School of Information and Communication Technology

Griffith University

7821ICT

Green City Situation Awareness User Manual – Public Transport

Proof of Concept Demonstrator for Sustainable Transport

18/10/2024 T2 2024

Industry Partner: City of Gold Coast

Client: Phillip Karfs

Team members:

Omkar Dhananjay Kulkarni (s5325810)

Vlesetty Vishal Kumar (s5314434)

Viet An Nguyen (s5321461)

Zhanrui Liao (s5290972)

Table of Content

Introduction

This user manual provides a comprehensive guide to analyzing public transportation data for the years 2022, 2023, and 2024. The analysis covers various aspects of public transport usage, such as mode share, year-over-year percentage changes, and peak versus off-peak month comparisons. The tools and methodologies employed in this manual include Python libraries like Pandas, NumPy, Matplotlib, Seaborn, and Statsmodels, which are utilized to process, visualize, and predict public transport data trends. By following this manual, users will gain a clear understanding of how to handle large datasets, perform time series analysis, and generate predictive models for public transport planning and optimization. This manual also illustrates how these insights can be utilized for decision-making, particularly in the context of policy changes, such as fare adjustments, that impact ridership.

Installing Anaconda and Starting Jupyter

Step 1: Download and Install Anaconda

1. Visit the official Anaconda website at https://www.anaconda.com.
2. Click on the "Download" button and choose the appropriate version for your operating system (Windows, macOS, or Linux).
3. After downloading, run the installer and follow the on-screen instructions. It is recommended to install Anaconda for "All Users" (Windows) or choose the default settings (macOS/Linux).

Step 2: Verifying Anaconda Installation

1. After installation, open the terminal (macOS/Linux) or the Anaconda Prompt (Windows).
2. Type the following command to verify the installation:

   conda –version

   You should see the version number of Conda, confirming a successful installation.

Step 3: Starting Jupyter Notebook

1. Open the Anaconda Navigator from your Applications menu or by typing anaconda-navigator in the terminal/Anaconda Prompt.
2. In the Anaconda Navigator, find and click on "Launch" under the Jupyter Notebook section.
3. Alternatively, you can start Jupyter directly by typing the following command in the terminal or Anaconda Prompt:

jupyter notebook

4. Your web browser will automatically open Jupyter's interface. You can now create or open .ipynb files to begin your work.

Step 4: Updating Anaconda and Jupyter

To ensure you're using the latest version of Anaconda and Jupyter, periodically update them with the following commands:

conda update conda
conda update anaconda
conda update jupyter

Data Analysis of Public Transport Data

Code:

```
#installing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from scipy import stats
```

This code imports the necessary libraries for analyzing public transport data. These include pandas for data manipulation, numpy for numerical calculations, matplotlib and seaborn for data visualization, statsmodels for time series analysis, and scipy for statistical functions.

Analysing '2022,2023,2024 Public Transport Data'

Code:

```
# Loading the Excel files
file_path_1 = '2022,2023,2024 Public Transport Data.xlsx'

# Loading the 2022-2024 Public Transport Data sheet
df = pd.read_excel(file_path_1, sheet_name='Passenger Journeys Gold Coast')
df['Fiscal Month'] = pd.to_datetime(df['Fiscal Month'], errors='coerce').dt.to_period('M')
```

df_clean = df.dropna()

print(df_clean.head())

```
# Loading the Excel files
file_path_1 = '2022,2023,2024 Public Transport Data.xlsx'

# Loading the 2022-2024 Public Transport Data sheet
df = pd.read_excel(file_path_1, sheet_name='Passenger Journeys Gold Coast')
df['Fiscal Month'] = pd.to_datetime(df['Fiscal Month'], errors='coerce').dt.to_period('M')
df_clean = df.dropna()

print(df_clean.head())
```

This code segment loads an Excel file containing public transport data from 2022 to 2024 and extracts a specific sheet, "Passenger Journeys Gold Coast." It converts the "Fiscal Month" column to a date format, specifically handling it as year and month. The data is then cleaned by removing any rows containing null values, and the first five rows of the cleaned data are printed for validation.

	Fiscal Month	Gold Coast Light Rail	Queensland Rail	Surfside Buslines	Total
0	2022-01	333770	92855	599234	1025859
1	2022-02	369991	116914	772641	1259546
2	2022-03	450552	142197	969454	1562203
3	2022-04	531790	165392	896297	1593479
4	2022-05	533585	186616	1009498	1729699

This table displays public transport passenger data from January to May 2022, including different transport modes such as "Gold Coast Light Rail," "Queensland Rail," and "Surfside Buslines." Each row represents the passenger volume for a specific month across these transport modes, as well as the total passenger count. It provides a quick overview of the monthly transport volume distribution.

Code:

df_clean.describe()

```
df_clean.describe()
```

	Gold Coast Light Rail	Queensland Rail	Surfside Buslines	Total
count	32.000000	32.000000	3.200000e+01	3.200000e+01
mean	666796.468750	204139.593750	1.073266e+06	1.944202e+06
std	120462.910832	40075.774524	1.468247e+05	2.946468e+05
min	333770.000000	92855.000000	5.992340e+05	1.025859e+06
25%	650538.000000	187906.750000	1.014983e+06	1.826918e+06
50%	690676.000000	208762.500000	1.083672e+06	2.001031e+06
75%	727407.250000	225461.750000	1.182554e+06	2.111634e+06
max	922271.000000	312974.000000	1.290216e+06	2.489767e+06

This section of code generates a summary of descriptive statistics for the cleaned dataset using the df_clean.describe() function. The resulting table provides key statistical metrics for the Gold Coast Light Rail, Queensland Rail, and Surfside Buslines, including the total counts, mean values, standard deviation, minimum and maximum values, as well as the 25th, 50th (median), and 75th percentiles. These metrics help in understanding the overall distribution of the data, such as the average ridership, variability, and range of public transport usage for each mode.

Code:

df_clean.info()

```
df_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 32 entries, 0 to 31
Data columns (total 5 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Fiscal Month          32 non-null     period[M]
 1   Gold Coast Light Rail  32 non-null     int64
 2   Queensland Rail        32 non-null     int64
 3   Surfside Buslines      32 non-null     int64
 4   Total                 32 non-null     int64
dtypes: int64(4), period[M](1)
memory usage: 1.5 KB
```

The code snippet uses the df_clean.info() function to display a concise summary of the cleaned dataset. This table provides essential information about the structure of the dataset, including the number of entries (32 in total), the names and data types of each column, as well as the count of non-null values for each column. The table shows that there are five columns: 'Fiscal Month', 'Gold Coast Light Rail', 'Queensland Rail', 'Surfside Buslines', and 'Total', with each column containing 32 non-null entries. The

data types are indicated, with the 'Fiscal Month' being a period type (monthly), and the remaining columns being of integer type (int64). This summary is useful for verifying the integrity and structure of the data before further analysis.

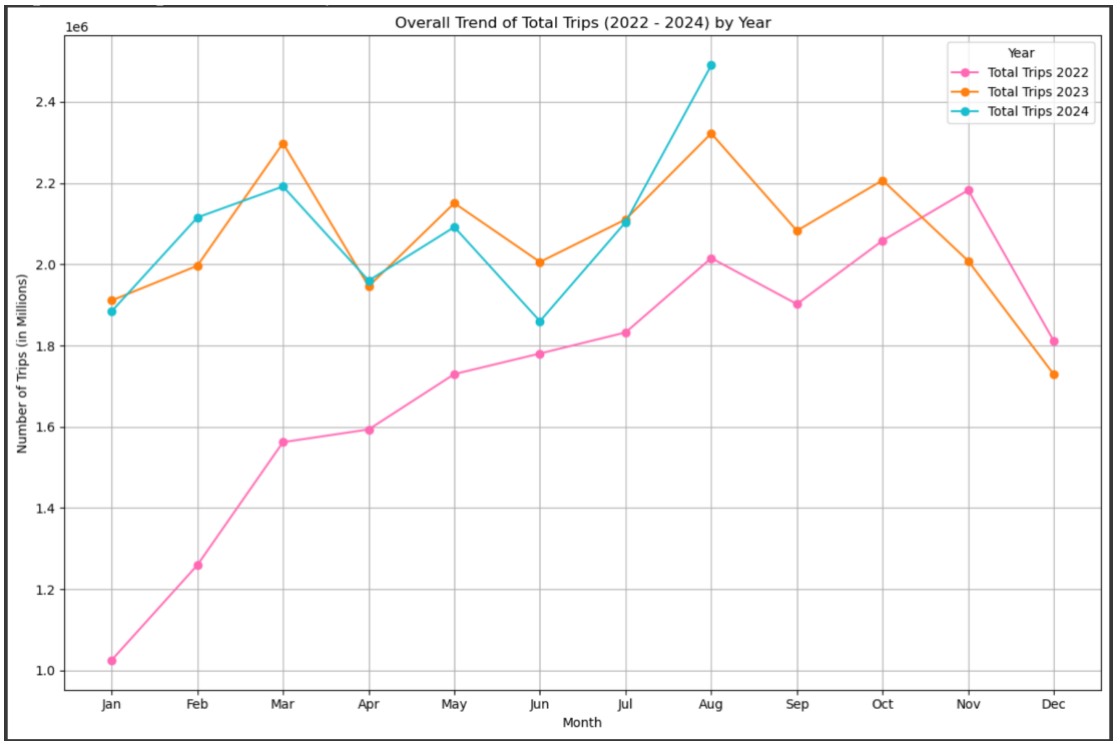
Overall Trend Analysis

Code:

```
# Creating a new column for year to distinguishing different years
df_clean['Year'] = df_clean['Fiscal Month'].dt.year

# Plotting
plt.figure(figsize=(12, 8))

colors = ['#ff69b4', '#ff7f0e', '#17becf']
for idx, year in enumerate(df_clean['Year'].unique()):
    year_data = df_clean[df_clean['Year'] == year]
    # Extracting only the month name for display on the x-axis
    plt.plot(year_data['Fiscal Month'].dt.strftime('%b'), year_data['Total'],
             marker='o', linestyle='-', color=colors[idx], label=f'Total Trips {year}')

plt.title('Overall Trend of Total Trips (2022 - 2024) by Year')
plt.xlabel('Month')
plt.ylabel('Number of Trips (in Millions)')
plt.grid(True)
plt.legend(title='Year')
plt.tight_layout()
plt.show()
```

This code first creates a new column Year to represent the year of each Fiscal Month. Then, using matplotlib, it generates a plot that visualizes the overall trend of total passenger trips from 2022 to 2024. Data for each year is plotted in a different color,

with the month displayed on the X-axis and the number of trips (in millions) on the Y-axis. The chart includes a title, axis labels, gridlines, and a legend to clearly present the overall trend.



Findings:

1. Overview: This graph shows the overall trend of total trips for three different years (2022, 2023, and 2024). The x-axis represents the months from January to December, while the y-axis represents the number of trips in millions.
2. Observation:
   A. 2022 had a steady rise in trips across the months, with no sharp fluctuations.
   B. 2023 had a relatively high start, peaked in June, and then had a notable decline by December.
   C. 2024 was following a similar pattern to 2023 but with more variability, especia lly in the latter part of the year, where August and September saw noticeable peak s.
3. Conclusion: The rise in August and September 2024 may correspond with the 50c trial, causing increased public transport usage during these months.

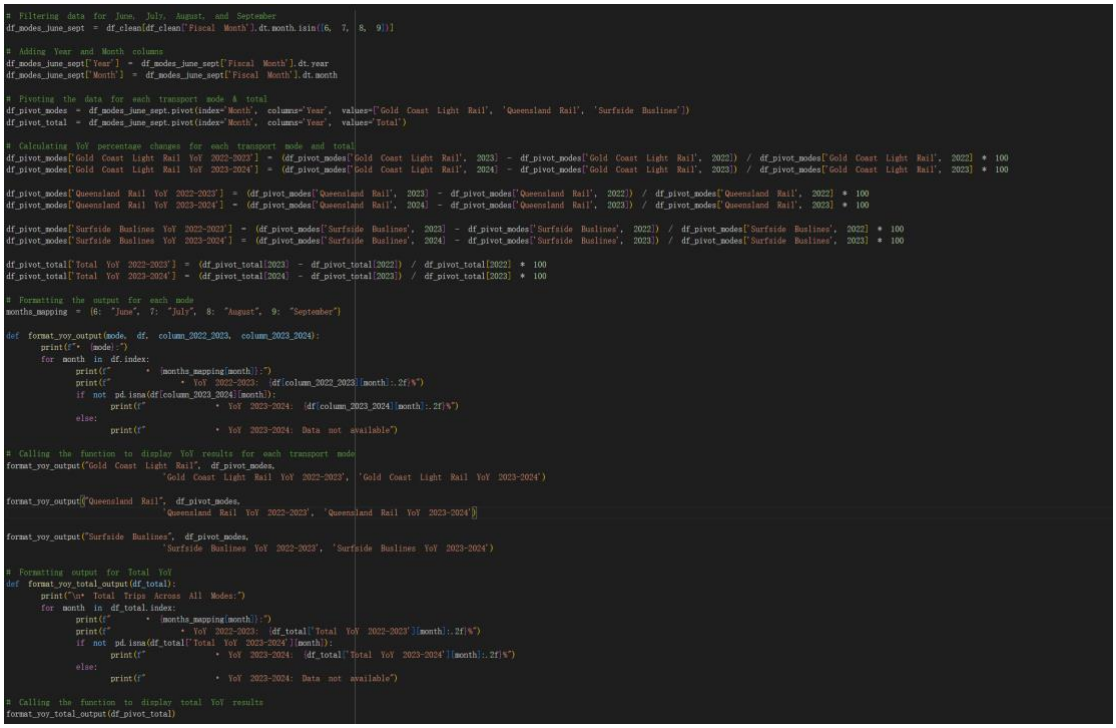
YoY % Change

Code:

```
# Filtering data for June, July, August, and September
df_modes_june_sept = df_clean[df_clean['Fiscal Month'].dt.month.isin([6, 7, 8, 9])]

# Adding Year and Month columns
df_modes_june_sept['Year'] = df_modes_june_sept['Fiscal Month'].dt.year
df_modes_june_sept['Month'] = df_modes_june_sept['Fiscal Month'].dt.month
```

```
# Pivoting the data for each transport mode & total
df_pivot_modes = df_modes_june_sept.pivot(index='Month', columns='Year', values=['Gold Coast Light Rail', 'Queensland Rail', 'Surfside Buslines'])
df_pivot_total = df_modes_june_sept.pivot(index='Month', columns='Year', values='Total')

# Calculating YoY percentage changes for each transport mode and total
df_pivot_modes['Gold Coast Light Rail YoY 2022-2023'] = (df_pivot_modes['Gold Coast Light Rail', 2023] - df_pivot_modes['Gold Coast Light Rail', 2022]) / df_pivot_modes['Gold Coast Light Rail', 2022] * 100
df_pivot_modes['Gold Coast Light Rail YoY 2023-2024'] = (df_pivot_modes['Gold Coast Light Rail', 2024] - df_pivot_modes['Gold Coast Light Rail', 2023]) / df_pivot_modes['Gold Coast Light Rail', 2023] * 100

df_pivot_modes['Queensland Rail YoY 2022-2023'] = (df_pivot_modes['Queensland Rail', 2023] - df_pivot_modes['Queensland Rail', 2022]) / df_pivot_modes['Queensland Rail', 2022] * 100
df_pivot_modes['Queensland Rail YoY 2023-2024'] = (df_pivot_modes['Queensland Rail', 2024] - df_pivot_modes['Queensland Rail', 2023]) / df_pivot_modes['Queensland Rail', 2023] * 100

df_pivot_modes['Surfside Buslines YoY 2022-2023'] = (df_pivot_modes['Surfside Buslines', 2023] - df_pivot_modes['Surfside Buslines', 2022]) / df_pivot_modes['Surfside Buslines', 2022] * 100
df_pivot_modes['Surfside Buslines YoY 2023-2024'] = (df_pivot_modes['Surfside Buslines', 2024] - df_pivot_modes['Surfside Buslines', 2023]) / df_pivot_modes['Surfside Buslines', 2023] * 100

df_pivot_total['Total YoY 2022-2023'] = (df_pivot_total[2023] - df_pivot_total[2022]) / df_pivot_total[2022] * 100
df_pivot_total['Total YoY 2023-2024'] = (df_pivot_total[2024] - df_pivot_total[2023]) / df_pivot_total[2023] * 100

# Formatting the output for each mode
months_mapping = {6: "June", 7: "July", 8: "August", 9: "September"}

def format_yoy_output(mode, df, column_2022_2023, column_2023_2024):
    print(f"• {mode}:")
    for month in df.index:
        print(f"    • {months_mapping[month]}:")
        print(f"        • YoY 2022-2023: {df[column_2022_2023][month]:.2f}%")
        if not pd.isna(df[column_2023_2024][month]):
            print(f"        • YoY 2023-2024: {df[column_2023_2024][month]:.2f}%")
```

