King Saud University

College of Business Administration

Management Information Systems Department

MIS 350- Decision Support Systems & Expert Systems

1st Semester 2024

Analyzing Traffic and Weather Patterns in Riyadh

Phase #3

Group #5

Section#	Name	ID
41459	Aljoharah Alsulami	443202090
	Roaa Almutairi	443200427
	Hussa Aldriweesh	443201577
	Reem Alotaibi	446200429

Problem:

The purpose of this project is to address the persistent issue of travel delays in major cities like Riyadh, which negatively impact productivity. By analyzing the root causes of these delays using historical traffic and weather pattern data, we aim to investigate factors such as rush hours and Riyadh Season traffic flow. This project will uncover insights and offer practical recommendations to reduce travel delays, supporting a more efficient decision-making process in urban traffic management.

Dataset Overview:

The database we have acquisitioned is a record of car traffic in Riyadh, Saudi Arabia, covering many factors, mainly in three categories: General Information, Traffic Data, and Weather Data. The dataset contains a total of 8,316 rows and 19 columns, all of which will go through a cleaning process, and the total number of rows will probably decrease.

Category	Column	Data Type	Description
General Information	City	Character: string	The city where the observation took place, like Riyadh
	Datetime	Numerical: date & time	Include the day, month and the year. And the hour when the observation took place
	Day	Character: string	The name of the day of the week
	Day Type	Character: string	Either weekday or weekend day
Traffic Data	TrafficIndexLive	Numerical: integer	Reflects the Traffic Congestion
	JamsCount	Numerical: integer	Indicate how many traffic jams have been spotted
	JamsDelay	Numerical: float	Represent the delay caused by the Jams (min)
	LamsLength	Numerical: float	The total length of all Jams (km)
	TrafficIndexWeekAgo	Numerical: float	Reflects the Traffic Congestion a week ago
	TrafficTimeHistoric	Numerical: float	Reflects average travel time in normal conditions
	TravelTimeLive	Numerical: float	Represents the travel time for a full trip
Weather Data	Temp	Numerical: float	The weather temperature (Celsius)

	FeelsLike	Numerical: float	The weather feeling that comes from wind, humidity and other factors
	Humidity	Numerical: float	The amount of water vapor present in the air
	WindSpeed	Numerical: float	Wind speed (kph)
	WindDirction	Numerical: integer	The direction from which the wind is blowing
	CouldCover	Numerical: integer	The percentage of the sky covered by clouds.
	Visibility	Numerical: integer	The distance a person can clearly see through in the road
	Conditions	Character: string	description of the weather (clear, cloudy or rainy)

KPIs:

- Track the average jam delay and the average jam count per hour in Riyadh in 2023
  Goal Improve traffic flow by addressing peak congestion hours such as rush hours.

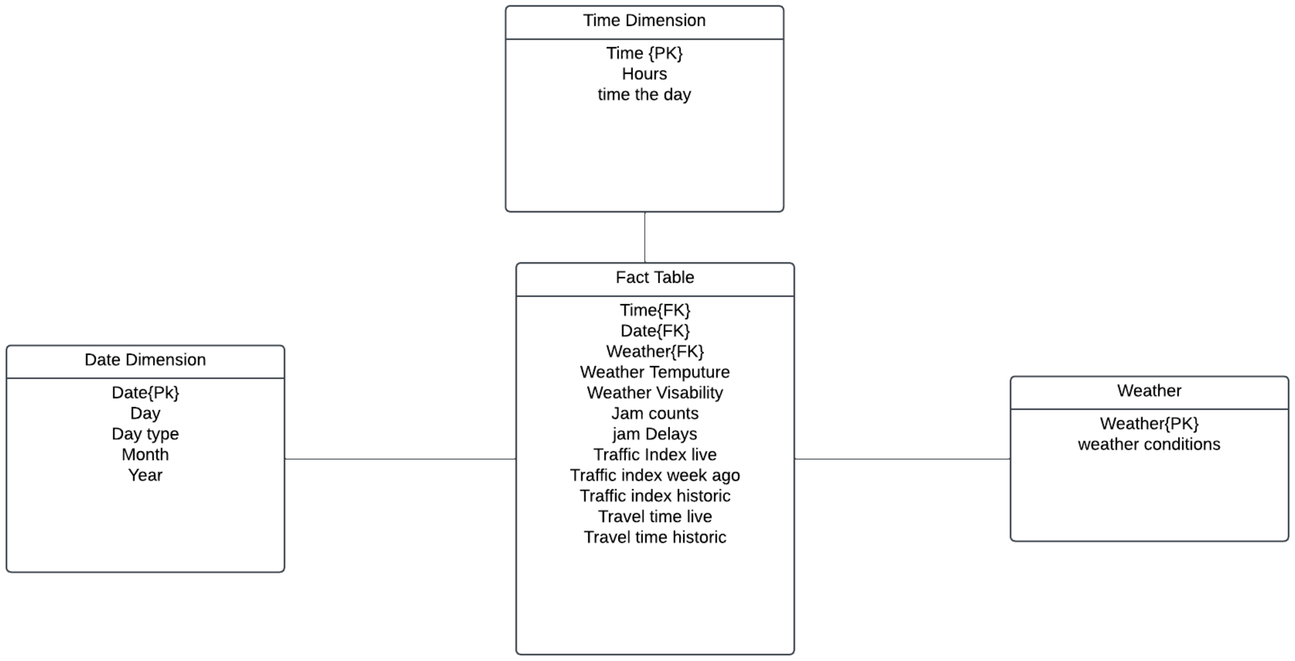
- Compare the Average Travel Time Live with the Average Travel Time Historic during the months of Riyadh Season
  Goal measures the overall change of travel time, whether it was increasing or decreasing.

