DATA SCIENCE PROJECT:

DELHI METRO NETWORK ANAYSIS

(By Mahak Bisht & Radhika Sharma)

DELHI METRO NETWORK ANALYSIS PROJECT SUMMARY

Our project provides a comprehensive exploration of the Delhi Metro system, focusing on the number of stations, their geographical locations, metro lines, and distances between stations. Utilizing Python for data analysis and visualization, we created an informative representation of the network.

Key Highlights:

A.Stations and Locations:

Mapped and analyzed all metro stations, revealing distribution and accessibility across the city.

B.Metro Lines:

Distinctly represented each line with unique colors and quantified station counts for a detailed network breakdown.

C.Distance Analysis:

Measured distances from first stations, using histograms to illustrate distribution.

Visualizations: Included pie charts and histograms to enhance data interpretability and summarize network features.

INPUT

```
# 1. TO READ DATA FROM DELHI METRO CSV FILE

import pandas as pd
df = pd.read_csv(r"C:\Users\DELL\Documents\Delhi metro data sheet.csv")
df
```

SUMMARY

This code will:

1. Import the numpy and pandas libraries.
2. Load the dataset from the specified path into a DataFrame named df.
3. Print the first five rows of the DataFrame to give you a quick look at the data.

OUTPUT

	ID (Station ID)	Station Names	Dist. From First Station(km)	Metro Line	Opened(Year)	Layout	Latitude	Longitude
0	1.0	Shaheed Sthal(First Station)	0.0	Red line	08-03-2019	Elevated	28.670611	77.415582
1	2.0	Hindon River	1.0	Red line	08-03-2019	Elevated	28.878965	77.415483
2	3.0	Arthala	2.5	Red line	08-03-2019	Elevated	28.676999	77.391892
3	4.0	Mohan Nagar	3.2	Red line	08-03-2019	Elevated	28.606319	77.106082
4	5.0	Shyam park	4.5	Red line	08-03-2019	Elevated	28.698807	28.698807
...	...	...	...	...	...	...	...	...
333	NaN	Conn: Pink	NaN	NaN	NaN	NaN	NaN	NaN
334	4.0	Delhi Aerocity	14.5	Orange line	15-08-2011	Underground	28.548810	77.120920
335	5.0	IGI Airport	17.9	Orange line	23-02-2011	Underground	28.556930	77.086690
336	6.0	Dwarka Sector 21	20.8	Orange line	23-02-2011	Underground	28.552260	77.058280
337	NaN	Conn: Blue	NaN	NaN	NaN	NaN	NaN	NaN

SUMMARY

To display the Data Frame of the data in the csv file

INPUT

```
# 3. DISPLAY INFORMATION ABOUT THE COLUMNS

import pandas as pd
df = pd.read_csv(r"C:\Users\DELL\Documents\Delhi metro data sheet.csv")
df.info()
```

SUMMARY

a.Import Libraries: Load NumPy and pandas.

b.Read CSV File: Load data from the CSV file into a DataFrame.

c.Display Information: Show details about the DataFrame, including column names, data types, and non-null counts.

OUTPUT

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 338 entries, 0 to 337
Data columns (total 8 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   ID (Station ID)                 285 non-null    float64
 1   Station Names                   335 non-null    object
 2   Dist. From First Station(km)    285 non-null    float64
 3   Metro Line                      285 non-null    object
 4   Opened(Year)                    285 non-null    object
 5   Layout                          285 non-null    object
 6   Latitude                        286 non-null    float64
 7   Longitude                       286 non-null    float64
dtypes: float64(4), object(4)
memory usage: 21.3+ KB
```

SUMMARY

Load CSV file into DataFrame and display its information

INPUT

```
# 5. SHOW NULL VALUES IN EACH COLUMN

import pandas as pd
df = pd.read_csv(r"C:\Users\DELL\Documents\Delhi metro data sheet.csv")
null=df.isnull().sum()
print('null values in each column is:\n',null)
```

SUMMARY

limport Libraries: Import the necessary libraries, numpy and pandas.

b.Load Data: Read the CSV file containing the Delhi Metro data into a DataFrame using pd.read_csv().

c.Check Missing Values: Use the isnull() method to identify missing values in the DataFrame and sum() to count the number of missing values in each column.

d.Print Summary: Output the summary of missing values for each column.

OUTPUT

