I. Cleaning the data set

At first, we tried to make do with the given data set without any modifications. However, due to the different date formats that were found in the data sets there were discrepancies that had to be addressed. So, we decided to use a simple python script to make all the months of the data sets uniform. We followed the format of the January and February data sets and applied the same format for every other data set. The changes were made in DATE_BOOKED columns and DEPARTURE_DATE columns. This is the script that we used with Python:

```
import pandas as pd

# Function to convert D/M/YYYY to M/D/YYYY (Swap day and month)

def swap_day_month(date_str):

  try:

    # Split the date by '/'

    day, month, year = date_str.split('/')


    # Return in M/D/YYYY format

    return f"{month}/{day}/{year}"

  except Exception as e:

    return None  # Return None if there is any issue with the date format


# Path for the October CSV file

input_file_path = r"C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\January-2024.csv"

output_file_path = r"C:\Users\labva\OneDrive\Desktop\DE Simulation\January-2024_clean.csv"


# Read the October CSV file into a DataFrame
```

```
df = pd.read_csv(input_file_path)


# Check if 'DATE_BOOKED' and 'DEPARTURE_DATE' columns exist and apply the conversion
if 'DateBooked' in df.columns:
    # Apply the swap function to the 'DATE_BOOKED' column
    df['DateBooked'] = df['DateBooked'].apply(swap_day_month)

if 'DepartureDate' in df.columns:
    # Apply the swap function to the 'DEPARTURE_DATE' column
    df['DepartureDate'] = df['DepartureDate'].apply(swap_day_month)

# Save the cleaned data into the output folder
df.to_csv(output_file_path, index=False)

print(f"Cleaned file saved at: {output_file_path}")
```

A. Creating the tables

We need to create the necessary tables to load our data into and assign the proper data type. This SQL script features a DROP IF TABLE EXISTS so it can be executed without any problem by our automated batch file ran by windows task scheduler later. This is the SQL script:

```
DROP TABLE IF EXISTS source_table;
DROP TABLE IF EXISTS staging_table;
DROP TABLE IF EXISTS analytical_business_table;

CREATE TABLE source_table (
    DATE_BOOKED VARCHAR(255),
```

```
    ORIGIN VARCHAR(255),
    DESTINATION VARCHAR(255),
    ORDER_REF VARCHAR(255),
    TICKET_NO VARCHAR(255),
    SEATNO VARCHAR(255),
    DATE_REDEEMED VARCHAR(255),
    EMAIL VARCHAR(255),
    MOBILENO VARCHAR(255),
    FARE VARCHAR(255),
    CONVENIENCE_FEE VARCHAR(255),
    DISCOUNT VARCHAR(255),
    DEPARTURE_DATE VARCHAR(255),
    DEPARTURE_TIME VARCHAR(255),
    BUS_TYPE VARCHAR(255),
    NUMBER_OF_VOUCHERS_BOOKED VARCHAR(255)
);

CREATE TABLE Analytical_Business_Table (
    DATE_BOOKED VARCHAR(255),
    ORIGIN VARCHAR(255),
    DESTINATION VARCHAR(255),
    ORDER_REF VARCHAR(255),
    TICKET_NO VARCHAR(255),
    SEATNO DECIMAL(10, 2),
    DATE_REDEEMED VARCHAR(255),
    EMAIL VARCHAR(255),
```

```
    MOBILENO VARCHAR(255),
    FARE DECIMAL(10, 2),
    CONVENIENCE_FEE DECIMAL(10, 2),
    DISCOUNT VARCHAR(255),
    DEPARTURE_DATE VARCHAR(255),
    DEPARTURE_TIME TIME,
    BUS_TYPE VARCHAR(255),
    NUMBER_OF_VOUCHERS_BOOKED VARCHAR(255),
    BOOK_DT DATE,
    BOOK_TM TIME,
    DEPARTURE_DAY VARCHAR(255),
    BOOKING_INFO VARCHAR(255),
    VOUCHER VARCHAR(255),
    ROUTE VARCHAR(255),
    TRIP_ID VARCHAR(255),
    TRIP_ID_DAY VARCHAR(255),
    REDEEMED_FLAG VARCHAR(255)
);

CREATE TABLE staging_table (
    DATE_BOOKED VARCHAR(255),
    ORIGIN VARCHAR(255),
    DESTINATION VARCHAR(255),
    ORDER_REF VARCHAR(255),
    TICKET_NO VARCHAR(255),
    SEATNO DECIMAL(10, 2),
```

```
  DATE_REDEEMED VARCHAR(255),
  EMAIL VARCHAR(255),
  MOBILENO VARCHAR(255),
  FARE DECIMAL(10, 2),
  CONVENIENCE_FEE DECIMAL(10, 2),
  DISCOUNT VARCHAR(255),
  DEPARTURE_DATE VARCHAR(255),
  DEPARTURE_TIME TIME,
  BUS_TYPE VARCHAR(255),
  NUMBER_OF_VOUCHERS_BOOKED VARCHAR(255),
  BOOK_DT DATE,
  BOOK_TM TIME,
  DEPARTURE_DAY VARCHAR(255),
  BOOKING_INFO VARCHAR(255),
  VOUCHER VARCHAR(255),
  ROUTE VARCHAR(255),
  TRIP_ID VARCHAR(255),
  TRIP_ID_DAY VARCHAR(255),
  REDEEMED_FLAG VARCHAR(255)
);
```