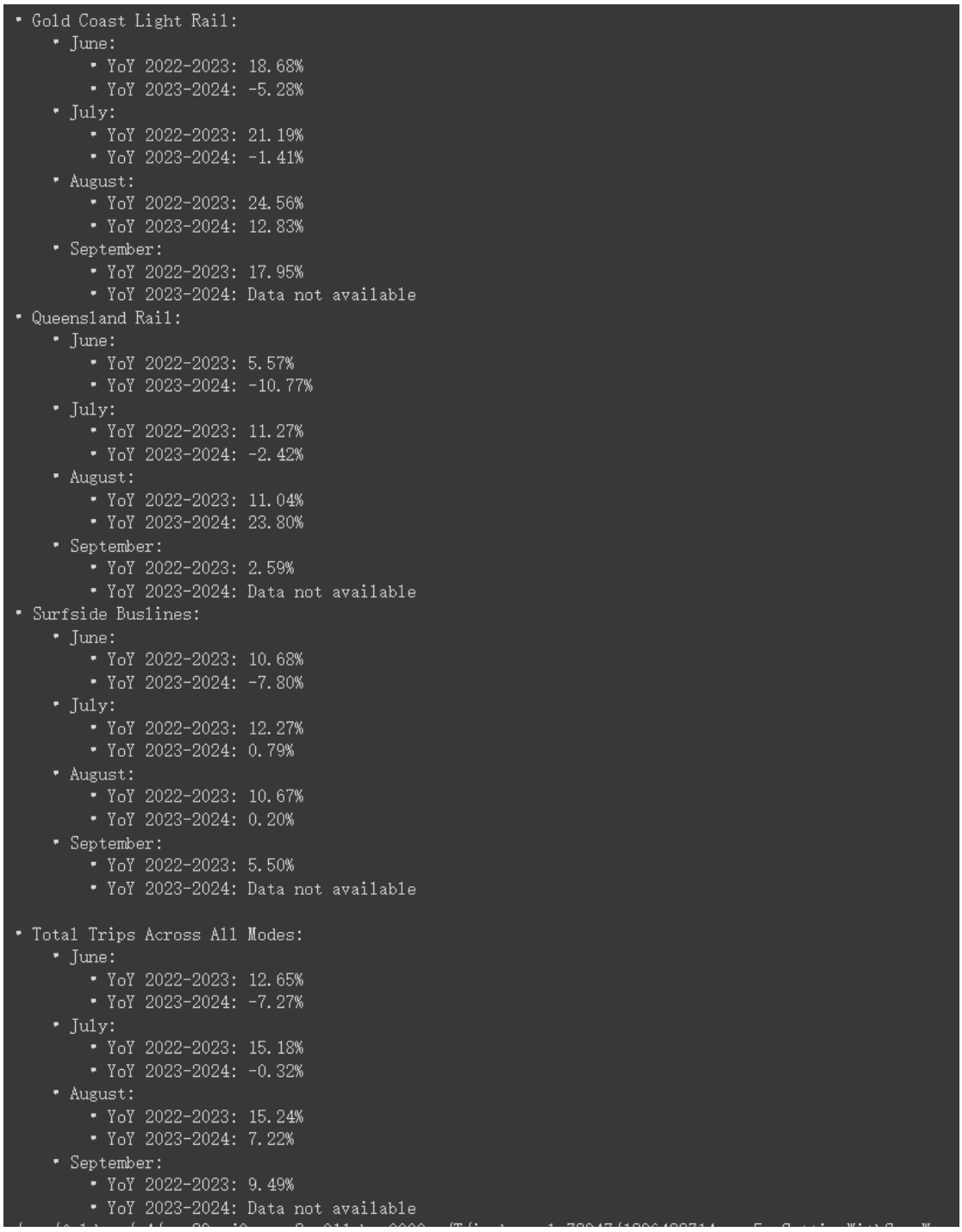
```
        else:
            print(f"          • YoY 2023-2024: Data not available")

# Calling the function to display YoY results for each transport mode
format_yoy_output("Gold Coast Light Rail", df_pivot_modes,
                  'Gold Coast Light Rail YoY 2022-2023', 'Gold Coast Light Rail YoY
2023-2024')

format_yoy_output("Queensland Rail", df_pivot_modes,
                  'Queensland Rail YoY 2022-2023', 'Queensland Rail YoY 2023-2024')

format_yoy_output("Surfside Buslines", df_pivot_modes,
                   'Surfside Buslines YoY 2022-2023', 'Surfside Buslines YoY 2023-
2024')

# Formatting output for Total YoY
def format_yoy_total_output(df_total):
    print("\n• Total Trips Across All Modes:")
    for month in df_total.index:
        print(f"    • {months_mapping[month]}:")
            print(f"              • YoY 2022-2023: {df_total['Total YoY 2022-
2023'][month]:.2f}%")
        if not pd.isna(df_total['Total YoY 2023-2024'][month]):
                print(f"              • YoY 2023-2024: {df_total['Total YoY 2023-
2024'][month]:.2f}%")
        else:
            print(f"          • YoY 2023-2024: Data not available")

# Calling the function to display total YoY results
format_yoy_total_output(df_pivot_total)
```



```
# Filtering data for June, July, August, and September
df_modes_june_sept = df_clean[df_clean['Fiscal Month'].dt.month.isin([6, 7, 8, 9])]

# Adding Year and Month columns
df_modes_june_sept['Year'] = df_modes_june_sept['Fiscal Month'].dt.year
df_modes_june_sept['Month'] = df_modes_june_sept['Fiscal Month'].dt.month

# Pivoting the data for each transport mode & total
df_pivot_modes = df_modes_june_sept.pivot(index='Month', columns='Year', values=['Gold Coast Light Rail', 'Queensland Rail', 'Surfside Buslines'])
df_pivot_total = df_modes_june_sept.pivot(index='Month', columns='Year', values='Total')

# Calculating YoY percentage changes for each transport mode and total
df_pivot_modes['Gold Coast Light Rail YoY 2022-2023'] = (df_pivot_modes['Gold Coast Light Rail', 2023] - df_pivot_modes['Gold Coast Light Rail', 2022]) / df_pivot_modes['Gold Coast Light Rail', 2022] * 100
df_pivot_modes['Gold Coast Light Rail YoY 2023-2024'] = (df_pivot_modes['Gold Coast Light Rail', 2024] - df_pivot_modes['Gold Coast Light Rail', 2023]) / df_pivot_modes['Gold Coast Light Rail', 2023] * 100

df_pivot_modes['Queensland Rail YoY 2022-2023'] = (df_pivot_modes['Queensland Rail', 2023] - df_pivot_modes['Queensland Rail', 2022]) / df_pivot_modes['Queensland Rail', 2022] * 100
df_pivot_modes['Queensland Rail YoY 2023-2024'] = (df_pivot_modes['Queensland Rail', 2024] - df_pivot_modes['Queensland Rail', 2023]) / df_pivot_modes['Queensland Rail', 2023] * 100

df_pivot_modes['Surfside Buslines YoY 2022-2023'] = (df_pivot_modes['Surfside Buslines', 2023] - df_pivot_modes['Surfside Buslines', 2022]) / df_pivot_modes['Surfside Buslines', 2022] * 100
df_pivot_modes['Surfside Buslines YoY 2023-2024'] = (df_pivot_modes['Surfside Buslines', 2024] - df_pivot_modes['Surfside Buslines', 2023]) / df_pivot_modes['Surfside Buslines', 2023] * 100

df_pivot_total['Total YoY 2022-2023'] = (df_pivot_total[2023] - df_pivot_total[2022]) / df_pivot_total[2022] * 100
df_pivot_total['Total YoY 2023-2024'] = (df_pivot_total[2024] - df_pivot_total[2023]) / df_pivot_total[2023] * 100

# Formatting the output for each mode
months_mapping = {6: "June", 7: "July", 8: "August", 9: "September"}

def format_yoy_output(mode, df, column_2022_2023, column_2023_2024):
    print(f"* {mode}:")
    for month in df.index:
        print(f"    - {months_mapping[month]}:")
        print(f"        - YoY 2022-2023: {df[column_2022_2023][month]:.2f}%")
        if not pd.isna(df[column_2023_2024][month]):
            print(f"        - YoY 2023-2024: {df[column_2023_2024][month]:.2f}%")
        else:
            print(f"        - YoY 2023-2024: Data not available")

# Calling the function to display YoY results for each transport mode
format_yoy_output("Gold Coast Light Rail", df_pivot_modes,
                  'Gold Coast Light Rail YoY 2022-2023', 'Gold Coast Light Rail YoY 2023-2024')

format_yoy_output("Queensland Rail", df_pivot_modes,
                  'Queensland Rail YoY 2022-2023', 'Queensland Rail YoY 2023-2024')

format_yoy_output("Surfside Buslines", df_pivot_modes,
                  'Surfside Buslines YoY 2022-2023', 'Surfside Buslines YoY 2023-2024')

# Formatting output for Total YoY
def format_yoy_total_output(df_total):
    print("\n* Total Trips Across All Modes:")
    for month in df_total.index:
        print(f"    - {months_mapping[month]}:")
        print(f"        - YoY 2022-2023: {df_total['Total YoY 2022-2023'][month]:.2f}%")
        if not pd.isna(df_total['Total YoY 2023-2024'][month]):
            print(f"        - YoY 2023-2024: {df_total['Total YoY 2023-2024'][month]:.2f}%")
        else:
            print(f"        - YoY 2023-2024: Data not available")

# Calling the function to display total YoY results
format_yoy_total_output(df_pivot_total)
```

This code first filters the data from df_clean to only include records for June, July, August, and September. It then adds new columns for Year and Month. After that, it pivots the data for each transport mode (Gold Coast Light Rail, Queensland Rail, Surfside Buslines) and total trips, organizing it by year.

Next, the code calculates the year-over-year (YoY) percentage change for each transport mode and total trips from 2022 to 2023 and 2023 to 2024. Finally, it uses the format_yoy_output and format_yoy_total_output functions to print the YoY results for each transport mode and the total number of trips month by month, providing a clear overview of the transport trend changes.



This image shows the year-over-year percentage changes from June to September 2022-2024 for Gold Coast Light Rail, Queensland Rail, and Surfside Buslines. Some data is missing for certain months. Overall, it helps analyze the trends in public transport usage across different years.

Explanation

Standard formula to calculate % change i.e. [(New value) - (Old Value)] / (Old Value) * 100

Interpreting the O/P

1. Positive O/P when New Value is greater than previous value e.g. August 2024

v/s August 2023

2. Negative O/P when New Value is lower than previous value e.g. June 2024 v/s June 2023
3. It would have been 0 if values were same. Not applicable for our case.

Code:

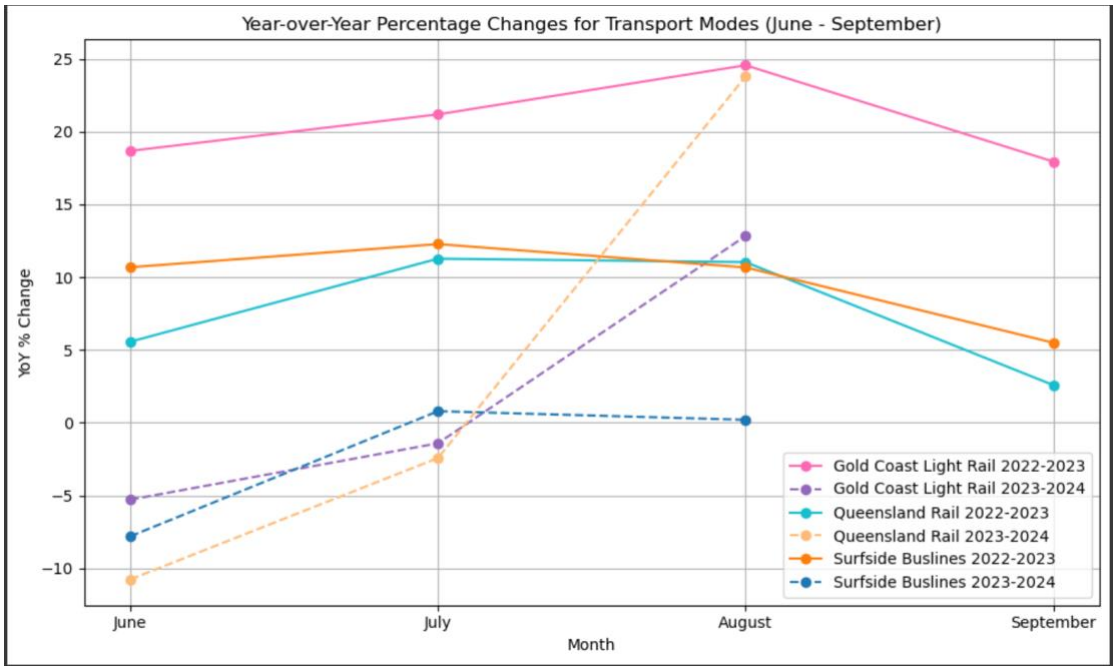
```
# Using 6 distinct colours
colors = {'pink': '#ff69b4', 'orange': '#ff7f0e', 'teal': '#17becf', 'purple': '#9467bd', 'yellow': '#ffbb78', 'blue': '#1f77b4'}

# Plotting the YoY percentage changes for each transport mode using real data
plt.figure(figsize=(10, 6))

# Plotting Gold Coast Light Rail YoY percentage changes
plt.plot(df_pivot_modes.index, df_pivot_modes['Gold Coast Light Rail YoY 2022-2023'],
         marker='o', linestyle='-', color=colors['pink'], label='Gold Coast Light Rail 2022-2023')
plt.plot(df_pivot_modes.index, df_pivot_modes['Gold Coast Light Rail YoY 2023-2024'],
         marker='o', linestyle='--', color=colors['purple'], label='Gold Coast Light Rail 2023-2024')

# Plotting Queensland Rail YoY percentage changes
plt.plot(df_pivot_modes.index, df_pivot_modes['Queensland Rail YoY 2022-2023'],
         marker='o', linestyle='-', color=colors['teal'], label='Queensland Rail 2022-2023')
plt.plot(df_pivot_modes.index, df_pivot_modes['Queensland Rail YoY 2023-2024'],
         marker='o', linestyle='--', color=colors['yellow'], label='Queensland Rail 2023-2024')

# Plotting Surfside Buslines YoY percentage changes
plt.plot(df_pivot_modes.index, df_pivot_modes['Surfside Buslines YoY 2022-2023'],
         marker='o', linestyle='-', color=colors['orange'], label='Surfside Buslines 2022-2023')
plt.plot(df_pivot_modes.index, df_pivot_modes['Surfside Buslines YoY 2023-2024'],
         marker='o', linestyle='--', color=colors['blue'], label='Surfside Buslines 2023-2024')

# Customising the plot
plt.title('Year-over-Year Percentage Changes for Transport Modes (June - September)')
plt.xlabel('Month')
plt.ylabel('YoY % Change')
plt.xticks(ticks=df_pivot_modes.index, labels=[months_mapping[m] for m in df_pivot_modes.index])
```

plt.grid(True)
plt.legend()
plt.tight_layout()

plt.show()

```
# Using 6 distinct colours
colors = {'pink': '#ff69b4', 'orange': '#ff7f0e', 'teal': '#17becf', 'purple': '#9467bd', 'yellow': '#ffbb78', 'blue': '#1f77b4'}

# Plotting the YoY percentage changes for each transport mode using real data
plt.figure(figsize=(10, 6))

# Plotting Gold Coast Light Rail YoY percentage changes
plt.plot(df_pivot_modes.index, df_pivot_modes['Gold Coast Light Rail YoY 2022-2023'],
         marker='o', linestyle='-', color=colors['pink'], label='Gold Coast Light Rail 2022-2023')
plt.plot(df_pivot_modes.index, df_pivot_modes['Gold Coast Light Rail YoY 2023-2024'],
         marker='o', linestyle='--', color=colors['purple'], label='Gold Coast Light Rail 2023-2024')

# Plotting Queensland Rail YoY percentage changes
plt.plot(df_pivot_modes.index, df_pivot_modes['Queensland Rail YoY 2022-2023'],
         marker='o', linestyle='-', color=colors['teal'], label='Queensland Rail 2022-2023')
plt.plot(df_pivot_modes.index, df_pivot_modes['Queensland Rail YoY 2023-2024'],
         marker='o', linestyle='--', color=colors['yellow'], label='Queensland Rail 2023-2024')

# Plotting Surfside Buslines YoY percentage changes
plt.plot(df_pivot_modes.index, df_pivot_modes['Surfside Buslines YoY 2022-2023'],
         marker='o', linestyle='-', color=colors['orange'], label='Surfside Buslines 2022-2023')
plt.plot(df_pivot_modes.index, df_pivot_modes['Surfside Buslines YoY 2023-2024'],
         marker='o', linestyle='--', color=colors['blue'], label='Surfside Buslines 2023-2024')

# Customising the plot
plt.title('Year-over-Year Percentage Changes for Transport Modes (June - September)')
plt.xlabel('Month')
plt.ylabel('YoY % Change')
plt.xticks(ticks=df_pivot_modes.index, labels=[months_mapping[m] for m in df_pivot_modes.index])
plt.grid(True)
plt.legend()
plt.tight_layout()

plt.show()
```

This code generates a line plot showing the year-over-year percentage changes for Gold Coast Light Rail, Queensland Rail, and Surfside Buslines from June to September for the years 2022-2024. Different colors and line styles are used to distinguish the changes for each transport mode across different years. The plot's title, axis labels, and grid are customized to provide a clear visual representation of the trends.



Findings:

1. Overview: This graph shows the year-over-year (YoY) percentage changes for three public transport modes (Gold Coast Light Rail, Queensland Rail, and Surfside Buslines) from June to September over two different time periods(2022 - 2023 and 2023 - 2024).
2. Observation:

A. The Gold Coast Light Rail saw the most significant increase from 2022 to 2023, and despite starting at a negative percentage in 2023-2024, it showed strong recovery by August.

B. Queensland Rail showed modest year-over-year changes, with a slight improvement in both periods, peaking in August 2024.

C. Surfside Buslines experienced a recovery from negative values in both periods, but the 2023-2024 increase in August was much more drastic.

3. Conclusion: The 50c fare trial in August 2024 significantly boosted public transport usage, particularly for the Gold Coast Light Rail and Surfside Buslines, both of which saw marked increases compared to previous months and years. While Queensland Rail also showed a modest rise, its growth was less pronounced. The sharp spike in August, followed by slight declines in September, indicates the trial had a substantial but possibly temporary impact on travel behavior.

Mode Share Analysis

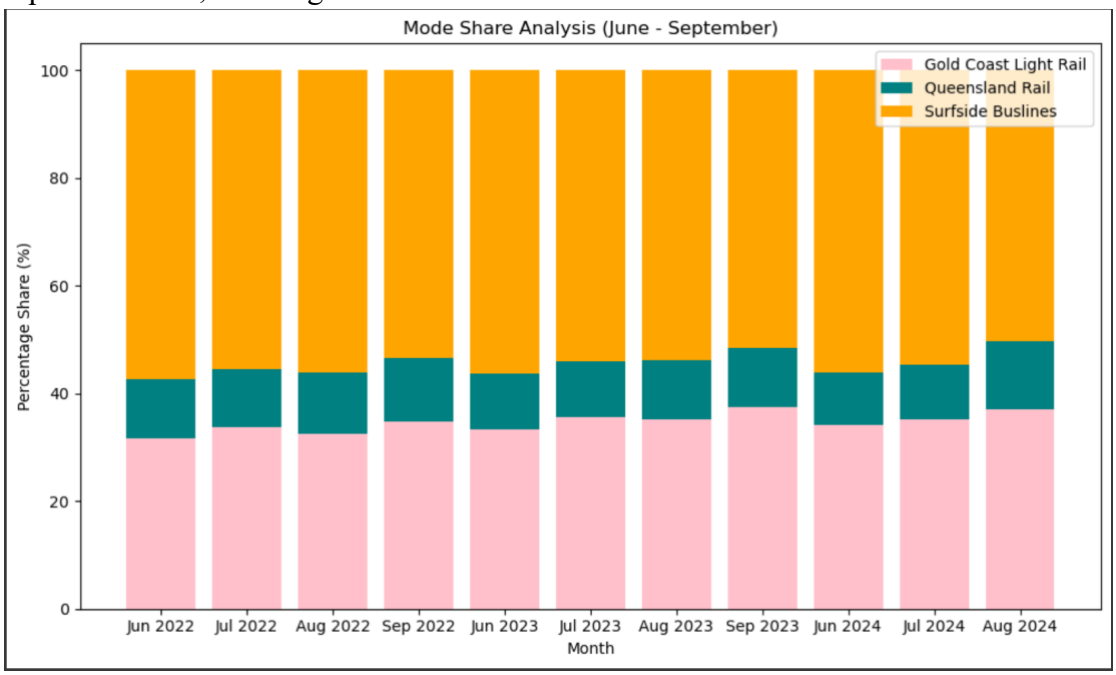
Code:

```
# Calculating the mode share (percentage of total trips) for each transport mode
df_modes_june_sept['Gold Coast Light Rail Share'] = (df_modes_june_sept['Gold Coast Light Rail'] / df_modes_june_sept['Total']) * 100
df_modes_june_sept['Queensland Rail Share'] = (df_modes_june_sept['Queensland Rail'] / df_modes_june_sept['Total']) * 100
df_modes_june_sept['Surfside Buslines Share'] = (df_modes_june_sept['Surfside Buslines'] / df_modes_june_sept['Total']) * 100

# Defining the months for X-axis
months = df_modes_june_sept['Fiscal Month'].dt.strftime('%b %Y')

# Plotting the stacked bar chart
plt.figure(figsize=(10, 6))
plt.bar(months, df_modes_june_sept['Gold Coast Light Rail Share'], color='pink', label='Gold Coast Light Rail')
plt.bar(months, df_modes_june_sept['Queensland Rail Share'], bottom=df_modes_june_sept['Gold Coast Light Rail Share'], color='teal', label='Queensland Rail')
plt.bar(months, df_modes_june_sept['Surfside Buslines Share'],
        bottom=df_modes_june_sept['Gold Coast Light Rail Share'] + df_modes_june_sept['Queensland Rail Share'],
        color='orange', label='Surfside Buslines')
```

```
# Customising the plot
plt.title('Mode Share Analysis (June - September)')
plt.xlabel('Month')
plt.ylabel('Percentage Share (%)')
plt.legend()
plt.tight_layout()

plt.show()
```

```
# Calculating the mode share (percentage of total trips) for each transport mode
df_modes_june_sept['Gold Coast Light Rail Share'] = (df_modes_june_sept['Gold Coast Light Rail'] / df_modes_june_sept['Total']) * 100
df_modes_june_sept['Queensland Rail Share'] = (df_modes_june_sept['Queensland Rail'] / df_modes_june_sept['Total']) * 100
df_modes_june_sept['Surfside Buslines Share'] = (df_modes_june_sept['Surfside Buslines'] / df_modes_june_sept['Total']) * 100

# Defining the months for X-axis
months = df_modes_june_sept['Fiscal Month'].dt.strftime('%b %Y')

# Plotting the stacked bar chart
plt.figure(figsize=(10, 6))
plt.bar(months, df_modes_june_sept['Gold Coast Light Rail Share'], color='pink', label='Gold Coast Light Rail')
plt.bar(months, df_modes_june_sept['Queensland Rail Share'], bottom=df_modes_june_sept['Gold Coast Light Rail Share'], color='teal', label='Queensland Rail')
plt.bar(months, df_modes_june_sept['Surfside Buslines Share'],
        bottom=df_modes_june_sept['Gold Coast Light Rail Share'] + df_modes_june_sept['Queensland Rail Share'],
        color='orange', label='Surfside Buslines')

# Customising the plot
plt.title('Mode Share Analysis (June - September)')
plt.xlabel('Month')
plt.ylabel('Percentage Share (%)')
plt.legend()
plt.tight_layout()

plt.show()
```

This code generates a stacked bar chart to illustrate the mode share for Gold Coast Light Rail, Queensland Rail, and Surfside Buslines from June to September. It calculates the percentage share for each transport mode relative to the total trips and visualizes them using different colors. The chart uses stacked bars, with each mode's share layered on top of the other, showing how each mode contributes to the total for each month.