```
null values in each column is:
 ID (Station ID)                  53
Station Names                      3
Dist. From First Station(km)      53
Metro Line                        53
Opened(Year)                      53
Layout                            53
Latitude                          52
Longitude                         52
dtype: int64
```

SUMMARY

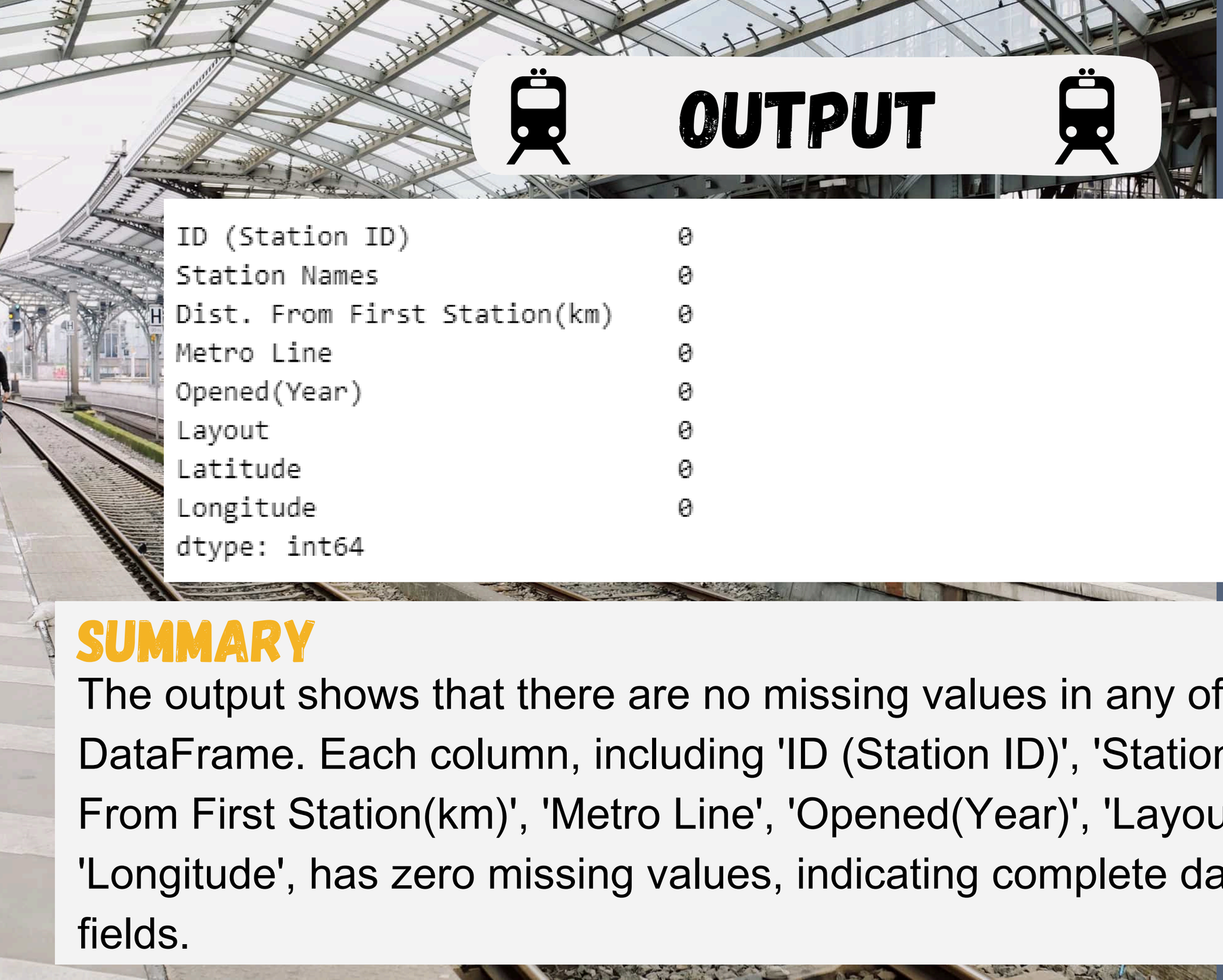
The output indicates the presence of null values in the DataFrame. Specifically, 'ID (Station ID)', 'Dist. From First Station(km)', 'Metro Line', 'Opened(Year)', and 'Layout' each have 53 null values. 'Station Names' has 3 null values, while 'Latitude' and 'Longitude' have 52 null values each. This highlights the columns with missing data that may need to be addressed for further analysis

INPUT

```
# 6. DROP ROWS WITH NULL VALUES

import pandas as pd
df = pd.read_csv(r"C:\Users\DELL\Documents\Delhi metro data sheet.csv")
df=df.dropna()
df.isnull().sum()
```

SUMMARY

This code will print the number of null values for each column in the DataFrame after dropping rows with any missing values.



OUTPUT

```
ID (Station ID)                  0
Station Names                    0
Dist. From First Station(km)     0
Metro Line                       0
Opened(Year)                     0
Layout                           0
Latitude                         0
Longitude                        0
dtype: int64
```

SUMMARY

The output shows that there are no missing values in any of the columns of the DataFrame. Each column, including 'ID (Station ID)', 'Station Names', 'Dist. From First Station(km)', 'Metro Line', 'Opened(Year)', 'Layout', 'Latitude', and 'Longitude', has zero missing values, indicating complete data for all these fields.

INPUT