B. Loading the data into MySQL

In order to import the data into the MySQL workbench we made a SQL script to load the data into their respective table and columns. This is the SQL query script:

```
LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\Data\\January-2023.csv'
```

```
INTO TABLE source_table
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\Data\\February-2023.csv'
INTO TABLE source_table
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\Data\\March-2023_clean.csv'
INTO TABLE source_table
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\Data\\April-2023_clean.csv'
INTO TABLE source_table
FIELDS TERMINATED BY ','
```

```
ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;


LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\Data\\May-2023_clean.csv'

INTO TABLE source_table

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;


LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\Data\\June-2023_clean.csv'

INTO TABLE source_table

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;


LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\Data\\July-2023_clean.csv'

INTO TABLE source_table

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;
```

```
LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\Data\\August-2023_clean.csv'

INTO TABLE source_table

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;


LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\Data\\September-2023_clean.csv'

INTO TABLE source_table

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;



LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\Data\\October-2023_clean.csv'

INTO TABLE source_table

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;
```

```
LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\Data\\November-2023_clean.csv'
INTO TABLE source_table
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\\Data\December-2023_clean.csv'
INTO TABLE source_table
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

C. Transform Staging_table

This SQL script is used to change the data types of certain columns to appeal to their proper formats. This is the SQL script I used:

```
INSERT INTO staging_table (
  DATE_BOOKED, ORIGIN, DESTINATION, ORDER_REF, TICKET_NO, SEATNO,
  DATE_REDEEMED, EMAIL, MOBILENO, FARE, CONVENIENCE_FEE, DISCOUNT,
  DEPARTURE_DATE, DEPARTURE_TIME, BUS_TYPE, NUMBER_OF_VOUCHERS_BOOKED,
  BOOK_DT, BOOK_TM, DEPARTURE_DAY, BOOKING_INFO, VOUCHER, ROUTE,
  TRIP_ID, TRIP_ID_DAY, REDEEMED_FLAG
)
```

```
SELECT
        DATE_BOOKED,
  ORIGIN, DESTINATION, ORDER_REF, TICKET_NO, SEATNO,
  DATE_REDEEMED, EMAIL, MOBILENO,
  NULLIF(FARE, '') AS FARE,
  NULLIF(CONVENIENCE_FEE, '') AS CONVENIENCE_FEE,
  DISCOUNT,
        DEPARTURE_DATE,
-- Direct extraction of DEPARTURE_TIME from DEPARTURE_DATE
  CASE
    WHEN DEPARTURE_TIME LIKE '%PM' OR DEPARTURE_TIME LIKE '%AM'
      THEN TIME_FORMAT(STR_TO_DATE(DEPARTURE_TIME, '%r'), '%H:%i:%s')
    WHEN LENGTH(DEPARTURE_TIME) = 8 AND DEPARTURE_TIME LIKE '%:%:%'
      THEN TIME_FORMAT(STR_TO_DATE(DEPARTURE_TIME, '%H:%i:%s'), '%H:%i:%s')
    WHEN LENGTH(DEPARTURE_TIME) = 7 AND DEPARTURE_TIME LIKE '%:%:%'
      THEN TIME_FORMAT(STR_TO_DATE(DEPARTURE_TIME, '%H:%i:%s'), '%H:%i:%s')
    WHEN LENGTH(DEPARTURE_TIME) = 5 AND DEPARTURE_TIME LIKE '%:%'
      THEN TIME_FORMAT(STR_TO_DATE(CONCAT(DEPARTURE_TIME, ':00'), '%H:%i:%s'),
'%H:%i:%s')
    ELSE NULL
  END AS DEPARTURE_TIME,
  BUS_TYPE, NUMBER_OF_VOUCHERS_BOOKED,
  NULL AS BOOK_DT,
  NULL AS BOOK_TM,
  NULL AS DEPARTURE_DAY,
  NULL AS BOOKING_INFO,
```

```
    NULL AS VOUCHER,
    NULL AS ROUTE,
    NULL AS TRIP_ID,
    NULL AS TRIP_ID_DAY,
    NULL AS REDEEMED_FLAG
FROM source_table;
```

D. Transform Analytical_business_table

Now, after adding the columns in the staging table, we transform it again to apply any necessary calculations or formula that appeals to the business needs/requirements. This SQL script oversees making the format of the data uniform. This is the SQL script:

```
INSERT INTO Analytical_Business_Table
(
    DATE_BOOKED,
    ORIGIN,
    DESTINATION,
    ORDER_REF,
    TICKET_NO,
    SEATNO,
    DATE_REDEEMED,
    EMAIL,
    MOBILENO,
    FARE,
    CONVENIENCE_FEE,
    DISCOUNT,
```

```
    DEPARTURE_DATE,
    DEPARTURE_TIME,
    BUS_TYPE,
    NUMBER_OF_VOUCHERS_BOOKED,
    BOOK_DT,
    BOOK_TM,
    DEPARTURE_DAY,
    BOOKING_INFO,
    VOUCHER,
    ROUTE,
    TRIP_ID,
    TRIP_ID_DAY,
    REDEEMED_FLAG
)
SELECT
        DATE_BOOKED,
    ORIGIN,
    DESTINATION,
    ORDER_REF,
    TICKET_NO,
    SEATNO,
    DATE_REDEEMED,
    EMAIL,
    MOBILENO,
    FARE,
    CONVENIENCE_FEE,
```

