UK TRAIN RIDES PROJECT Week 1

Build data model, Data cleaning and preprocessing

Comprehensive Report on the Work Submitted for Week One

1. Overview

The work submitted for the first week consists of :

- Python code in a Jupyter Notebook for processing and cleaning the "railway.csv" dataset related to UK train journeys.
- A set of SQL queries to analyze data stored in the "railway" database.
- Exploratory Data Analysis (EDA) performed in Excel, supplemented with Python analysis.

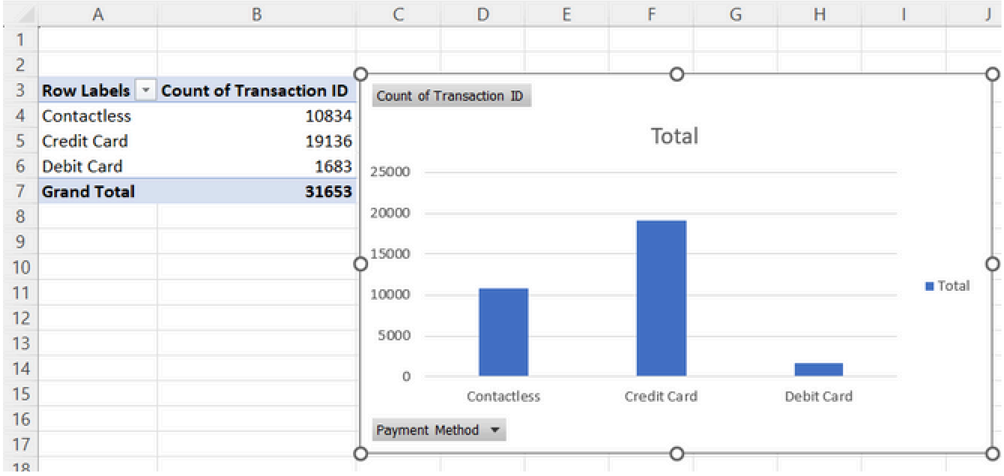
The objective is to clean the raw data, perform initial exploratory analysis, and extract analytical insights using Excel, Python, and SQL to understand patterns in train journeys, pricing, delays, and customer preferences.

1. Exploratory Data Analysis (EDA) Using Excel

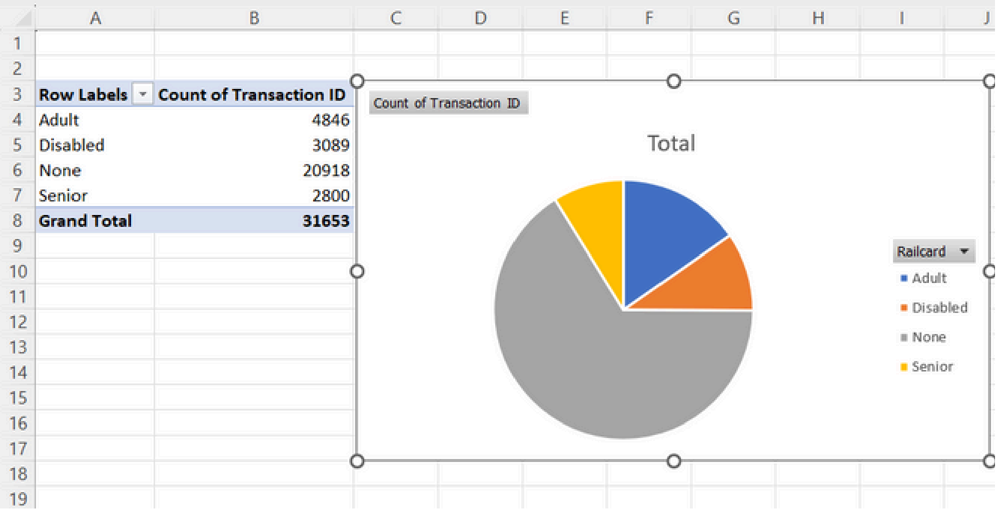
by Basem Fady

A. Visualizations and Insights

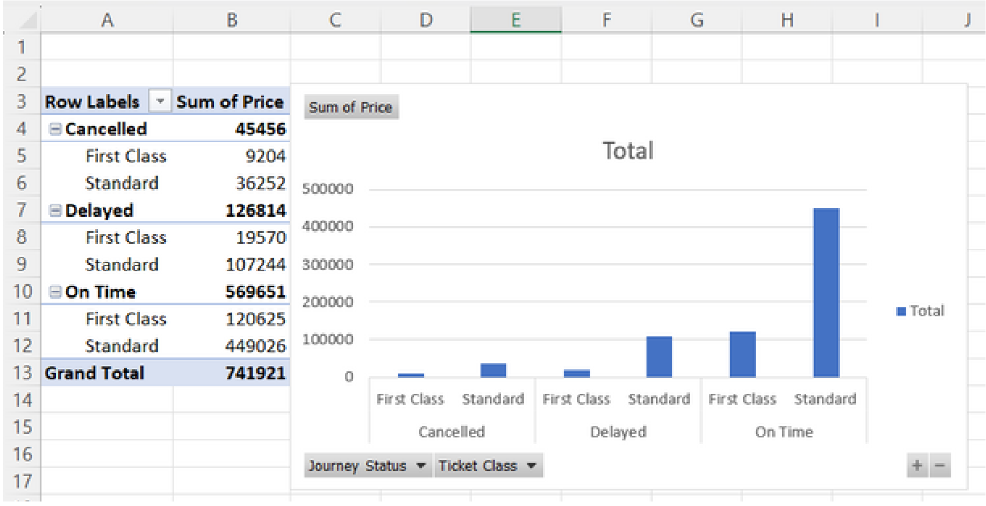
The EDA in Excel utilized Pivot Tables and Charts to summarize key aspects of the dataset. The following visualizations were created:



Row Labels	Count of Transaction ID
Contactless	10834
Credit Card	19136
Debit Card	1683
Grand Total	**31653**



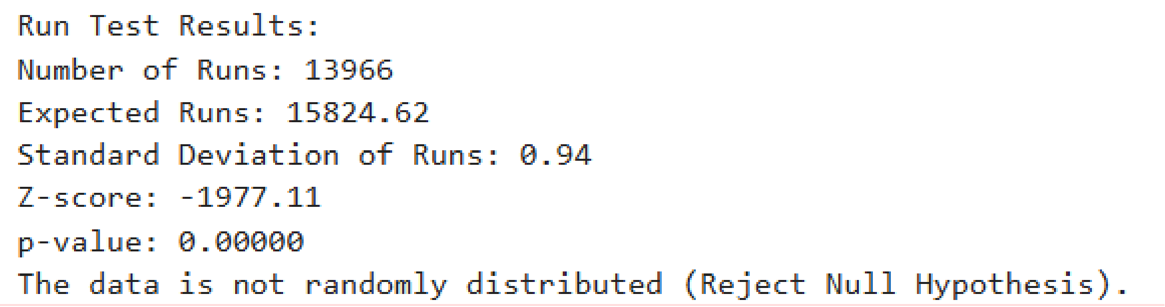
Row Labels	Count of Transaction ID
Adult	4846
Disabled	3089
None	20918
Senior	2800
Grand Total	**31653**



Row Labels	Sum of Price
⊟ Cancelled	**45456**
First Class	9204
Standard	36252
⊟ Delayed	**126814**
First Class	19570
Standard	107244
⊟ On Time	**569651**
First Class	120625
Standard	449026
Grand Total	**741921**

Run-Test Algorithm and Results

by Basem Fady



```
Run Test Results:
Number of Runs: 13966
Expected Runs: 15824.62
Standard Deviation of Runs: 0.94
Z-score: -1977.11
p-value: 0.00000
The data is not randomly distributed (Reject Null Hypothesis).
```

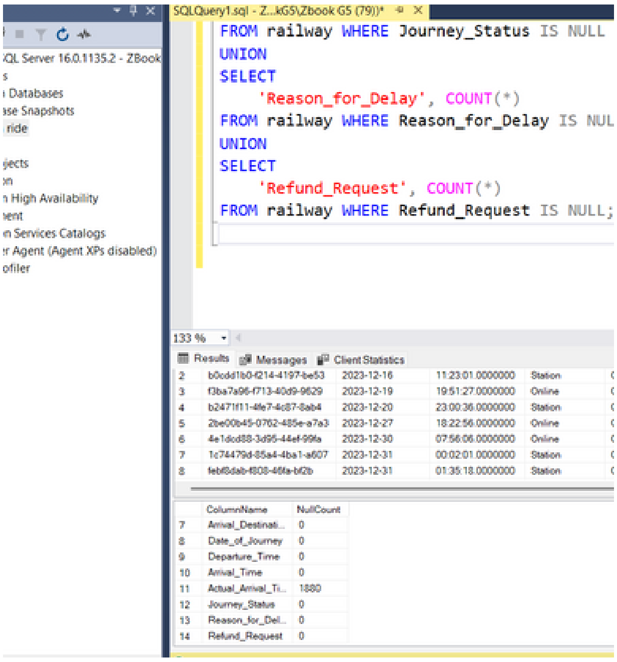
Conclusion: The price values in the dataset are not randomly distributed, suggesting the presence of a pattern in ticket prices.

2. Cleaning & Analysis Using SQL

by Mariam Ahmed and Alaa Ebrahim.