```
# 7. PRINT UNIQUE VALUES IN 'METRO LINE' COLUMN

import pandas as pd
df = pd.read_csv(r"C:\Users\DELL\Documents\Delhi metro data sheet.csv")
print(df['Metro Line'].unique())
```

SUMMARY

This code snippet demonstrates how to load data from a CSV file into a pandas DataFrame and then print the unique values in the 'Metro Line' column. This can be useful for understanding the different metro lines present in the dataset.

OUTPUT

```
['Red line' nan 'Yellow line' 'Blue line' 'Blue line branch'
 'Green line branch' 'Green line' 'Rapid Metro' 'Voilet line'
 'Magenta line' 'Pink line' 'Aqua line' 'Gray line' 'Orange line']
```

SUMMARY

Display all the unique metro line in the data frame

INPUT

```
# 8. TO IDENTIFY THE PIE CHART FOR METRO LINE

import pandas as pd
import matplotlib.pyplot as plt

# Create a dictionary to map Metro Line names to colors
line_colors = {
    'Red line': 'red',
    'Yellow line': 'yellow',
    'Blue line': 'blue',
    'Blue line branch': 'navy',
    'Green line branch': 'forestgreen',
    'Green line': 'green',
    'Rapid Metro': 'silver',
    'Voilet line': 'purple',
    'Magenta line': 'magenta',
    'Pink line': 'pink',
    'Aqua line': 'skyblue',
    'Gray line': 'gray',
    'Orange line': 'orange'
}

# Calculate value counts for each Metro Line
line_counts = df['Metro Line'].value_counts()

# Create a pie chart using custom colors
plt.pie(line_counts, labels=line_counts.index, autopct='%1.1f%%', colors=[line_colors[line] for line in line_counts.index])