```
  DISCOUNT,
        DEPARTURE_DATE,
  DEPARTURE_TIME,
  BUS_TYPE,
  NUMBER_OF_VOUCHERS_BOOKED,
  -- Extract only the date part from DATE_BOOKED
  DATE(
    CASE
      WHEN DATE_BOOKED REGEXP '^[0-9]{1,2}/[0-9]{1,2}/[0-9]{4} [0-9]{1,2}:[0-9]{1,2}$'
THEN
        STR_TO_DATE(DATE_BOOKED, '%m/%d/%Y %H:%i')
      WHEN DATE_BOOKED REGEXP '^[0-9]{1,2}/[0-9]{1,2}/[0-9]{4} [0-9]{1,2}:[0-9]{1,2}$'
THEN
        STR_TO_DATE(DATE_BOOKED, '%d/%m/%Y %H:%i')
      WHEN DATE_BOOKED REGEXP '^[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-9]{2}:[0-9]{2}$'
THEN
        STR_TO_DATE(DATE_BOOKED, '%Y-%m-%d %H:%i:%s')
      ELSE NULL
    END
  ) AS BOOK_DT,
  -- Extract only the time part
  TIME(
    CASE
      WHEN DATE_BOOKED REGEXP '^[0-9]{1,2}/[0-9]{1,2}/[0-9]{4} [0-9]{1,2}:[0-9]{1,2}$'
THEN
        STR_TO_DATE(DATE_BOOKED, '%m/%d/%Y %H:%i')
```

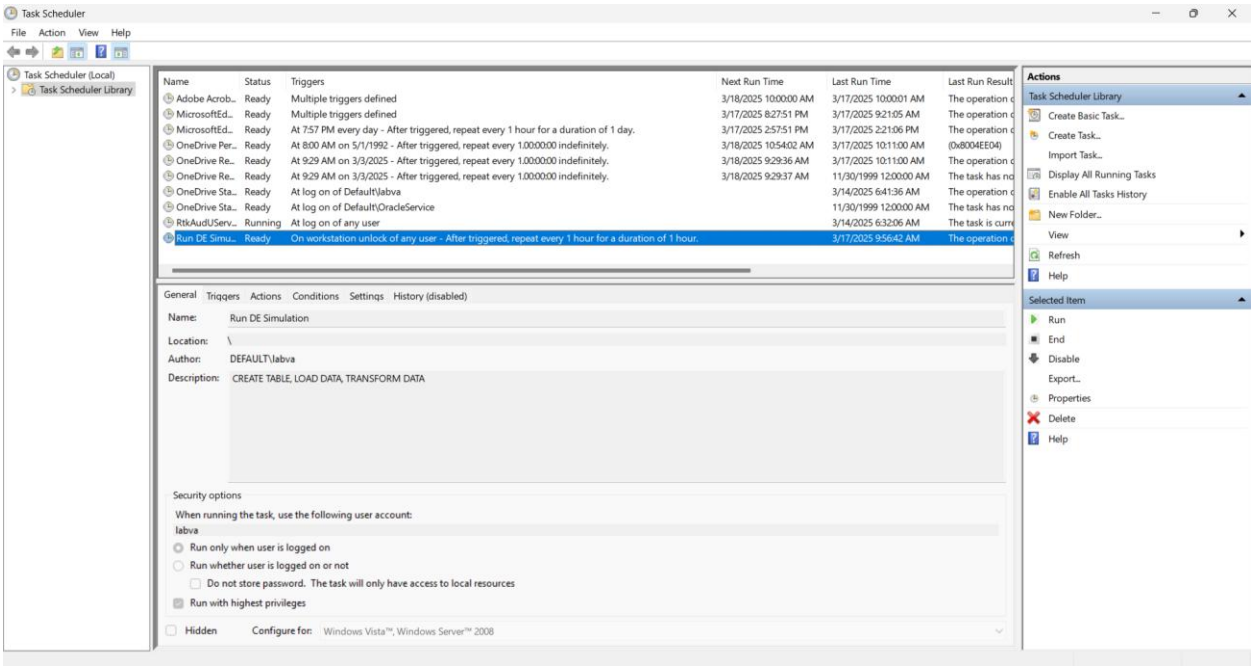
```
        WHEN DATE_BOOKED REGEXP '^[0-9]{1,2}/[0-9]{1,2}/[0-9]{4} [0-9]{1,2}:[0-9]{1,2}$'
THEN
          STR_TO_DATE(DATE_BOOKED, '%d/%m/%Y %H:%i')
        WHEN DATE_BOOKED REGEXP '^[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-9]{2}:[0-9]{2}$'
THEN
          STR_TO_DATE(DATE_BOOKED, '%Y-%m-%d %H:%i:%s')
        ELSE NULL
      END
  ) AS BOOK_TM,
  LEFT(DAYNAME(DEPARTURE_DATE), 3) AS DEPARTURE_DAY,
  CASE
    WHEN ORDER_REF LIKE '%OFF%' THEN 'Offline'
    ELSE 'Online'
  END AS BOOKING_INFO,
  CASE
    WHEN LEFT(NUMBER_OF_VOUCHERS_BOOKED, 1) = '1' THEN 'Individual'