Findings:

1. Overview: This graph shows the year-over-year (YoY) percentage changes for three public transport modes (Gold Coast Light Rail, Queensland Rail, and Surfside Buslines) from June to September over two different time periods(2022 - 2023 and 2023 - 2024).

2. Observation:
   A. The Gold Coast Light Rail consistently holds around 30% of the mode share.
   B. Queensland Rail has a steady share, making up around 10-15% of the total mode. Its contribution remains stable, similar to the light rail.
   C. Surfside Buslines consistently dominates the mode share, accounting for around 60% or more. This indicates that bus usage is the most prevalent mode of transport compared to the other two.
3. Conclusion: The graph highlights the stable distribution of public transport mode share in the Gold Coast region, with Surfside Buslines consistently holding the largest share, followed by Gold Coast Light Rail and Queensland Rail. Despite policy changes like the 50c fare trial in August 2024, the relative distribution of these modes remains largely unchanged, indicating strong preferences for bus transport.

Peak vs Off Peak Month Analysis

Code:

```
# Finding the Peak and Off-Peak months for each transport mode and total trips
peak_off_peak_summary = {}

# For each mode, finding the month with the highest and lowest trips
modes = ['Gold Coast Light Rail', 'Queensland Rail', 'Surfside Buslines', 'Total']

for mode in modes:
    peak_month = df_modes_june_sept.loc[df_modes_june_sept[mode].idxmax(), 'Fiscal Month'].strftime('%b %Y')
    off_peak_month = df_modes_june_sept.loc[df_modes_june_sept[mode].idxmin(), 'Fiscal Month'].strftime('%b %Y')

    peak_off_peak_summary[mode] = {
        'Peak Month': peak_month,
        'Off-Peak Month': off_peak_month,
        'Peak Trips': df_modes_june_sept[mode].max(),
        'Off-Peak Trips': df_modes_june_sept[mode].min()
    }

# Displaying the peak and off-peak summary
peak_off_peak_df = pd.DataFrame(peak_off_peak_summary).T
print(peak_off_peak_df)
```

```
# Finding the Peak and Off-Peak months for each transport mode and total trips
peak_off_peak_summary = {}

# For each mode, finding the month with the highest and lowest trips
modes = ['Gold Coast Light Rail', 'Queensland Rail', 'Surfside Buslines', 'Total']

for mode in modes:
        peak_month = df_modes_june_sept.loc[df_modes_june_sept[mode].idxmax(), 'Fiscal Month'].strftime('%b %Y')
        off_peak_month = df_modes_june_sept.loc[df_modes_june_sept[mode].idxmin(), 'Fiscal Month'].strftime('%b %Y')

        peak_off_peak_summary[mode] = {
                'Peak Month': peak_month,
                'Off-Peak Month': off_peak_month,
                'Peak Trips': df_modes_june_sept[mode].max(),
                'Off-Peak Trips': df_modes_june_sept[mode].min()
        }

# Displaying the peak and off-peak summary
peak_off_peak_df = pd.DataFrame(peak_off_peak_summary).T
print(peak_off_peak_df)
```

This code aims to determine the peak and off-peak months for Gold Coast Light Rail, Queensland Rail, Surfside Buslines, and total trips. It iterates through each transport mode, identifies the months with the highest and lowest trips, and records the corresponding trip counts. Finally, the information is aggregated into a dictionary and converted to a DataFrame, displaying the peak and off-peak months along with the respective trip counts for each mode.

	Peak Month	Off-Peak Month	Peak Trips	Off-Peak Trips
Gold Coast Light Rail	Aug 2024	Jun 2022	922271	563579
Queensland Rail	Aug 2024	Jun 2024	312974	183718
Surfside Buslines	Aug 2024	Jul 2022	1254522	1016811
Total	Aug 2024	Jun 2022	2489767	1780385

Findings:

1. Overview: This table provides an analysis of peak and off-peak trips for three transport modes.
2. Observation and Conclusion:
   A. All transport modes reached their peak usage in August 2024, which aligns with the 50c fare trial, suggesting that the trial significantly boosted ridership.
   B. The off-peak months vary, with Gold Coast Light Rail experiencing its lowest ridership in June 2022, while Queensland Rail's off-peak was in June 2024 and Surfside Buslines in July 2022.
   C. The difference between peak and off-peak trips is most notable for Surfside Buslines, which consistently maintains higher ridership even during off-peak months.
   D. The data strongly suggests that the 50c fare trial in August 2024 drove substantial increases in public transport usage across all modes, with the highest trip counts recorded during that month. However, trip numbers during off-peak months were relatively lower.

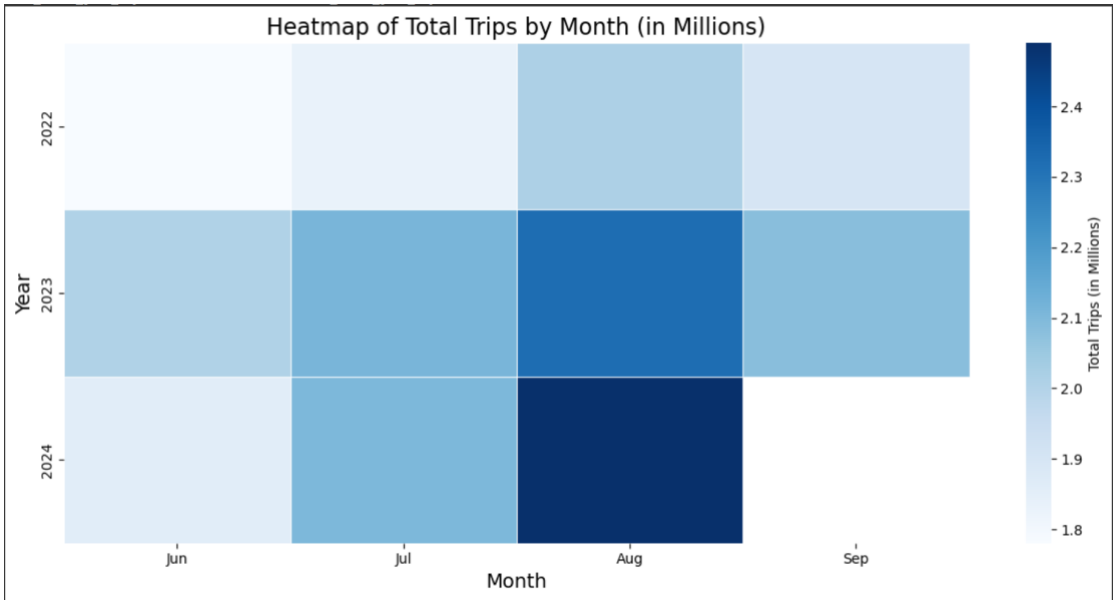
Code:

```
import seaborn as sns
# Dividing 'Total' by 1 million to get 'Total in Millions'
df_modes_june_sept['Total in Millions'] = df_modes_june_sept['Total'] / 1e6
```

```
# Preparing the data for the heatmap (with trips shown in millions)
heatmap_data_full = df_modes_june_sept.pivot_table(index=df_modes_june_sept['Fiscal Month'].dt.strftime('%Y'),
                                                   columns=df_modes_june_sept['Fiscal Month'].dt.strftime('%b'),
                                                   values='Total in Millions')

# Reindexing to ensure all months (Jun, Jul, Aug, Sep) are included
months_order = ['Jun', 'Jul', 'Aug', 'Sep']
heatmap_data_full = heatmap_data_full.reindex(columns=months_order)

# Plotting the heatmap
plt.figure(figsize=(12, 6))
ax = sns.heatmap(heatmap_data_full, annot=False, cmap='Blues', cbar_kws={'label': 'Total Trips (in Millions)'}, linewidths=0.5)

# Customising the plot
ax.set_title('Heatmap of Total Trips by Month (in Millions)', fontsize=16)
ax.set_xlabel('Month', fontsize=14)
ax.set_ylabel('Year', fontsize=14)

ax.set_xticklabels(ax.get_xticklabels(), rotation=0, horizontalalignment='center')

plt.tight_layout()

plt.show()
```

The purpose of this code is to generate a heatmap showing the total trips across specific months (June to September). First, the total trips are converted into millions, and the pivot_table method is used to structure the data with years and months as indices. The heatmap visualizes the yearly travel trends, using different shades to represent trip

volume differences, and the plot is then customized with appropriate labels and formatting.



Findings:

1. Overview: This heatmap shows the total trips by month (in millions) for June to September over the years 2022, 2023, and 2024.
2. Observation:
   A. June to September is consistently light in color, indicating that total trips were lower in 2022 (around 1.8 to 2 million trips across these months).
   B. In 2023 there is a moderate increase in trips, especially in July and August, which are darker compared to June and September. These months likely saw over 2.2 million trips.
   C. August 2024 is the darkest square on the heatmap, indicating the highest trip count, exceeding 2.4 million trips.
   D. June, July, and September 2024 are lighter, indicating trip counts closer to 2 million, but still higher than 2022 levels.
3. Conclusion: The heatmap clearly highlights that August 2024 saw a significant spike in public transport usage, with over 2.4 million trips, which coincides with the 50c fare trial. Other months, particularly in 2022 and early 2024, show fewer total trips, reinforcing the impact of fare changes and other factors on ridership behavior.

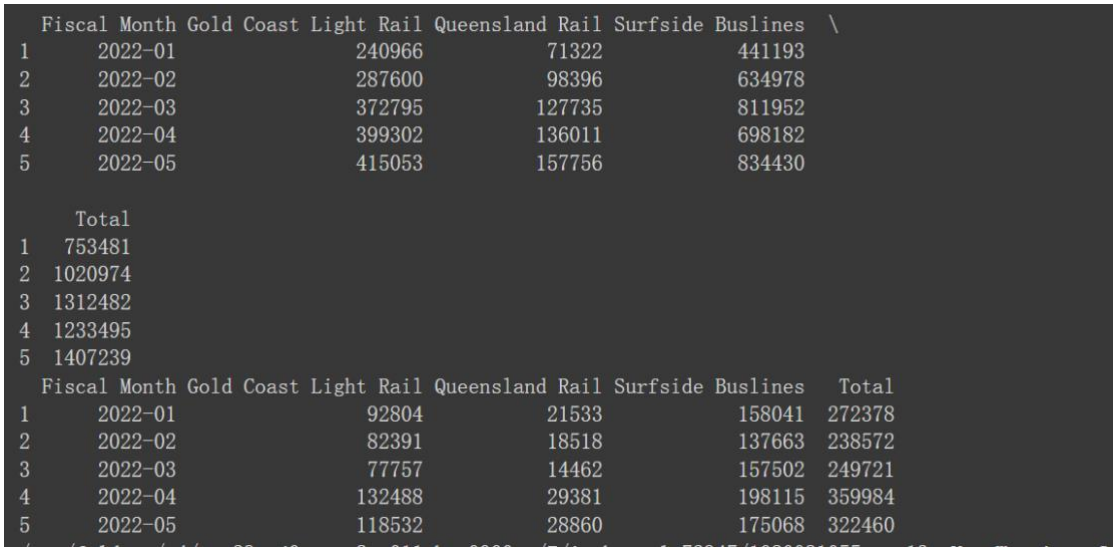
Analysing 'Weekday vs Weekend Public Transport Data'

Code:

```
# Loading the Excel files
file_path_2 = 'Weekday vs Weekend Public Transport Data.xlsx'

# Loading the Weekday and Weekend data
df_weekday = pd.read_excel(file_path_2, sheet_name='Weekday')
```

```
df_weekend = pd.read_excel(file_path_2, sheet_name='Weekend')

# Renaming columns
df_weekday.columns = ['Fiscal Month', 'Gold Coast Light Rail', 'Queensland Rail', 'Surfside Buslines', 'Total']
df_weekend.columns = ['Fiscal Month', 'Gold Coast Light Rail', 'Queensland Rail', 'Surfside Buslines', 'Total']

# Converting 'Fiscal Month' for both datasets to month-year format
df_weekday['Fiscal Month'] = pd.to_datetime(df_weekday['Fiscal Month'], errors='coerce').dt.to_period('M')
df_weekend['Fiscal Month'] = pd.to_datetime(df_weekend['Fiscal Month'], errors='coerce').dt.to_period('M')

# Droping rows with missing data
df_weekday_clean = df_weekday.dropna()
df_weekend_clean = df_weekend.dropna()

# Displaying the first few rows of each dataframe
print(df_weekday_clean.head())
print(df_weekend_clean.head())
```

This code loads weekday and weekend public transport data from an Excel file and processes the data. It starts by loading the data from two worksheets, renaming the columns, and converting the "Fiscal Month" column into a month-year format. The code also drops any rows containing missing data, and displays the first few rows of the cleaned datasets.



	Fiscal Month	Gold Coast Light Rail	Queensland Rail	Surfside Buslines	\
1	2022-01	240966	71322	441193	
2	2022-02	287600	98396	634978	
3	2022-03	372795	127735	811952	
4	2022-04	399302	136011	698182	
5	2022-05	415053	157756	834430	

	Total
1	753481
2	1020974
3	1312482
4	1233495
5	1407239

	Fiscal Month	Gold Coast Light Rail	Queensland Rail	Surfside Buslines	Total
1	2022-01	92804	21533	158041	272378
2	2022-02	82391	18518	137663	238572
3	2022-03	77757	14462	157502	249721
4	2022-04	132488	29381	198115	359984
5	2022-05	118532	28860	175068	322460

This image shows the first few rows of the loaded weekday and weekend public transport data. Each table presents the total number of passengers for different transport modes (Gold Coast Light Rail, Queensland Rail, Surfside Buslines, and Total) from January to May 2022. The tables display the initial few cleaned records, giving users an overview of the data's structure and contents.

Code:

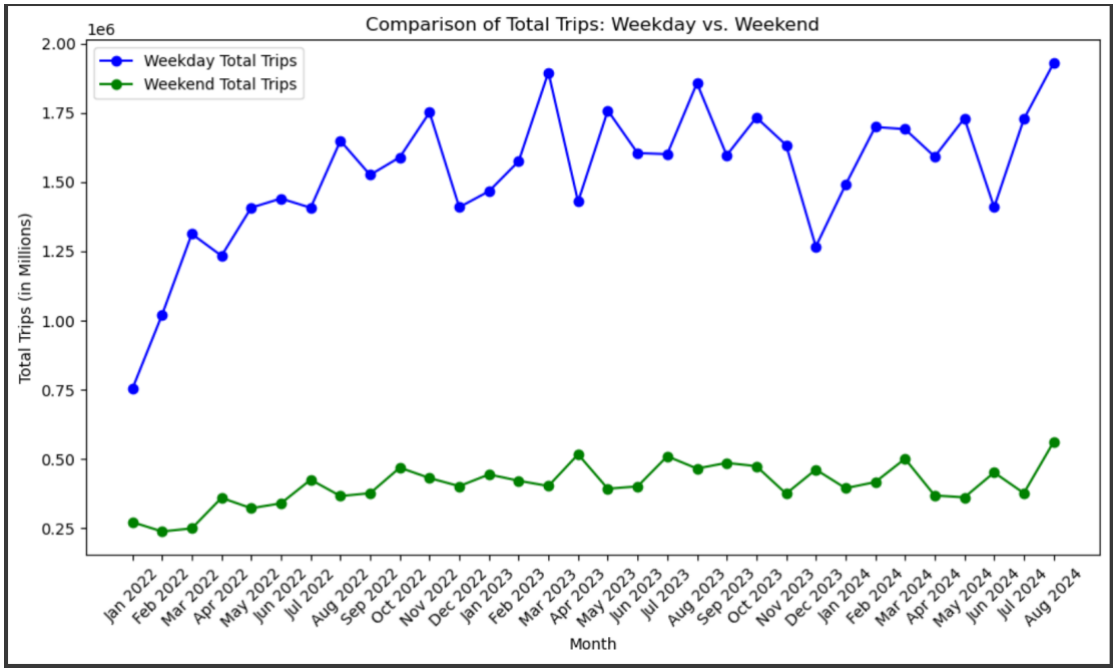
```
# Aggregating the total trips for weekdays and weekends to compare over time
weekday_total_trips = df_weekday_clean.groupby(df_weekday_clean['Fiscal Month']).sum()['Total']
weekend_total_trips = df_weekend_clean.groupby(df_weekend_clean['Fiscal Month']).sum()['Total']

# Creating a combined dataframe for both weekday and weekend total trips
comparison_df = pd.DataFrame({
    'Weekday Total Trips': weekday_total_trips,
    'Weekend Total Trips': weekend_total_trips
}).reset_index()

# Displaying
print(comparison_df)

# Visualizing the comparison with a line plot
plt.figure(figsize=(10, 6))
plt.plot(comparison_df['Fiscal Month'].dt.strftime('%b %Y'), comparison_df['Weekday Total Trips'], label='Weekday Total Trips', marker='o', color='blue')
plt.plot(comparison_df['Fiscal Month'].dt.strftime('%b %Y'), comparison_df['Weekend Total Trips'], label='Weekend Total Trips', marker='o', color='green')

# Customising the plot
```

plt.title('Comparison of Total Trips: Weekday vs. Weekend')
plt.xlabel('Month')
plt.ylabel('Total Trips (in Millions)')
plt.xticks(rotation=45)
plt.legend()

plt.tight_layout()
plt.show()

```
# Aggregating the total trips for weekdays and weekends to compare over time
weekday_total_trips = df_weekday_clean.groupby(df_weekday_clean['Fiscal Month']).sum()['Total']
weekend_total_trips = df_weekend_clean.groupby(df_weekend_clean['Fiscal Month']).sum()['Total']

# Creating a combined dataframe for both weekday and weekend total trips
comparison_df = pd.DataFrame({
        'Weekday Total Trips': weekday_total_trips,
        'Weekend Total Trips': weekend_total_trips
}).reset_index()

# Displaying
print(comparison_df)

# Visualizing the comparison with a line plot
plt.figure(figsize=(10, 6))
plt.plot(comparison_df['Fiscal Month'].dt.strftime('%b %Y'), comparison_df['Weekday Total Trips'], label='Weekday Total Trips', marker='o', color='blue')
plt.plot(comparison_df['Fiscal Month'].dt.strftime('%b %Y'), comparison_df['Weekend Total Trips'], label='Weekend Total Trips', marker='o', color='green')

# Customising the plot
plt.title('Comparison of Total Trips: Weekday vs. Weekend')
plt.xlabel('Month')
plt.ylabel('Total Trips (in Millions)')
plt.xticks(rotation=45)
plt.legend()

plt.tight_layout()
plt.show()
```

	Fiscal Month	Weekday Total Trips	Weekend Total Trips
0	2022-01	753481	272378
1	2022-02	1020974	238572
2	2022-03	1312482	249721
3	2022-04	1233495	359984
4	2022-05	1407239	322460
5	2022-06	1440215	340170
6	2022-07	1406820	425422
7	2022-08	1648552	366616
8	2022-09	1525410	376670
9	2022-10	1589260	468998
10	2022-11	1750978	431385
11	2022-12	1408818	402127
12	2023-01	1466512	444473
13	2023-02	1574723	421655
14	2023-03	1895393	402616
15	2023-04	1428894	516950
16	2023-05	1757618	392874
17	2023-06	1604438	401246
18	2023-07	1600390	510048
19	2023-08	1856500	465696
20	2023-09	1596185	486326
21	2023-10	1732687	474122
22	2023-11	1633199	375049
23	2023-12	1266720	462073
24	2024-01	1489558	394782
25	2024-02	1698382	416840
26	2024-03	1690587	500992
27	2024-04	1591376	368443
28	2024-05	1729825	361744
29	2024-06	1408526	451406
30	2024-07	1726686	376952
31	2024-08	1928351	561416



Observation:

1. The number of weekday total trips (represented by the blue line) is consistently higher than the number of weekend total trips (represented by the green line) over the observed period.
2. There is a noticeable upward trend in weekday trips starting from January 2022, reaching a peak around June 2023, with fluctuations afterward. The graph shows a slight increase again in mid-2024.
3. The weekday total trips exhibit more variability, with regular peaks and troughs. This indicates fluctuating travel patterns on weekdays, possibly influenced by seasonal changes, events, or holidays.
4. The weekend total trips show less variability compared to weekday trips, but there is a gradual increase over the same period. However, weekend trips seem more stable overall, without drastic fluctuations.
5. Both weekday and weekend trips show an increase starting from early 2024, with weekday trips having sharper increases and peaks, especially around mid-2024.
6. The gap between weekday and weekend travel volumes remains significant throughout the period, with weekday trips being approximately 3-4 times higher than weekend trips. This implies that public transport usage is heavily reliant on weekday commuters, likely driven by work-related travel.