 # Set aspect ratio to be equal, so the pie is drawn as a circle.
 plt.axis('equal')

 # Add a title
 plt.title('Metro Line Distribution')

 # Display the pie chart
 plt.show()
```

SUMMARY

- Import Libraries: The code imports the pandas library as pd for data manipulation and matplotlib.pyplot as plt for creating visualizations.

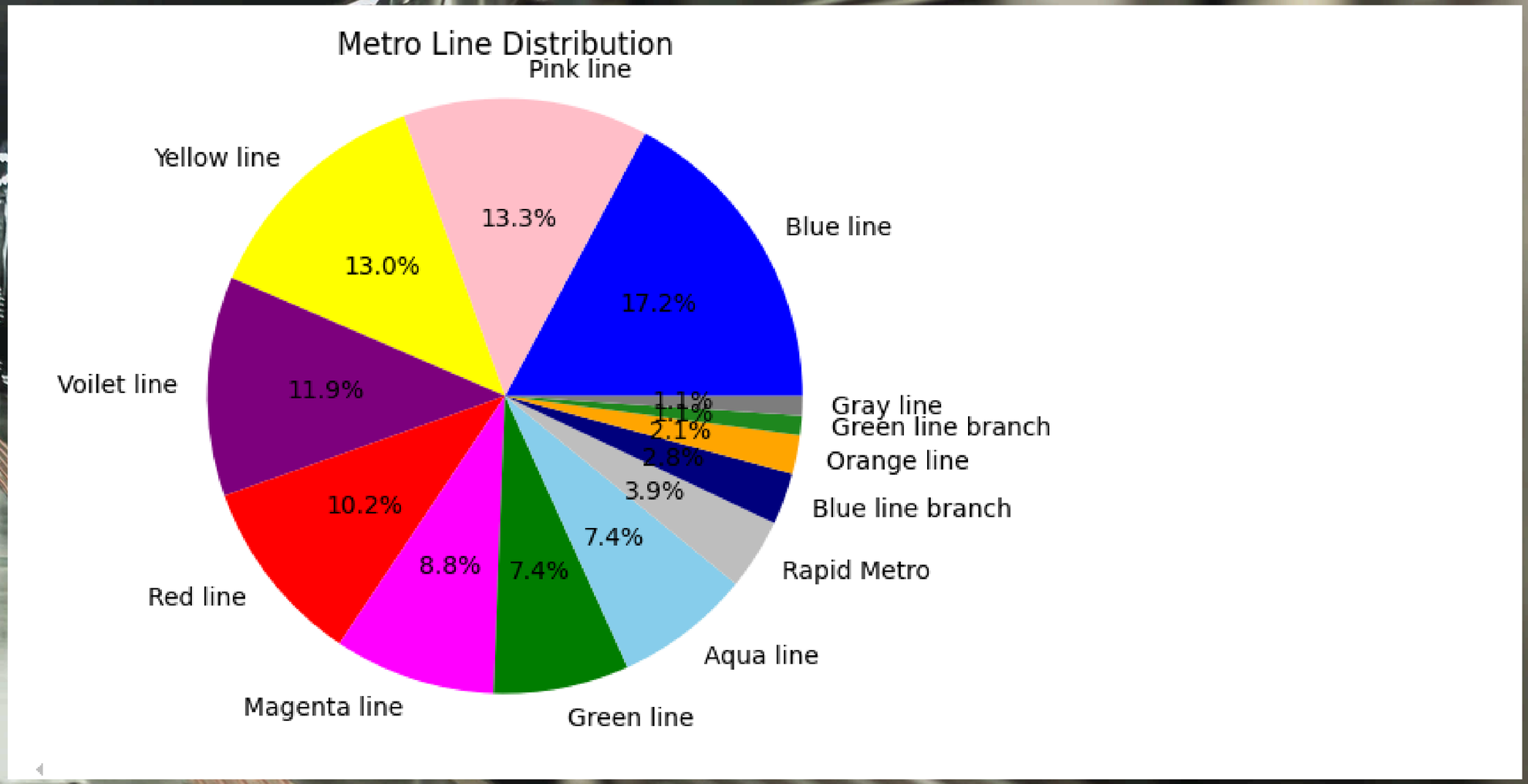
- Define Color Mapping: A dictionary named line_colors maps each Delhi Metro Line to a specific color. This will be used to ensure that each segment of the pie chart has a consistent and meaningful color.

- Calculate Value Counts: The code calculates the count of occurrences for each Metro Line in the DataFrame df_cleaned. The resulting series (line_counts) contains the Metro Line names as the index and their respective counts as the values.

- Create Pie Chart:

- The plt.pie() function is used to create a pie chart. The line_counts series provides the data for the chart, with the labels parameter using the index of the series to label each segment.
- The autopct parameter is set to '%1.1f%%' to display the percentage of each segment.
- The colors parameter uses a list comprehension to assign the colors from the line_colors dictionary to each Metro Line in line_counts.index.
- Aspect Ratio and Title:
  The plt.axis('equal') command ensures the pie chart is drawn as a circle.
  The plt.title() function sets the title of the chart to "Metro Line Distribution".
  Display Chart: The plt.show() function displays the pie chart.

OUTPUT


Metro Line Distribution
Pink line
13.3%
Blue line
17.2%
Gray line
1.1%
Green line branch
2.1%
Orange line
2.8%
Blue line branch
3.9%
Rapid Metro
7.4%
Aqua line
7.4%
Green line
8.8%
Magenta line
10.2%
Red line
11.9%
Voilet line
13.0%
Yellow line

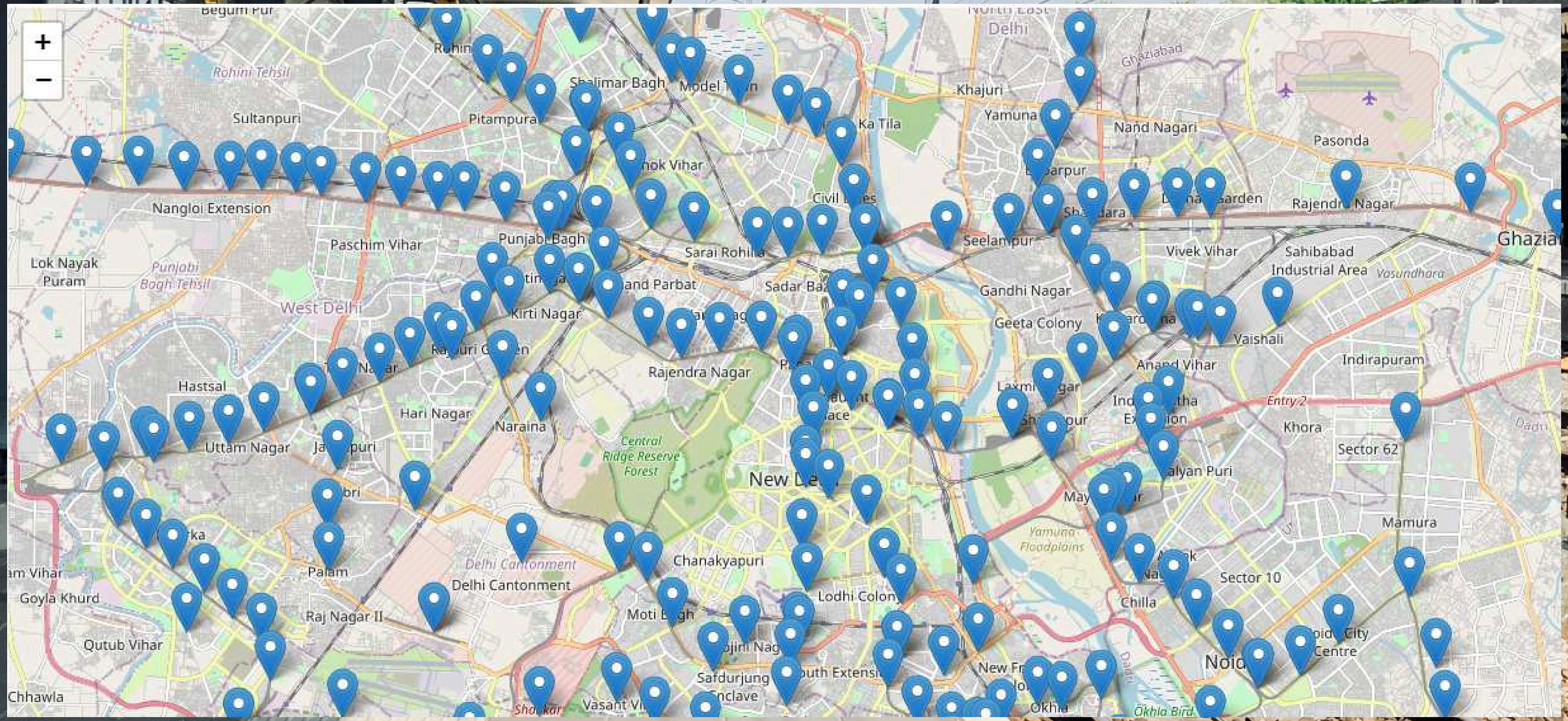
INPUT

