Task 2.2

1. **Create a list of the elements you'd like to have on it—it's OK if you want to use the proposed structure in the Exercise, but if you feel strongly about a different format/plot, feel free to approach the research question in your own way.**

Dashboard Plan:

1. **Title and Introduction:** A prominent title and a brief introduction will set the context for the dashboard, explaining the purpose and objectives of the analysis.
2. **Data Summary:** Display key summary statistics of the dataset, such as the total number of records, average values, and other relevant metrics.

Data Visualization:

1. **Line Chart:** Visualize the trend of a numeric variable over time to identify any patterns or fluctuations.
2. **Bar Chart:** Compare categorical variables to understand distribution and identify dominant categories.
3. **Pie Chart:** Display the composition of a categorical variable to show the proportion of each category.

Interactive Filters:

1. **Dropdowns and Sliders:** Add interactive filters to allow users to choose specific time periods, regions, or other relevant dimensions to explore the data in more detail.
2. **Map Visualization:** Plot the data on a map to visualize geographical patterns and spatial distribution.
3. **Data Table:** Present the raw data in a tabular format, enabling users to view specific data points in detail.

2. **Write down some questions to guide your analysis in a new word-processing document and explain how you intend to visualize the result to answer each of your questions.**

Analysing these questions and to guide our analysis, here are some questions to explore

Questions:

How does bike usage vary by time of day and day of the week?
Visualization: Create a heatmap or line plot showing the bike trip counts on the y-axis and time of day (hourly intervals) on the x-axis. Use different colors or line styles for each day of the week.

What are the most popular starting and ending stations for bike trips?
Visualization: Create a bar chart or scatter plot with the station names on the x-axis and the number of trips on the y-axis. Use color-coding to differentiate between starting and ending stations.

Is there a relationship between bike usage and weather conditions?
Visualization: Plot bike trip counts on the y-axis and weather variables (e.g., temperature, precipitation) on the x-axis using line plots or scatter plots. Use different colors or markers to represent different weather conditions.

Are bike trips influenced by the day's weather forecast?
Visualization: Create a line chart or bar chart comparing bike trip counts against weather forecasts (e.g., sunny, rainy, cloudy). Use color-coding or different bar patterns for each weather forecast category.

How does bike usage differ between members and casual users?
Visualization: Generate a bar chart with user types on the x-axis and the number of trips on the y-axis. Use different colors for member and casual user categories.

Does bike usage change across different seasons of the year?
Visualization: Create a line plot or bar chart with the number of trips on the y-axis and months on the x-axis to observe seasonal variations in bike usage.

What are the most common trip durations?
Visualization: Generate a histogram of trip durations with trip duration intervals on the x-axis and the count of trips falling within each interval on the y-axis.

Are there specific regions or neighbourhoods with high bike usage?
Visualization: Utilize a heatmap or choropleth map to display bike trip density in different regions of the city. Use colour gradients to indicate trip intensity.

Does bike usage change during holidays or special events?
Visualization: Plot bike trip counts on the y-axis and dates (including holidays and events) on the x-axis using line plots or bar charts. Use annotations or color-coding to highlight holidays and events.

These questions and visualization techniques can provide valuable insights into the patterns of bike usage in correlation with weather conditions, user types, and other variables. We should also consider that data exploration and visualization are iterative processes, and you may discover new questions or insights as you delve deeper into the data.

3. **In a new notebook, import all necessary libraries, read in your data, and join it. Hint: what's the most effective way to import and join data in such a format?**

The most effective way to import and join data in such a format (multiple CSV files) depends on the size of the data, available system resources, and the specific requirements of the task. Here are some effective strategies to consider:

Pandas pd.concat(): The Pandas pd.concat() function used in the provided code is a straightforward and efficient way to concatenate Data Frames vertically (along rows). It works well when the number of CSV files and the size of individual files are relatively small. For smaller datasets, this method is simple and effective.

Dask: If the dataset is too large to fit into memory, Dask can be a more efficient solution. Dask is a parallel computing library that can handle larger-than-memory datasets by breaking them into smaller partitions and processing them in parallel.

SQL Database: If you have access to an SQL database, you can import each CSV file into separate tables and then perform SQL JOIN operations to combine the data efficiently. SQL databases are optimized for handling large datasets and can perform joins and other operations efficiently.

Apache Spark: For even larger datasets and distributed computing, Apache Spark can be used. Spark allows you to work with data across a cluster of machines and perform distributed data processing, making it suitable for big data scenarios.

Data Pre-processing: Depending on your specific use case, you might also consider pre-processing the data before combining it. For example, if the data is too large to fit into memory, you could filter and aggregate the data in each CSV file separately, saving the processed results as intermediate files, and then merge those intermediate files.

In conclusion, the choice of the most effective method for importing and joining data from multiple CSV files depends on the size of the data and the available computing resources. For smaller datasets, Pandas' pd.concat() is often sufficient. As the dataset size grows, Dask, SQL databases, Apache Spark, or data pre-processing techniques may be more appropriate to handle the data efficiently.

The dataset which we have been provided for the analysis is large in size so we will be using Dask and pd.concat for data merging task.