    WHEN LEFT(NUMBER_OF_VOUCHERS_BOOKED, 1) = '2' THEN 'Pair'
    WHEN CAST(LEFT(NUMBER_OF_VOUCHERS_BOOKED, 1) AS UNSIGNED) >= 3 THEN
'Group'
    ELSE NULL
  END AS VOUCHER,
  CONCAT(ORIGIN, ' to ', DESTINATION) AS ROUTE,
  CONCAT(
    UPPER(LEFT(ORIGIN, 3)),
    '-',
    UPPER(LEFT(DESTINATION, 3)),
    '-',
```

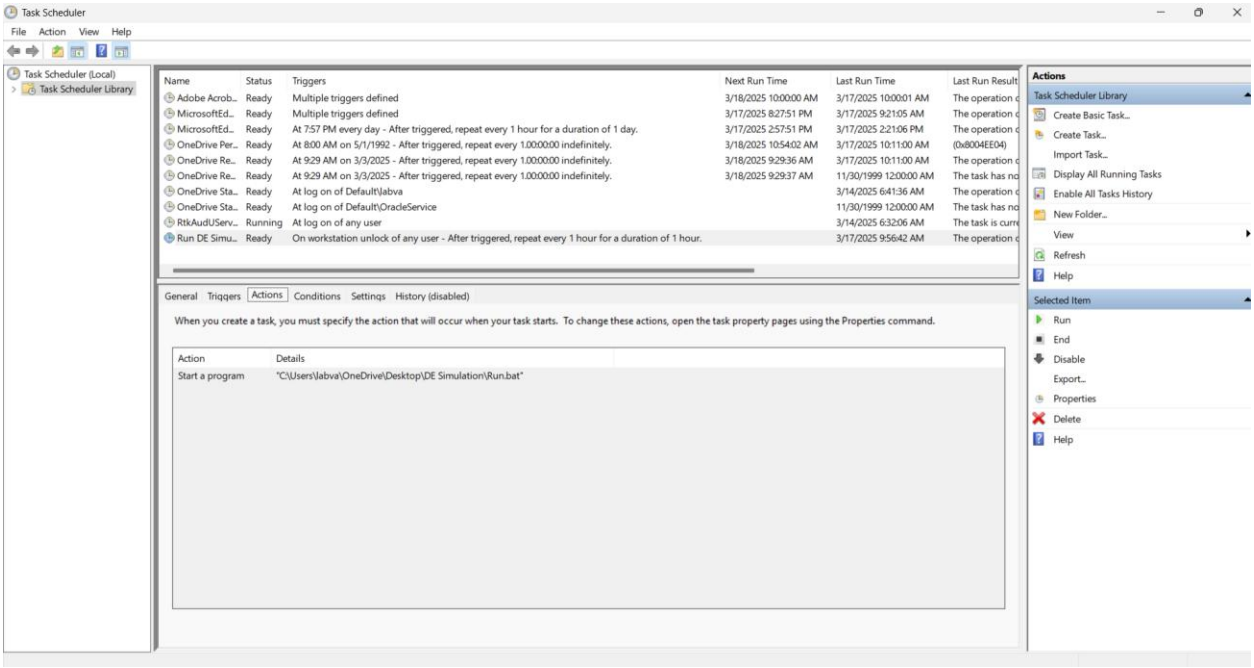
```
    UPPER(LEFT(DAYNAME(DEPARTURE_DATE), 3)),
    '-',
    IFNULL(LEFT(DEPARTURE_TIME, 2), '00')
  ) AS TRIP_ID,
  CONCAT(
    UPPER(LEFT(ORIGIN, 3)),
    '-',
    UPPER(LEFT(DESTINATION, 3)),
    '-',
    UPPER(LEFT(DAYNAME(DEPARTURE_DATE), 3))
  ) AS TRIP_ID_DAY,
  CASE
    WHEN DATE_REDEEMED IS NULL OR DATE_REDEEMED = 'Unredeemed' THEN 'N'
    ELSE 'Y'
  END AS REDEEMED_FLAG
FROM staging_table;
```

II. Automation – Windows Task Scheduler & Batch File

A. Windows Task Scheduler Configuration

Upon user logon, the windows task scheduler will automatically execute the batch file with admin privileges (required for MySQL). The batch file will be ran again after 1 hour and every time a user logs in.





B. Batch File

This batch file has the necessary user requirements to access MySQL. This batch file will run each query one by one. It will start with dropping existing tables, then creating new tables and populating them and, finally, transforming them according to the business requirements. Additionally, it provides a user interface to alert the user if it's done executing each task.