```
# 8. CREATING AN INTERACTIVE MAP OF DELHI METRO STATION

import pandas as pd
import folium

# Create a base map centered on Delhi
delhi_map = folium.Map(location=[28.6139, 77.2090], zoom_start=12)

# Add markers for each metro station
for index, row in df.iterrows():
    folium.Marker([row['Latitude'], row['Longitude']], popup=row['Station Names']).add_to(delhi_map)

# Display the map
display(delhi_map)
```

SUMMARY

This code snippet demonstrates how to create an interactive map of Delhi using the Folium library. The map is centered on Delhi, and markers are added for each metro station using the latitude and longitude coordinates from the dataset. Each marker includes a popup displaying the station name. The map is displayed interactively within the Jupyter notebook environment.

OUTPUT


Rohini Tehsil
Sultanpuri
Pitampura
Shalimar Bagh
Nangloi Extension
Paschim Vihar
Punjabi Bagh
Punjabi Bagh Tehsil
Lok Nayak Puram
West Delhi
Kirti Nagar
Rajendra Nagar
Sarai Rohilla
Hastsal
Uttam Nagar
Hari Nagar
Naraina
Central Ridge Reserve Forest
New Delhi
Delhi Cantonment
Chanakyapuri
Lodhi Colony
Moti Bagh
Palam
Goyla Khurd
Raj Nagar II
Qutub Vihar
Chhawla
Vasant Vihar
Safdurjung Enclave
Okhla
Okhla Bird
Yamuna Floodplains
Khajuri
Yamuna
Nand Nagari
Pasonda
Seelampur
Vivek Vihar
Sahibabad Industrial Area
Vasundhara
Gandhi Nagar
Geeta Colony
Vaishali
Anand Vihar
Indirapuram
Khora
Sector 62
Mamura
Sector 10
Chilla
Noida
Delhi
Ghaziabad

INPUT

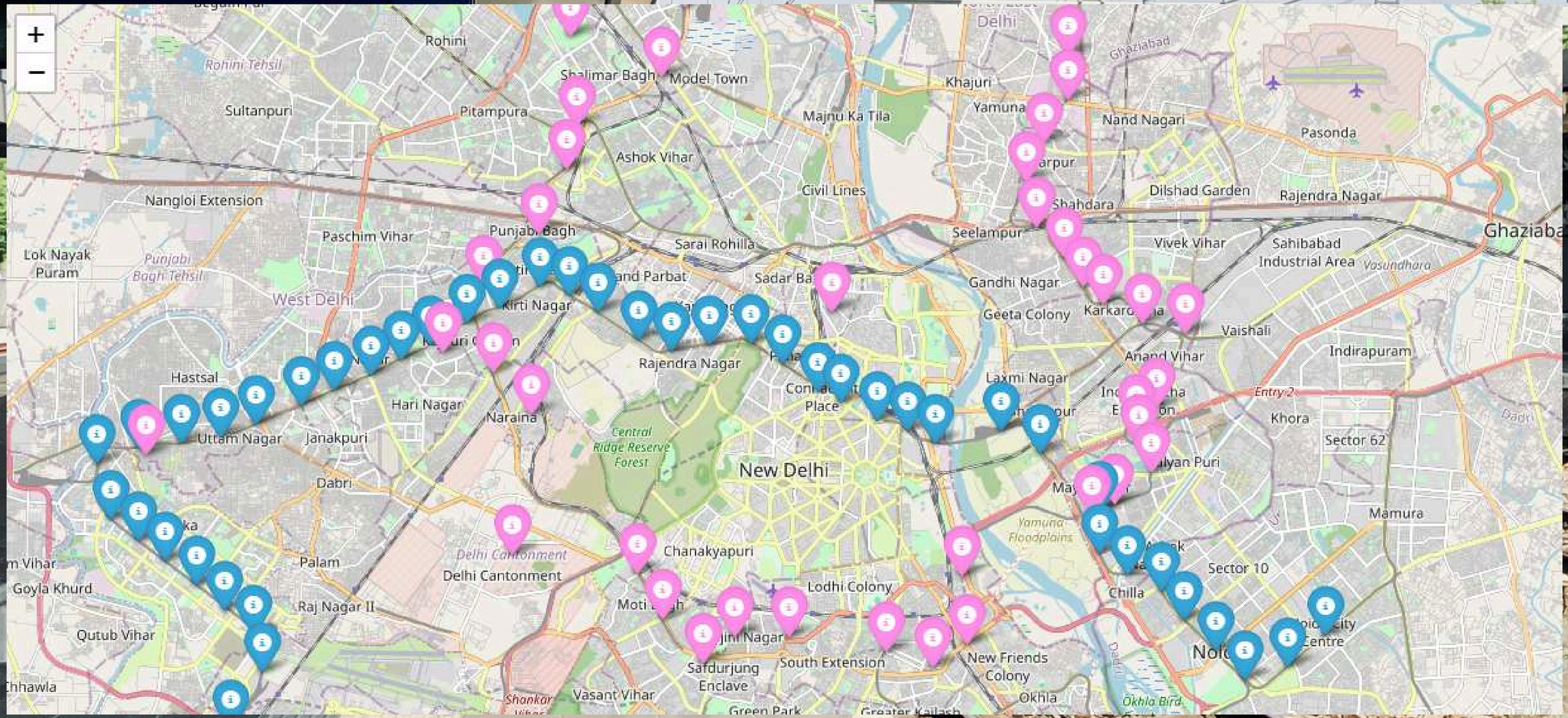
```
# 10. TO DISPLAY INTERACTIVE MAP OF BLUE AND PINK LINE