1. Checking for Null Values

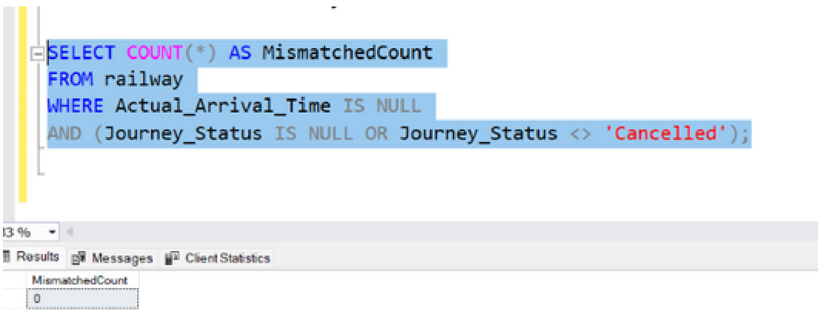
- Checked for missing values in multiple columns such as Payment Method, Ticket Type, Price, Stations, Journey Date, Departure & Arrival Times, Journey Status, and Reason for Delay.
- Performed an additional check to identify null values in Actual_Arrival_Time when the journey status is not "Cancelled".



```sql
FROM railway WHERE Journey_Status IS NULL
UNION
SELECT
    'Reason_for_Delay', COUNT(*)
FROM railway WHERE Reason_for_Delay IS NUL
UNION
SELECT
    'Refund_Request', COUNT(*)
FROM railway WHERE Refund_Request IS NULL;
```

```
2   b0cdd1b0-f214-4197-be53   2023-12-16   11:23:01.0000000   Station
3   f3ba7a96-f713-40d9-9629   2023-12-19   19:51:27.0000000   Online
4   b2471f11-4fe7-4c87-8ab4   2023-12-20   23:00:36.0000000   Station
5   2be00b45-0762-485e-a7a3   2023-12-27   18:22:56.0000000   Online
6   4e1dcd88-3d95-44ef-99fa   2023-12-30   07:56:06.0000000   Online
7   1c74479d-85a4-4ba1-a607   2023-12-31   00:02:01.0000000   Station
8   febf8dab-f808-46fa-bf2b   2023-12-31   01:35:18.0000000   Station

    ColumnName          NullCount
7   Arrival_Destinati...  0
8   Date_of_Journey       0
9   Departure_Time        0
10  Arrival_Time          0
11  Actual_Arrival_Ti...  1880
12  Journey_Status        0
13  Reason_for_Del...     0
14  Refund_Request        0
```

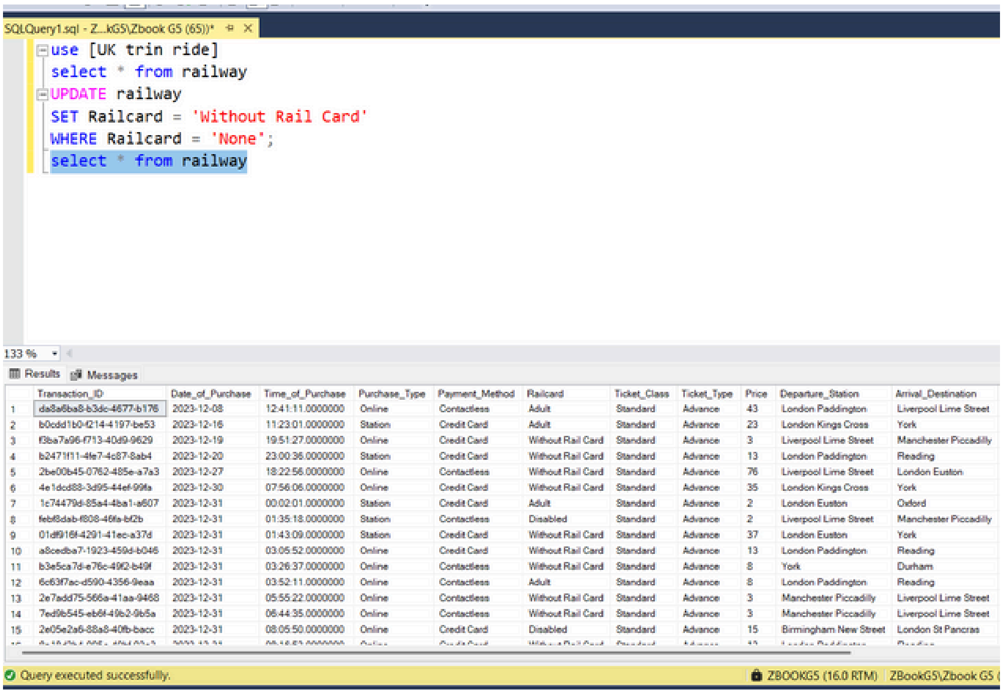


```sql
SELECT COUNT(*) AS MismatchedCount
FROM railway
WHERE Actual_Arrival_Time IS NULL
AND (Journey_Status IS NULL OR Journey_Status <> 'Cancelled');
```

```
MismatchedCount
0
```

2. Handling Null Values

- Reason_for_Delay: Replaced null values with "No Delay".
- Railcard: Replaced "None" with "Without Rail Card".
- Actual_Arrival_Time: Set null values to "1900-01-01 00:00:00" for cancelled journeys only.



```sql
use [UK trin ride]
select * from railway
UPDATE railway
SET Railcard = 'Without Rail Card'
WHERE Railcard = 'None';
select * from railway
```