```
@echo off

set MYSQL_CMD="C:\Program Files\MySQL\MySQL Server 8.0\bin\mysql.exe"

set USER="root"

set PASSWORD="Jian1234"

set DATABASE="simul"


echo Running CREATE_TABLES.sql...

%MYSQL_CMD% -u %USER% -p%PASSWORD% %DATABASE% <
"C:\Users\labva\OneDrive\Desktop\DE Simulation\CREATE_TABLES.sql" >> "C:\Program
Files\MySQL\MySQL Server 8.0\bin\log.txt" 2>&1

IF %ERRORLEVEL% NEQ 0 (

   echo Error executing CREATE_TABLES.sql. Exiting.

   pause

   exit /b

)


echo Running LOAD_MONTHS.sql...

%MYSQL_CMD% -u %USER% -p%PASSWORD% %DATABASE% <
"C:\Users\labva\OneDrive\Desktop\DE Simulation\LOAD_MONTHS.sql" >> "C:\Program
Files\MySQL\MySQL Server 8.0\bin\log.txt" 2>&1

IF %ERRORLEVEL% NEQ 0 (

   echo Error executing LOAD_MONTHS.sql. Exiting.

   pause

   exit /b

)


echo Running TRANSFORM_STAGINGTABLE.sql...
```

```
%MYSQL_CMD% -u %USER% -p%PASSWORD% %DATABASE% < "C:\Users\labva\OneDrive\Desktop\DE Simulation\TRANSFORM_STAGINGTABLE.sql" >> "C:\Program Files\MySQL\MySQL Server 8.0\bin\log.txt" 2>&1

IF %ERRORLEVEL% NEQ 0 (
   echo Error executing TRANSFORM_STAGINGTABLE.sql. Exiting.
   pause
   exit /b
)


echo Running TRANSFORM_ANALYTICALBUSINESSTABLE.sql...

%MYSQL_CMD% -u %USER% -p%PASSWORD% %DATABASE% < "C:\Users\labva\OneDrive\Desktop\DE Simulation\TRANSFORM_ANALYTICALBUSINESSTABLE.sql" >> "C:\Program Files\MySQL\MySQL Server 8.0\bin\log.txt" 2>&1

IF %ERRORLEVEL% NEQ 0 (
   echo Error executing TRANSFORM_ANALYTICALBUSINESSTABLE.sql. Exiting.
   pause
   exit /b
)


echo All SQL scripts executed successfully.