- Average Traffic Jams Count per weekend day in Riyadh
  Goal: Analyze the effect of non-working days on traffic.

- Compare the average Traffic index live to the Traffic index a week ago for last week
  Goal Identify deviations in travel time trends to implement timely corrective measures.

- Track the average Traffic index live and the average jam delay per weather conditions in Riyadh
  Goal Close flood-affected roads for safety and to reduce traffic disruptions.

Star Schema:



Descriptions of the Fact and Dimension tables:

Fact Table:

- Traffic Index: traffic congestion indicator, which reflects the current level of traffic congestion. A higher number indicates more severe traffic congestion.
- Jam counts: Number of traffic jams reported at that specific date and time. This metric indicates how many separate traffic jam incidents were observed
- Jam Delays: Total delay caused by the traffic jams, typically measured in minutes. It represents the extra time drivers are stuck in traffic compared to usual.
- Travel time live: Current travel time at the time the data was recorded. It represents how long it currently takes to travel a certain distance
- Travel time historic: The average travel time under normal conditions, based on historical data. Use this as a baseline to see how current travel times differ.
- Time (FK): Foreign key linking to the Dimension Time table, indicating the specific time associated with the traffic data.
- Date (FK): Foreign key linking to the Dimension Date table, indicating the specific date of the traffic data.
- Weather (FK): Foreign key linking to the Dimension Weather table, indicating the weather conditions at the time of traffic data collection.
- Weather temperature: The temperature at the time of data recording measured in degrees Celsius.
- Weather visibility: The distance you can clearly see, measured in kilometers. Lower values indicate foggy or hazy conditions.

Dimension tables:

Date Dimension:

- Date (PK): Primary key, uniquely identifying each date entry.
- Day: The day of the week (e.g., Sunday, Monday), indicating when the data was recorded.
- Day type: Specifies whether it's a weekday or weekend.
- Month: Specifies the month of the measures.
- Year: Specifies the year of the measures.

Time Dimension:

- Time (PK): Primary key, uniquely identifying each time entry. This could represent time in intervals (e.g., hours, minutes).
- Time of day: Specifies the time of day such as early morning, morning, afternoon and evening.
- Hour: Specifies the hour of the measure.

Weather Dimension:

- Weather (PK): Primary key, uniquely identifying each weather condition entry.
- Weather conditions: A brief description of the weather (e.g., clear, cloudy, rainy), providing context for the traffic data.

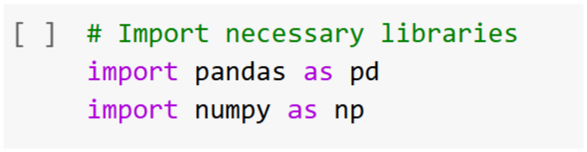
ETL process:

We performed an ETL (Extract, Transform, Load) process using a dedicated Python script to transform the dataset into a cleansed, analyzable format. The initial dataset contained 16,100 rows and 19 columns. As the first step, we removed all rows where the city was not Riyadh, narrowing the data to focus on relevant insights. We then addressed missing data by eliminating 42 rows with null values across 7 columns, leaving us with 16,058 rows. Duplicate rows were also identified and removed.

Additionally, we found 4 negative values, which were deemed invalid and dropped. Temperature records exceeding 60 degrees, which were unrealistic upon investigation, were also removed. Using the data.describe() method, we generated a statistical summary for validation. Data was further processed into datetime formats by year, month, and hour. Finally, all data was merged into fact tables. The resulting dataset now contains 8,311 rows and 24 columns, ready for analysis.

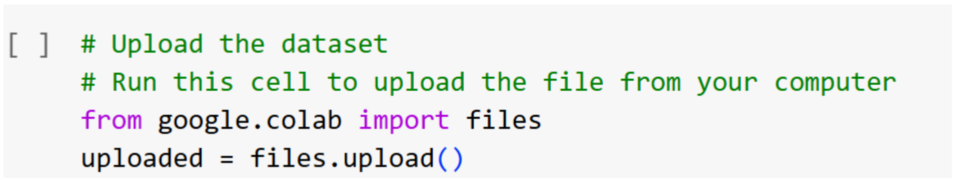
Loading the Data:

Code starts by importing the necessary libraries for data ETL, which are NumPy and Pandas.



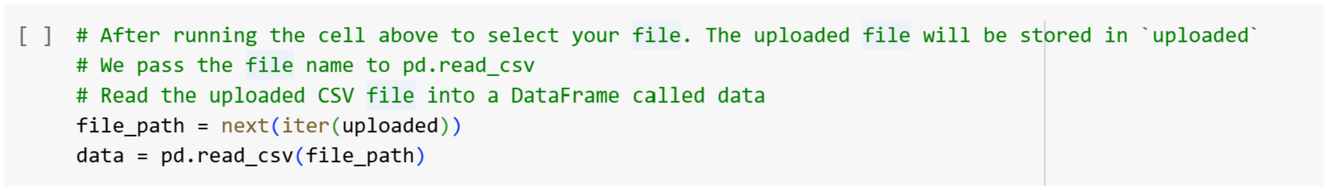
```
[ ]  # Import necessary libraries
     import pandas as pd
     import numpy as np
```

Uploading the CSV file that contains the data prior to cleaning.