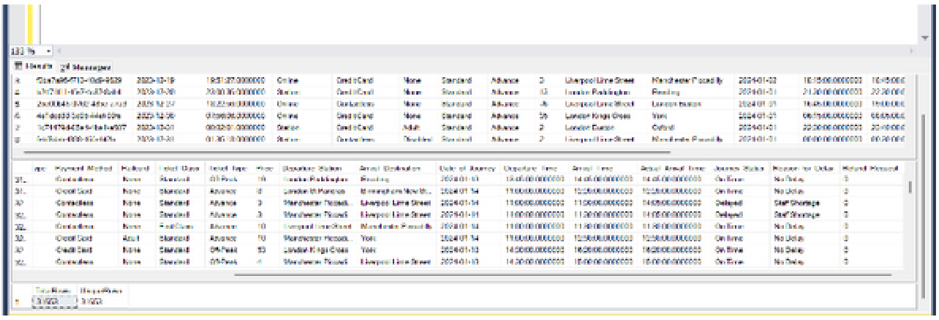
	Transaction_ID	Date_of_Purchase	Time_of_Purchase	Purchase_Type	Payment_Method	Railcard	Ticket_Class	Ticket_Type	Price	Departure_Station	Arrival_Destination
1	da8a6ba8-b3dc-4677-b176	2023-12-08	12:41:11.0000000	Online	Contactless	Adult	Standard	Advance	43	London Paddington	Liverpool Lime Street
2	b0cdd1b0-f214-4197-be53	2023-12-16	11:23:01.0000000	Station	Credit Card	Adult	Standard	Advance	23	London Kings Cross	York
3	f3ba7a96-f713-40d9-9629	2023-12-19	19:51:27.0000000	Online	Credit Card	Without Rail Card	Standard	Advance	3	Liverpool Lime Street	Manchester Piccadilly
4	b2471f11-4fe7-4c87-8ab4	2023-12-20	23:00:36.0000000	Station	Credit Card	Without Rail Card	Standard	Advance	13	London Paddington	Reading
5	2be00b45-0762-485e-a7a3	2023-12-27	18:22:56.0000000	Online	Contactless	Without Rail Card	Standard	Advance	76	Liverpool Lime Street	London Euston
6	4e1dcd88-3d95-44ef-99fa	2023-12-30	07:56:06.0000000	Online	Credit Card	Without Rail Card	Standard	Advance	35	London Kings Cross	York
7	1c74479d-85a4-4ba1-a607	2023-12-31	00:02:01.0000000	Station	Credit Card	Adult	Standard	Advance	2	London Euston	Oxford
8	feb6dab-f808-46fa-bf2b	2023-12-31	01:35:18.0000000	Station	Contactless	Disabled	Standard	Advance	2	Liverpool Lime Street	Manchester Piccadilly
9	01df916f-4291-41ec-a37d	2023-12-31	01:43:09.0000000	Station	Credit Card	Without Rail Card	Standard	Advance	37	London Euston	York
10	a8cedba7-1923-459d-b046	2023-12-31	03:05:52.0000000	Online	Credit Card	Without Rail Card	Standard	Advance	13	London Paddington	Reading
11	b3e5ca7d-e76c-49f2-b49f	2023-12-31	03:26:37.0000000	Online	Contactless	Without Rail Card	Standard	Advance	8	York	Durham
12	6c63f7ac-d590-4356-9eaa	2023-12-31	03:52:11.0000000	Online	Contactless	Adult	Standard	Advance	8	London Paddington	Reading
13	2e7add75-566a-41aa-9468	2023-12-31	05:55:22.0000000	Online	Contactless	Without Rail Card	Standard	Advance	3	Manchester Piccadilly	Liverpool Lime Street
14	7ed9b545-eb6f-49b2-9b5a	2023-12-31	06:44:35.0000000	Online	Contactless	Without Rail Card	Standard	Advance	3	Manchester Piccadilly	Liverpool Lime Street
15	2e05e2a6-88a8-40fb-bacc	2023-12-31	08:05:50.0000000	Online	Credit Card	Disabled	Standard	Advance	15	Birmingham New Street	London St Pancras

Query executed successfully.

