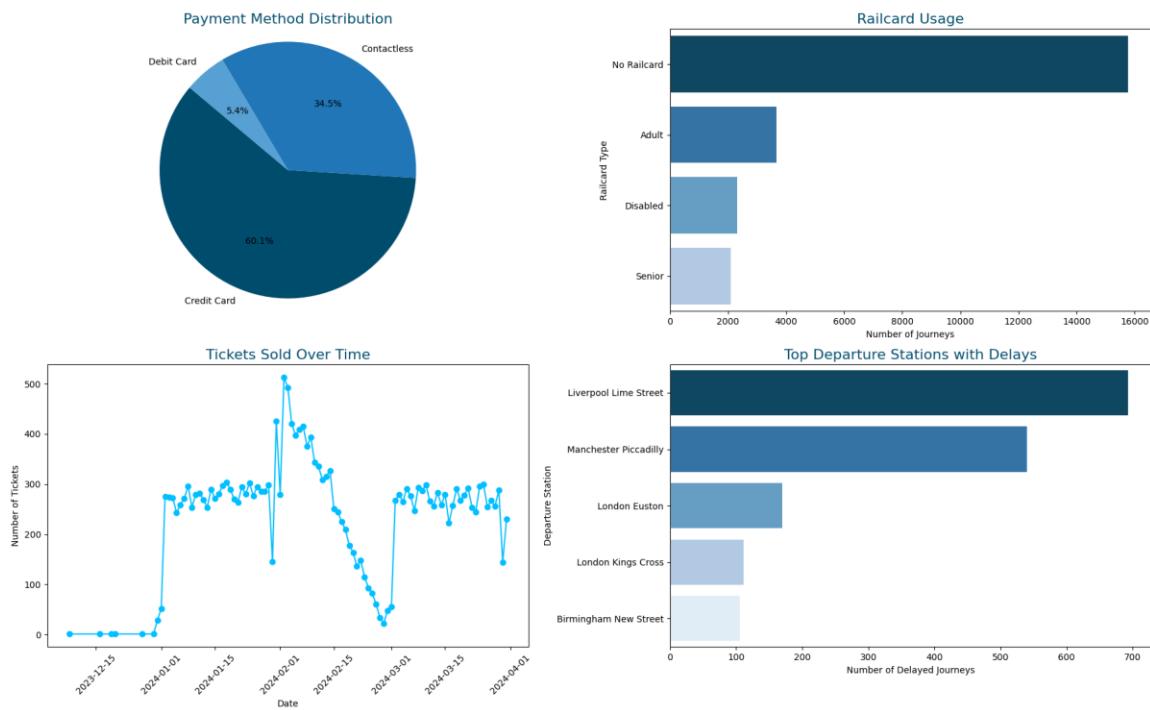
```
# 4- أكثر محطات المغادرة التي بها تأخيرات
delayed_stations = df[df['Journey Status'] ==
'Delayed']['Departure Station'].value_counts().head(5)
sns.barplot(x=delayed_stations.values,
y=delayed_stations.index, palette=custom_palette,
ax=axs2[3])
```

تم عرض أكثر محطات المغادرة التي تتكرر بها التأخيرات باستخدام رسم بياني عمودي.

```
# إعداد عنوان الدashboard الثاني
plt.suptitle('Railway Dashboard - Part 2', fontsize=20,
color='#004c6d', y=1.02)
plt.tight_layout(rect=[0, 0, 1, 0.96])
fig2.savefig('railway_dashboard_part2.png', dpi=300,
bbox_inches='tight')
plt.show()
```

تم إضافة عنوان موحد للدشبورد الثاني، وضبط المسافات بين المخططات، ثم حفظ الداشبورد الثاني كصورة PNG عالية الجودة.

Railway Dashboard - Part 2



```
# توحيد كتابة 'Signal failure' إلى 'Signal Failure'  
df['Reason for Delay'] = df['Reason for Delay'].replace('Signal failure', 'Signal Failure')
```

يبحث الكود في عمود Reason for Delay.

كل قيمة مكتوبة حرف صغير في كلمة Signal Failure يتم استبدالها بالقيمة الصحيحة.

```
# إزالة التكرارات  
df = df.drop_duplicates()
```

الكود يبحث عن أي صفوف مكررة تماماً داخل الداتا، ويقوم بحذفها تلقائياً مع الاحتفاظ بأول نسخة فقط من الصف المكرر.

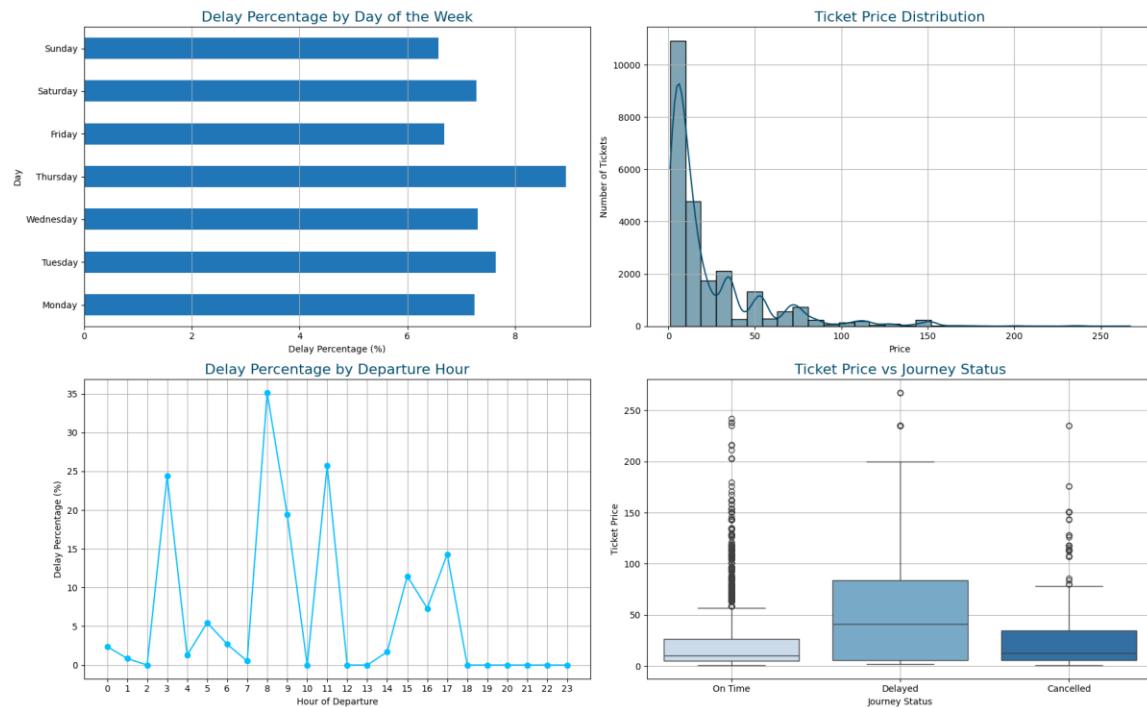
بعد الكود ده الداتا (df) تصبح خالية ١٠٠٪ من التكرارات.

```
#dashboard 3
```

```
sns.set_style("whitegrid")
plt.style.use('default')
```

يتم هنا ضبط شكل الرسوم البيانية بحيث تكون الخلفية شبكية (whitegrid)، والأسلوب العام للرسم هو الافتراضي البسيط لضمان وضوح البيانات.