import pandas as pd
import folium
from IPython.display import display

# Load the data from your CSV file
df = pd.read_csv(r"C:\Users\DELL\Documents\Delhi metro data sheet.csv")

# Filter the DataFrame to include only the blue and pink line stations
stations_to_plot = df[(df['Metro Line']=='Blue line') |(df['Metro Line']=='Pink line')]

# Debug: Print the filtered DataFrame
#print(stations_to_plot)

# Check if the necessary columns exist
required_columns = ['Station Names', 'Latitude', 'Longitude', 'Metro Line']
for column in required_columns:
    if column not in df.columns:
        raise ValueError(f"Column '{column}' not found in the DataFrame")

# Check for any missing values in the required columns
if stations_to_plot[required_columns].isnull().any().any():
    raise ValueError("There are missing values in the required columns")

# Create a base map centered on Delhi
delhi_map = folium.Map(location=[28.6139, 77.2090], zoom_start=12)
```

```
# Add markers for each metro station with different colors based on the 'Line' column
for index, row in stations_to_plot.iterrows():
    color = 'blue' if row['Metro Line'] == 'Blue line' else 'pink'
    folium.Marker(
       location=[row['Latitude'], row['Longitude']],
        popup=row['Station Names'],
        icon=folium.Icon(color=color)
    ).add_to(delhi_map)