ZBOOKG5 (16.0 RTM) | ZBookG5\Zbook G5 (

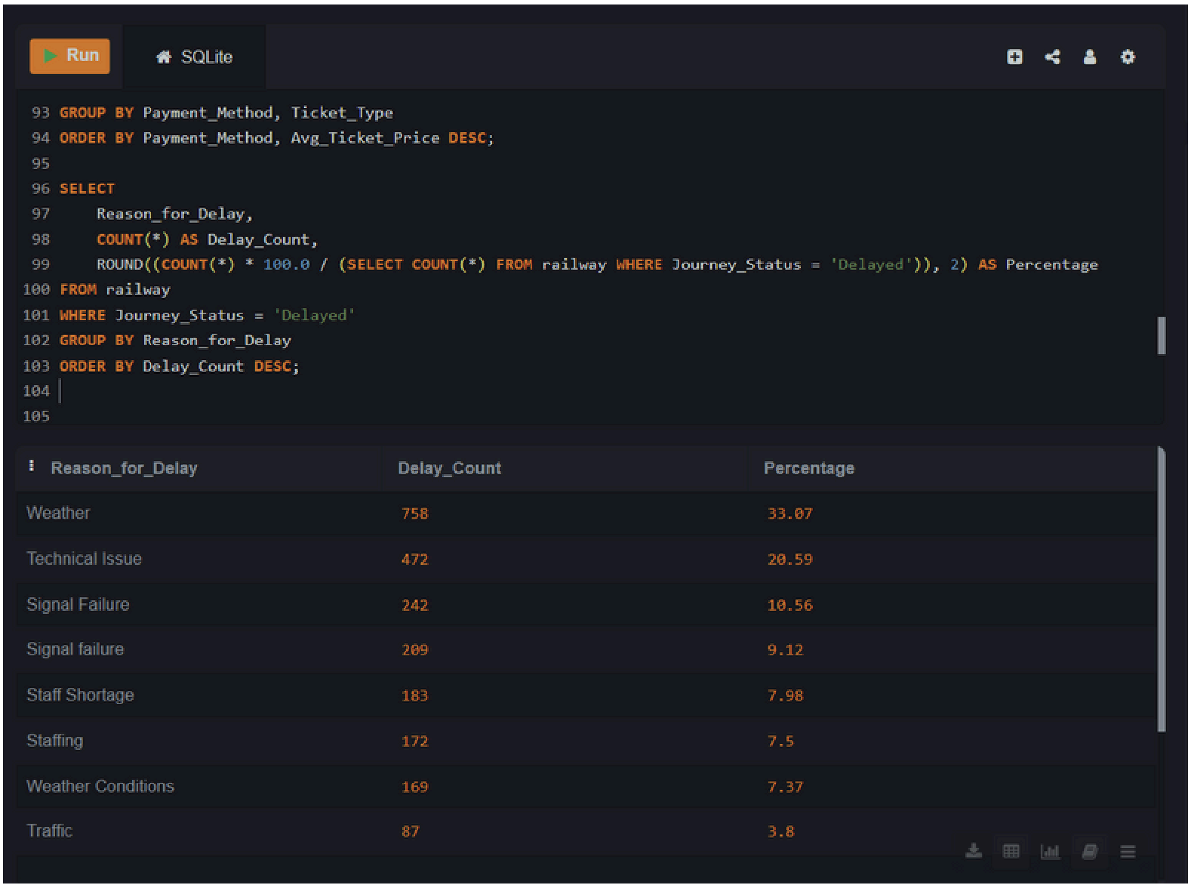
3. Checking for Duplicates

- Compared the total row count with the number of distinct Transaction_ID values to detect potential duplicates.



	TotalRows	UniqueRows
1	31653	31653

1. Average Price by Purchase Type
   - Calculates the average ticket price based on the Purchase_Type.
   - Orders the results in ascending order of price.
2. Ticket Price & Count by Journey Status
   - Computes the average ticket price and total number of tickets for journeys that were either Delayed or On Time.
   - Results are sorted in descending order by average ticket price.
3. Ticket Count by Payment Method & Ticket Type
   - Counts the number of tickets sold based on Payment_Method and Ticket_Type.
   - Orders the results by Payment_Method and ticket count (descending order).
4. Journey Frequency by Departure & Arrival Stations
   - Counts the total number of journeys between each departure and arrival station.
   - Orders the results by the most frequent journeys.
5. Departures & Arrivals Count
   - Counts the total number of departures per station.
   - Counts the total number of arrivals per destination.
   - Orders the results by the most frequent departure and arrival locations.



```
GROUP BY Payment_Method, Ticket_Type
ORDER BY Payment_Method, Avg_Ticket_Price DESC;

SELECT
    Reason_for_Delay,
    COUNT(*) AS Delay_Count,
    ROUND((COUNT(*) * 100.0 / (SELECT COUNT(*) FROM railway WHERE Journey_Status = 'Delayed')), 2) AS Percentage
FROM railway
WHERE Journey_Status = 'Delayed'
GROUP BY Reason_for_Delay
ORDER BY Delay_Count DESC;

```

Reason_for_Delay	Delay_Count	Percentage
Weather	758	33.07
Technical Issue	472	20.59
Signal Failure	242	10.56
Signal failure	209	9.12
Staff Shortage	183	7.98
Staffing	172	7.5
Weather Conditions	169	7.37
Traffic	87	3.8

3. Data Processing Using Python

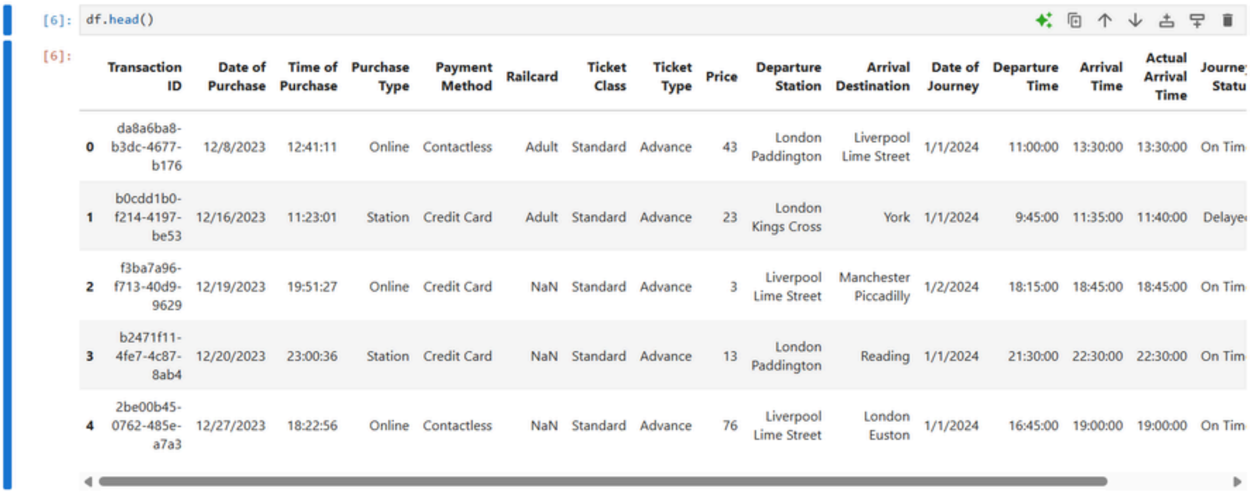
by Alaa Ebrahim, Saif AboElmagd and Basem Fady.