pause
```

III. Visualization and Insights Generation (PowerBI)

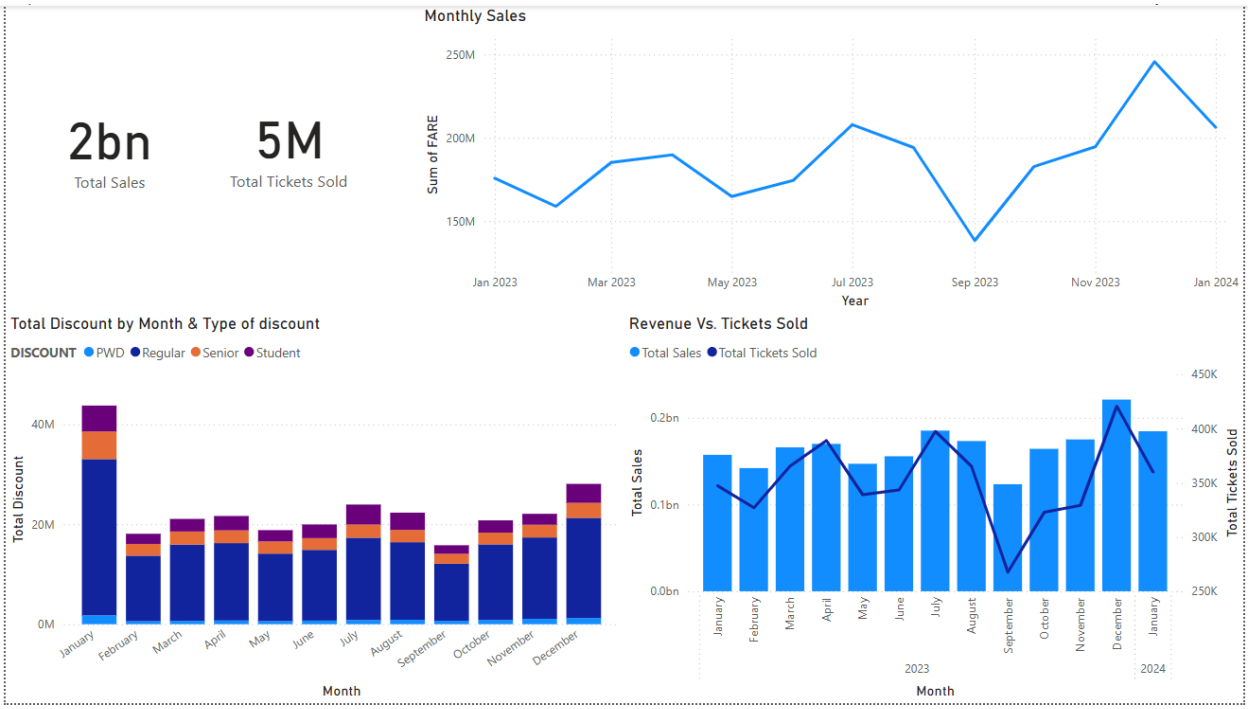
A. Visualization

1. Executive Summary Dashboard

- Required KPI's:

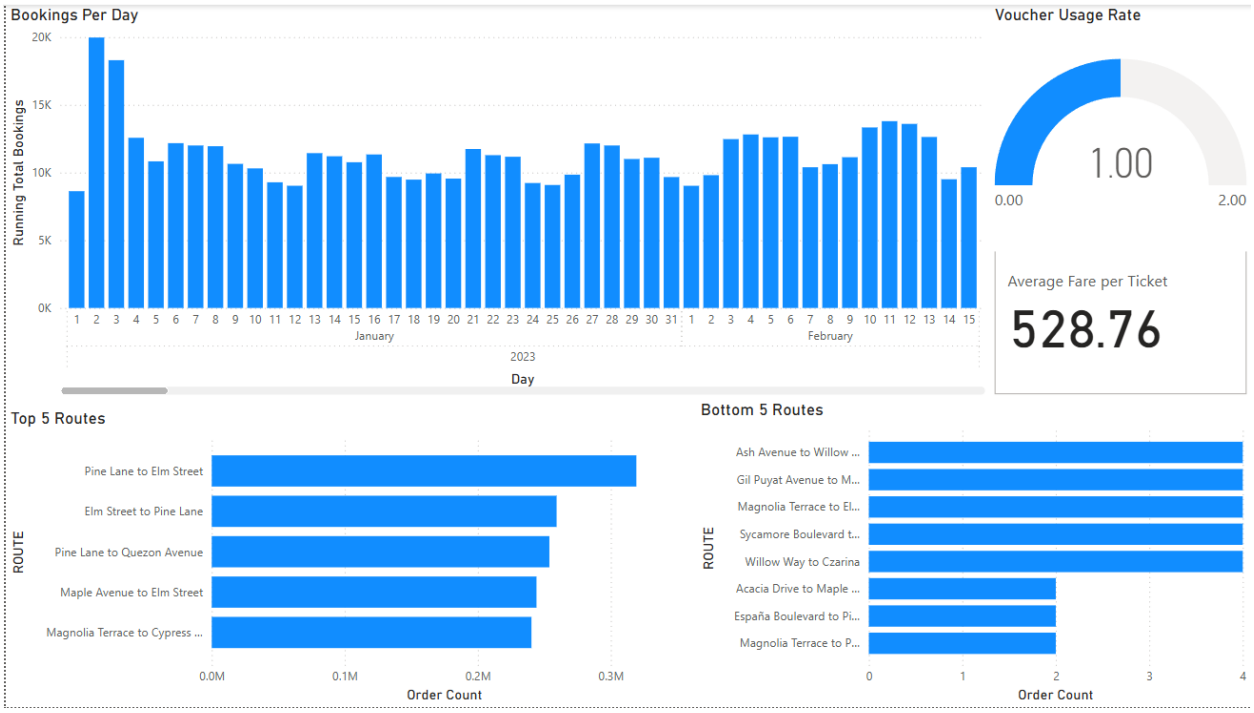
- Total Sales (Card)
- Total Tickets Sold (Card)
- Monthly Sales (Line Chart)

- Added Elements
  - Revenue Vs. Tickets Sold (Line and Clustered Column Chart)
  - Total Discount by Month & Type of discount (Stacked Column Chart)

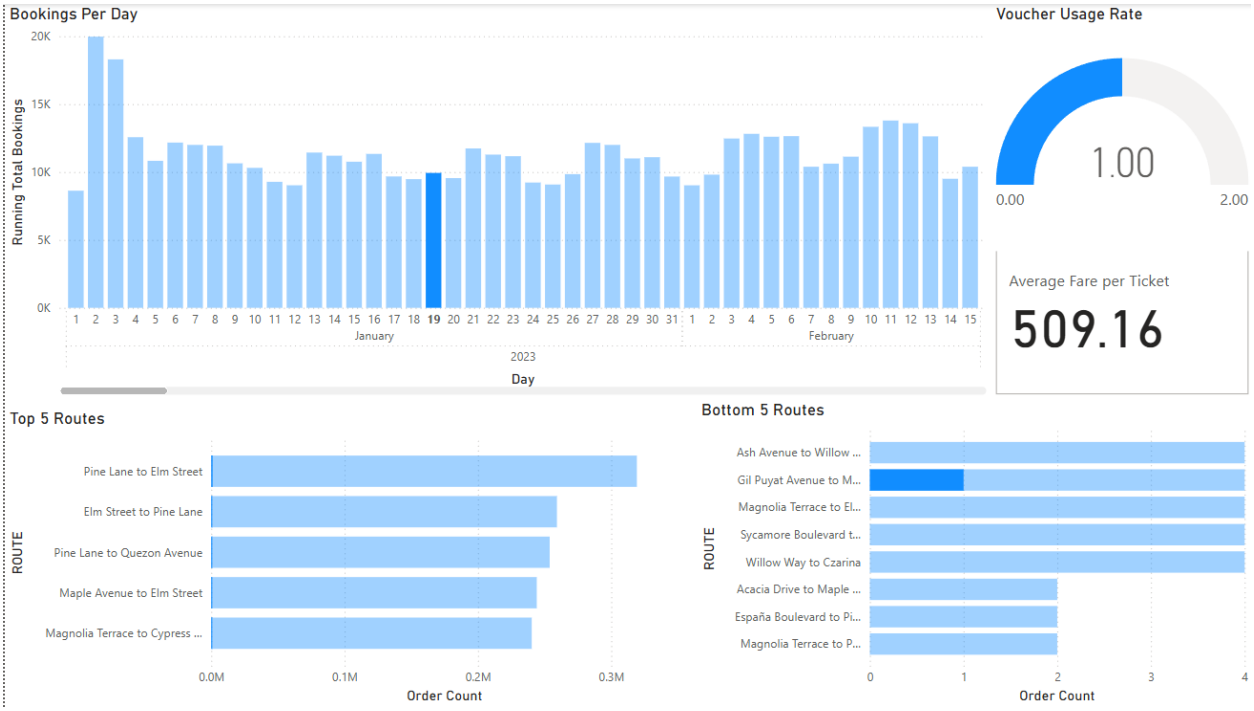


2. Trips Dashboard

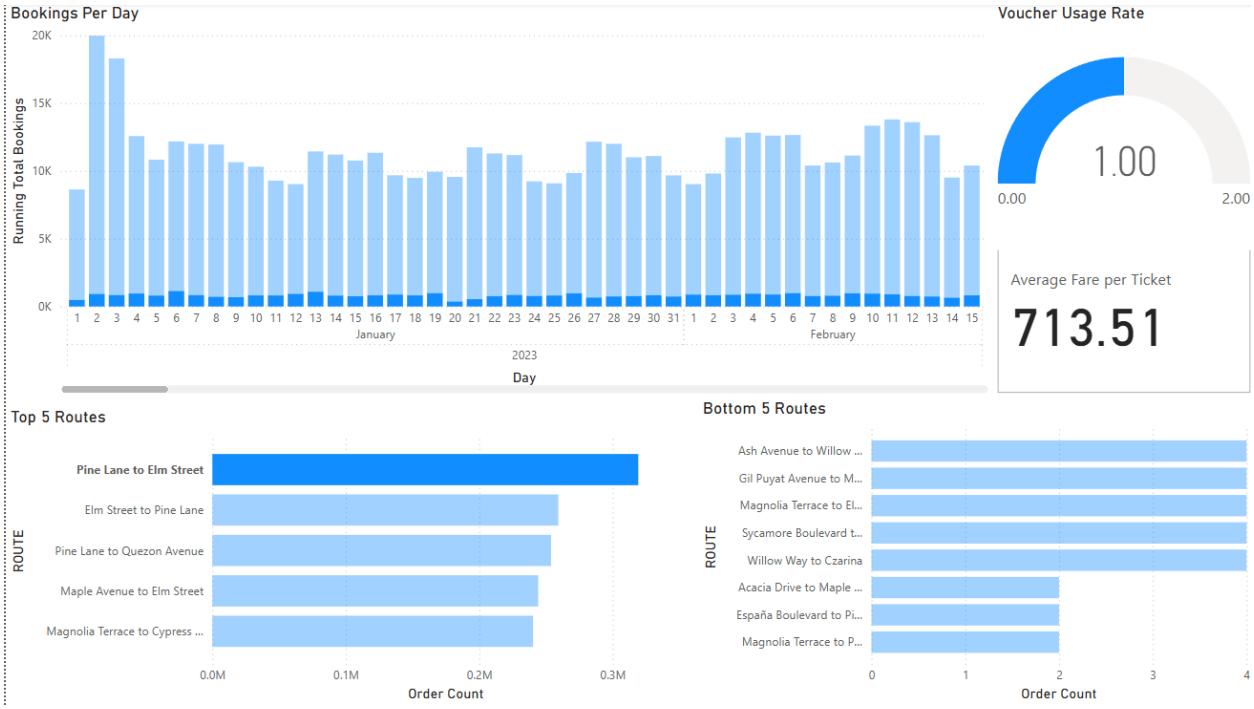
- Required KPI's:
  - Top 5 Routes (Clustered Bar Chart)
  - Bottom 5 Routes (Clustered Bar Chart)
- Added Elements
  - Bookings Per Day (Stacked Column Chart)
  - Voucher Usage Rate (Gauge)
  - Average Fare Per Ticket (Card)



Clicking a certain bar will show you statistics associated with that bar. Here, I clicked January 19, 2023. It shows which route was taken and how much the average fair was for all the bookings made that day.



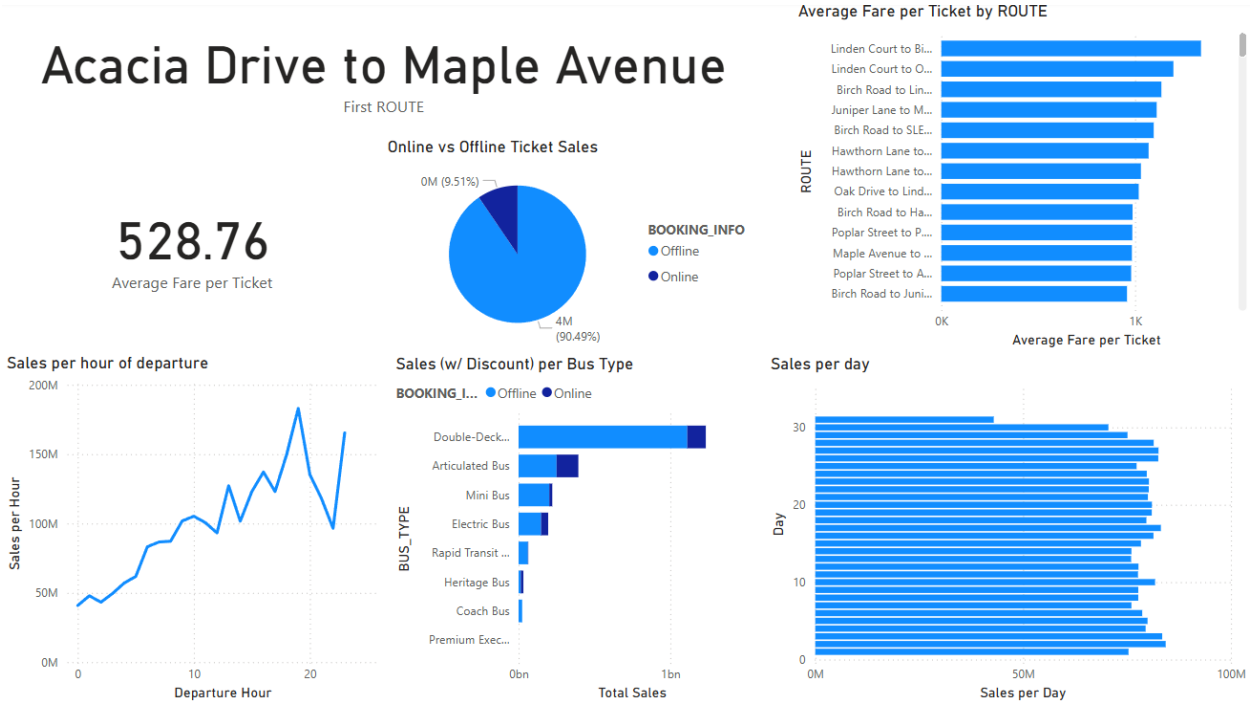
Here I selected the top 1 route, it shows how many bookings were made to this route in that particular day.



3. Sales Dashboard

- Required KPI's:

        - Online vs Offline Ticket Sales (Pie Chart)
        - Sales per hour of departure (Line Chart)
        - Sales per day (Clustered Bar Chart)
    - Added Elements
        - Sales per Bus Type (Stacked Bar Chart)
        - Average Fare per Ticket (Card)
        - Route Name (Card)
        - Average Fare per Ticket by Route (Stacked Bar Chart)



B. Insights

1. Visualization Insights

a) Executive Summary Dashboard

1. **Total Sales (Card)**

Displays the total revenue generated from ticket sales, helping gauge overall business performance.

2. **Total Tickets Sold (Card)**

Shows the total number of tickets sold, providing a measure of customer demand.