Railway Dashboard - Part 3



```
fig3, axs3 = plt.subplots(2, 2, figsize=(18, 12))
axs3 = axs3.flatten()
```

يتم إنشاء شكل (Figure) يحتوي على شبكة من 2 صف × 2 عمود، أي 4 أماكن (Subplots) للرسم. يتم تحويلهم إلى مصفوفة مسطحة (flatten()) لتسهيل التعامل مع كل رسم بياني على حدة.

```
df['Day_of_Week'] = df['Date of Journey'].dt.day_name()
delay_by_day = df.groupby('Day_of_Week')['Journey
Status'].apply(lambda x: (x == 'Delayed').mean() * 100)

delay_by_day = delay_by_day.reindex([
    'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
    'Saturday', 'Sunday'
])

delay_by_day.plot(kind='barh', color='#2176b7', ax=axs3[0])
```

يتم هنا إنشاء عمود جديد Day_of_Week يحول تاريخ الرحلة إلى اسم اليوم. بعد ذلك يتم حساب نسبة الرحلات المتأخرة في كل يوم باستخدام groupby و lambda. يتم ترتيب الأيام ترتيباً طبيعياً بدأيةً من الاثنين. أخيراً يتم رسم النتائج على هيئة رسم بياني أفقي (barh) باللون الأزرق.

```
sns.histplot(df['Price'], bins=30, kde=True,
color='#004c6d', ax=axs3[1])
```

يتم رسم توزيع الأسعار باستخدام histplot مع 30 فئة (bins)، بالإضافة إلى رسم منحنى الكثافة الاحتمالية (KDE) لتوضيح شكل توزيع الأسعار، الرسم يتم باللون الأزرق الداكن. (#004c6d)

```
df['Departure_Hour'] = pd.to_datetime(df['Departure Time'],
errors='coerce').dt.hour
delay_by_hour = df.groupby('Departure_Hour')['Journey
Status'].apply(lambda x: (x == 'Delayed').mean() * 100)
delay_by_hour.plot(kind='line', marker='o',
color='#00BFFF', ax=axs3[2])
```

يتم تحويل وقت المغادرة إلى ساعة (رقم من 0 إلى 23) وحفظه في عمود جديد Departure_Hour. يتم حساب نسبة التأخير في كل ساعة. يتم رسم خط بياني (Line Plot) يوضح كيف تتغير نسبة التأخير خلال ساعات اليوم، مع نقاط بارزة 'o'.

```
sns.boxplot(x='Journey Status', y='Price', data=df,  
palette='Blues', ax=axs3[3])
```

يتم رسم مخطط صندوقي (Boxplot) لمقارنة توزيع أسعار التذاكر حسب حالة الرحلة (On Time, Delayed, Cancelled). يوضح هذا التحليل الفروقات المحتملة في الأسعار حسب حالة الرحلة باستخدام تدرج لوني أزرق Blues palette).

```
plt.suptitle('Railway Dashboard - Part 3, fontsize=20,  
color='#004c6d', y=1.02)  
plt.tight_layout(rect=[0, 0, 1, 0.96])
```

يتم إضافة عنوان رئيسي للداشبورد (Railway Dashboard - Part 3: Advanced Analysis) ويتم ضبط المسافات حول الرسومات باستخدام tight_layout بحيث لا تتدخل العناوين مع الرسومات.

```
fig3.savefig('railway_dashboard_part3.png', dpi=300,  
bbox_inches='tight')  
plt.show()
```

يتم حفظ الداشبورد كصورة عالية الجودة باسم railway_dashboard_part3.png. وأخيراً يتم عرض الشكل باستخدام plt.show().

شرح الداشبورد الأول 1 : Railway Dashboard - Part 1

تحليل أداء الرحلات وأسباب التأخير والمحطات الرئيسية

في هذا الجزء من التحليل البصري للبيانات، تم إعداد لوحة معلومات بعنوان "Railway Dashboard - Part 1" ، والتي تهدف إلى استكشاف الأداء العام للرحلات من حيث الالتزام بالمواعيد، وتحليل أسباب التأخير، وتحديد المحطات الأكثر نشاطاً سواءً من حيث الانطلاق أو الوصول. وقد تمثلت مكونات هذه اللوحة في أربعة رسوم بيانية رئيسية كما يلي:

1-توزيع حالة الرحلات (Journey Status Distribution)

في هذا الرسم البياني، تم عرض توزيع الرحلات إلى ثلاثة فئات رئيسية:

- رحلات وصلت في الموعد المحدد (On Time).
- رحلات تعرضت للتأخير (Delayed).
- رحلات تم إلغاؤها (Cancelled).

وقد أظهرت النتائج أن الغالبية العظمى من الرحلات وصلت في موعدها دون تأخير، بينما كانت نسبة الرحلات المتأخرة أو الملغاة أقل بكثير، مما يعكس مؤشرات إيجابية حول التزام القطارات بمواعيدها.

2-أسباب التأخير (Reasons for Delay)

يقدم هذا الرسم البياني تحليلًا لأكثر الأسباب شيوعًا التي أدت إلى تأخير الرحلات.
وقد تبين أن:

- الأحوال الجوية (Weather) كانت السبب الرئيسي للتأخير،
- تلتها مشكلات مثل فشل الإشارة (Signal Failure) ونقص الموظفين (Staff Shortage) والمشكلات التقنية (Technical Issue).

ويُعد هذا التحليل ضروريًا لدعم قرارات تحسين الأداء التشغيلي ومعالجة الأسباب المتكررة للتأخير.

3-المحطات الخمسة الأعلى مغادرة (Top 5 Departure Stations)

يعرض هذا الرسم البياني أكثر خمس محطات شهدت أكبر عدد من الرحلات المغادرة.
وقد جاءت محطة **London Euston** في صدارة القائمة، تلتها محطات أخرى بارزة مثل **Manchester Piccadilly** و **Liverpool Lime Street**.
هذا التحليل يساعد في تحديد المحطات الرئيسية التي تتطلب إدارة تشغيلية فعالة ودعماً إضافياً في أوقات الذروة.