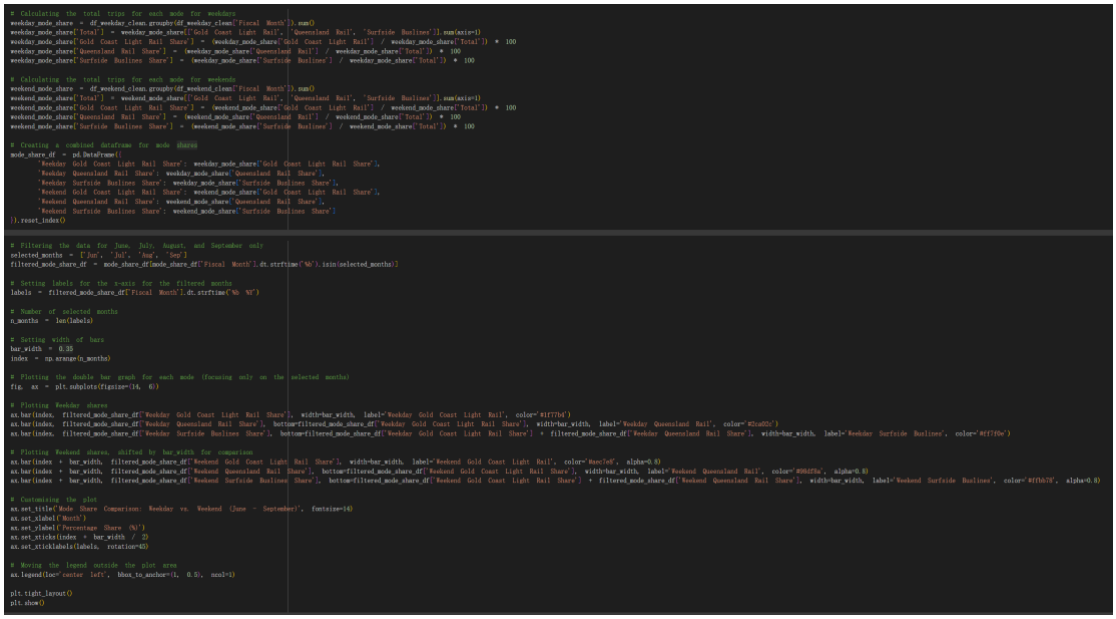
Mode share Analysis (Weekday vs Weekend)

Code:

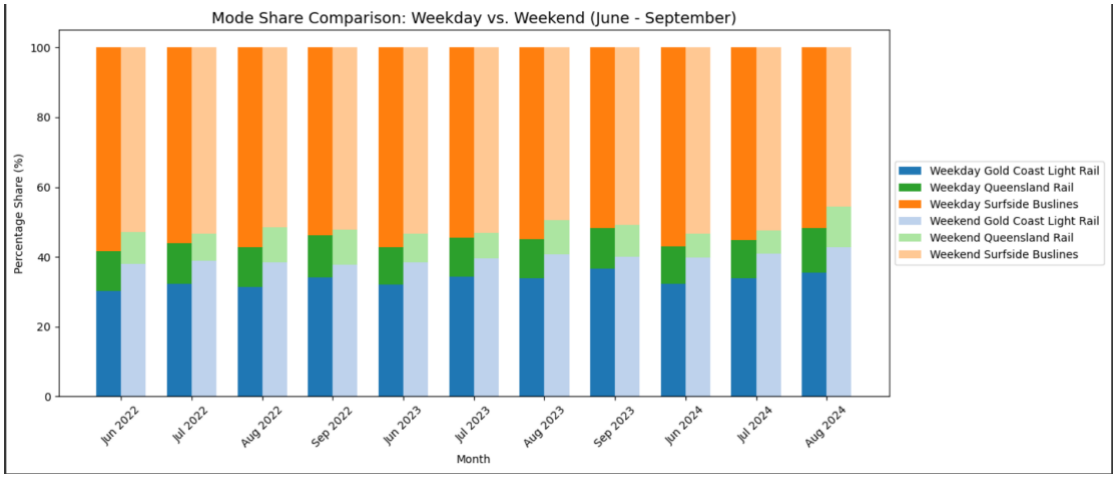
```
# Calculating the total trips for each mode for weekdays
weekday_mode_share = df_weekday_clean.groupby(df_weekday_clean['Fiscal Month']).sum()
weekday_mode_share['Total'] = weekday_mode_share[['Gold Coast Light Rail',
```

```
'Queensland Rail', 'Surfside Buslines']].sum(axis=1)
weekday_mode_share['Gold Coast Light Rail Share'] = (weekday_mode_share['Gold Coast Light Rail'] / weekday_mode_share['Total']) * 100
weekday_mode_share['Queensland Rail Share'] = (weekday_mode_share['Queensland Rail'] / weekday_mode_share['Total']) * 100
weekday_mode_share['Surfside Buslines Share'] = (weekday_mode_share['Surfside Buslines'] / weekday_mode_share['Total']) * 100

# Calculating the total trips for each mode for weekends
weekend_mode_share = df_weekend_clean.groupby(df_weekend_clean['Fiscal Month']).sum()
weekend_mode_share['Total'] = weekend_mode_share[['Gold Coast Light Rail', 'Queensland Rail', 'Surfside Buslines']].sum(axis=1)
weekend_mode_share['Gold Coast Light Rail Share'] = (weekend_mode_share['Gold Coast Light Rail'] / weekend_mode_share['Total']) * 100
weekend_mode_share['Queensland Rail Share'] = (weekend_mode_share['Queensland Rail'] / weekend_mode_share['Total']) * 100
weekend_mode_share['Surfside Buslines Share'] = (weekend_mode_share['Surfside Buslines'] / weekend_mode_share['Total']) * 100

# Creating a combined dataframe for mode shares
mode_share_df = pd.DataFrame({
    'Weekday Gold Coast Light Rail Share': weekday_mode_share['Gold Coast Light Rail Share'],
    'Weekday Queensland Rail Share': weekday_mode_share['Queensland Rail Share'],
     'Weekday Surfside Buslines Share': weekday_mode_share['Surfside Buslines Share'],
    'Weekend Gold Coast Light Rail Share': weekend_mode_share['Gold Coast Light Rail Share'],
    'Weekend Queensland Rail Share': weekend_mode_share['Queensland Rail Share'],
     'Weekend Surfside Buslines Share': weekend_mode_share['Surfside Buslines Share']
}).reset_index()
# Filtering the data for June, July, August, and September only
selected_months = ['Jun', 'Jul', 'Aug', 'Sep']
filtered_mode_share_df = mode_share_df[mode_share_df['Fiscal Month'].dt.strftime('%b').isin(selected_months)]

# Setting labels for the x-axis for the filtered months
labels = filtered_mode_share_df['Fiscal Month'].dt.strftime('%b %Y')

# Number of selected months
n_months = len(labels)
```

```
# Setting width of bars
bar_width = 0.35
index = np.arange(n_months)

# Plotting the double bar graph for each mode (focusing only on the selected months)
fig, ax = plt.subplots(figsize=(14, 6))

# Plotting Weekday shares
ax.bar(index, filtered_mode_share_df['Weekday Gold Coast Light Rail Share'], width=bar_width, label='Weekday Gold Coast Light Rail', color='#1f77b4')
ax.bar(index, filtered_mode_share_df['Weekday Queensland Rail Share'], bottom=filtered_mode_share_df['Weekday Gold Coast Light Rail Share'], width=bar_width, label='Weekday Queensland Rail', color='#2ca02c')
ax.bar(index, filtered_mode_share_df['Weekday Surfside Buslines Share'], bottom=filtered_mode_share_df['Weekday Gold Coast Light Rail Share'] + filtered_mode_share_df['Weekday Queensland Rail Share'], width=bar_width, label='Weekday Surfside Buslines', color='#ff7f0e')

# Plotting Weekend shares, shifted by bar_width for comparison
ax.bar(index + bar_width, filtered_mode_share_df['Weekend Gold Coast Light Rail Share'], width=bar_width, label='Weekend Gold Coast Light Rail', color='#aec7e8', alpha=0.8)
ax.bar(index + bar_width, filtered_mode_share_df['Weekend Queensland Rail Share'], bottom=filtered_mode_share_df['Weekend Gold Coast Light Rail Share'], width=bar_width, label='Weekend Queensland Rail', color='#98df8a', alpha=0.8)
ax.bar(index + bar_width, filtered_mode_share_df['Weekend Surfside Buslines Share'], bottom=filtered_mode_share_df['Weekend Gold Coast Light Rail Share'] + filtered_mode_share_df['Weekend Queensland Rail Share'], width=bar_width, label='Weekend Surfside Buslines', color='#ffbb78', alpha=0.8)

# Customising the plot
ax.set_title('Mode Share Comparison: Weekday vs. Weekend (June - September)', fontsize=14)
ax.set_xlabel('Month')
ax.set_ylabel('Percentage Share (%)')
ax.set_xticks(index + bar_width / 2)
ax.set_xticklabels(labels, rotation=45)

# Moving the legend outside the plot area
ax.legend(loc='center left', bbox_to_anchor=(1, 0.5), ncol=1)

plt.tight_layout()
plt.show()
```



This code calculates and visualizes the mode share of different public transport modes (Gold Coast Light Rail, Queensland Rail, Surfside Buslines) for weekdays and weekends. It first calculates the total trips and percentage shares for each mode on both weekdays and weekends. Then, it filters the data for the months of June to September, creating a dataset comparing weekday and weekend mode shares. Finally, it presents the comparison using a double bar chart, clearly illustrating the differences in mode usage between weekdays and weekends.



Observation:

The mode share comparison between weekdays and weekends from June to September across multiple years shows that **Surfside Buslines** (both weekday and weekend) consistently dominate the public transport mode share, occupying the largest portion of the total trips. **Queensland Rail** and **Gold Coast Light Rail** have a relatively stable share but are more prominent on weekdays compared to weekends. The weekday mode share for both **Gold Coast Light Rail** and **Queensland Rail** is significantly higher than their weekend counterparts, indicating higher reliance on these modes during weekdays, likely due to commuting patterns. Across the years, there is no drastic shift

in mode preferences, but some seasonal variations are visible.

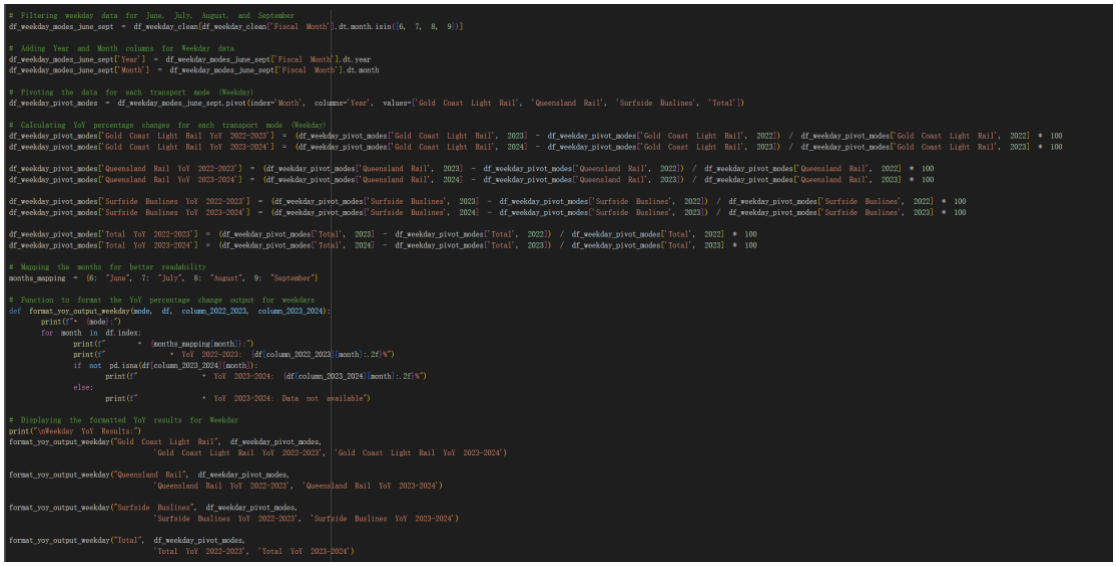
YoY % Change Analysis

Weekday Analysis

Code:

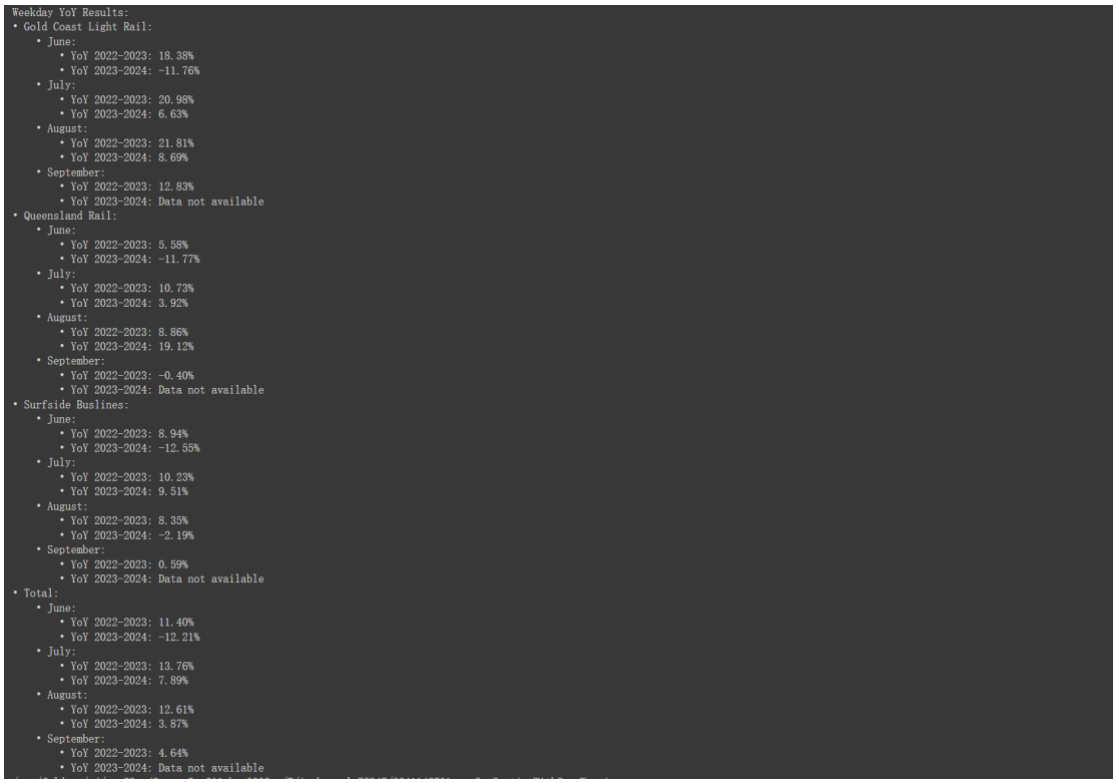
```
# Filtering weekday data for June, July, August, and September
df_weekday_modes_june_sept = df_weekday_clean[df_weekday_clean['Fiscal Month'].dt.month.isin([6, 7, 8, 9])]

# Adding Year and Month columns for Weekday data
df_weekday_modes_june_sept['Year'] = df_weekday_modes_june_sept['Fiscal Month'].dt.year
df_weekday_modes_june_sept['Month'] = df_weekday_modes_june_sept['Fiscal Month'].dt.month

# Pivoting the data for each transport mode (Weekday)
df_weekday_pivot_modes = df_weekday_modes_june_sept.pivot(index='Month', columns='Year', values=['Gold Coast Light Rail', 'Queensland Rail', 'Surfside Buslines', 'Total'])

# Calculating YoY percentage changes for each transport mode (Weekday)
df_weekday_pivot_modes['Gold Coast Light Rail YoY 2022-2023'] = (df_weekday_pivot_modes['Gold Coast Light Rail', 2023] - df_weekday_pivot_modes['Gold Coast Light Rail', 2022]) / df_weekday_pivot_modes['Gold Coast Light Rail', 2022] * 100
df_weekday_pivot_modes['Gold Coast Light Rail YoY 2023-2024'] = (df_weekday_pivot_modes['Gold Coast Light Rail', 2024] - df_weekday_pivot_modes['Gold Coast Light Rail', 2023]) / df_weekday_pivot_modes['Gold Coast Light Rail', 2023] * 100

df_weekday_pivot_modes['Queensland Rail YoY 2022-2023'] = (df_weekday_pivot_modes['Queensland Rail', 2023] - df_weekday_pivot_modes['Queensland Rail', 2022]) / df_weekday_pivot_modes['Queensland Rail', 2022] * 100
df_weekday_pivot_modes['Queensland Rail YoY 2023-2024'] = (df_weekday_pivot_modes['Queensland Rail', 2024] - df_weekday_pivot_modes['Queensland Rail', 2023]) / df_weekday_pivot_modes['Queensland Rail', 2023] * 100

df_weekday_pivot_modes['Surfside Buslines YoY 2022-2023'] = (df_weekday_pivot_modes['Surfside Buslines', 2023] -
```

```
df_weekday_pivot_modes['Surfside Buslines', 2022]) /
df_weekday_pivot_modes['Surfside Buslines', 2022] * 100
df_weekday_pivot_modes['Surfside Buslines YoY 2023-2024'] =
(df_weekday_pivot_modes['Surfside Buslines', 2024] -
df_weekday_pivot_modes['Surfside Buslines', 2023]) /
df_weekday_pivot_modes['Surfside Buslines', 2023] * 100

df_weekday_pivot_modes['Total YoY 2022-2023'] =
(df_weekday_pivot_modes['Total', 2023] - df_weekday_pivot_modes['Total', 2022]) /
df_weekday_pivot_modes['Total', 2022] * 100
df_weekday_pivot_modes['Total YoY 2023-2024'] =
(df_weekday_pivot_modes['Total', 2024] - df_weekday_pivot_modes['Total', 2023]) /
df_weekday_pivot_modes['Total', 2023] * 100

# Mapping the months for better readability
months_mapping = {6: "June", 7: "July", 8: "August", 9: "September"}

# Function to format the YoY percentage change output for weekdays
def format_yoy_output_weekday(mode, df, column_2022_2023, column_2023_2024):
    print(f"• {mode}:")
    for month in df.index:
        print(f"    • {months_mapping[month]}:")
        print(f"        • YoY 2022-2023: {df[column_2022_2023][month]:.2f}%")
        if not pd.isna(df[column_2023_2024][month]):
            print(f"        • YoY 2023-2024: {df[column_2023_2024][month]:.2f}%")
        else:
            print(f"        • YoY 2023-2024: Data not available")

# Displaying the formatted YoY results for Weekday
print("\nWeekday YoY Results:")
format_yoy_output_weekday("Gold Coast Light Rail", df_weekday_pivot_modes,
                  'Gold Coast Light Rail YoY 2022-2023', 'Gold Coast Light Rail YoY
2023-2024')

format_yoy_output_weekday("Queensland Rail", df_weekday_pivot_modes,
                  'Queensland Rail YoY 2022-2023', 'Queensland Rail YoY 2023-2024')

format_yoy_output_weekday("Surfside Buslines", df_weekday_pivot_modes,
                   'Surfside Buslines YoY 2022-2023', 'Surfside Buslines YoY 2023-
2024')

format_yoy_output_weekday("Total", df_weekday_pivot_modes,
                  'Total YoY 2022-2023', 'Total YoY 2023-2024')
```



```
# Filtering weekday data for June, July, August, and September
df_weekday_modes_june_sept = df_weekday_clean[df_weekday_clean['Fiscal Month'].dt.month.isin([6, 7, 8, 9])]

# Adding Year and Month columns for Weekday data
df_weekday_modes_june_sept['Year'] = df_weekday_modes_june_sept['Fiscal Month'].dt.year
df_weekday_modes_june_sept['Month'] = df_weekday_modes_june_sept['Fiscal Month'].dt.month

# Pivoting the data for each transport mode (Weekday)
df_weekday_pivot_modes = df_weekday_modes_june_sept.pivot(index='Month', columns='Year', values=['Gold Coast Light Rail', 'Queensland Rail', 'Surfside Buslines', 'Total'])

# Calculating YoY percentage changes for each transport mode (Weekday)
df_weekday_pivot_modes['Gold Coast Light Rail YoY 2022-2023'] = (df_weekday_pivot_modes['Gold Coast Light Rail', 2023] - df_weekday_pivot_modes['Gold Coast Light Rail', 2022]) / df_weekday_pivot_modes['Gold Coast Light Rail', 2022] * 100
df_weekday_pivot_modes['Gold Coast Light Rail YoY 2023-2024'] = (df_weekday_pivot_modes['Gold Coast Light Rail', 2024] - df_weekday_pivot_modes['Gold Coast Light Rail', 2023]) / df_weekday_pivot_modes['Gold Coast Light Rail', 2023] * 100

df_weekday_pivot_modes['Queensland Rail YoY 2022-2023'] = (df_weekday_pivot_modes['Queensland Rail', 2023] - df_weekday_pivot_modes['Queensland Rail', 2022]) / df_weekday_pivot_modes['Queensland Rail', 2022] * 100
df_weekday_pivot_modes['Queensland Rail YoY 2023-2024'] = (df_weekday_pivot_modes['Queensland Rail', 2024] - df_weekday_pivot_modes['Queensland Rail', 2023]) / df_weekday_pivot_modes['Queensland Rail', 2023] * 100

df_weekday_pivot_modes['Surfside Buslines YoY 2022-2023'] = (df_weekday_pivot_modes['Surfside Buslines', 2023] - df_weekday_pivot_modes['Surfside Buslines', 2022]) / df_weekday_pivot_modes['Surfside Buslines', 2022] * 100
df_weekday_pivot_modes['Surfside Buslines YoY 2023-2024'] = (df_weekday_pivot_modes['Surfside Buslines', 2024] - df_weekday_pivot_modes['Surfside Buslines', 2023]) / df_weekday_pivot_modes['Surfside Buslines', 2023] * 100

df_weekday_pivot_modes['Total YoY 2022-2023'] = (df_weekday_pivot_modes['Total', 2023] - df_weekday_pivot_modes['Total', 2022]) / df_weekday_pivot_modes['Total', 2022] * 100
df_weekday_pivot_modes['Total YoY 2023-2024'] = (df_weekday_pivot_modes['Total', 2024] - df_weekday_pivot_modes['Total', 2023]) / df_weekday_pivot_modes['Total', 2023] * 100

# Mapping the months for better readability
months_mapping = {6: "June", 7: "July", 8: "August", 9: "September"}

# Function to format the YoY percentage change output for weekdays
def format_yoy_output_weekday(mode, df, column_2022_2023, column_2023_2024):
    print(f"- {mode}:")
    for month in df.index:
        print(f"    - {months_mapping[month]}:")
        print(f"        - YoY 2022-2023: {df[column_2022_2023][month]:.2f}%")
        if not pd.isna(df[column_2023_2024][month]):
            print(f"        - YoY 2023-2024: {df[column_2023_2024][month]:.2f}%")
        else:
            print(f"        - YoY 2023-2024: Data not available")

# Displaying the formatted YoY results for Weekday
print("\nWeekday YoY Results:")
format_yoy_output_weekday("Gold Coast Light Rail", df_weekday_pivot_modes,
                          'Gold Coast Light Rail YoY 2022-2023', 'Gold Coast Light Rail YoY 2023-2024')

format_yoy_output_weekday("Queensland Rail", df_weekday_pivot_modes,
                          'Queensland Rail YoY 2022-2023', 'Queensland Rail YoY 2023-2024')

format_yoy_output_weekday("Surfside Buslines", df_weekday_pivot_modes,
                          'Surfside Buslines YoY 2022-2023', 'Surfside Buslines YoY 2023-2024')

format_yoy_output_weekday("Total", df_weekday_pivot_modes,
                          'Total YoY 2022-2023', 'Total YoY 2023-2024')
```

This code calculates and outputs the year-over-year (YoY) percentage changes for different public transport modes (Gold Coast Light Rail, Queensland Rail, Surfside Buslines) on weekdays during June to September. The code first filters the data and uses pivot tables to calculate the YoY percentage changes for 2022-2023 and 2023-2024. Then, it formats the results using the format_yoy_output_weekday function, displaying the percentage changes for each transport mode and the total trips, organized by month.



This image presents the Year-over-Year (YoY) changes calculated from weekday data. The chart illustrates the percentage changes for various public transport modes (e.g., Gold Coast Light Rail, Queensland Rail, and Surfside Buslines) between 2022-2023 and 2023-2024 for the months of June to September. These YoY changes provide insights into the monthly passenger trends, helping to understand the shifts in transport usage across different years and months.

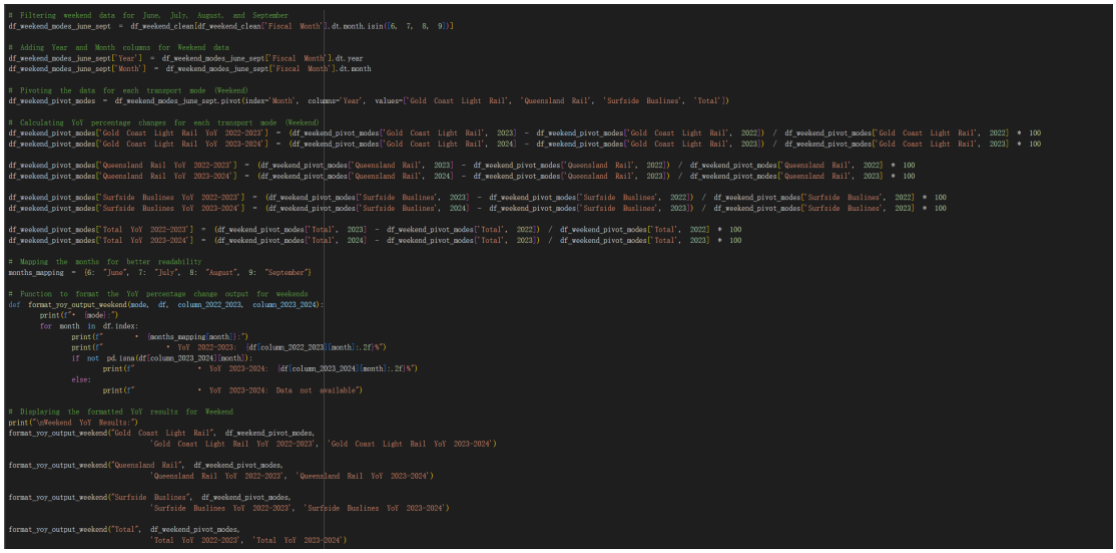
Weekend Analysis

Code:

```
# Filtering weekend data for June, July, August, and September
df_weekend_modes_june_sept = df_weekend_clean[df_weekend_clean['Fiscal Month'].dt.month.isin([6, 7, 8, 9])]

# Adding Year and Month columns for Weekend data
df_weekend_modes_june_sept['Year'] = df_weekend_modes_june_sept['Fiscal Month'].dt.year
df_weekend_modes_june_sept['Month'] = df_weekend_modes_june_sept['Fiscal Month'].dt.month

# Pivoting the data for each transport mode (Weekend)
df_weekend_pivot_modes = df_weekend_modes_june_sept.pivot(index='Month', columns='Year', values=['Gold Coast Light Rail', 'Queensland Rail', 'Surfside Buslines', 'Total'])
```

```
# Calculating YoY percentage changes for each transport mode (Weekend)
df_weekend_pivot_modes['Gold Coast Light Rail YoY 2022-2023'] =
(df_weekend_pivot_modes['Gold Coast Light Rail', 2023] -
df_weekend_pivot_modes['Gold Coast Light Rail', 2022]) /
df_weekend_pivot_modes['Gold Coast Light Rail', 2022] * 100
df_weekend_pivot_modes['Gold Coast Light Rail YoY 2023-2024'] =
(df_weekend_pivot_modes['Gold Coast Light Rail', 2024] -
df_weekend_pivot_modes['Gold Coast Light Rail', 2023]) /
df_weekend_pivot_modes['Gold Coast Light Rail', 2023] * 100

df_weekend_pivot_modes['Queensland Rail YoY 2022-2023'] =
(df_weekend_pivot_modes['Queensland Rail', 2023] -
df_weekend_pivot_modes['Queensland Rail', 2022]) /
df_weekend_pivot_modes['Queensland Rail', 2022] * 100
df_weekend_pivot_modes['Queensland Rail YoY 2023-2024'] =
(df_weekend_pivot_modes['Queensland Rail', 2024] -
df_weekend_pivot_modes['Queensland Rail', 2023]) /
df_weekend_pivot_modes['Queensland Rail', 2023] * 100

df_weekend_pivot_modes['Surfside Buslines YoY 2022-2023'] =
(df_weekend_pivot_modes['Surfside Buslines', 2023] -
df_weekend_pivot_modes['Surfside Buslines', 2022]) /
df_weekend_pivot_modes['Surfside Buslines', 2022] * 100
df_weekend_pivot_modes['Surfside Buslines YoY 2023-2024'] =
(df_weekend_pivot_modes['Surfside Buslines', 2024] -
df_weekend_pivot_modes['Surfside Buslines', 2023]) /
df_weekend_pivot_modes['Surfside Buslines', 2023] * 100

df_weekend_pivot_modes['Total YoY 2022-2023'] =
(df_weekend_pivot_modes['Total', 2023] - df_weekend_pivot_modes['Total', 2022]) /
df_weekend_pivot_modes['Total', 2022] * 100
df_weekend_pivot_modes['Total YoY 2023-2024'] =
(df_weekend_pivot_modes['Total', 2024] - df_weekend_pivot_modes['Total', 2023]) /
df_weekend_pivot_modes['Total', 2023] * 100

# Mapping the months for better readability
months_mapping = {6: "June", 7: "July", 8: "August", 9: "September"}

# Function to format the YoY percentage change output for weekends
def format_yoy_output_weekend(mode, df, column_2022_2023, column_2023_2024):
    print(f"• {mode}:")
    for month in df.index:
        print(f"    • {months_mapping[month]}:")
        print(f"        • YoY 2022-2023: {df[column_2022_2023][month]:.2f}%")
```

```
        if not pd.isna(df[column_2023_2024][month]):
            print(f"          • YoY 2023-2024: {df[column_2023_2024][month]:.2f}%")
        else:
            print(f"          • YoY 2023-2024: Data not available")

# Displaying the formatted YoY results for Weekend
print("\nWeekend YoY Results:")
format_yoy_output_weekend("Gold Coast Light Rail", df_weekend_pivot_modes,
                          'Gold Coast Light Rail YoY 2022-2023', 'Gold Coast Light Rail YoY
2023-2024')

format_yoy_output_weekend("Queensland Rail", df_weekend_pivot_modes,
                          'Queensland Rail YoY 2022-2023', 'Queensland Rail YoY 2023-2024')

format_yoy_output_weekend("Surfside Buslines", df_weekend_pivot_modes,
                           'Surfside Buslines YoY 2022-2023', 'Surfside Buslines YoY 2023-
2024')

format_yoy_output_weekend("Total", df_weekend_pivot_modes,
                          'Total YoY 2022-2023', 'Total YoY 2023-2024')
```



This code filters weekend data and calculates the Year-over-Year (YoY) changes for the Gold Coast Light Rail, Queensland Rail, and Surfside Buslines from 2022 to 2024 for June to September. The code first pivots the data for each transport mode and calculates the YoY percentage changes for 2022-2023 and 2023-2024. It then maps the months for readability, formats the output, and displays the YoY changes for each mode on weekends.

This image illustrates the year-over-year (YoY) percentage changes for different transport modes on weekends. The chart covers three modes: Gold Coast Light Rail, Queensland Rail, and Surfside Buslines, as well as the total trips. The YoY changes are shown month by month, detailing the growth or decline between 2022-2023 and 2023-2024. It provides insights into the trends of public transport usage over different years.

Plotting for Weekday

Code:

```
# Prepare data for plotting the YoY changes (Weekday and Weekend)
months = ['June', 'July', 'August', 'September']
colors_2022_2023 = ['blue', 'green', 'orange', 'purple']   # Colors for YoY 2022-2023
colors_2023_2024 = ['cyan', 'lime', 'coral', 'magenta']    # Colors for YoY 2023-2024

# Plotting the Weekday YoY changes
plt.figure(figsize=(12, 6))

plt.plot(months, df_weekday_pivot_modes['Gold Coast Light Rail YoY 2022-2023'], label='Weekday Gold Coast Light Rail YoY 2022-2023', marker='o', color=colors_2022_2023[0])
plt.plot(months, df_weekday_pivot_modes['Gold Coast Light Rail YoY 2023-2024'], label='Weekday Gold Coast Light Rail YoY 2023-2024', linestyle='--', marker='o', color=colors_2023_2024[0])

plt.plot(months, df_weekday_pivot_modes['Queensland Rail YoY 2022-2023'], label='Weekday Queensland Rail YoY 2022-2023', marker='o', color=colors_2022_2023[1])
plt.plot(months, df_weekday_pivot_modes['Queensland Rail YoY 2023-2024'], label='Weekday Queensland Rail YoY 2023-2024', linestyle='--', marker='o',
```

color=colors_2023_2024[1])

plt.plot(months, df_weekday_pivot_modes['Surfside Buslines YoY 2022-2023'], label='Weekday Surfside Buslines YoY 2022-2023', marker='o', color=colors_2022_2023[2])
plt.plot(months, df_weekday_pivot_modes['Surfside Buslines YoY 2023-2024'], label='Weekday Surfside Buslines YoY 2023-2024', linestyle='--', marker='o', color=colors_2023_2024[2])

plt.plot(months, df_weekday_pivot_modes['Total YoY 2022-2023'], label='Weekday Total YoY 2022-2023', marker='o', color=colors_2022_2023[3])
plt.plot(months, df_weekday_pivot_modes['Total YoY 2023-2024'], label='Weekday Total YoY 2023-2024', linestyle='--', marker='o', color=colors_2023_2024[3])

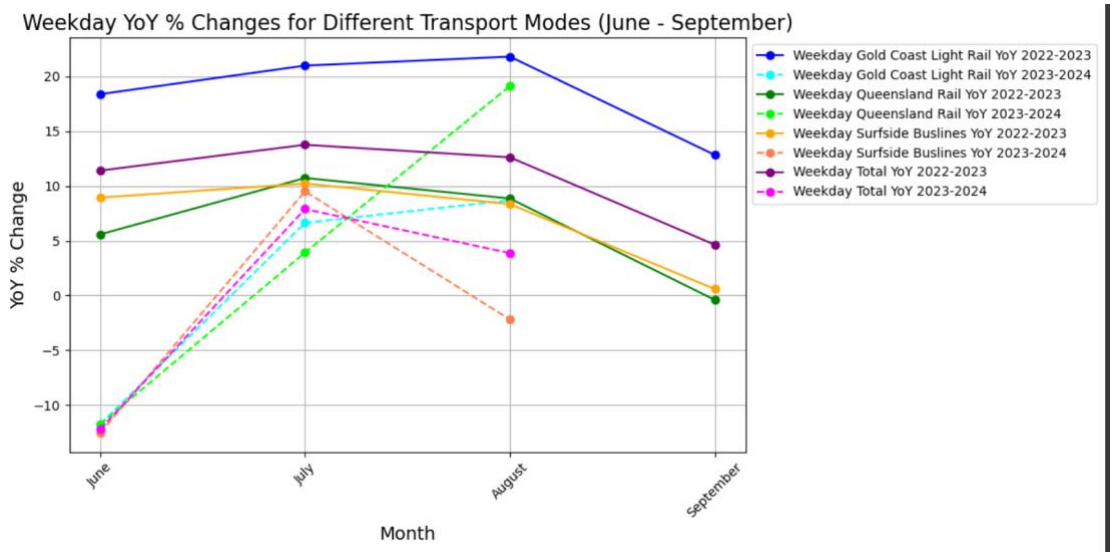
# Customising the plot for Weekday YoY
plt.title('Weekday YoY % Changes for Different Transport Modes (June - September)', fontsize=16)
plt.xlabel('Month', fontsize=14)
plt.ylabel('YoY % Change', fontsize=14)
plt.xticks(rotation=45)
plt.legend(loc='upper left', bbox_to_anchor=(1, 1)) # Put the legend outside the plot for clarity
plt.grid(True)

# Show the Weekday plot
plt.tight_layout()
plt.show()

```
# Prepare data for plotting the YoY changes (Weekday and Weekend)
months = ['June', 'July', 'August', 'September']
colors_2022_2023 = ['blue', 'green', 'orange', 'purple']   # Colors for YoY 2022-2023
colors_2023_2024 = ['cyan', 'lime', 'coral', 'magenta']    # Colors for YoY 2023-2024

# Plotting the Weekday YoY changes
plt.figure(figsize=(12, 6))

plt.plot(months, df_weekday_pivot_modes['Gold Coast Light Rail YoY 2022-2023'], label='Weekday Gold Coast Light Rail YoY 2022-2023', marker='o', color=colors_2022_2023[0])
plt.plot(months, df_weekday_pivot_modes['Gold Coast Light Rail YoY 2023-2024'], label='Weekday Gold Coast Light Rail YoY 2023-2024', linestyle='--', marker='o', color=colors_2023_2024[0])

plt.plot(months, df_weekday_pivot_modes['Queensland Rail YoY 2022-2023'], label='Weekday Queensland Rail YoY 2022-2023', marker='o', color=colors_2022_2023[1])
plt.plot(months, df_weekday_pivot_modes['Queensland Rail YoY 2023-2024'], label='Weekday Queensland Rail YoY 2023-2024', linestyle='--', marker='o', color=colors_2023_2024[1])

plt.plot(months, df_weekday_pivot_modes['Surfside Buslines YoY 2022-2023'], label='Weekday Surfside Buslines YoY 2022-2023', marker='o', color=colors_2022_2023[2])
plt.plot(months, df_weekday_pivot_modes['Surfside Buslines YoY 2023-2024'], label='Weekday Surfside Buslines YoY 2023-2024', linestyle='--', marker='o', color=colors_2023_2024[2])

plt.plot(months, df_weekday_pivot_modes['Total YoY 2022-2023'], label='Weekday Total YoY 2022-2023', marker='o', color=colors_2022_2023[3])
plt.plot(months, df_weekday_pivot_modes['Total YoY 2023-2024'], label='Weekday Total YoY 2023-2024', linestyle='--', marker='o', color=colors_2023_2024[3])

# Customising the plot for Weekday YoY
plt.title('Weekday YoY % Changes for Different Transport Modes (June - September)', fontsize=16)
plt.xlabel('Month', fontsize=14)
plt.ylabel('YoY % Change', fontsize=14)
plt.xticks(rotation=45)
plt.legend(loc='upper left', bbox_to_anchor=(1, 1))  # Put the legend outside the plot for clarity
plt.grid(True)

# Show the Weekday plot
plt.tight_layout()
plt.show()
```

This code plots a line graph showing the Year-over-Year (YoY) percentage changes for various transport modes on weekdays, covering the four months from June to September. Different colors are used to distinguish between the changes from 2022-2023 and 2023-2024 for Gold Coast Light Rail, Queensland Rail, Surfside Buslines, and the total trips. Each line represents the YoY change for a transport mode across different months, using different colors and styles (solid and dashed) to represent the

trends over the two time periods.



