# 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}")

## 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)

);

## 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;

## 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;

## 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;

# Automation – Windows Task Scheduler & Batch File

## 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.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## 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

# Visualization and Insights Generation (PowerBI)

## Visualization

### 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)

A screenshot of a graph

AI-generated content may be incorrect.

### 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)

A screenshot of a graph

AI-generated content may be incorrect.

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.

A screenshot of a graph

AI-generated content may be incorrect.

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

A screenshot of a graph

AI-generated content may be incorrect.

### 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)

A close-up of a graph

AI-generated content may be incorrect.

## Insights

### Visualization Insights

#### Executive Summary Dashboard

1. **Total Sales (Card)**

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

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

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

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

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

1. **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.

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

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

#### Trips Dashboard

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

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

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

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

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

Shows daily demand fluctuations, helping optimize scheduling and staffing.

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

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

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

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

#### Sales Dashboard

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

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

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

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

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

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

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

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

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

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

1. **Route Name (Card)**

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

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

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

### 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.

1. **Personalized Promotions**

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

1. **AI Chatbots & Virtual Assistants**

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

1. **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.