# Display the map in a Jupyter Notebook or other interactive environment
display(delhi_map)
```

SUMMARY

The script loads a CSV file containing Delhi Metro data into a Pandas DataFrame and filters it to include only the Blue and Pink line stations. It ensures the required columns ('Station Names', 'Latitude', 'Longitude', 'Metro Line') are present and checks for any missing values in these columns. Using the Folium library, it creates an interactive map centered on Delhi, plotting each metro station with markers colored blue for Blue line stations and pink for Pink line stations. Finally, it displays the map in an interactive environment such as a Jupyter Notebook.

OUTPUT


Rohini
Rohini Tehsil
Sultanpuri
Pitampura
Shalimar Bagh
Model Town
Majnu Ka Tila
Ashok Vihar
Civil Lines
Nangloi Extension
Paschim Vihar
Punjabi Bagh
Sarai Rohilla
Lok Nayak Puram
Punjabi Bagh Tehsil
West Delhi
Kirti Nagar
Rajendra Nagar
Hastsal
Hari Nagar
Naraina
Uttam Nagar
Janakpuri
Central Ridge Reserve Forest
New Delhi
Dabri
Palam
Delhi Cantonment
Chanakyapuri
Goyla Khurd
Raj Nagar II
Qutub Vihar
Lodhi Colony
Safdurjung Enclave
South Extension
Vasant Vihar
Shankar Vihar
Green Park
Greater Kailash
New Friends Colony
Okhla
Okhla Bird
Delhi
Khajuri
Yamuna
Nand Nagari
Pasonda
Shahdara
Dilshad Garden
Rajendra Nagar
Seelampur
Vivek Vihar
Sahibabad Industrial Area
Vasundhara
Gandhi Nagar
Geeta Colony
Vaishali
Anand Vihar
Indirapuram
Laxmi Nagar
Khora
Sector 62
Yamuna Floodplains
Mamura
Sector 10
Chilla

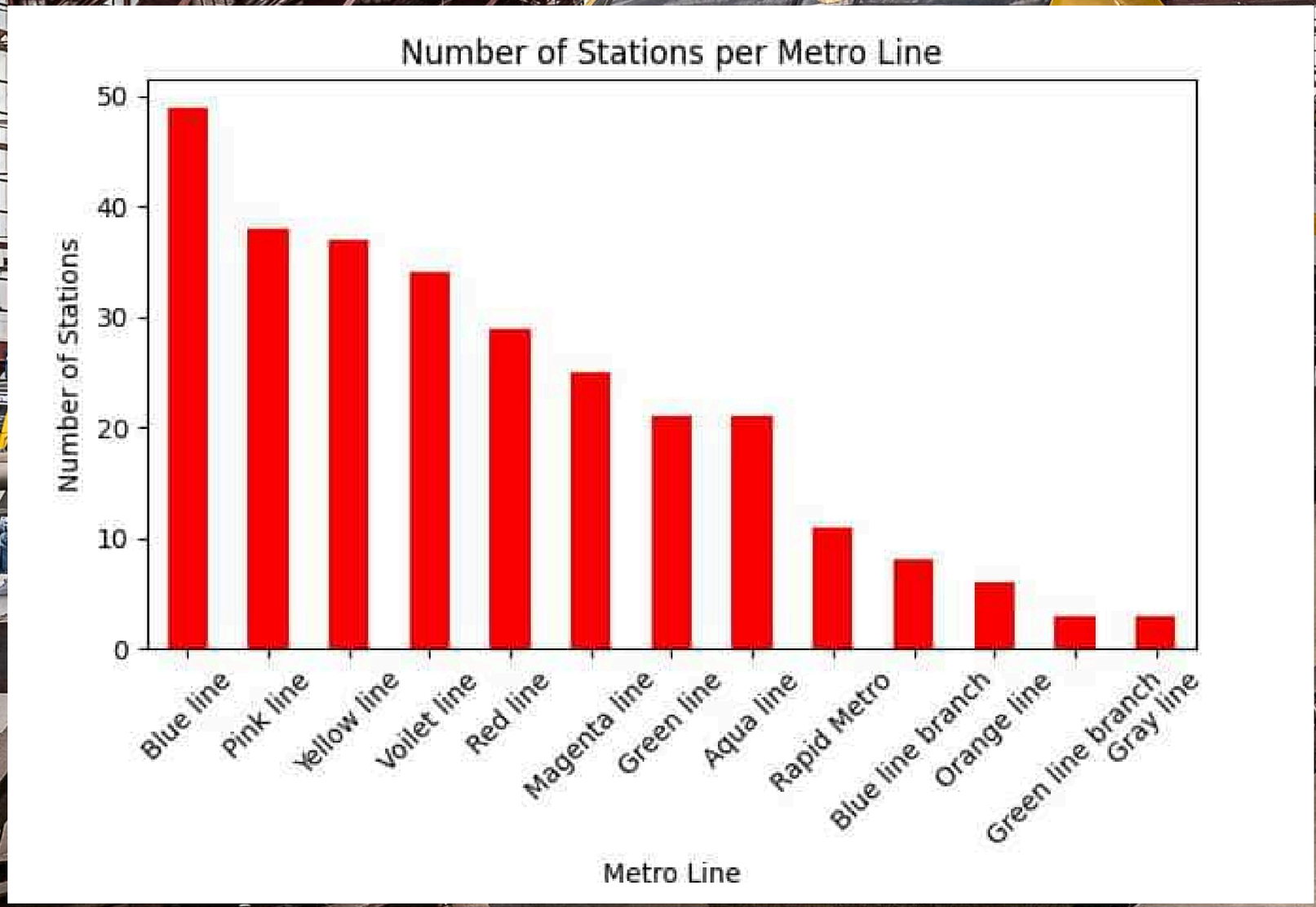
INPUT

```
# 12.BAR PLOT OF STATION COUNT BY METRO LINE

df['Metro Line'].value_counts().plot(kind='bar',color='red')
plt.xlabel('Metro Line')
plt.ylabel('Number of Stations')
plt.title('Number of Stations per Metro Line')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

SUMMARY

The code generates a bar chart using the Pandas DataFrame to visualize the number of stations per Metro Line. It uses the value_counts() method to count the number of stations for each Metro Line, then plots these counts as a bar chart with red bars. The x-axis is labeled "Metro Line," the y-axis is labeled "Number of Stations," and the title of the chart is "Number of Stations per Metro Line." The x-axis labels are rotated 45 degrees for better readability, and the layout is adjusted to ensure all elements fit well. Finally, the chart is displayed using plt.show().