Findings:

1. Overview: This line graph presents the Year-over-Year (YoY) percentage changes for weekday trips across different transport modes from June to September, comparing the years 2022-2023 and 2023-2024.
2. Observation:
   A. Gold Coast Light Rail 2022 - 23 (Blue) shows steady growth, peaking at around 20% in August, before a slight decline in September.
   B. Gold Coast Light Rail 2023 - 24 (Green) starts lower in June (~5% YoY change), increases significantly in August (~15% growth), and then declines in September.
   C. Queensland Rail 2022 - 23 (Dark Green) shows a steady increase from 5% in June to around 10% in August, followed by a small decline in September.
   D. Queensland Rail 2023 - 24 (Light Green) shows a similar pattern to 2022-2023, starting low (~3%) in June and peaking in August (~10%), but slightly more volatile, with a notable drop after August.
   E. Surfside Buslines 2022 - 23 (Solid Orange) shows consistent growth, peaking at around 12% in August, with a slight dip in September.
   F. Surfside Buslines 2023 - 24 (Dashed Orange) starts low with a -10% change in June but recovers quickly, peaking in August (~15%) before dipping again in September.
   G. Total YoY 2022 - 23 (Purple) shows a steady growth from June (~7%) to August (~10%) before a decline in September.
   H. Total YoY 2023 - 24 (Pink) starts with a -10% decline in June but recovers strongly in August to about 8%, followed by another dip in September.
3. Conclusion: The August 2024 data (when the 50c fare trial occurred) shows a clear peak across all transport modes, particularly for Gold Coast Light Rail and Surfside Buslines, indicating that the trial led to a strong growth in ridership.

Plotting for Weekend

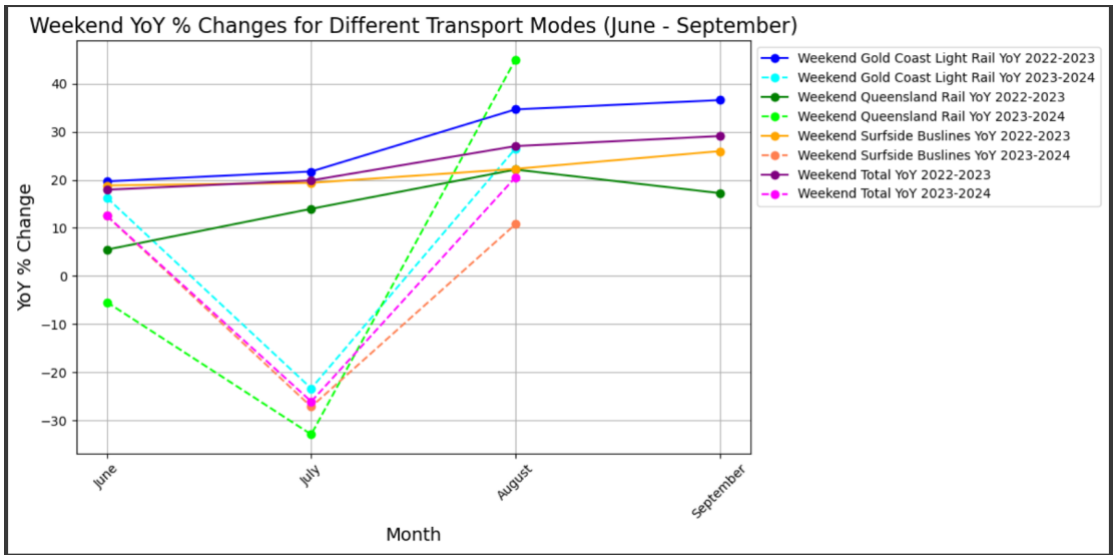
Code:

```
# Plotting the Weekend YoY changes
plt.figure(figsize=(12, 6))

plt.plot(months, df_weekend_pivot_modes['Gold Coast Light Rail YoY 2022-2023'], label='Weekend Gold Coast Light Rail YoY 2022-2023', marker='o', color=colors_2022_2023[0])
plt.plot(months, df_weekend_pivot_modes['Gold Coast Light Rail YoY 2023-2024'], label='Weekend Gold Coast Light Rail YoY 2023-2024', linestyle='--', marker='o', color=colors_2023_2024[0])

plt.plot(months, df_weekend_pivot_modes['Queensland Rail YoY 2022-2023'], label='Weekend Queensland Rail YoY 2022-2023', marker='o', color=colors_2022_2023[1])
plt.plot(months, df_weekend_pivot_modes['Queensland Rail YoY 2023-2024'], label='Weekend Queensland Rail YoY 2023-2024', linestyle='--', marker='o', color=colors_2023_2024[1])

plt.plot(months, df_weekend_pivot_modes['Surfside Buslines YoY 2022-2023'], label='Weekend Surfside Buslines YoY 2022-2023', marker='o', color=colors_2022_2023[2])
plt.plot(months, df_weekend_pivot_modes['Surfside Buslines YoY 2023-2024'], label='Weekend Surfside Buslines YoY 2023-2024', linestyle='--', marker='o', color=colors_2023_2024[2])

plt.plot(months, df_weekend_pivot_modes['Total YoY 2022-2023'], label='Weekend Total YoY 2022-2023', marker='o', color=colors_2022_2023[3])
plt.plot(months, df_weekend_pivot_modes['Total YoY 2023-2024'], label='Weekend Total YoY 2023-2024', linestyle='--', marker='o', color=colors_2023_2024[3])

# Customising the plot for Weekend YoY
plt.title('Weekend YoY % Changes for Different Transport Modes (June - September)', fontsize=16)
plt.xlabel('Month', fontsize=14)
plt.ylabel('YoY % Change', fontsize=14)
plt.xticks(rotation=45)
plt.legend(loc='upper left', bbox_to_anchor=(1, 1))  # Put the legend outside the plot for clarity
plt.grid(True)

# Show the Weekend plot
plt.tight_layout()
```

plt.show()

```
# Plotting the Weekend YoY changes
plt.figure(figsize=(12, 6))

plt.plot(months, df_weekend_pivot_modes['Gold Coast Light Rail YoY 2022-2023'], label='Weekend Gold Coast Light Rail YoY 2022-2023', marker='o', color=colors_2022_2023[0])
plt.plot(months, df_weekend_pivot_modes['Gold Coast Light Rail YoY 2023-2024'], label='Weekend Gold Coast Light Rail YoY 2023-2024', linestyle='--', marker='o', color=colors_2023_2024[0])

plt.plot(months, df_weekend_pivot_modes['Queensland Rail YoY 2022-2023'], label='Weekend Queensland Rail YoY 2022-2023', marker='o', color=colors_2022_2023[1])
plt.plot(months, df_weekend_pivot_modes['Queensland Rail YoY 2023-2024'], label='Weekend Queensland Rail YoY 2023-2024', linestyle='--', marker='o', color=colors_2023_2024[1])

plt.plot(months, df_weekend_pivot_modes['Surfside Buslines YoY 2022-2023'], label='Weekend Surfside Buslines YoY 2022-2023', marker='o', color=colors_2022_2023[2])
plt.plot(months, df_weekend_pivot_modes['Surfside Buslines YoY 2023-2024'], label='Weekend Surfside Buslines YoY 2023-2024', linestyle='--', marker='o', color=colors_2023_2024[2])

plt.plot(months, df_weekend_pivot_modes['Total YoY 2022-2023'], label='Weekend Total YoY 2022-2023', marker='o', color=colors_2022_2023[3])
plt.plot(months, df_weekend_pivot_modes['Total YoY 2023-2024'], label='Weekend Total YoY 2023-2024', linestyle='--', marker='o', color=colors_2023_2024[3])

# Customising the plot for Weekend YoY
plt.title('Weekend YoY % Changes for Different Transport Modes (June - September)', fontsize=16)
plt.xlabel('Month', fontsize=14)
plt.ylabel('YoY % Change', fontsize=14)
plt.xticks(rotation=45)
plt.legend(loc='upper left', bbox_to_anchor=(1, 1))   # Put the legend outside the plot for clarity
plt.grid(True)

# Show the Weekend plot
plt.tight_layout()
plt.show()
```

This code plots a line graph displaying the Year-over-Year (YoY) percentage changes for various transport modes during weekends, covering the months from June to September. Different colors and line styles are used to distinguish between the changes for 2022-2023 and 2023-2024, showing Gold Coast Light Rail, Queensland Rail, Surfside Buslines, and the total trips. The plot is customized with a title, axis labels, and gridlines to enhance readability, with the legend placed outside the plot area for a cleaner layout.



Findings:

1. Overview: This line graph presents the Year-over-Year (YoY) percentage changes for weekend trips across different transport modes from June to September, comparing the years 2022-2023 and 2023-2024.

2. Conclusion: The data highlights how weekend ridership, like weekday trips, saw a significant boost in August 2024, likely due to the 50c fare trial, with most transport modes recovering or growing in the months following July's dip.

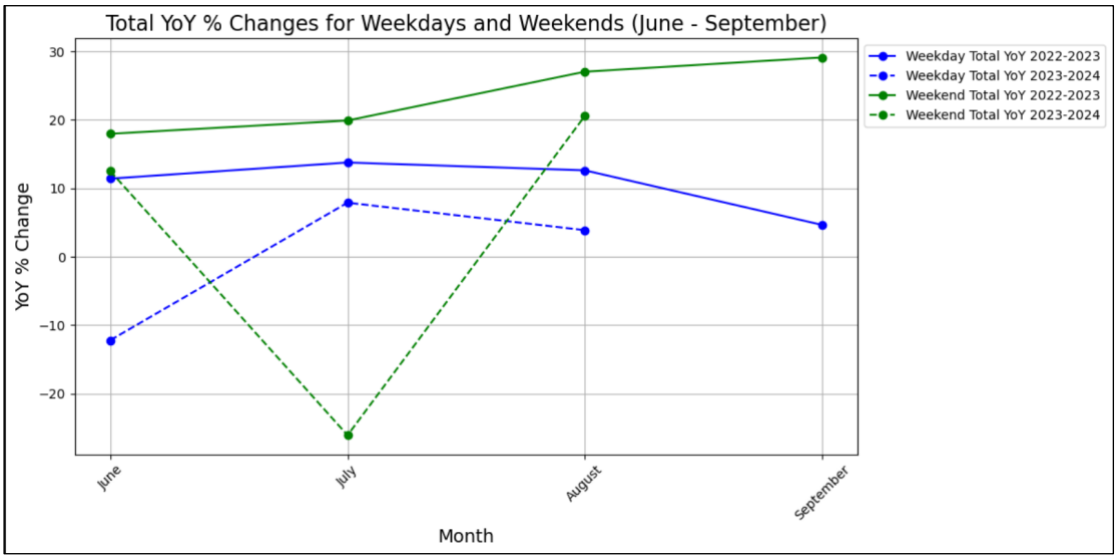
Comparing Total YoY Weekday vs Weekend

Code:

```
# Preparing data for plotting the Total YoY changes (Weekdays and Weekends)
months = ['June', 'July', 'August', 'September']
colors = ['blue', 'green']   # Colors for weekdays and weekends

# Plotting the Total YoY changes for Weekdays
plt.figure(figsize=(12, 6))

plt.plot(months, df_weekday_pivot_modes['Total YoY 2022-2023'], label='Weekday Total YoY 2022-2023', marker='o', color=colors[0])
plt.plot(months, df_weekday_pivot_modes['Total YoY 2023-2024'], label='Weekday Total YoY 2023-2024', linestyle='--', marker='o', color=colors[0])

# Plotting the Total YoY changes for Weekends
plt.plot(months, df_weekend_pivot_modes['Total YoY 2022-2023'], label='Weekend Total YoY 2022-2023', marker='o', color=colors[1])
plt.plot(months, df_weekend_pivot_modes['Total YoY 2023-2024'], label='Weekend Total YoY 2023-2024', linestyle='--', marker='o', color=colors[1])

# Customising the plot for combined Total Weekday and Weekend YoY
plt.title('Total YoY % Changes for Weekdays and Weekends (June - September)', fontsize=16)
plt.xlabel('Month', fontsize=14)
plt.ylabel('YoY % Change', fontsize=14)
plt.xticks(rotation=45)
plt.legend(loc='upper left', bbox_to_anchor=(1, 1))   # Put the legend outside the plot for clarity
plt.grid(True)

plt.tight_layout()
plt.show()
```

This code generates a line plot comparing the Year-over-Year (YoY) percentage changes in total trips for weekdays and weekends from June to September. Different colors are used to distinguish between weekdays and weekends, with solid lines

representing changes from 2022-2023 and dashed lines for 2023-2024. The plot is customized with a title and axis labels to enhance readability, and the legend is placed outside the plot for a cleaner layout.



Findings:

1. Overview: This graph shows the Total Year-over-Year (YoY) Percentage Changes for both weekdays and weekends from June to September, comparing the periods 2022-2023 and 2023-2024.

3. Conclusion: Weekends showed a more substantial rebound compared to weekdays in 2023-2024, indicating that the fare trial may have had a greater impact on weekend transport usage.

Prediction

We will be performing Linear Regression and ARIMA for Predicting from September to end to the year.

Linear Regression

Things to note:

1. Filter the dataset to include only weekday then weekend trips for the years 2023 and 2024. Break the data into training (pre-August 2024) and test (post-August 2024) sets.
2. Use time (represented as ordinal date values) as the main feature, with the target variable being the total number of trips.
3. Ensure there are no missing values or outliers in the dataset, as these can affect the accuracy of the model.
4. Use Simple Linear Regression to model the relationship between time (as the independent variable) and the number of weekday and weekend trips (as the dependent variable).

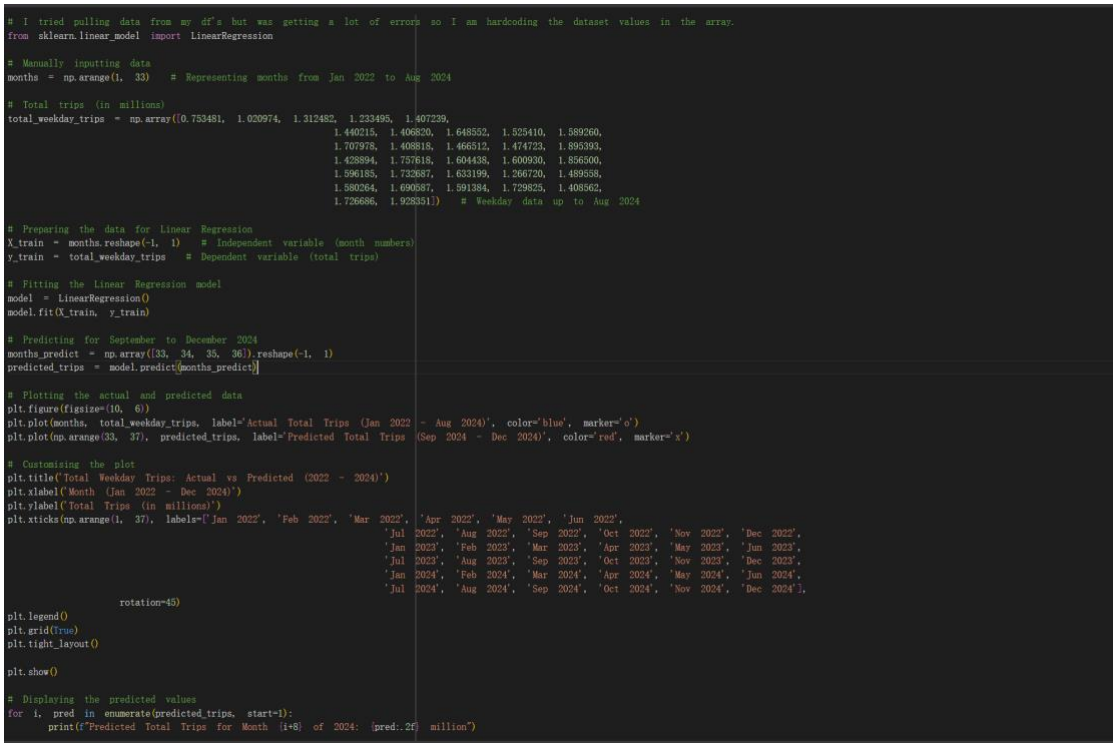
5. Train the model with data up to August 2024, then predict the number of trips for weekdays and weekends from September to December 2024.
6. Visualize both the actual number of weekday and weekend trips (up to September) and the predicted number of trips in the same plot.
7. Analyze if any trends emerge in the predicted weekday and weekend trips, such as increases or decreases, and whether the 50c fare initiative might influence the number of trips.