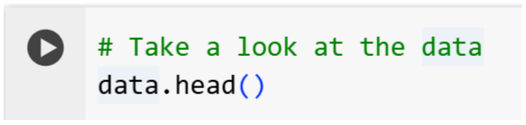
```
[ ]  # Upload the dataset
     # Run this cell to upload the file from your computer
     from google.colab import files
     uploaded = files.upload()
```

Next, it's necessary to read the data into a file to be accessed and save it to a variable, in our case it was 'data'



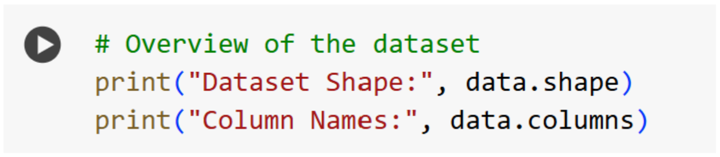
```
[ ]  # After running the cell above to select your file. The uploaded file will be stored in `uploaded`
     # We pass the file name to pd.read_csv
     # Read the uploaded CSV file into a DataFrame called data
     file_path = next(iter(uploaded))
     data = pd.read_csv(file_path)
```

The following code was used to see the overall structure of the dataset and a full display of all column names and look at the first five rows of the dataset.



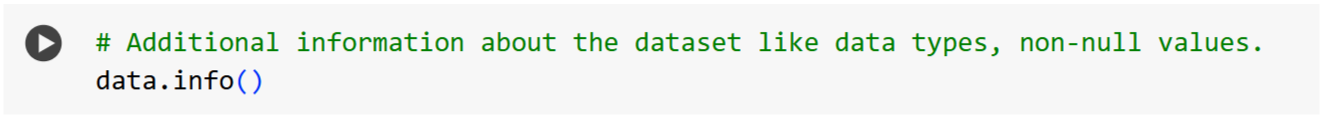
```
# Take a look at the data
data.head()
```

The data shape is columns x rows, which gives a general view of the data set volume. Data columns however print out the names of the attributes (column names). The dataset includes columns such as City, Datetime, Day, Day Type, TrafficIndexLive, JamsCount, JamsDelay, JamsLength, TrafficIndexWeekAgo, TravelTimeLive, TravelTimeHistoric, Temp, FeelsLike, Humidity, WindSpeed, WindDirection, CloudCover, Visibility, and Conditions. This initial exploration helps in understanding the dataset's structure, types of data, and potential insights for further analysis. And the Dataset Shape is: 16100 rows, and 19 columns.



```
# Overview of the dataset
print("Dataset Shape:", data.shape)
print("Column Names:", data.columns)
```

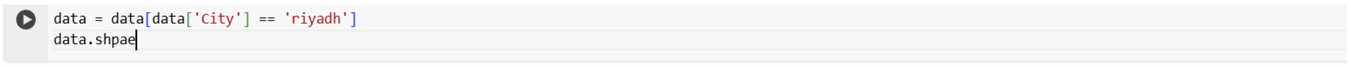
Provides an overview of the data types in the data set and the number of null cells present in the dataset. The dataset is a Pandas DataFrame consisting of 16,100 entries across 19 columns. Each column represents a specific attribute, with data types including object for categorical data and float64 for numerical data. The columns City, Datetime, Day, Day Type, Temp, FeelsLike, Humidity, WindSpeed, WindDirection, CloudCover, Visibility, and Conditions have complete data (16,100 non-null values). However, columns such as TrafficIndexLive, JamsCount, JamsDelay, JamsLength, TrafficIndexWeekAgo, TravelTimeLive, and TravelTimeHistoric have some missing values, with 16,058 non-null entries each. This summary indicates that while most columns are well-populated, further handling of missing values will be necessary for accurate analysis of traffic and weather-related metrics.



```
# Additional information about the dataset like data types, non-null values.
data.info()
```

Cleaning the data:

We dropped all the columns where the city was not Riyadh, we were left with 8,369 rows, and 19 columns after looking at the shape of the dataset.



```
data = data[data['City'] == 'riyadh']
data.shpae
```

We have checked the missing values in our dataset by locating columns that contain empty entries. then it showed the total count of missing values for each column. The dataset contains missing values in the following columns: TrafficIndexLive, JamsCount, JamsDelay, JamsLength, TrafficIndexWeekAgo, TravelTimeLive, and TravelTimeHistoric, with each having 42 missing entries.

```
# Handle Missing Values
# Identify columns with missing values
data.isnull().sum()
```

All rows in the dataset that contain missing values have been removed in this step, and the dataset shape after dropping missing values: (8327, 19).Thereafter, it printed the new shape of the dataset to show how many rows and columns remain after cleaning.

```
# Drop rows with missing values
data.dropna(inplace=True)
print("Dataset Shape after dropping missing values:", data.shape)
```

Identified any rows that are identical to others. Then it showed the total count of duplicate rows. And we found that the dataset contained 12 null values.

```
# Check for duplicates to remove them if found
data.duplicated().sum()
```

Removed all redundant rows from the dataset, and the dataset shape after dropping duplicates: 8,315 rows and 19 columns. Thereafter, it printed the updated shape of the dataset.

```
# Drop duplicate rows
data.drop_duplicates(inplace=True)
print("Dataset Shape after dropping duplicates:", data.shape)
```