Library Imports:

- Imported pandas, numpy, matplotlib.pyplot, and seaborn for data manipulation and visualization.

Data Loading:

- Loaded "railway.csv" into a DataFrame using pd.read_csv().



```
[6]: df.head()
```

[6]:

	Transaction ID	Date of Purchase	Time of Purchase	Purchase Type	Payment Method	Railcard	Ticket Class	Ticket Type	Price	Departure Station	Arrival Destination	Date of Journey	Departure Time	Arrival Time	Actual Arrival Time	Journe Statu
0	da8a6ba8-b3dc-4677-b176	12/8/2023	12:41:11	Online	Contactless	Adult	Standard	Advance	43	London Paddington	Liverpool Lime Street	1/1/2024	11:00:00	13:30:00	13:30:00	On Tim
1	b0cdd1b0-f214-4197-be53	12/16/2023	11:23:01	Station	Credit Card	Adult	Standard	Advance	23	London Kings Cross	York	1/1/2024	9:45:00	11:35:00	11:40:00	Delaye
2	f3ba7a96-f713-40d9-9629	12/19/2023	19:51:27	Online	Credit Card	NaN	Standard	Advance	3	Liverpool Lime Street	Manchester Piccadilly	1/2/2024	18:15:00	18:45:00	18:45:00	On Tim
3	b2471f11-4fe7-4c87-8ab4	12/20/2023	23:00:36	Station	Credit Card	NaN	Standard	Advance	13	London Paddington	Reading	1/1/2024	21:30:00	22:30:00	22:30:00	On Tim
4	2be00b45-0762-485e-a7a3	12/27/2023	18:22:56	Online	Contactless	NaN	Standard	Advance	76	Liverpool Lime Street	London Euston	1/1/2024	16:45:00	19:00:00	19:00:00	On Tim

Initial Data Exploration

- Displayed the first few rows using df.head().
- Checked dataset structure with df.info() and missing values using df.isnull().sum().
- The dataset has 31,653 rows and 18 columns.

Checking for Duplicates

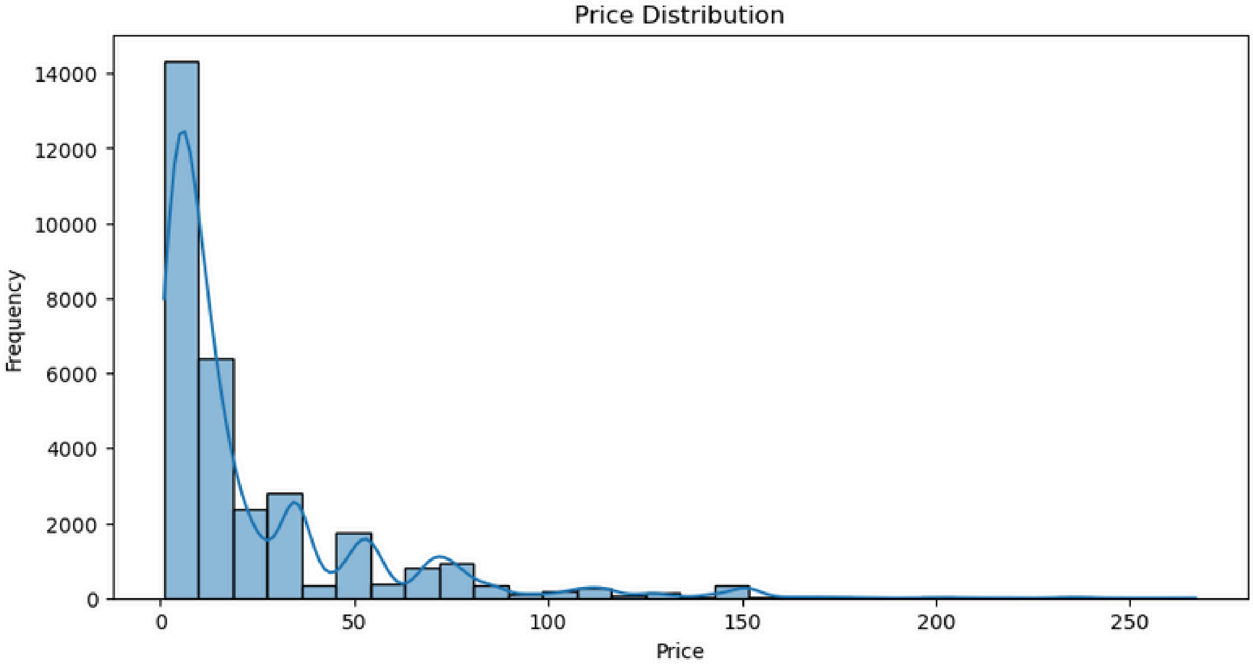
- Verified if duplicate rows exist using df.duplicated().sum(), and confirmed there were no duplicates.