4-المحطات الخمسة الأعلى وصولاً (Top 5 Arrival Destinations)

يسلط هذا الرسم الضوء على أكثر خمس محطات استقبلت الرحلات.
وقد احتلت محطة **Birmingham New Street** المركز الأول كأكبر محطة وصول ازدحاماً، متقدمة بمحطات أخرى مثل **Liverpool Lime Street** و **York**.

بعد هذا النوع من التحليل مهمًا للتخطيط المستقبلي وتحسين الخدمات المقدمة في المحطات الحيوية.

شرح الداشبورد الثاني : Railway Dashboard - Part 2

تحليل أنماط الدفع، استخدام البطاقات، مبيعات التذاكر، وتأخيرات المحطات

يتناول هذا الجزء من التحليل البصري مجموعة من المؤشرات المهمة التي تعكس سلوك الركاب من حيث طرق الدفع المستخدمة، استخدام البطاقات المخفضة، تطور مبيعات التذاكر عبر الزمن، بالإضافة إلى تحليل المحطات الأكثر تعرضاً للتأخير. وقد تم تجميع هذه التحليلات في أربعة رسوم بيانية رئيسية كما يلي:

1-توزيع وسائل الدفع (Payment Method Distribution)

يعرض هذا الرسم البياني نسب استخدام وسائل الدفع المختلفة عند شراء التذاكر، وتشمل:

- بطاقات الائتمان (Credit Card)
- الدفع اللااتلامسي (Contactless)
- بطاقات الخصم (Debit Card)

وقد أظهرت النتائج أن بطاقات الائتمان كانت الوسيلة الأكثر شيوعاً بين الركاب، تليها طريقة الدفع اللااتلامسي، بينما جاءت بطاقات الخصم بنسبة أقل بكثير. هذا التحليل يساعد في فهم تفضيلات الركاب ومن ثم تطوير الخدمات المالية المقدمة لهم.

2-استخدام البطاقات المخفضة (Railcard Usage)

يوضح هذا الرسم البياني توزيع استخدام أنواع البطاقات المخفضة المختلفة مثل:

- (عدم استخدام بطاقة تخفيض) No Railcard
- (بطاقة البالغين) Adult Railcard
- (بطاقة ذوي الاحتياجات الخاصة) Disabled Railcard
- (بطاقة كبار السن) Senior Railcard.

وقد تبين أن النسبة الأكبر من الركاب لا يستخدمون بطاقات تخفيض، في حين جاءت بطاقة البالغين في المرتبة الثانية من حيث الاستخدام.

يوفر هذا التحليل معلومات مهمة عند تصميم الحملات الترويجية والعروض الخاصة بالركاب.

3- مبيعات التذاكر عبر الزمن (Tickets Sold Over Time)

يُستعرض هذا الرسم البياني تطور عدد التذاكر المباعة يومياً خلال الفترة الزمنية المشمولة في الدراسة.

وقد لاحظنا:

- وجود ارتفاع ملحوظ في مبيعات التذاكر في بداية العام،
- نلاه انخفاض تدريجي قد يكون مرتبًا بالعوامل الموسمية أو فترات العطلات.

هذا التحليل مفيد جدًا في التخطيط لتوزيع الموارد البشرية واللوجستية خلال فترات الذروة والانخفاض.

4- المحطات الخمسة الأعلى في التأخيرات (Top Departure Stations with Delays)

يعرض هذا الرسم البياني المحطات التي شهدت أكبر عدد من الرحلات المتأخرة.

وقد جاءت محطة **Liverpool Lime Street** في المرتبة الأولى، تليها محطات أخرى مثل **Manchester**, **London Euston**, و **Piccadilly**.

يمكن هذا التحليل إدارات المحطات من دراسة الأسباب الخاصة بالتأخيرات وتحسين إجراءات التشغيل لتقليل معدلات التأخير مستقبلاً.

الملاحظات الخاصة بـ 1: Dashboard

1- ملاحظة على أسباب التأخير: (Reasons for Delay)

للحظ ضمن تصنيف أسباب التأخير وجود فئة عامة تحت اسم **Other**.
تشير هذه الفئة إلى مجموعة الأسباب التي لم يتم تصنيفها ضمن الأسباب الرئيسية مثل الطقس أو فشل الإشارة أو المشكلات التقنية.

يُوصى عند عرض هذه النتائج بذكر أن فئة "Other" تشمل أسباباً متنوعة وغير محددة بدقة في البيانات الأصلية.

الملاحظات الخاصة بـ 2: Dashboard

1- ملاحظة على استخدام بطاقات التخفيض: (Railcard Usage)

أظهرت البيانات أن نسبة كبيرة من الركاب لا يستخدمون بطاقات تخفيض مصنفين تحت ("No Railcard").
وهذا يعكس واقعاً قد يرتبط بطبيعة الفئات المستهدفة أو أن بعض المسافرين يفضلون الحجز دون استخدام بطاقات خصم.
عند عرض هذا التحليل، من المفيد الإشارة إلى هذه الملاحظة لتوضيح التركيبة السكانية للركاب.

2- ملاحظة على مبيعات التذاكر عبر الزمن: (Tickets Sold Over Time)

تبين من خلال الرسم البياني وجود تذبذب ملحوظ في أعداد التذاكر المباعة على مدار الفترة الزمنية، مع وجود فجوات في بعض الأيام. من المحتمل أن تكون هذه الفجوات ناتجة عن:

- فترات الإجازات الرسمية،
- أو انخفاض الطلب الموسمي،
- أو نقص في تسجيل بعض البيانات.

ينبغي عند تفسير هذا الرسم الإشارة إلى هذا التذبذب باعتباره نمطاً طبيعياً في حركة مبيعات التذاكر على مدار السنة.

شرح "Railway Dashboard - Part 3"

تتكون هذه الداشبورد من أربع تحليلات تفصيلية تهدف إلى فهم العوامل المؤثرة على تأخير الرحلات وأسعار التذاكر في بيانات السكك الحديدية. فيما يلي شرح مفصل لكل جزء:

١. تحليل نسبة التأخير حسب يوم الأسبوع

Delay Percentage by Day of the Week

- يوضح هذا الرسم البياني الأفقي نسبة الرحلات المتأخرة في كل يوم من أيام الأسبوع.
- نلاحظ أن يوم الخميس يظهر فيه أعلى نسبة تأخير مقارنة ببقية الأيام، في حين تكون نسبة التأخير أقل في أيام أخرى مثل الأحد والجمعة.
- يساعد هذا التحليل في التعرف على الأيام التي تتطلب تعزيزات إضافية لتحسين الخدمة.