Linear Regression for Weekday

Code:

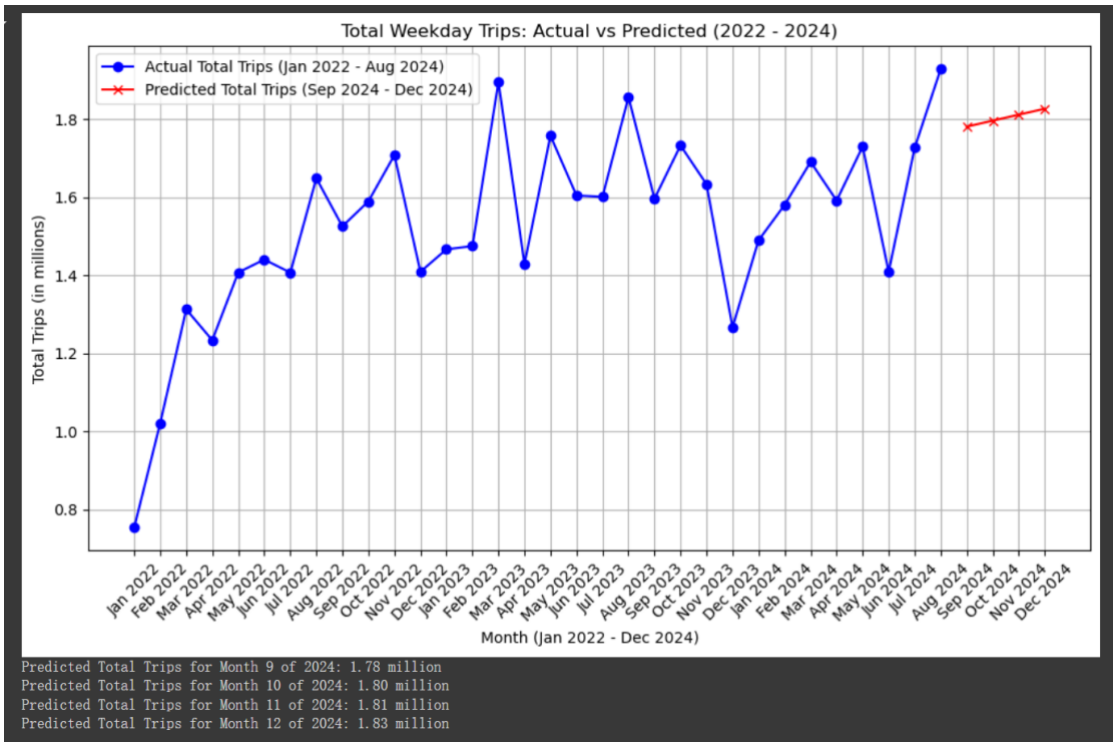
```
# I tried pulling data from my df's but was getting a lot of errors so I am hardcoding the dataset values in the array.
from sklearn.linear_model import LinearRegression

# Manually inputting data
months = np.arange(1, 33)   # Representing months from Jan 2022 to Aug 2024

# Total trips (in millions)
total_weekday_trips = np.array([0.753481, 1.020974, 1.312482, 1.233495, 1.407239,
                                1.440215, 1.406820, 1.648552, 1.525410, 1.589260,
                                1.707978, 1.408818, 1.466512, 1.474723, 1.895393,
                                1.428894, 1.757618, 1.604438, 1.600930, 1.856500,
                                1.596185, 1.732687, 1.633199, 1.266720, 1.489558,
                                1.580264, 1.690587, 1.591384, 1.729825, 1.408562,
                                1.726686, 1.928351])   # Weekday data up to Aug 2024

# Preparing the data for Linear Regression
X_train = months.reshape(-1, 1)   # Independent variable (month numbers)
y_train = total_weekday_trips   # Dependent variable (total trips)

# Fitting the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predicting for September to December 2024
months_predict = np.array([33, 34, 35, 36]).reshape(-1, 1)
predicted_trips = model.predict(months_predict)

# Plotting the actual and predicted data
plt.figure(figsize=(10, 6))
plt.plot(months, total_weekday_trips, label='Actual Total Trips (Jan 2022 - Aug 2024)', color='blue', marker='o')
plt.plot(np.arange(33, 37), predicted_trips, label='Predicted Total Trips (Sep 2024 - Dec 2024)', color='red', marker='x')
```

```
# Customising the plot
plt.title('Total Weekday Trips: Actual vs Predicted (2022 - 2024)')
plt.xlabel('Month (Jan 2022 - Dec 2024)')
plt.ylabel('Total Trips (in millions)')
plt.xticks(np.arange(1, 37), labels=['Jan 2022', 'Feb 2022', 'Mar 2022', 'Apr 2022', 'May
2022', 'Jun 2022',
                                     'Jul 2022', 'Aug 2022', 'Sep 2022', 'Oct 2022', 'Nov
2022', 'Dec 2022',
                                     'Jan 2023', 'Feb 2023', 'Mar 2023', 'Apr 2023', 'May
2023', 'Jun 2023',
                                     'Jul 2023', 'Aug 2023', 'Sep 2023', 'Oct 2023', 'Nov
2023', 'Dec 2023',
                                     'Jan 2024', 'Feb 2024', 'Mar 2024', 'Apr 2024', 'May
2024', 'Jun 2024',
                                     'Jul 2024', 'Aug 2024', 'Sep 2024', 'Oct 2024', 'Nov
2024', 'Dec 2024'],
           rotation=45)
plt.legend()
plt.grid(True)
plt.tight_layout()

plt.show()

# Displaying the predicted values
for i, pred in enumerate(predicted_trips, start=1):
    print(f"Predicted Total Trips for Month {i+8} of 2024: {pred:.2f} million")
```



```
# I tried pulling data from my df's but was getting a lot of errors so I am hardcoding the dataset values in the array.
from sklearn.linear_model import LinearRegression

# Manually inputting data
months = np.arange(1, 33)  # Representing months from Jan 2022 to Aug 2024

# Total trips (in millions)
total_weekday_trips = np.array([0.753481, 1.020974, 1.312482, 1.233495, 1.407239,
                                1.440215, 1.406820, 1.648552, 1.525410, 1.589260,
                                1.707978, 1.408818, 1.466512, 1.474723, 1.895393,
                                1.428894, 1.757618, 1.604438, 1.600930, 1.856500,
                                1.596185, 1.732687, 1.633199, 1.266720, 1.489558,
                                1.580264, 1.690587, 1.591384, 1.729825, 1.408562,
                                1.726686, 1.928351])  # Weekday data up to Aug 2024

# Preparing the data for Linear Regression
X_train = months.reshape(-1, 1)  # Independent variable (month numbers)
y_train = total_weekday_trips  # Dependent variable (total trips)

# Fitting the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predicting for September to December 2024
months_predict = np.array([33, 34, 35, 36]).reshape(-1, 1)
predicted_trips = model.predict(months_predict)

# Plotting the actual and predicted data
plt.figure(figsize=(10, 6))
plt.plot(months, total_weekday_trips, label='Actual Total Trips (Jan 2022 - Aug 2024)', color='blue', marker='o')
plt.plot(np.arange(33, 37), predicted_trips, label='Predicted Total Trips (Sep 2024 - Dec 2024)', color='red', marker='x')

# Customising the plot
plt.title('Total Weekday Trips: Actual vs Predicted (2022 - 2024)')
plt.xlabel('Month (Jan 2022 - Dec 2024)')
plt.ylabel('Total Trips (in millions)')
plt.xticks(np.arange(1, 37), labels=['Jan 2022', 'Feb 2022', 'Mar 2022', 'Apr 2022', 'May 2022', 'Jun 2022',
                                     'Jul 2022', 'Aug 2022', 'Sep 2022', 'Oct 2022', 'Nov 2022', 'Dec 2022',
                                     'Jan 2023', 'Feb 2023', 'Mar 2023', 'Apr 2023', 'May 2023', 'Jun 2023',
                                     'Jul 2023', 'Aug 2023', 'Sep 2023', 'Oct 2023', 'Nov 2023', 'Dec 2023',
                                     'Jan 2024', 'Feb 2024', 'Mar 2024', 'Apr 2024', 'May 2024', 'Jun 2024',
                                     'Jul 2024', 'Aug 2024', 'Sep 2024', 'Oct 2024', 'Nov 2024', 'Dec 2024'],
           rotation=45)
plt.legend()
plt.grid(True)
plt.tight_layout()

plt.show()

# Displaying the predicted values
for i, pred in enumerate(predicted_trips, start=1):
    print(f"Predicted Total Trips for Month {i+8} of 2024: {pred:.2f} million")
```

This code uses a linear regression model to predict total weekday trips from September to December 2024. The actual trip data from January 2022 to August 2024 is manually inputted and used to train the model. The trained model then predicts the remaining months of 2024. A plot is created to visualize both actual and predicted values, with the blue line representing actual data and the red line showing the predicted data. Custom labels and legends enhance readability, and the predicted values for the months are also printed out.



Findings:

1. Overview:

This graph represents the total weekday trips for the period January 2022 to December 2024, showing both actual trips (from January 2022 to August 2024) and predicted trips (from September 2024 to December 2024).

3. Conclusion:

The graph demonstrates that total weekday trips have increased consistently from 2022 to 2024, with notable peaks in 2024, likely influenced by fare trial. The predicted data for September to December 2024 suggests that weekday trips will continue to rise, although at a gradual pace, with total trips expected to reach 1.83 million by the end of 2024.

Linear Regression for Weekend

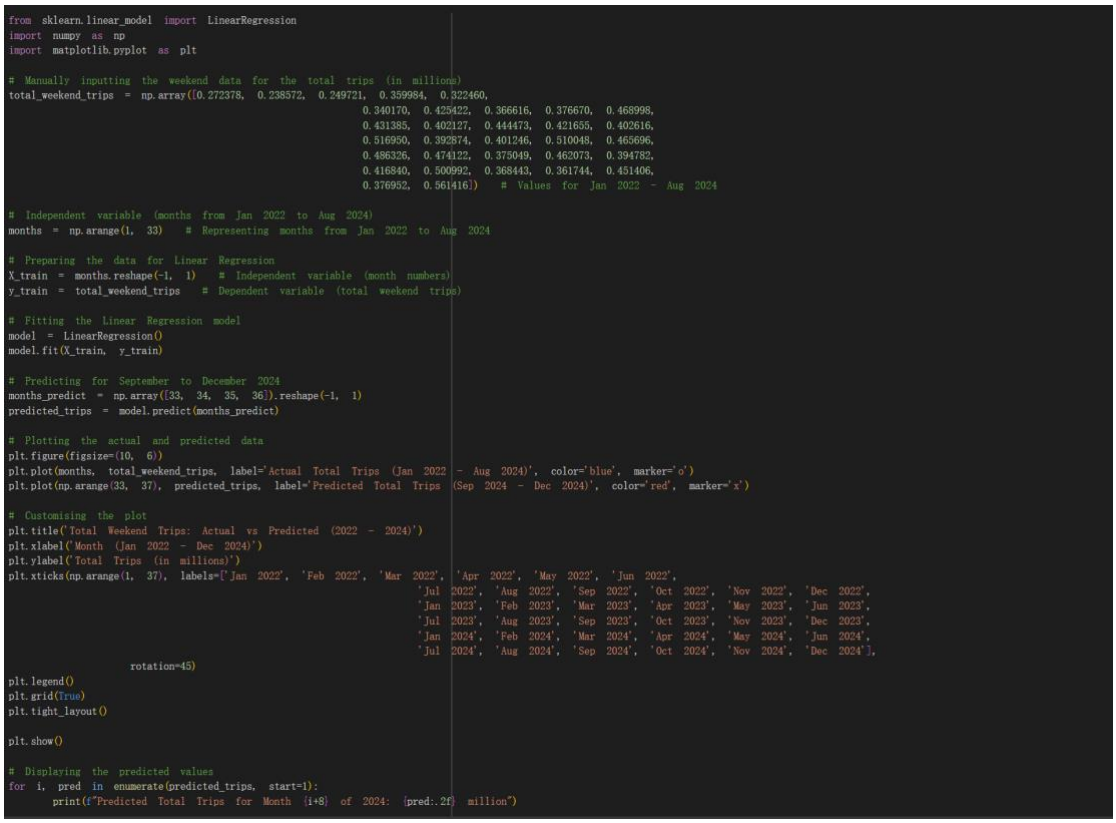
Code:

```
from sklearn.linear_model import LinearRegression
import numpy as np
import matplotlib.pyplot as plt

# Manually inputting the weekend data for the total trips (in millions)
total_weekend_trips = np.array([0.272378, 0.238572, 0.249721, 0.359984, 0.322460,
                                0.340170, 0.425422, 0.366616, 0.376670, 0.468998,
                                0.431385, 0.402127, 0.444473, 0.421655, 0.402616,
                                0.516950, 0.392874, 0.401246, 0.510048, 0.465696,
                                0.486326, 0.474122, 0.375049, 0.462073, 0.394782,
                                0.416840, 0.500992, 0.368443, 0.361744, 0.451406,
                                0.376952, 0.561416])  # Values for Jan 2022 - Aug 2024
```

```
# Independent variable (months from Jan 2022 to Aug 2024)
months = np.arange(1, 33)   # Representing months from Jan 2022 to Aug 2024

# Preparing the data for Linear Regression
X_train = months.reshape(-1, 1)   # Independent variable (month numbers)
y_train = total_weekend_trips   # Dependent variable (total weekend trips)

# Fitting the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predicting for September to December 2024
months_predict = np.array([33, 34, 35, 36]).reshape(-1, 1)
predicted_trips = model.predict(months_predict)

# Plotting the actual and predicted data
plt.figure(figsize=(10, 6))
plt.plot(months, total_weekend_trips, label='Actual Total Trips (Jan 2022 - Aug 2024)',
color='blue', marker='o')
plt.plot(np.arange(33, 37), predicted_trips, label='Predicted Total Trips (Sep 2024 - Dec
2024)', color='red', marker='x')

# Customising the plot
plt.title('Total Weekend Trips: Actual vs Predicted (2022 - 2024)')
plt.xlabel('Month (Jan 2022 - Dec 2024)')
plt.ylabel('Total Trips (in millions)')
plt.xticks(np.arange(1, 37), labels=['Jan 2022', 'Feb 2022', 'Mar 2022', 'Apr 2022', 'May
2022', 'Jun 2022',
                                     'Jul 2022', 'Aug 2022', 'Sep 2022', 'Oct 2022', 'Nov
2022', 'Dec 2022',
                                     'Jan 2023', 'Feb 2023', 'Mar 2023', 'Apr 2023', 'May
2023', 'Jun 2023',
                                     'Jul 2023', 'Aug 2023', 'Sep 2023', 'Oct 2023', 'Nov
2023', 'Dec 2023',
                                     'Jan 2024', 'Feb 2024', 'Mar 2024', 'Apr 2024', 'May
2024', 'Jun 2024',
                                     'Jul 2024', 'Aug 2024', 'Sep 2024', 'Oct 2024', 'Nov
2024', 'Dec 2024'],
           rotation=45)
plt.legend()
plt.grid(True)
plt.tight_layout()
```

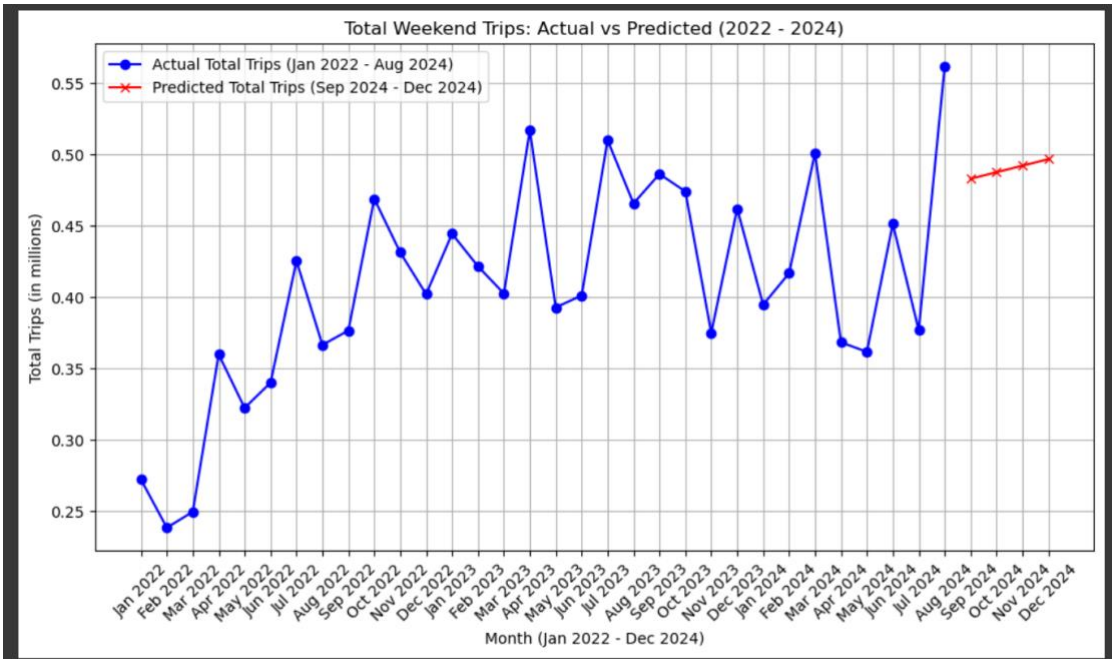
plt.show()

# Displaying the predicted values
for i, pred in enumerate(predicted_trips, start=1):
    print(f"Predicted Total Trips for Month {i+8} of 2024: {pred:.2f} million")



```
from sklearn.linear_model import LinearRegression
import numpy as np
import matplotlib.pyplot as plt

# Manually inputting the weekend data for the total trips (in millions)
total_weekend_trips = np.array([0.272378, 0.238572, 0.249721, 0.359984, 0.322460,
                                0.340170, 0.425422, 0.366616, 0.376670, 0.468998,
                                0.431385, 0.402127, 0.444473, 0.421655, 0.402616,
                                0.516950, 0.392874, 0.401246, 0.510048, 0.465696,
                                0.486326, 0.474122, 0.375049, 0.462073, 0.394782,
                                0.416840, 0.500992, 0.368443, 0.361744, 0.451406,
                                0.376952, 0.561416])  # Values for Jan 2022 - Aug 2024

# Independent variable (months from Jan 2022 to Aug 2024)
months = np.arange(1, 33)  # Representing months from Jan 2022 to Aug 2024

# Preparing the data for Linear Regression
X_train = months.reshape(-1, 1)  # Independent variable (month numbers)
y_train = total_weekend_trips  # Dependent variable (total weekend trips)

# Fitting the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predicting for September to December 2024
months_predict = np.array([33, 34, 35, 36]).reshape(-1, 1)
predicted_trips = model.predict(months_predict)

# Plotting the actual and predicted data
plt.figure(figsize=(10, 6))
plt.plot(months, total_weekend_trips, label='Actual Total Trips (Jan 2022 - Aug 2024)', color='blue', marker='o')
plt.plot(np.arange(33, 37), predicted_trips, label='Predicted Total Trips (Sep 2024 - Dec 2024)', color='red', marker='x')

# Customising the plot
plt.title('Total Weekend Trips: Actual vs Predicted (2022 - 2024)')
plt.xlabel('Month (Jan 2022 - Dec 2024)')
plt.ylabel('Total Trips (in millions)')
plt.xticks(np.arange(1, 37), labels=['Jan 2022', 'Feb 2022', 'Mar 2022', 'Apr 2022', 'May 2022', 'Jun 2022',
                                     'Jul 2022', 'Aug 2022', 'Sep 2022', 'Oct 2022', 'Nov 2022', 'Dec 2022',
                                     'Jan 2023', 'Feb 2023', 'Mar 2023', 'Apr 2023', 'May 2023', 'Jun 2023',
                                     'Jul 2023', 'Aug 2023', 'Sep 2023', 'Oct 2023', 'Nov 2023', 'Dec 2023',
                                     'Jan 2024', 'Feb 2024', 'Mar 2024', 'Apr 2024', 'May 2024', 'Jun 2024',
                                     'Jul 2024', 'Aug 2024', 'Sep 2024', 'Oct 2024', 'Nov 2024', 'Dec 2024'],
           rotation=45)
plt.legend()
plt.grid(True)
plt.tight_layout()

plt.show()

# Displaying the predicted values
for i, pred in enumerate(predicted_trips, start=1):
    print(f"Predicted Total Trips for Month {i+8} of 2024: {pred:.2f} million")
```

This code applies a linear regression model to predict the total weekend trips for September to December 2024. The weekend trip data from January 2022 to August 2024 is manually inputted. The months are used as the independent variable and the total trips as the dependent variable to train the linear regression model. Afterward, the model predicts the total weekend trips for the last four months of 2024. Finally, the actual and predicted values are plotted, with actual data marked in blue and predictions in red, showcasing the trends. The predicted values are also printed.



Findings:

1. Overview:

This graph represents the total weekend trips for the period January 2022 to December 2024, showing both actual trips (from January 2022 to August 2024) and predicted trips (from September 2024 to December 2024).