Identified rows where the TrafficIndexLive or Temperature columns had negative values. The dataset contains 4 rows with negative values in certain columns. A sample of these rows includes: A record from Riyadh on April 30, 2023, at 10:00 AM, where the Temp column has a value of -32, a record from Riyadh on October 10, 2023, at 10:00 AM, where the Temp column has a value of -34, a record from Jeddah on October 5, 2023, at 2:00 PM, where the TrafficIndexLive column has a value of -23 and a record from Dammam on August 27, 2023, at 2:00 AM, where the TrafficIndexLive column has a value of -3. These negative values are likely errors, however given that they are only four records, we have decided to drop them. Then we looked at the dataset shape, there are 8,311 rows, and 19 columns.

```
# Display rows with negative values in either of the two specified columns
negative_rows = data[(data['TrafficIndexLive'] < 0) | (data['Temp'] < 0)]

# Display the number of rows found and show a preview
print(f"Total Rows with Negative Values: {negative_rows.shape[0]}")
print("Sample of Rows with Negative Values:")
negative_rows.style.set_properties(**{'background-color': 'white', 'color': 'red'})
```

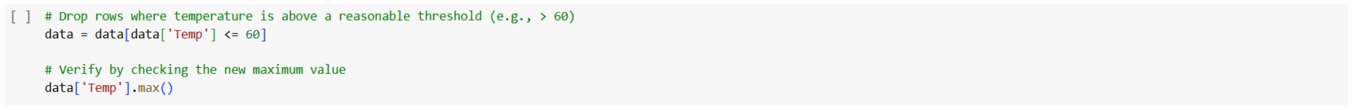
```
# Drop rows with negative values
data = data[(data['TrafficIndexLive'] >= 0) & (data['Temp'] >= 0)]
data.shape
```

Identify the rows with incorrect and unrealistic temperatures, the dataset includes records with unusual values in specific columns. For example: a record from Jeddah on October 9, 2023, at 5:00 AM shows Temp as 290.0, which is excessively high and likely incorrect. And a record from Dammam on October 14, 2023, at 10:00 AM has Temp listed as 136.0, another improbable value for temperature.

These anomalies suggest errors in data collection or input. Given that they are only two, we decided to drop them

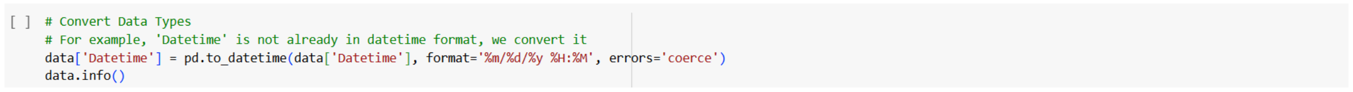
```
# Identify rows with unrealistic temperature values (e.g., > 60)
unrealistic_temps = data[data['Temp'] > 60]
print(unrealistic_temps)
```

By dropping rows where the temperature exceeds 60, the dataset is refined to exclude unrealistic temperature values, such as those exceeding typical ranges for most locations. This adjustment ensures that only plausible temperature data remains. After applying this filter, the maximum temperature value in the dataset will no longer exceed 60 and by verifying we discovered that the maximum temputure is 49.7 , effectively removing any outliers or data entry errors related to temperature.



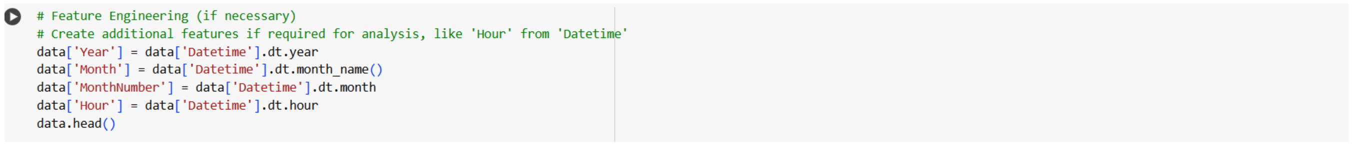
```
[ ] # Drop rows where temperature is above a reasonable threshold (e.g., > 60)
    data = data[data['Temp'] <= 60]

    # Verify by checking the new maximum value
    data['Temp'].max()
```

We have identified that the “Datetime” column is a string; when it should be a datetime type. We used the pandas function “to_datetime” to convert the type, and we specified the format. Then we printed a consice summary of the data.



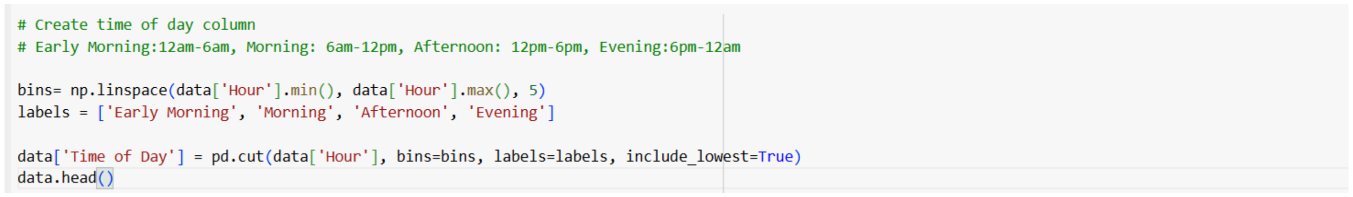
```
[ ] # Convert Data Types
    # For example, 'Datetime' is not already in datetime format, we convert it
    data['Datetime'] = pd.to_datetime(data['Datetime'], format='%m/%d/%y %H:%M', errors='coerce')
    data.info()
```