3. **Monthly Sales (Line Chart)**

Highlights revenue trends over time, identifying peak and off-peak months for ticket sales.

4. **Revenue vs. Tickets Sold (Line & Clustered Column Chart)**

Helps compare sales volume vs. revenue, revealing whether higher ticket sales always result in higher revenue or if pricing variations affect earnings.

5. **Total Discount by Month & Type of Discount (Stacked Column Chart)**

Shows how discounts (PWD, student, senior citizen) impact revenue monthly, helping assess promotional effectiveness.

---

b) Trips Dashboard

1. **Top 5 Routes (Clustered Bar Chart)**

Identifies the most popular routes, allowing optimization of fleet allocation and marketing strategies.

2. **Bottom 5 Routes (Clustered Bar Chart)**

Reveals underperforming routes, indicating possible service adjustments or targeted promotions.

3. **Bookings Per Day (Stacked Column Chart)**

Shows daily demand fluctuations, helping optimize scheduling and staffing.

4. **Voucher Usage Rate (Gauge)**

Measures the proportion of bookings using vouchers, indicating customer reliance on promotional offers.

5. **Average Fare Per Ticket (Card)**

Provides a quick reference for average pricing, helping monitor pricing strategies and customer affordability.

---

c) Sales Dashboard

1. **Online vs Offline Ticket Sales (Pie Chart)**

Compares digital and physical sales, helping assess the effectiveness of online ticketing channels.

2. **Sales Per Hour of Departure (Line Chart)**

Identifies peak departure hours, allowing better pricing and scheduling adjustments.

3. **Sales Per Day (Clustered Bar Chart)**

Shows daily revenue trends, helping plan promotions or operational improvements.

4. **Sales Per Bus Type (Stacked Bar Chart)**

Compares revenue across different bus types, supporting pricing decisions and fleet utilization strategies.