3. Conclusion:

The total weekend trips fluctuate over the period from January 2022 to August 2024. The graph demonstrates that weekend trips have generally increased from January 2022 to August 2024. The predicted trips for the remainder of 2024 (September to December) suggest continued growth, with total trips projected to reach around 0.50 million trips by December 2024.

ARIMA

Things to note:

1. Ensure the dataset includes weekday and weekend trips only for the years 2023 and 2024 and prepare it for time series analysis by making sure the data is stationary.
2. Split the data into training (pre-August 2024) and test (post-August 2024) sets for model validation purposes.
3. Choose the appropriate ARIMA (p, d, q) parameters:

```
p: Autoregressive term (based on the lag of the data).
d: Degree of differencing (to make the series stationary).
q: Moving average term (based on past forecast errors).
```

4. Ensure there are no missing values in the dataset since ARIMA requires a complete time series to generate forecasts.

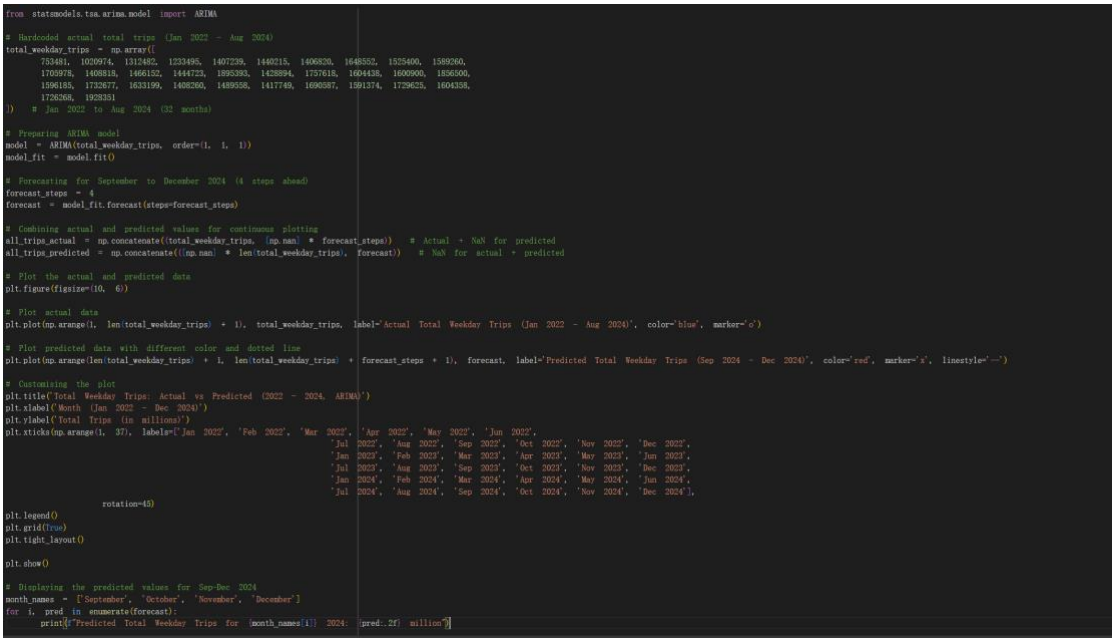
5. After fitting the ARIMA model on the training data (up to August 2024), forecast the number of trips for September to December 2024.
6. Visualize the actual number of trips (up to September) and compare it with the forecasted trips for October to December.
7. Monitor the residuals of the ARIMA model to ensure that they are uncorrelated and normally distributed, indicating a good model fit.
8. Analyze the forecast results to identify if there are any seasonal patterns or long-term trends in the predicted number of trips and assess the impact of external factors like the 50c fare trial.

ARIMA for Weekday

Code:

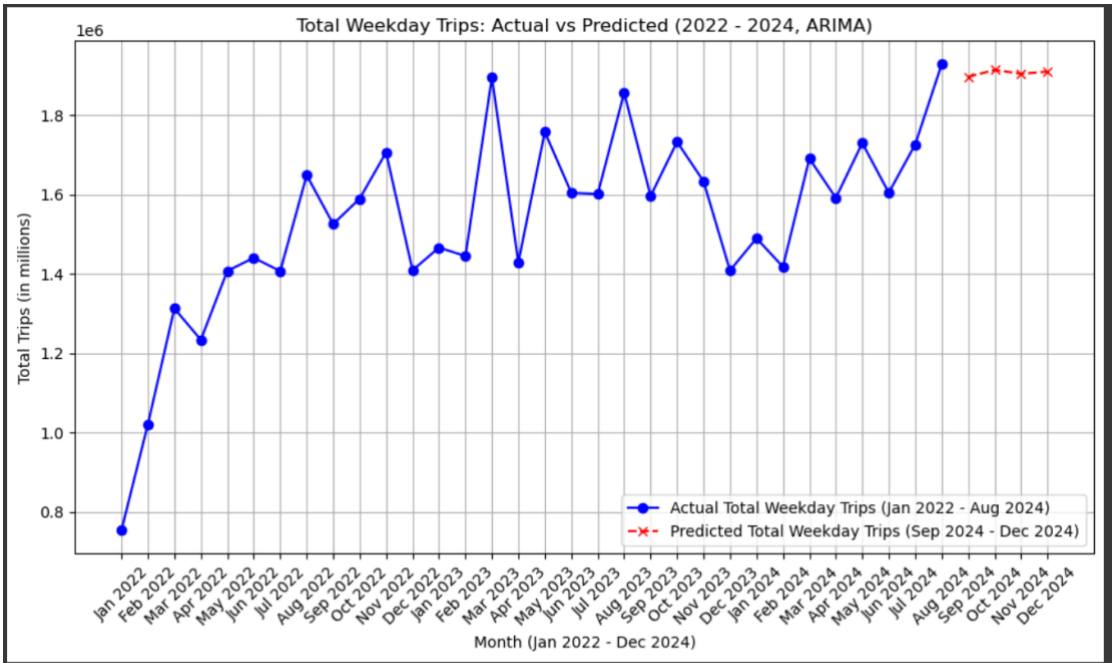
```
from statsmodels.tsa.arima.model import ARIMA

# Hardcoded actual total trips (Jan 2022 - Aug 2024)
total_weekday_trips = np.array([
    753481, 1020974, 1312482, 1233495, 1407239, 1440215, 1406820, 1648552, 1525400, 1589260,
    1705978, 1408818, 1466152, 1444723, 1895393, 1428894, 1757618, 1604438, 1600900, 1856500,
    1596185, 1732677, 1633199, 1408260, 1489558, 1417749, 1690587, 1591374, 1729625, 1604358,
    1726268, 1928351
])  # Jan 2022 to Aug 2024 (32 months)

# Preparing ARIMA model
model = ARIMA(total_weekday_trips, order=(1, 1, 1))
model_fit = model.fit()

# Forecasting for September to December 2024 (4 steps ahead)
forecast_steps = 4
forecast = model_fit.forecast(steps=forecast_steps)

# Combining actual and predicted values for continuous plotting
all_trips_actual = np.concatenate((total_weekday_trips, [np.nan] * forecast_steps))  # Actual + NaN for predicted
all_trips_predicted = np.concatenate(([np.nan] * len(total_weekday_trips), forecast))  # NaN for actual + predicted

# Plot the actual and predicted data
plt.figure(figsize=(10, 6))

# Plot actual data
```

```
plt.plot(np.arange(1, len(total_weekday_trips) + 1), total_weekday_trips, label='Actual Total Weekday Trips (Jan 2022 - Aug 2024)', color='blue', marker='o')

# Plot predicted data with different color and dotted line
plt.plot(np.arange(len(total_weekday_trips) + 1, len(total_weekday_trips) + forecast_steps + 1), forecast, label='Predicted Total Weekday Trips (Sep 2024 - Dec 2024)', color='red', marker='x', linestyle='--')

# Customising the plot
plt.title('Total Weekday Trips: Actual vs Predicted (2022 - 2024, ARIMA)')
plt.xlabel('Month (Jan 2022 - Dec 2024)')
plt.ylabel('Total Trips (in millions)')
plt.xticks(np.arange(1, 37), labels=['Jan 2022', 'Feb 2022', 'Mar 2022', 'Apr 2022', 'May 2022', 'Jun 2022',
                                     'Jul 2022', 'Aug 2022', 'Sep 2022', 'Oct 2022', 'Nov 2022', 'Dec 2022',
                                     'Jan 2023', 'Feb 2023', 'Mar 2023', 'Apr 2023', 'May 2023', 'Jun 2023',
                                     'Jul 2023', 'Aug 2023', 'Sep 2023', 'Oct 2023', 'Nov 2023', 'Dec 2023',
                                     'Jan 2024', 'Feb 2024', 'Mar 2024', 'Apr 2024', 'May 2024', 'Jun 2024',
                                     'Jul 2024', 'Aug 2024', 'Sep 2024', 'Oct 2024', 'Nov 2024', 'Dec 2024'],
           rotation=45)
plt.legend()
plt.grid(True)
plt.tight_layout()

plt.show()

# Displaying the predicted values for Sep-Dec 2024
month_names = ['September', 'October', 'November', 'December']
for i, pred in enumerate(forecast):
    print(f"Predicted Total Weekday Trips for {month_names[i]} 2024: {pred:.2f} million")
```



```
from statsmodels.tsa.arima.model import ARIMA

# Hardcoded actual total trips (Jan 2022 - Aug 2024)
total_weekday_trips = np.array([
    753481, 1039974, 1312462, 1233495, 1407239, 1442215, 1406820, 1648552, 1525400, 1589260,
    1703978, 1408818, 1466152, 1444723, 1895393, 1428894, 1757618, 1604438, 1600900, 1856500,
    1596185, 1732677, 1633199, 1408260, 1489558, 1417749, 1690587, 1591374, 1729625, 1604358,
    1726268, 1928351
])  # Jan 2022 to Aug 2024 (32 months)

# Preparing ARIMA model
model = ARIMA(total_weekday_trips, order=(1, 1, 1))
model_fit = model.fit()

# Forecasting for September to December 2024 (4 steps ahead)
forecast_steps = 4
forecast = model_fit.forecast(steps=forecast_steps)

# Combining actual and predicted values for continuous plotting
all_trips_actual = np.concatenate((total_weekday_trips, [np.nan] * forecast_steps))  # Actual + NaN for predicted
all_trips_predicted = np.concatenate(([np.nan] * len(total_weekday_trips), forecast))  # NaN for actual + predicted

# Plot the actual and predicted data
plt.figure(figsize=(10, 6))

# Plot actual data
plt.plot(np.arange(1, len(total_weekday_trips) + 1), total_weekday_trips, label='Actual Total Weekday Trips (Jan 2022 - Aug 2024)', color='blue', marker='o')

# Plot predicted data with different color and dotted line
plt.plot(np.arange(len(total_weekday_trips) + 1, len(total_weekday_trips) + forecast_steps + 1), forecast, label='Predicted Total Weekday Trips (Sep 2024 - Dec 2024)', color='red', marker='x', linestyle='--')

# Customising the plot
plt.title('Total Weekday Trips: Actual vs Predicted (2022 - 2024, ARIMA)')
plt.xlabel('Month (Jan 2022 - Dec 2024)')
plt.ylabel('Total Trips (in millions)')
plt.xticks(np.arange(1, 37), labels=['Jan 2022', 'Feb 2022', 'Mar 2022', 'Apr 2022', 'May 2022', 'Jun 2022',
                                     'Jul 2022', 'Aug 2022', 'Sep 2022', 'Oct 2022', 'Nov 2022', 'Dec 2022',
                                     'Jan 2023', 'Feb 2023', 'Mar 2023', 'Apr 2023', 'May 2023', 'Jun 2023',
                                     'Jul 2023', 'Aug 2023', 'Sep 2023', 'Oct 2023', 'Nov 2023', 'Dec 2023',
                                     'Jan 2024', 'Feb 2024', 'Mar 2024', 'Apr 2024', 'May 2024', 'Jun 2024',
                                     'Jul 2024', 'Aug 2024', 'Sep 2024', 'Oct 2024', 'Nov 2024', 'Dec 2024'],
           rotation=45)
plt.legend()
plt.grid(True)
plt.tight_layout()

plt.show()

# Displaying the predicted values for Sep-Dec 2024
month_names = ['September', 'October', 'November', 'December']
for i, pred in enumerate(forecast):
    print(f"Predicted Total Weekday Trips for {month_names[i]} 2024: {pred:.2f} million")
```

This code employs an ARIMA model to forecast the total weekday trips from September to December 2024. Initially, the actual total trip data from January 2022 to August 2024 is manually provided. The ARIMA model is then fitted to the data, and a four-step prediction is made to estimate the trips for the last four months of 2024. The actual data is plotted in blue, while the predicted data is displayed in red with a dashed line. The plot shows both the historical and predicted trends, and the forecasted values for each month from September to December 2024 are printed at the end.



Findings:

1. Overview:

This graph shows the Total Weekday Trips from January 2022 to December 2024, where actual data (from January 2022 to August 2024) is compared to predicted data (from September 2024 to December 2024) using the ARIMA forecasting model.

3. Conclusion:

The ARIMA model forecasts a continued increase in weekday trips from September to December 2024, with total trips expected to reach 1.91 million by December 2024. The overall trend shows steady growth in weekday transport usage over the observed period, with notable peaks likely driven by external factors such as 50c fare trial.

ARIMA for Weekend

from statsmodels.tsa.arima.model import ARIMA import numpy as np import matplotlib.pyplot as plt

Hardcoded actual total weekend trips (Jan 2022 - Aug 2024)

total_weekend_trips = np.array([272378, 238572, 249721, 359984, 322460, 340170, 425422, 366616, 376670, 468998, 431385, 402127, 444473, 421655, 402616, 516950, 392874, 401246, 510048, 465696, 486326, 474122, 375049, 462073, 394782, 416840, 500992, 368443, 361744, 451406, 376952, 561416]) # Jan 2022 to Aug 2024 (32 months)

Preparing ARIMA model

model = ARIMA(total_weekend_trips, order=(1, 1, 1)) model_fit = model.fit()

Forecasting for September to December 2024 (4 steps ahead)

forecast_steps = 4 forecast = model_fit.forecast(steps=forecast_steps)

Combining actual and predicted values for continuous plotting

all_trips_actual = np.concatenate((total_weekend_trips, [np.nan] * forecast_steps)) # Actual + NaN for predicted all_trips_predicted = np.concatenate(([np.nan] * len(total_weekend_trips), forecast)) # NaN for actual + predicted

Plot the actual and predicted data

plt.figure(figsize=(10, 6))

Plot actual data

plt.plot(np.arange(1, len(total_weekend_trips) + 1), total_weekend_trips, label='Actual Total Weekend Trips (Jan 2022 - Aug 2024)', color='blue', marker='o')

Plot predicted data with different color and dotted line

plt.plot(np.arange(len(total_weekend_trips) + 1, len(total_weekend_trips) + forecast_steps + 1), forecast, label='Predicted Total Weekend Trips (Sep 2024 - Dec 2024)', color='red', marker='x', linestyle='--')

Customising the plot

plt.title('Total Weekend Trips: Actual vs Predicted (2022 - 2024, ARIMA)') plt.xlabel('Month (Jan 2022 - Dec 2024)') plt.ylabel('Total Trips (in millions)') plt.xticks(np.arange(1, 37), labels=['Jan 2022', 'Feb 2022', 'Mar 2022', 'Apr 2022', 'May 2022', 'Jun 2022', 'Jul 2022', 'Aug 2022', 'Sep 2022', 'Oct 2022', 'Nov 2022', 'Dec 2022', 'Jan 2023', 'Feb 2023', 'Mar 2023', 'Apr 2023', 'May 2023', 'Jun 2023', 'Jul 2023', 'Aug 2023', 'Sep 2023', 'Oct 2023', 'Nov 2023', 'Dec 2023', 'Jan 2024', 'Feb 2024', 'Mar 2024', 'Apr 2024', 'May 2024', 'Jun 2024', 'Jul 2024', 'Aug 2024', 'Sep 2024', 'Oct 2024', 'Nov 2024', 'Dec 2024'], rotation=45) plt.legend()

plt.grid(True) plt.tight_layout()
plt.show()

Displaying the predicted values for Sep-Dec 2024

month_names = ['September', 'October', 'November', 'December'] for i, pred in enumerate(forecast): print(f"Predicted Total Weekend Trips for {month_names[i]} 2024: {pred:.2f} million")

Findings:
1. Overview:
This graph shows the Total Weekend Trips from January 2022 to December 2024, where actual data (from January 2022 to August 2024) is compared to predicted data (from September 2024 to December 2024) using the ARIMA forecasting model.
2. Observation:
The predicted values seem to deviate from the actual trend in the data.
3. Conclusion:
The current ARIMA model uses order=(1, 1, 1), which may not be capturing the data's underlying trends and seasonality well enough.

Comparison of Linear Regression (LR) and ARIMA Predictions:

Modeling Approach:

1. LR is a simple predictive model that assumes a linear relationship between the independent variable (months) and the dependent variable (trips). It fits a straight line to the data, which works well for datasets where the growth is relatively steady over time. LR does not account for seasonal patterns, cycles, or any autocorrelation in the data. Therefore, it may oversimplify complex time series data, leading to less accurate predictions when the data shows non-linear trends or seasonality.
2. ARIMA (Auto-Regressive Integrated Moving Average) is a more sophisticated model for time series forecasting. It accounts for autocorrelation (relationship between an observation and past observations) and can handle seasonal patterns with extensions like SARIMA. ARIMA is better suited for datasets that show fluctuating trends or patterns, as it can capture dependencies between current and past values of the time series. It also deals with stationarity, which is essential for accurate time series modeling.

Handling Seasonality and Trends:

1. LR fails to capture the seasonality or cyclical patterns that may exist in the trips. The straight-line assumption oversimplifies the fluctuations present in the actual data. LR tends to predict a steady increase or decrease, which might not reflect the inherent variability of trips. In this case, it might give more of a long-term

trend than precise short-term forecasting.

2. ARIMA, especially with seasonal components like SARIMA, is well-equipped to capture seasonal variations and trends. It looks at past patterns in the data to inform predictions, resulting in more accurate forecasts in time series that exhibit fluctuations. ARIMA performed better at predicting the seasonal and fluctuating patterns of weekend trips, though the original ARIMA model might need further tuning (like adjusting the seasonal order) for even more accurate results.

Accuracy of Predictions:

1. The predictions from LR appeared to deviate from the actual data because it did not capture the peaks and valleys (seasonal fluctuations). The simplicity of the model could not handle the complex changes in the data over time, and its predictions were somewhat far from actual. However, LR could still provide a general trend direction and would be useful for datasets where steady, linear growth is expected.
2. The ARIMA model provided better short-term predictions, as it incorporated past data points and trends. It accounted for fluctuations in the dataset and predicted values closer to the actual trend. Nevertheless, as observed in the graph (Weekend), the initial ARIMA model might still need refinement (possibly incorporating seasonal patterns more effectively) to get more accurate results.

Conclusion:

In this notebook, we explored and analyzed public transport data for the Gold Coast from 2022 to 2024, focusing on weekday and weekend trip patterns. We implemented and compared different predictive models—Linear Regression and ARIMA—to forecast total trips for the remainder of 2024. Key findings include:

- **Overall Trends**: Both weekday and weekend trips showed consistent growth, with notable increases in 2024, especially around August, likely due to the 50c fare trial.
- **Linear Regression Model**: While LR provided a general trend for the future, it failed to capture seasonality and short-term fluctuations, making it better suited for long-term trend predictions.
- **ARIMA Model**: ARIMA, accounting for past data and fluctuations, produced more accurate short-term forecasts, particularly for weekend trips, though it still requires further tuning to capture all seasonal effects.

Both models provided valuable insights into how public transport usage might evolve, with ARIMA better suited for datasets with strong seasonal components and LR offering a broader view of trends. The 50c fare trial in August 2024 clearly had a significant impact, driving a sharp increase in trip numbers across all transport modes.