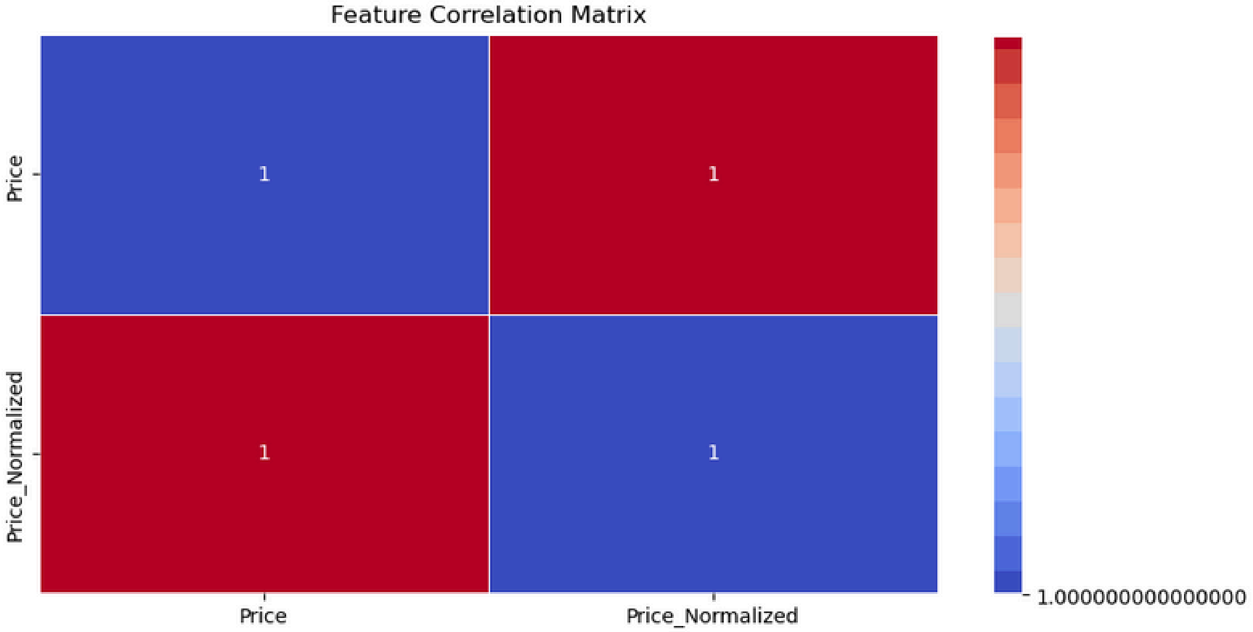
Handling Missing Values

- The notebook appears to include steps to manage missing data, but further details would be in later cells.

Normalization

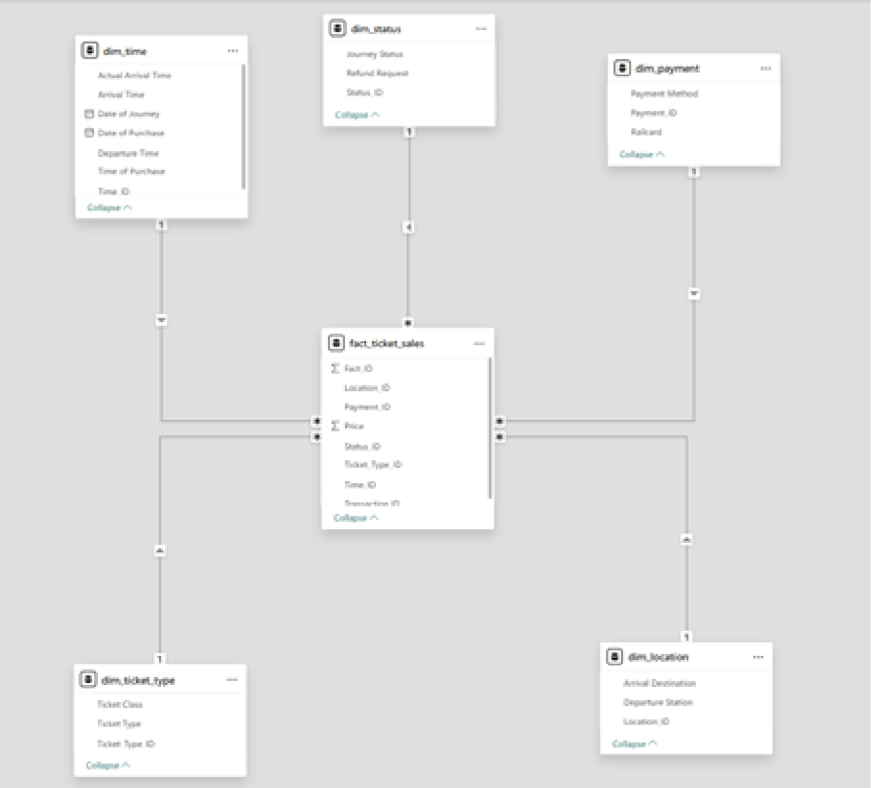
- Min-Max Scaling for normalization using MinMaxScaler from sklearn.preprocessing:


Price Distribution
Frequency
14000
12000
10000
8000
6000
4000
2000
0
0
50
100
150
200
250
Price


Feature Correlation Matrix
Price
Price_Normalized
1
1
1
1
Price
Price_Normalized
1.000000000000000