٢. توزيع أسعار التذاكر

Ticket Price Distribution

- يمثل هذا الرسم التوزيع الإحصائي لأسعار التذاكر.
- نلاحظ أن معظم التذاكر تقع ضمن نطاقات سعرية منخفضة (0-50 تقريراً)، مع وجود عدد قليل جداً من التذاكر ذات الأسعار المرتفعة.

- يظهر منحنى الكثافة (KDE) فوق المدرج البياني مما يوضح اتجاه توزيع الأسعار بشكل أكثر سلاسة.
 - يساعد هذا التحليل في فهم طبيعة التسعير ومدى تنوع الأسعار المقدمة للعملاء.
-

٣. تحليل نسبة التأخير حسب ساعة المغادرة

Delay Percentage by Departure Hour

- يعرض هذا الرسم العلاقة بين ساعة انطلاق الرحلة ونسبة حدوث التأخير.
 - نلاحظ وجود بعض الساعات، مثل **الساعة 9 صباحاً والساعة 12 ظهراً**، ترتفع فيها نسبة التأخير مقارنة بساعات أخرى.
 - هذا النوع من التحليل يفيد في تحسين جدولة القطارات عبر زيادة عدد القطارات أو تحسين أوقات الانطلاق في ساعات الذروة.
-

٤. مقارنة سعر التذكرة حسب حالة الرحلة

Ticket Price vs Journey Status

- يتم مقارنة توزيع أسعار التذاكر بناءً على حالة الرحلة (Boxplot) باستخدام مخطط الصندوق (Boxplot) باستخدامة **On Time**، **Delayed**، **Cancelled**، (ملغاة).
- يوضح المخطط أن الرحلات المتأخرة عموماً لها تذاكر بسعر متوسط أعلى قليلاً مقارنة بالرحلات في موعدها.
- كما يظهر وجود بعض القيم الشاذة (outliers) بأسعار مرتفعة في جميع الفئات.
- يساهم هذا التحليل في فحص ما إذا كان هناك علاقة بين الأسعار وجودة الخدمة.

```
# استيراد المكتبات
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# تحميل البيانات
df = pd.read_csv(r"railway 2 .csv")

# نظافة البيانات
print(df.head())
print(df.tail())
print(df.describe())
print(df.info())
print(df.isnull().sum())
print(df.dtypes)
print(df.duplicated().sum())

# إزالة المسافات الزائدة
df.columns = df.columns.str.strip()
df = df.applymap(lambda x: x.strip() if isinstance(x, str) else x)

# إزالة التكرارات
df = df.drop_duplicates()

# تنضيف النصوص
str_cols = df.select_dtypes(include=['object']).columns
df[str_cols] = df[str_cols].apply(lambda x: x.str.strip())

# تحويل التواريخ
df['Date of Journey'] = pd.to_datetime(df['Date of Journey'],
                                         errors='coerce')
df['Date of Purchase'] = pd.to_datetime(df['Date of Purchase'],
                                         errors='coerce')

# معالجة القيم الفارغة
df.loc[(df['Journey Status'] == 'On Time') & (df['Reason for Delay'].isna()), 'Reason for Delay'] = 'No Delay'
df.loc[(df['Journey Status'] == 'Delayed') & (df['Reason for Delay'].isna()), 'Reason for Delay'] = 'Unknown'
df['Actual Arrival Time'] = df['Actual Arrival Time'].fillna('Cancelled')
```

```

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

# تعديل بعض القيم الموحدة
df['Reason for Delay'] = df['Reason for Delay'].replace({
    'Staffing': 'Staff Shortage',
    'Weather Conditions': 'Weather'
})

# 'Signal failure' إلى 'Signal Failure'
df['Reason for Delay'] = df['Reason for Delay'].replace('Signal
failure', 'Signal Failure')

print(df.isnull().sum())
print(df.dtypes)

# حفظ نسخة من البيانات بعد التنظيف
df.to_csv(r"D:\\railway_2_cleaned.csv", index=False)

# -----
# عمل وعرض وحفظ THE DASHBOARDS

sns.set_style("whitegrid")
plt.style.use('default')

# ألوان متدرجة
custom_palette = ['#004c6d', '#2176b7', '#57a0d3', '#a9c9eb',
'#dceefb']

# -----
# داشبورد 1
fig1, axs1 = plt.subplots(2, 2, figsize=(18, 12))
axs1 = axs1.flatten()

# Journey Status Distribution
sns.countplot(x='Journey Status', data=df, palette=custom_palette,
ax=axs1[0])
axs1[0].set_title('Journey Status Distribution', fontsize=16,
color='#004c6d')
axs1[0].set_xlabel('Status')
axs1[0].set_ylabel('Number of Journeys')

# Reasons for Delay

```

```

delay_reasons = df[df['Reason for Delay'] != 'No Delay']['Reason for Delay'].value_counts()
sns.barplot(x=delay_reasons.values, y=delay_reasons.index,
palette=custom_palette, ax=axs1[1])
axs1[1].set_title('Reasons for Delay', fontsize=16, color='#004c6d')
axs1[1].set_xlabel('Number of Journeys')
axs1[1].set_ylabel('Reason')

# Top 5 Departure Stations
top_departure = df['Departure Station'].value_counts().head(5)
sns.barplot(x=top_departure.values, y=top_departure.index,
palette=custom_palette, ax=axs1[2])
axs1[2].set_title('Top 5 Departure Stations', fontsize=16,
color='#004c6d')
axs1[2].set_xlabel('Number of Journeys')
axs1[2].set_ylabel('Departure Station')

# Top 5 Arrival Destinations
top_arrival = df['Arrival Destination'].value_counts().head(5)
sns.barplot(x=top_arrival.values, y=top_arrival.index,
palette=custom_palette, ax=axs1[3])
axs1[3].set_title('Top 5 Arrival Destinations', fontsize=16,
color='#004c6d')
axs1[3].set_xlabel('Number of Journeys')
axs1[3].set_ylabel('Arrival Destination')

plt.suptitle('Railway Dashboard - Part 1', fontsize=20,
color='#004c6d', y=1.02)
plt.tight_layout(rect=[0, 0, 1, 0.96])

fig1.savefig('railway_dashboard_part1.png', dpi=300,
bbox_inches='tight')
plt.show()

# -----
# 2 بورڈ اش
fig2, axs2 = plt.subplots(2, 2, figsize=(18, 12))
axs2 = axs2.flatten()

# Payment Method Distribution
payment_methods = df['Payment Method'].value_counts()
axs2[0].pie(payment_methods.values, labels=payment_methods.index,
autopct='%.1f%%', startangle=140,
colors=custom_palette, textprops={'color':'black'})