We created new columns, Year, Month, Month number and Hour. Pulling the data from the Datetime column, the dataset now includes additional columns such as Year, Month, MonthNumber, and Hour. then we looked at the first five rows of the data.



```
# Feature Engineering (if necessary)
# Create additional features if required for analysis, like 'Hour' from 'Datetime'
data['Year'] = data['Datetime'].dt.year
data['Month'] = data['Datetime'].dt.month_name()
data['MonthNumber'] = data['Datetime'].dt.month
data['Hour'] = data['Datetime'].dt.hour
data.head()
```

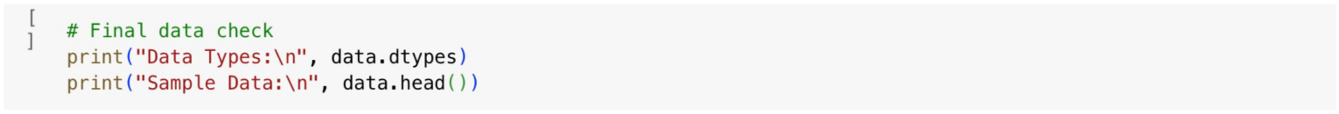
Created five bins that split the range of hours into sections: Early morning, Morning, Afternoon and Evening. We defined labels for each time period and applied them to the Time-of-Day column. The dataset now has a new column, Time of Day, which categorizes the time of each entry into specific parts of the day, such as "Early Morning", "Morning", and "Afternoon. Finaly, we previewed the first five rows of the data.



```
# Create time of day column
# Early Morning:12am-6am, Morning: 6am-12pm, Afternoon: 12pm-6pm, Evening:6pm-12am

bins= np.linspace(data['Hour'].min(), data['Hour'].max(), 5)
labels = ['Early Morning', 'Morning', 'Afternoon', 'Evening']

data['Time of Day'] = pd.cut(data['Hour'], bins=bins, labels=labels, include_lowest=True)
data.head()
```

Saving the data:

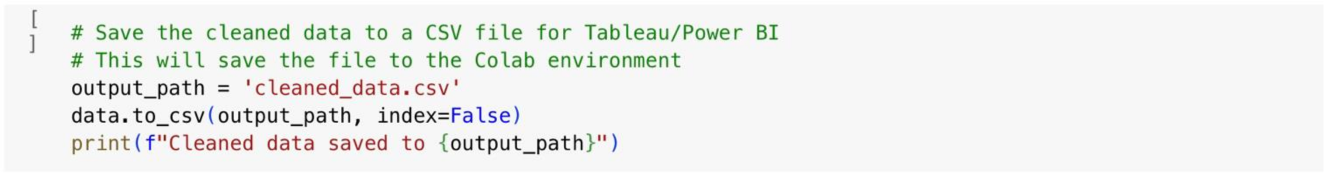
Check that data for the final time to understand the structure of data. And the dataset contains 8,311 rows and 24 columns, including traffic, weather, and time-related data. Key columns include TrafficIndexLive, JamsCount, Temp, Humidity, and WindSpeed, all of type float64, while Datetime is of type datetime64[ns]. Categorical data like City and Conditions are stored as objects, and Time of Day is categorized. It provides hourly traffic and weather details for "Riyadh", with additional temporal features like Year, MonthNumber, Hour, and Time of Day. This structure supports time-series analysis



```python
# Final data check
print("Data Types:\n", data.dtypes)
print("Sample Data:\n", data.head())
```

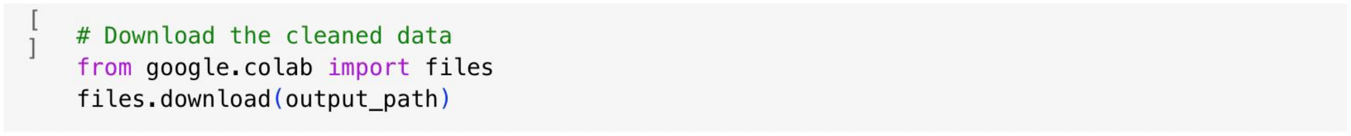
and forecasting.

Save the cleaned data to a in CSV format.