4. Data Modeling by Power BI

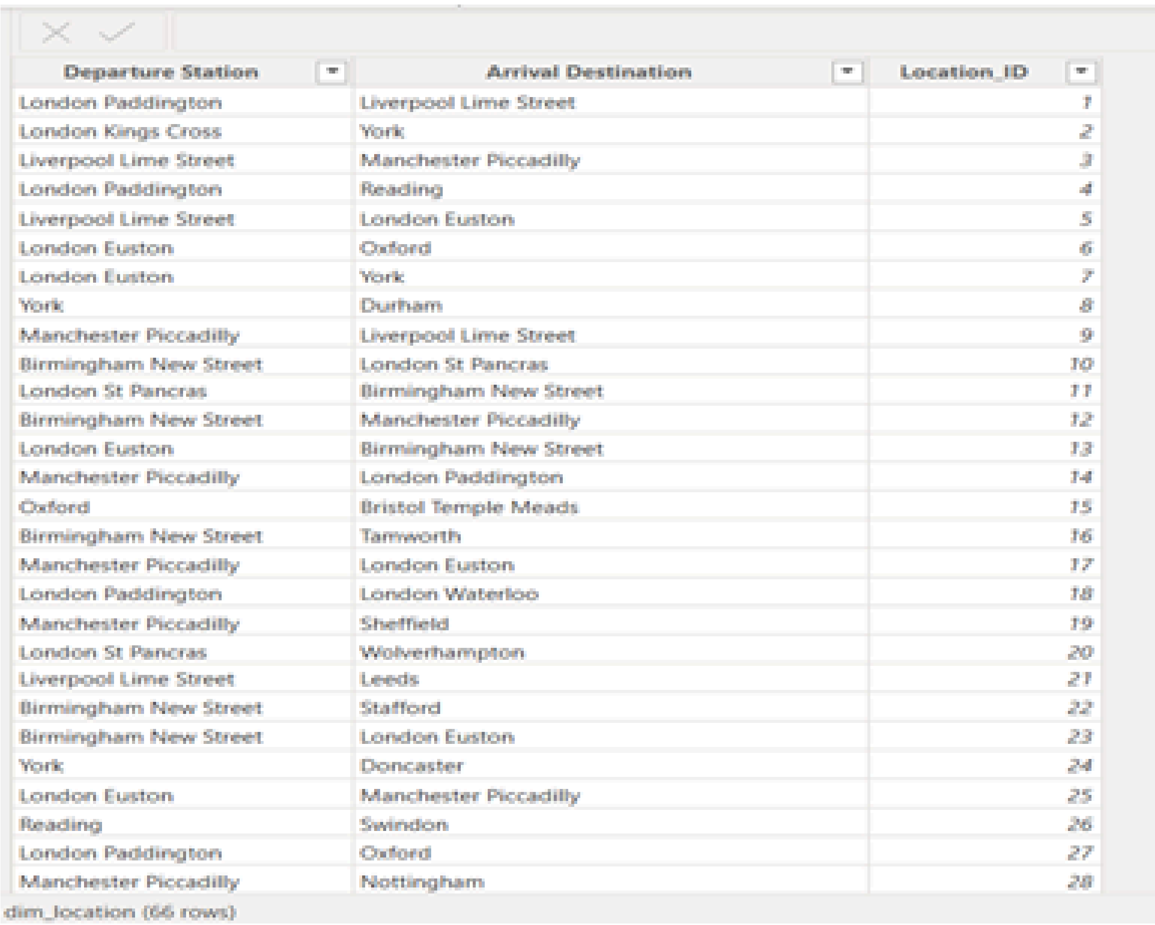
by Alaa Ebrahim.



To analyze UK train data efficiently, we used Power BI's data modeling to structure and relate datasets. The model follows a star schema, linking key tables:

- dim_location (stations with Location_ID)
- dim_ticket_type (ticket categories)
- fact_journey (train journeys)
- dim_payment (payment methods)
- dim_date (time dimension)

This structure ensures data integrity, optimized queries, and efficient visualization. The example table shows station mappings, enabling seamless analysis of travel patterns and station usage.



Departure Station	Arrival Destination	Location_ID
London Paddington	Liverpool Lime Street	1
London Kings Cross	York	2
Liverpool Lime Street	Manchester Piccadilly	3
London Paddington	Reading	4
Liverpool Lime Street	London Euston	5
London Euston	Oxford	6
London Euston	York	7
York	Durham	8
Manchester Piccadilly	Liverpool Lime Street	9
Birmingham New Street	London St Pancras	10
London St Pancras	Birmingham New Street	11
Birmingham New Street	Manchester Piccadilly	12
London Euston	Birmingham New Street	13
Manchester Piccadilly	London Paddington	14
Oxford	Bristol Temple Meads	15
Birmingham New Street	Tamworth	16
Manchester Piccadilly	London Euston	17
London Paddington	London Waterloo	18
Manchester Piccadilly	Sheffield	19
London St Pancras	Wolverhampton	20
Liverpool Lime Street	Leeds	21
Birmingham New Street	Stafford	22
Birmingham New Street	London Euston	23
York	Doncaster	24
London Euston	Manchester Piccadilly	25
Reading	Swindon	26
London Paddington	Oxford	27
Manchester Piccadilly	Nottingham	28

dim_location (66 rows)