```

```

    axs2[0].set_title('Payment Method Distribution', fontsize=16,
color='#004c6d')
    axs2[0].axis('equal')

# Railcard Usage
railcard_usage = df['Railcard'].value_counts()
sns.barplot(x=railcard_usage.values, y=railcard_usage.index,
palette=custom_palette, ax=axs2[1])
axs2[1].set_title('Railcard Usage', fontsize=16, color='#004c6d')
axs2[1].set_xlabel('Number of Journeys')
axs2[1].set_ylabel('Railcard Type')

# Tickets Sold Over Time
sales_over_time = df['Date of Purchase'].value_counts().sort_index()
axs2[2].plot(sales_over_time.index, sales_over_time.values,
marker='o', color='#00BFFF')
axs2[2].set_title('Tickets Sold Over Time', fontsize=16,
color='#004c6d')
axs2[2].set_xlabel('Date')
axs2[2].set_ylabel('Number of Tickets')
axs2[2].tick_params(axis='x', rotation=45)

# Top Departure Stations with Delays
delayed_stations = df[df['Journey Status'] == 'Delayed']['Departure
Station'].value_counts().head(5)
sns.barplot(x=delayed_stations.values, y=delayed_stations.index,
palette=custom_palette, ax=axs2[3])
axs2[3].set_title('Top Departure Stations with Delays', fontsize=16,
color='#004c6d')
axs2[3].set_xlabel('Number of Delayed Journeys')
axs2[3].set_ylabel('Departure Station')

plt.suptitle('Railway Dashboard - Part 2', fontsize=20,
color='#004c6d', y=1.02)
plt.tight_layout(rect=[0, 0, 1, 0.96])

fig2.savefig('railway_dashboard_part2.png', dpi=300,
bbox_inches='tight')
plt.show()

# -----
# داشبورد 3
# ضبط ستایل العرض

```

```

sns.set_style("whitegrid")
plt.style.use('default')

إنشاء الشكل # تحليل نسبة التأخير حسب يوم الأسبوع
fig3, axs3 = plt.subplots(2, 2, figsize=(18, 12))
axs3 = axs3.flatten()

df['Day_of_Week'] = df['Date of Journey'].dt.day_name()
delay_by_day = df.groupby('Day_of_Week')[['Journey
Status']].apply(lambda x: (x == 'Delayed').mean() * 100)

delay_by_day = delay_by_day.reindex([
    'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
'Saturday', 'Sunday'
]) # ترتيب الأيام بشكل صحيح

delay_by_day.plot(kind='barh', color='#2176b7', ax=axs3[0])
axs3[0].set_title('Delay Percentage by Day of the Week',
fontsize=16, color='#004c6d')
axs3[0].set_xlabel('Delay Percentage (%)')
axs3[0].set_ylabel('Day')
axs3[0].grid(axis='x')

توزيع أسعار التذاكر # تحليل وقت المغادرة مقابل نسبة التأخير
sns.histplot(df['Price'], bins=30, kde=True, color='#004c6d',
ax=axs3[1])
axs3[1].set_title('Ticket Price Distribution', fontsize=16,
color='#004c6d')
axs3[1].set_xlabel('Price')
axs3[1].set_ylabel('Number of Tickets')
axs3[1].grid(True)

df['Departure_Hour'] = pd.to_datetime(df['Departure Time'],
errors='coerce').dt.hour
delay_by_hour = df.groupby('Departure_Hour')[['Journey
Status']].apply(lambda x: (x == 'Delayed').mean() * 100)

delay_by_hour.plot(kind='line', marker='o', color="#00BFFF",
ax=axs3[2])
axs3[2].set_title('Delay Percentage by Departure Hour', fontsize=16,
color='#004c6d')
axs3[2].set_xlabel('Hour of Departure')
axs3[2].set_ylabel('Delay Percentage (%)')

```

```
axs3[2].grid(True)
axs3[2].set_xticks(range(0, 24))

# تحليل العلاقة بين سعر التذكرة وحالة الرحلة
sns.boxplot(x='Journey Status', y='Price', data=df, palette='Blues',
ax=axs3[3])
axs3[3].set_title('Ticket Price vs Journey Status', fontsize=16,
color='#004c6d')
axs3[3].set_xlabel('Journey Status')
axs3[3].set_ylabel('Ticket Price')
axs3[3].grid(True)

# تنسيق الشكل العام
plt.suptitle('Railway Dashboard - Part 3', fontsize=20,
color='#004c6d', y=1.02)
plt.tight_layout(rect=[0, 0, 1, 0.96])

# حفظ الصورة
fig3.savefig('railway_dashboard_part3.png', dpi=300,
bbox_inches='tight')
plt.show()
```

Power bi

شرح لوحة المعلومات الرئيسية (Overview Dashboard) لتحليل بيانات القطارات UK Train

1- نظرة عامة عامة

تعرض هذه اللوحة نظرة شاملة على الأداء العام لشبكة القطارات خلال فترة زمنية محددة، حيث تشمل أهم المؤشرات التشغيلية والمالية.

2- المؤشرات الرئيسية (KPIs)

• عدد الركاب (Passenger):

عدد الرحلات المจองة من قبل الركاب، ويبلغ حوالي 31.65 ألف.

• الإيرادات (Revenue):

مجموع الإيرادات الناتجة عن بيع التذاكر ويبلغ 742 ألف وحدة نقدية.

• متوسط سعر التذكرة (Avg Ticket Price):

السعر الوسيط للتذكرة ويبلغ 23 وحدة نقدية.

• عدد المسارات (Route):

تم تحليل 65 مساراً مختلفاً.

• عدد الرحلات الملغاة (Cancelled):

بلغ عدد الرحلات الملغاة 1.88 ألف.

• الرحلات في الوقت المحدد (On Time):

عدد الرحلات التي وصلت أو غادرت في وقتها دون تأخير بلغ 27.48 ألف.

(Purchase Type): طرق شراء التذاكر

• **أونلاين (Online):** تم شراء 18,521 تذكرة عبر الإنترنت.

• **من المحطة (Station):** تم شراء 13,132 تذكرة من محطات القطار.

• **إجمالي المعاملات:** 31,653 معاملة.

(Revenue by Departure Station): الإيرادات حسب محطة المغادرة

يوضح الرسم البياني أن محطة **London Kings Cross** سجلت أعلى الإيرادات (~200K)، تليها **London Euston** و **Liverpool Lime Street**.