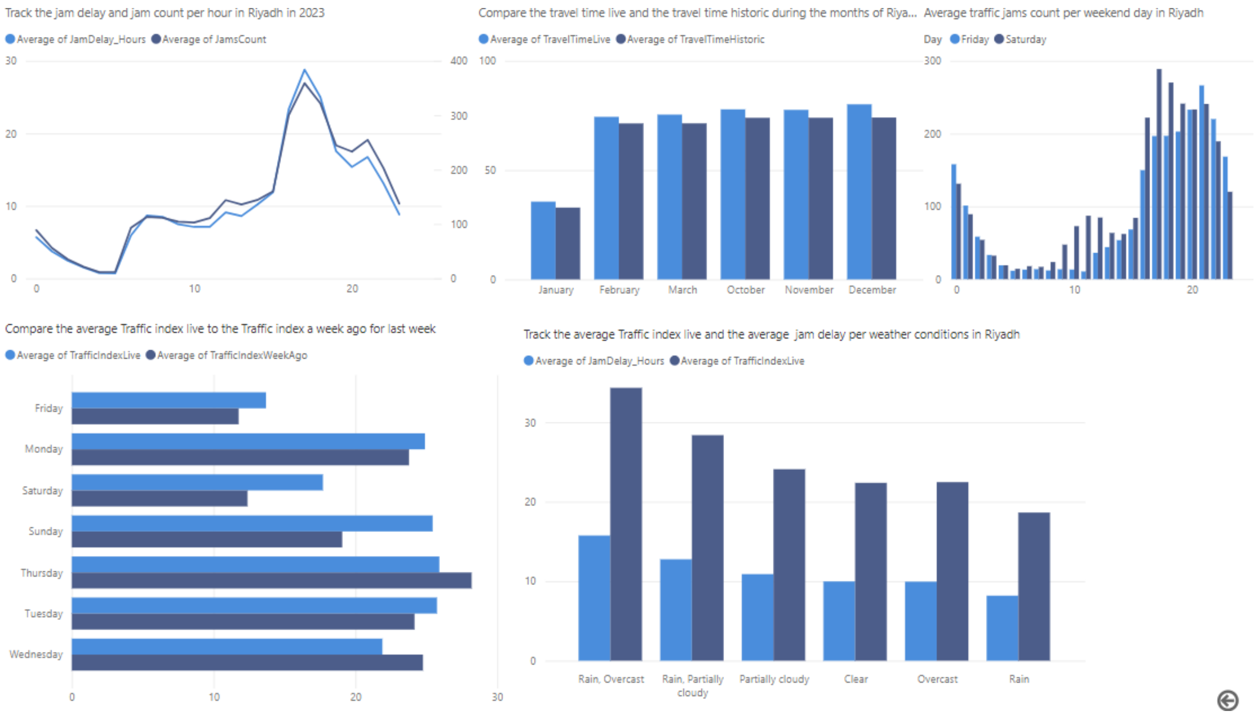
```python
# Save the cleaned data to a CSV file for Tableau/Power BI
# This will save the file to the Colab environment
output_path = 'cleaned_data.csv'
data.to_csv(output_path, index=False)
print(f"Cleaned data saved to {output_path}")
```

Download the cleaned data.



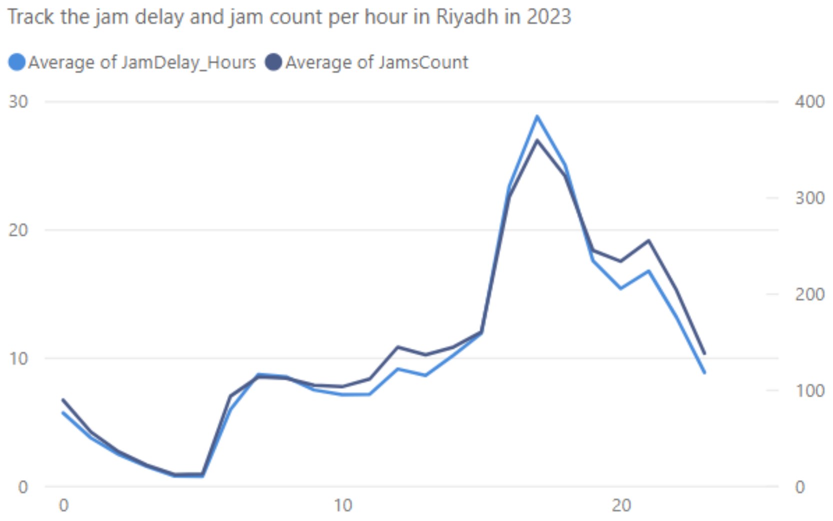
```python
# Download the cleaned data
from google.colab import files
files.download(output_path)
```

Dashboard:



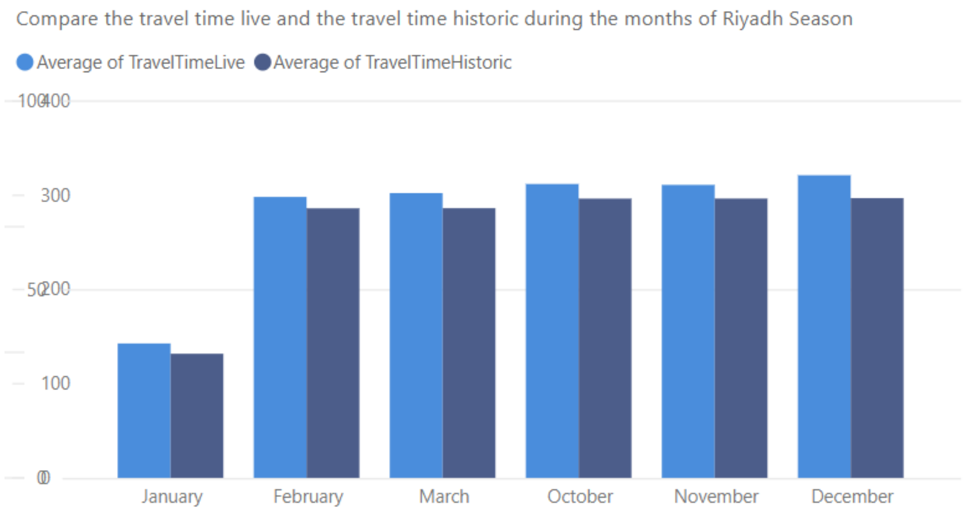
Link

Track the jam delay and jam count per hour in Riyadh in 2023:



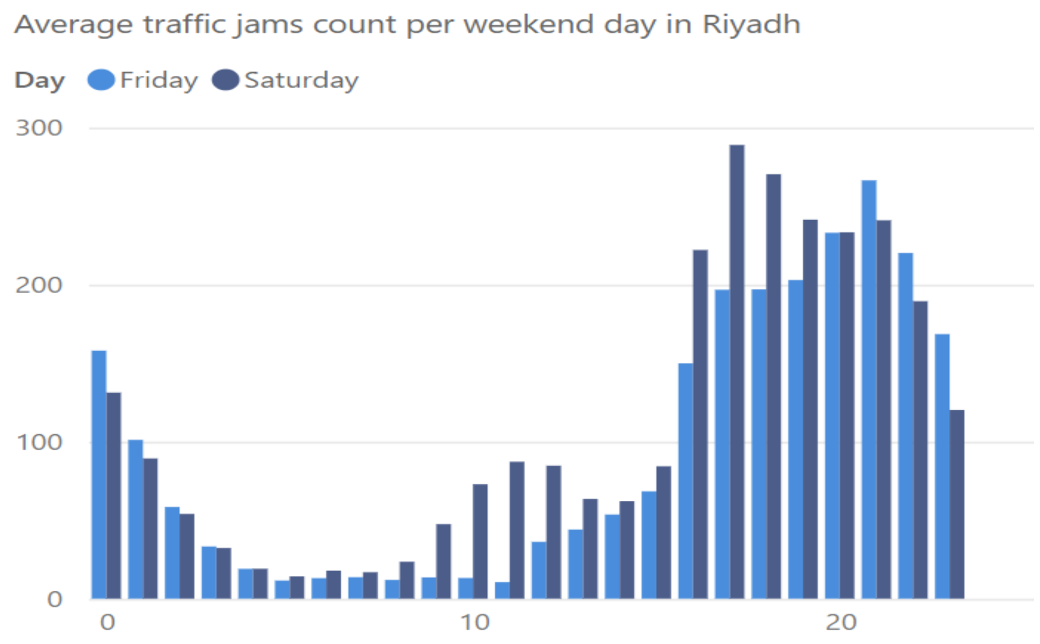
We can clearly see when traffic is at its highest in Riyadh, such as at 5:00 PM, due to factors like the time when most people commute home after work. Notably, at 12:00 PM, there are more jam counts, but these have less impact on actual delays. In contrast, at 5:00 PM, fewer jam counts cause significantly higher delays, highlighting the intensity of rush-hour congestion. Based on these observations, proposed recommendations include adjusting traffic signal timings to prioritize high-traffic periods, increasing public transportation options to reduce dependency on personal vehicles, and encouraging remote work to alleviate congestion during peak hours.

Compare the travel time live and the travel time historic during the months of Riyadh Season:



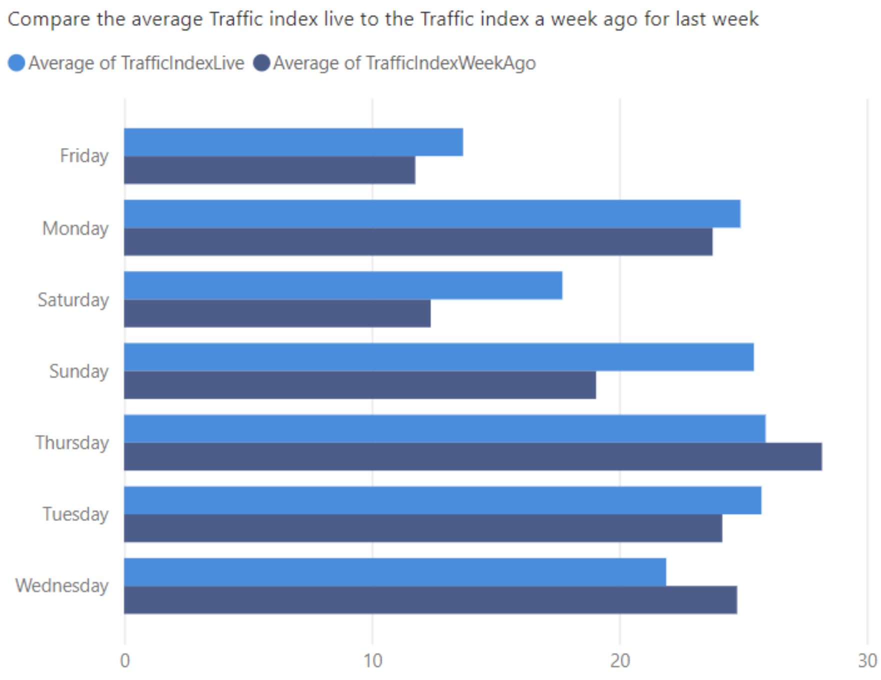
We found that the average Travel time after Riyadh Season was implemented increased, thus making the travel time for visitors worse. Our recommendation is to use apps and social media to provide real-time updates on traffic conditions and alternative routes, also to offer incentives for attendees who use public transport to attend the event to promote public transport.