5. **Average Fare Per Ticket (Card)**

Helps track average ticket price changes and their effect on sales.

6. **Route Name (Card)**

Displays route-specific data when a user selects a specific date or booking.

7. **Average Fare Per Ticket by Route (Stacked Bar Chart)**

Compares ticket pricing across different routes, helping optimize fare structures.

---

2. Solution

The world that we currently live in now has easy access to AI through services like ChatGPT. When utilizing these services it may lead to a boost in sales. Here are some of the benefits that AI may be able to give us.

1. **AI-Powered Dynamic Pricing**

- Use AI to adjust fares based on demand, offering discounts during off-peak times and increasing prices during high demand.

2. **Personalized Promotions**

- Offer targeted discounts to specific customers based on their booking history instead of across-the-board reductions.

3. **AI Chatbots & Virtual Assistants**

- Improve customer engagement and sales by integrating AI-powered ticket booking chatbots.

4. **Loyalty & Referral Programs**

- Instead of reducing discounts, introduce reward points for repeat customers to maintain customer retention.

However, integrating AI with our systems may take some time and proper application. So, an alternative immediate way we can propose is lowering the discount rates by 5%.

Discount Type	**Current Rate**	**Proposed Rate**
Regular	10	5
Student	15	10
PWD	20	15
Senior	25	20

This would give us more revenue per ticket and potentially increase total sales by around 5% more. Moreover, since regular fares make up majority of the sales it would mean that total sales will increase. However, sales volume might drop due to some of the passengers relying on the discount and there might be backlash and some customers might prefer other competitors.