(Trips by Departure Station): عدد الرحلات حسب محطة المغادرة

تُظهر البيانات أن **Manchester Piccadilly** هي الأعلى من حيث عدد الرحلات المغادرة (~5.7K)، تليها **London Euston** و **Liverpool Lime Street**.

(Revenue by Arrival Station): الإيرادات حسب محطة الوصول

أعلى الإيرادات جاءت من محطة **York** (~185K)، تليها **Birmingham** و **London Euston**.

(Trips by Arrival Destination): عدد الرحلات حسب محطة الوصول

سجلت **Birmingham** أعلى عدد وصول (~7.7K)، ثم **York** و **Liverpool**.

8- توزيع عدد الرحلات عبر الأشهر (Trip Count):

تمثل البيانات التغير الشهري في عدد الرحلات، حيث كان أعلى عدد في فبراير (269 رحلة)، بينما شهد أبريل انخفاضاً نسبياً إلى 250.



تحليل التأخيرات (Delay Analysis Dashboard)

1-مؤشرات الأداء الرئيسية (KPIs):

- عدد الرحلات المتأخرة: (Delayed Trips)
بلغ عدد الرحلات التي تعرضت لتأخير 2.29 ألف رحلة.
- إجمالي دقائق التأخير: (Delay Minutes)
مجموع عدد دقائق التأخير بلغ 672.23 دقيقة.
- متوسط مدة التأخير: (Avg Delay Minutes)
متوسط التأخير لكل رحلة بلغ 42 دقيقة.
- نسبة معدل التأخير: (Delay Rate Percentage)
يمثل نسبة الرحلات المتأخرة من إجمالي الرحلات، وتبلغ 7.24%.

2- الرحلات المتأخرة حسب وجهة الوصول (Arrival Destination):

- أعلى نسبة تأخير كانت في London Euston: عدد الرحلات المتأخرة 1,064 رحلة بنسبة 68% تأخير وصلت إلى.
- سجلتا نسب تأخير أقل بكثير Doncaster و Leeds.

3- الرحلات المتأخرة حسب محطة المغادرة (Departure Station):

- المحطات التي شهدت أكبر عدد من الرحلات المتأخرة:

ـ حوالى 0.90 ألف رحلة Liverpool Lime Street: .
ـ حوالى 0.67 ألف رحلة Manchester Piccadilly: .
ـ بينما Edinburgh Waverley و London Paddington سجلتا عدداً قليلاً نسبياً من التأخيرات.

- ـ 4- تحليل عدد الرحلات ونسب التأخير حسب محطة المغادرة:
- ـ سجلت أكبر عدد من الرحلات (5,650) Manchester Piccadilly: . مع 672 رحلة متأخرة.
 - ـ سجلت 4,561 رحلة، منها 900 متأخرة Liverpool Lime Street: .
 - ـ المحطات الأخرى مثل Oxford و Birmingham New Street سجلت تأخيرات معتدلة.
-

- ـ 5- أسباب التأخير: (Delay Count by Reason)
- ـ أهم الأسباب المسجلة للتأخير: .
 - ـ الطقس 758 (Weather): حالة.
 - ـ مشكلة تقنية 472 (Technical Issue): حالة.
 - ـ فشل الإشارة 451 (Signal Failure): حالة.
 - ـ نقص الموظفين، والظروف الجوية، والازدحام المروري كانت من الأسباب الأقل تأثيراً.

UK Train Rides GROUP 2 • Last saved: Today at 11:31 AM

Search 217959724@mavenbi.org X

File Home Insert Modeling View Optimize Help Share ▾

Cut Copy Format painter Get Excel OneLake SQL Enter Dataverse Recent sources Transform Refresh New Text More visual New visual box quick calculations Sensitivity Publish Copilot

Clipboard Data Queries Insert Calculations Sensitivity Share Copilot

Delayed trips 2.29K Delay percentage 672.23 Avg Delay Minutes 42 Delay Rate Percentage 7.24

UK Train

Overview Delay Analysis Payment & Ticket Analysis Time-Based Analysis Advanced insights 1 Advanced insights 2

Trips, Delayed Trips and Delay Percentage by Arrival Destination

Delayed trips by Departure Station

Trips, Delayed Trips and Delay Percentage by Departure Station

Delay Count by Reason For Delay

UK Train Overview Delay Analysis X Payment & Ticket Analysis Time-Based Analysis Advanced Insights Advanced insights 2 +

Page 3 of 7 UK_TRAIN_TRAIN RIDES[22].docx - Word Storage Mode: Mixed 84%

| ملخص التحليلات

تم إنشاء لوحة تحكم تفاعلية باستخدام Power BI تشمل:

- عدد طلبات الاسترداد: 1.118K
- قيمة المبالغ المسترددة: 38.70K
- معدل الاسترداد: 3.53%

| تفاصيل التحليلات

-1. تحليل التأخير (Delay Analysis)

- أكبر عدد مستخدمين متاخرين بدون بطاقة خصم.
- البالغين أكثر تعرضاً للتأخير مقارنة بكبار السن أو ذوي الإعاقة.

-2. تحليل التذاكر والدفع (Payment & Ticket Analysis)

• أنواع التذاكر الأعلى إيراداً:

Advance: 309K

Anytime: 209K

• طرق الدفع الأكثر استخداماً:

Credit Card: 469K

Contactless: 219K

-3. تحليل شهري للإيرادات (Time-Based Analysis)

• أعلى إيراد تحقق في مارس 2024 (593K) للدرجة العاديّة.

رؤى متقدمة | Advanced Insights

- معدل طلبات الاسترداد منخفض نسبياً (3.5%)، ما يدل على رضا عام.
- الدخل الأكبر من التذاكر العاديّة رغم ارتفاع سعر الدرجة الأولى.

UK Train Rides GROUP 2 • Last saved: Today at 11:31 AM ▾

Search

217959724@mavenbi.org

File Home Insert Modeling View Optimize Help

Share

Cut Copy Format painter

Paste Get data Data workbook catalog Server Recent sources Transform Refresh data New visual Text box More visual Calculations Sensitivity Publish Copilot

Clipboard Data Queries Insert Calculations Sensitivity Share Copilot

Refund Count
1.118K

Total Refund Value
38.70K

Refund Rate
3.53

Delayed Users and Total Users by Railcard

Railcard Type	Delayed Users	Total Users
None	2K	1.4K
Adult	0.7K	0.5K
Disabled	0.1K	0.1K
Senior	0.1K	0.1K

Total Paid and Usage Count by Railcard

Railcard Type	Total Paid	Usage Count
None	574K	21K
Adult	86K	5K
Disabled	52K	3K
Senior	30K	3K

Ticket Type Total Revenue Trip Count

Ticket Type	Total Revenue	Trip Count
Advance	309274	17561
Anytime	209309	5340
Off-Peak	223338	8752
Total	741921	31653