Average traffic jams count per weekend day in Riyadh:



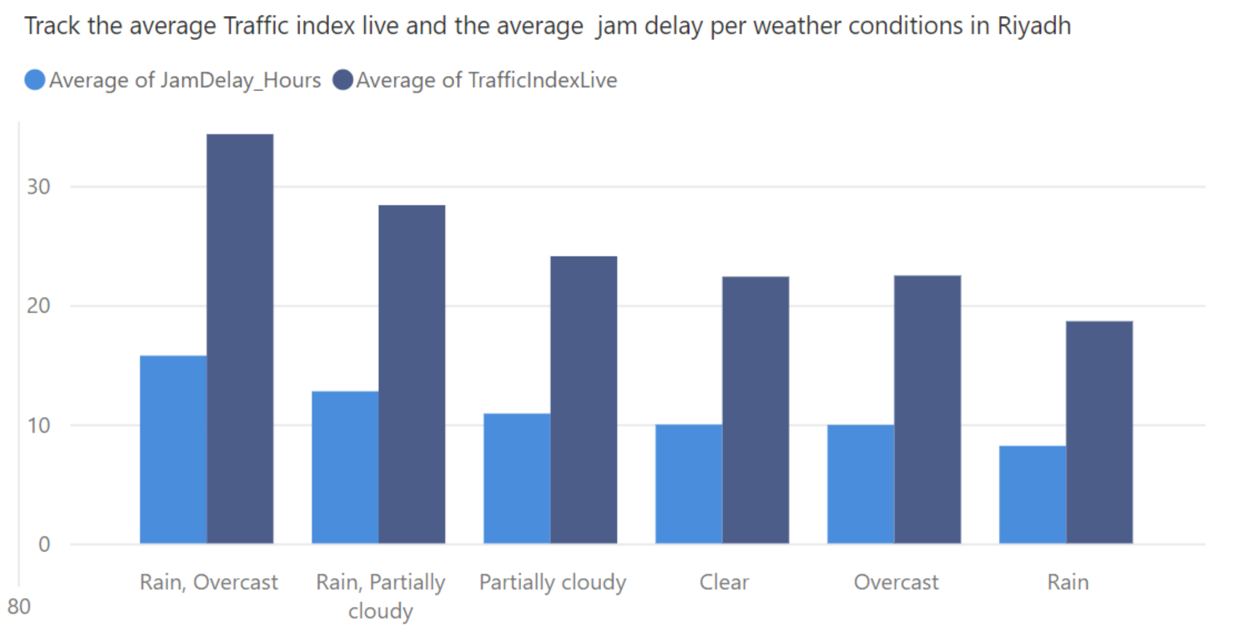
We found that the maximum value of jam counts happened to be at 5:00PM on Saturday, in which the average jam count was 288.8. Also, we found that generally the jam counts increased on both days after 3:00PM. Our recommendation is to work with businesses and recreational venues to offer discounts or promotions during off-peak hours. Also, to engage with shopping malls, sports arenas, and entertainment venues to adjust event timings and reduce congestion at peak hours.

Compare the average Traffic index live to the Traffic index a week ago for last week:



We found that the Traffic index live has increased slightly compared to the Traffic index a week ago for last week, which could be due to reasons such as accidents, construction, or special events. We only looked at records when the weather condition was clear, ruling out weather abnormalities as a possible cause. Additionally, we can drill down to see the hours of the day for more granular insights, allowing us to pinpoint specific times when Traffic index live had spiked. Some recommendations include enhancing emergency response by deploying dedicated quick-response teams to accident-prone areas to promptly clear minor accidents and restore traffic flow, leveraging communication channels such as social media, radio, and traffic apps to inform the public about ongoing construction and suggest alternate routes, and collaborating with event organizers to develop traffic management strategies, including designated parking areas and shuttle services for attendees.

Track the average Traffic index live and the average jam delay per weather conditions in Riyadh:



We found that rainy, overcast day had the highest average in both the Traffic index and jam delay, while the lowest was on rainy days only, based on this we recommend, installing advanced drainage systems and flood barriers at critical locations especially for rainy days, and using weather forecasting systems to preemptively alert authorities and the public about potential disruptions.

Conclusion

In conclusion, this project aimed to tackle the persistent issue of travel delays in Riyadh by analyzing historical traffic and weather data, with the goal of improving urban traffic management and productivity. Through an extensive ETL process, we cleansed the dataset, identified key patterns in traffic congestion, and developed Key Performance Indicators (KPIs) to assess factors such as rush hour delays, Riyadh Season impacts, weekend traffic trends, and the effect of weather conditions on traffic flow. This project also involved creating a comprehensive dashboard to visualize the KPIs, providing a clear and interactive way to track traffic patterns and performance metrics in Riyadh.

Our findings revealed critical insights, including peak traffic times, the influence of Riyadh Season events on travel times, and the correlation between weather conditions and congestion levels. Based on these insights, we proposed several actionable recommendations to alleviate traffic delays, such as optimizing traffic signal timings, promoting public transportation, adjusting event timings, enhancing emergency response strategies, and improving infrastructure to manage weather-related disruptions.

Work Distribution:

Name	ID	Tasks completed
Aljoharah Alsulami	443202090	-Star Schema -Applied data cleaning techniques using Python and documented a part of the process. -Created the shared Power BI file and collaborated with teammates. -Wrote the entire data cleaning process correctly -Fact and Dimension table descriptions -Wrote KPI #3, #4 and #5 -Created the dashboard -Visualized KPI #2, #3, #4 and #5 - Wrote all recommendations except #1 - Applied the Edits after receiving feedback to all parts of the project. -Added drill down ability on chart #4
Roaa Almutairi	443200427	-Problem/opportunity identification -Applied data cleaning techniques using Python and documented the data saving -Accessed the shared Power BI file. -Developed the presentation design and structure. -Formulated the conclusion of the Project
Hussa Aldriweesh	443201577	-Wrote Data Overview - Practiced Data Cleaning using Python, and documented the data loading - Logged into the shared Power BI file. - Applied the Edits after receiving feedback.
Reem Alotaibi	446200429	-KPIs #1 and #2 -Data cleaning (ETL) -Chart of rush hour -Wrote recommendation #1