Requests by Refund Request

Revenue and Trips by Ticket Class

Ticket Class	Revenue	Trips
Standard	593K	29K
First Class	149K	3K

Payment Method Transactions Total Revenue avg price

Payment Method	Transactions	Total Revenue	avg price
Debit Card	1683	52966	31
Credit Card	19136	469511	24
Contactless	10834	219444	20
Total	31653	741921	75

Revenue by month and Ticket Class

UK Train Overview Delay Analysis Payment & Ticket Analysis Time-Based Analysis Advanced Insights Advanced insights 2

Filters

Page 4 of 7

Storage Mode: Mixed 84%

تحليل زمني (Time-Based Analysis Dashboard)

1- عدد الرحلات حسب أيام الأسبوع: (Trips by Journey Day)

- الأيام الأكثر ازدحاماً:
- يوم الثلاثاء (Tuesday) سجل أعلى عدد رحلات بـ 4,692 رحلة.
- يليه الأحد (Sunday) والأربعاء (Wednesday).
- أقل الأيام في عدد الرحلات:
- الخميس (Friday) والجمعة (Thursday) سجلاً أقل عدد رحلات.

2- المشتريات حسب يوم الشراء: (Purchases by Purchase Day)

- الأيام الأكثر في عدد عمليات الشراء:
- الأحد (Sunday) والأربعاء (Wednesday).
- أقل أيام الشراء:
- كان يوم الخميس (Thursday) بأقل عدد مشتريات.

3- عدد الرحلات في تواریخ محددة: (Trip Count by Date)

- أكثر التواریخ التي شهدت عدد رحلات كبير:
- 24 يناير 2024: سجلت 269 رحلة.
- 10 أبريل و 30 يناير: سجلت كل منهما 250 و 265 رحلة على التوالي.

4- الإيرادات حسب الشهر ونوع التذكرة (Revenue by Month and Ticket Class):

- أعلى الإيرادات سُجلت في:
- يناير 2024: 200K (إجمالي 159K من الدرجة العادية + 41K من الدرجة الأولى).
- يليها مارس وأبريل بنفس النمط.
- الدرجة العادية (Standard) تمثل النسبة الأكبر من الإيرادات بالمقارنة مع الدرجة الأولى.

5- الرحلات والإيرادات حسب الشهر (Trips and Revenue by Journey Month):

- يناير ومارس هما الأشهر أعلى في عدد الرحلات (8.1K).
- مارس (2024-03) حقق أعلى إيرادات شهرية بـ 195K.

6- إجمالي المشتريات حسب ساعة الشراء (Sum of Purchases by Purchase Hour):

- أعلى أوقات الشراء كانت خلال:
- الساعة 15:00 (3 مساءً): 2.7K (عملية شراء).
- الساعة 19:00 (7 مساءً): 2.2K (عملية).
- أقل الأوقات: بين الساعة 00:00 و 06:00 صباحاً.

UK Train Rides GROUP 2 • Last saved: Today at 11:31 AM ▾

File Home Insert Modeling View Optimize Help

Share ▾

Cut Copy Format painter Get Excel OneLake SQL Enter Dataverse Recent sources Data Transform Refresh data New visual Text box More visuals Insert Calculations Sensitivity New visual calculation New measure Quick measure Publish Copilot

Clipboard

UK Train

Trips by Journey Day

Journey Day	Trips
Wednesday	4692
Tuesday	4607
Sunday	4580
Thursday	4580
Monday	4436
Saturday	4407
Friday	4351

Purchases by Purchase Day

Purchase Day	Purchases
Sunday	4600
Friday	4676
Wednesday	4627
Saturday	4602
Monday	4477
Tuesday	4455
Thursday	4454

Date of Journey Trip Count

Date of Journey	Trip Count
1/7/2024 12:00:00 AM	248
1/24/2024 12:00:00 AM	269
1/30/2024 12:00:00 AM	265
3/24/2024 12:00:00 AM	248
4/10/2024 12:00:00 AM	250

Revenue by month and Ticket Class

month	First Class	Standard
2024-01	41K	159K
2024-03	41K	154K
2024-04	37K	150K
2024-02	30K	129K

Trips and Revenue by Journey Month

Journey Month	Trips	Revenue
1	8.1K	159K
2	7.6K	195K
3	8.1K	188K
4	7.8K	150K

Sum of Purchases by Purchase Hour

Purchase Hour	Sum of Purchases
0	0.9K
1	1.2K
2	0.7K
3	1.1K
4	2.7K
5	1.9K
6	1.2K
7	0.5K
8	2.2K
9	0.7K
10	0.7K
11	1.1K
12	2.7K
13	1.2K
14	0.5K
15	1.9K
16	1.1K
17	2.7K
18	1.2K
19	0.5K
20	2.2K
21	0.7K
22	1.1K
23	2.7K
24	1.2K
25	0.5K
26	1.9K
27	1.1K
28	2.7K
29	1.2K
30	0.5K
31	1.9K
32	1.1K
33	2.7K
34	1.2K
35	0.5K
36	1.9K
37	1.1K
38	2.7K
39	1.2K
40	0.5K
41	1.9K
42	1.1K
43	2.7K
44	1.2K
45	0.5K
46	1.9K
47	1.1K
48	2.7K
49	1.2K
50	0.5K
51	1.9K
52	1.1K
53	2.7K
54	1.2K
55	0.5K
56	1.9K
57	1.1K
58	2.7K
59	1.2K
60	0.5K
61	1.9K
62	1.1K
63	2.7K
64	1.2K
65	0.5K
66	1.9K
67	1.1K
68	2.7K
69	1.2K
70	0.5K
71	1.9K
72	1.1K
73	2.7K
74	1.2K
75	0.5K
76	1.9K
77	1.1K
78	2.7K
79	1.2K
80	0.5K
81	1.9K
82	1.1K
83	2.7K
84	1.2K
85	0.5K
86	1.9K
87	1.1K
88	2.7K
89	1.2K
90	0.5K
91	1.9K
92	1.1K
93	2.7K
94	1.2K
95	0.5K
96	1.9K
97	1.1K
98	2.7K
99	1.2K
100	0.5K

UK Train Overview Delay Analysis Payment & Ticket Analysis Time-Based Analysis Advanced Insights Advanced insights 1 Advanced insights 2

Storage Mode: Mixed 84%

1- الحجز المسبق والتأخير (Booking in Advance vs Delays)

- أغلب الحجوزات تمت قبل الرحلة بـ 0 يوم (K14) رحلة، بينما الحجوزات المسبقة لأيام لاحقة تكاد تكون معدومة.
- رغم الحجز المسبق، تظهر نسبة تأخير شبه ثابتة عبر الأيام.

2- نسبة التأخير حسب نوع التذكرة (Ticket Type)

- أعلى نسبة تأخير كانت لتذاكر Anytime (9.1%) رغم قلة عدد الرحلات.
- تذاكر Advance و Off-Peak لديهما عدد رحلات أعلى مع نسب تأخير متقاربة (6.9% و 6.8%).

3- متوسط السعر حسب نوع التذكرة

- الأعلى بمتوسط سعر 39 Anytime:
- متوسط 25 Off-Peak:
- الأرخص بـ 17 Advance:

4- الزمن وتأثيره على التأخير (Time Slots)

- فترة الصباح الأولى في عدد الرحلات والتأخيرات (K11) رحلة و 1.3% متأخرة).
- التأخير في الفترة المسائية معدوم تقريباً.

5- الساعة وتأثيرها على التأخير (Hour-Based Delays)

- أعلى نسب تأخير حدثت في ساعات الذروة: 8 صباحاً (33%) و 13 ظهراً (26%).
- الساعات الأخرى تظهر تأخيرات أقل بوضوح.

UK Train Rides GROUP 2 • Last saved: Today at 11:31 AM

Search

217959724@mavenbi.org

File Home Insert Modeling View Optimize Help

Share

Cut Copy Format painter

Paste Get data Data workbook catalog OneLake SQL Enter Server Data Transform Refresh New Text More New visual New measure Quick Sensitivity Publish Copilot

Clipboard Data Queries Insert Calculations Sensitivity Share Copilot

UK Train

Bookings by Days in Advance

Days in Advance	Bookings
0	14K
1	OK
2	OK
3	OK
4	OK
5	OK
6	OK
7	OK
8	OK
9	OK
10	OK
11	OK
12	OK
13	OK
14	OK
15	OK
16	OK
17	OK
18	OK
19	OK
20	OK
21	OK
22	OK
23	OK
24	OK
25	OK
26	OK
27	OK
28	OK
29	OK
30	OK

Trips, Delayed Trips and Delay Percentage by Ticket Type

Ticket Type	Total Trips	Delayed Trips	Delay Percentage
Advance	18K	1K	6.9%
Off-Peak	9K	1K	6.8%
Anytime	5K	0K	9.1%

Avg price by Ticket Type

Ticket Type	Avg Price
Advance	17
Anytime	39
Off-Peak	25

Trips and Delayed Trips by Days in Advance

Days in Advance	Trips	Delayed Trips
0	100%	0%
1	95%	5%
2	90%	10%
3	85%	15%
4	80%	20%
5	75%	25%
6	70%	30%
7	65%	35%
8	60%	40%
9	55%	45%
10	50%	50%
11	45%	55%
12	40%	60%
13	35%	65%
14	30%	70%
15	25%	75%
16	20%	80%
17	15%	85%
18	10%	90%
19	5%	95%
20	0%	100%

Trips and Delayed Trips by Time Slot

Time Slot	Total Trips	Delayed Trips
Morning	11.0K	1.3K
Afternoon	9.3K	0.7K
Evening	6.6K	0.0K
Night	4.7K	0.2K

Trips, Delayed Trips and Delay Percentage by Departure Hour

Departure Hour	Total Trips	Delayed Trips	Delay Percentage
0	2K	0K	0%
1	2K	0K	0%
2	2K	0K	0%
3	2K	0K	0%
4	3K	1K	33%
5	3K	1K	33%
6	3K	1K	33%
7	3K	1K	33%
8	3K	1K	33%
9	2K	0K	0%
10	2K	0K	0%
11	2K	0K	0%
12	2K	0K	0%
13	26K	1K	4%
14	2K	0K	0%
15	2K	0K	0%
16	2K	0K	0%
17	2K	0K	0%
18	13K	0K	0%
19	2K	0K	0%
20	2K	0K	0%
21	2K	0K	0%
22	2K	0K	0%
23	2K	0K	0%
24	2K	0K	0%

UK Train Overview Delay Analysis Payment & Ticket Analysis Time-Based Analysis Advanced insights 1 Advanced insights 2 Advanced Insights Advanced insights 2 +

Page 6 of 7 Storage Mode: Mixed 84%

رؤى متقدمة 2 | Advanced Insights 2

1- نسبة التأخير حسب نوع التذكرة

• تذكرة Anytime هي الأعلى من حيث نسبة التأخير، رغم أن عدد رحلاتها أقل.

2- تحليل حسب المحطة (Transactions, Refunds & Rates by Station)

- أكثر المحطات من حيث المعاملات:
 - London Euston: 7 . 7K ° مع نسبة استرداد منخفضة.
 - London St Pancras: 3 . 9K °
 - Birmingham New Street: 5 . 7K °
- لها أعلى نسبة استرداد (100%) رغم قلة عدد المعاملات.
 - Edinburgh Waverley °
- أعلى عدد استردادات فعليًا كان في:
 - Liverpool Lime Street: 0 . 3K ° من أصل 4 . 6K
 - Manchester Piccadilly: 0 . 2K ° من أصل 5 . 7K

UK Train Rides GROUP 2 • Last saved: Today at 11:31 AM

File Home Insert Modeling View Optimize Help

Clipboard Data Queries Insert Calculations Sensitivity Share Copilot

Ticket Type Total Trips Delayed Trips Delay Percentage

Ticket Type	Total Trips	Delayed Trips	Delay Percentage
Advance	17561	1211	6.90
Anytime	5340	486	9.10
Off-Peak	8752	595	6.80

Transactions, Refund Count and Refund Rate by station

Refund count, Transactions and Refund rate by Station

arrival destination departure station Sum of Total transactions

arrival destination	departure station	Sum of Total transactions
Birmingham New Street	Liverpool Lime Street	14
Birmingham New Street	London Euston	4209
Birmingham New Street	London St Pancras	3471
Birmingham New Street	Reading	32
Birmingham New Street	York	16
Bristol Temple Meads	Oxford	144
Cardiff Central	Bristol Temple Meads	16
Coventry	Liverpool Lime Street	65
Crewe	Liverpool Lime Street	193
Didcot	Reading	48
Doncaster	York	211
Dunham	York	258
Edinburgh	Birmingham New Street	16
Edinburgh Waverley	London Kings Cross	163
Edinburgh Waverley	York	15
Jets	Liverpool Lime Street	96
Total		31653

UK Train Overview Delay Analysis Payment & Ticket Analysis Time-Based Analysis Advanced insights 1 Advanced insights 2 Advanced insights 2

Page 7 of 7 Storage Mode: Mixed 84%