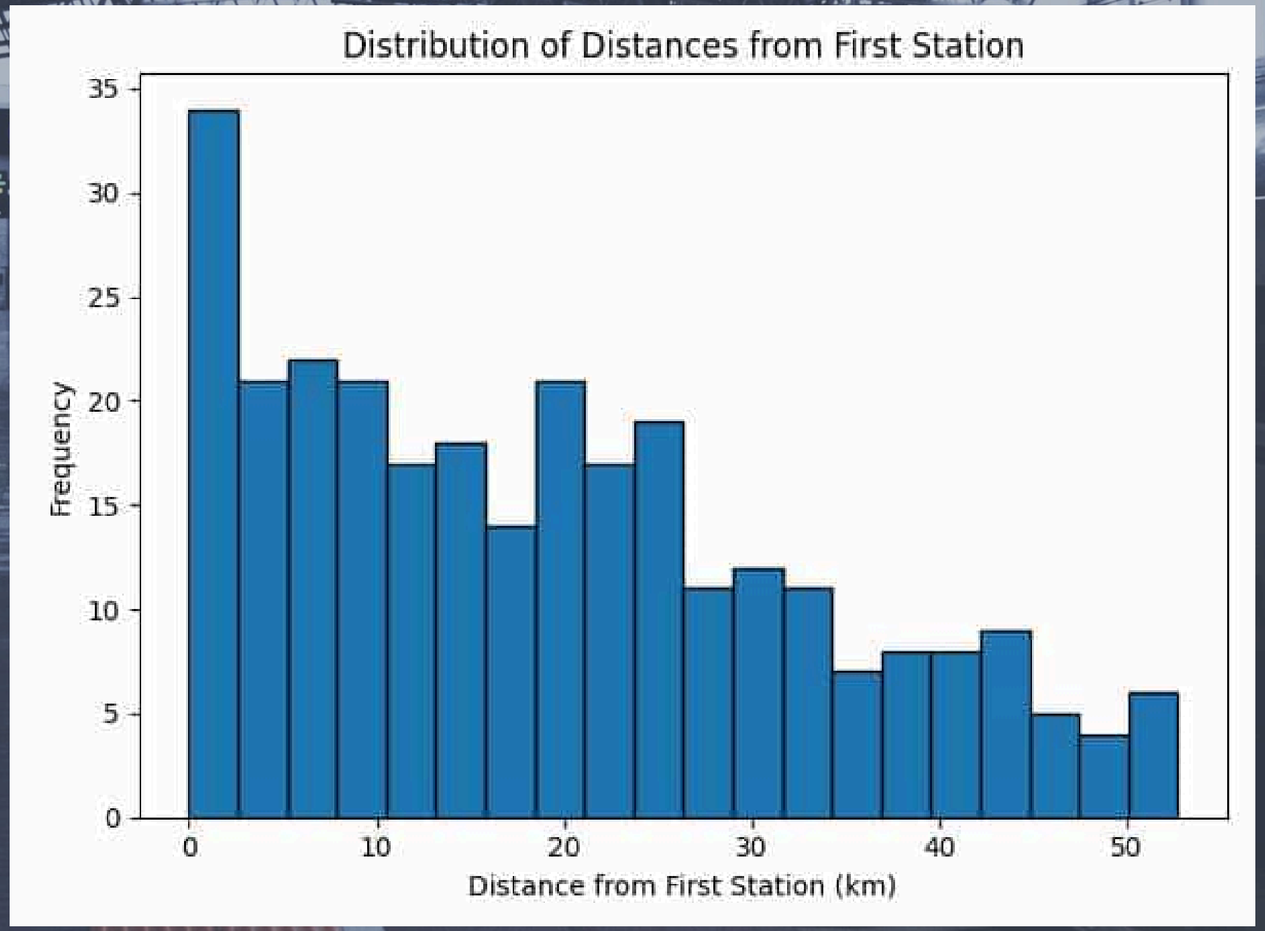
OUTPUT


Number of Stations per Metro Line
Number of Stations
0
10
20
30
40
50
Blue line
Pink line
Yellow line
Voilet line
Red line
Magenta line
Green line
Aqua line
Rapid Metro
Blue line branch
Orange line
Green line branch
Gray line
Metro Line

INPUT

```
# 13.  DISPLAY DISTRIBUTION OF DISTANCES FROM FIRST STATION

import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv(r"C:\Users\DELL\Documents\Delhi metro data sheet.csv")

plt.hist(df['Dist. From First Station(km)'], bins=20,edgecolor='black')
plt.xlabel('Distance from First Station (km)')
plt.ylabel('Frequency')
plt.title('Distribution of Distances from First Station')
plt.tight_layout()
plt.show()
```

OUTPUT



SUMMARY

Import Libraries: The code assumes pandas and matplotlib.pyplot have already been imported.
Create Histogram: The plt.hist() function is used to create a histogram.
The data for the histogram is taken from the Dist. From First Station(km) column of the DataFrame df_cleaned.
The bins parameter is set to 20, which divides the data into 20 intervals (bins).
The edgecolor parameter is set to 'black', which adds a black border around each bin for better visual separation.
Add Labels and Title:
The plt.xlabel() function sets the label for the x-axis to 'Distance from First Station (km)'.
The plt.ylabel() function sets the label for the y-axis to 'Frequency'.
The plt.title() function sets the title of the chart to 'Distribution of Distances from First Station'.
Adjust Layout: The plt.tight_layout() function adjusts the padding of the plot to make sure everything fits within the figure area.
Display Histogram: The plt.show() function displays the histogram.

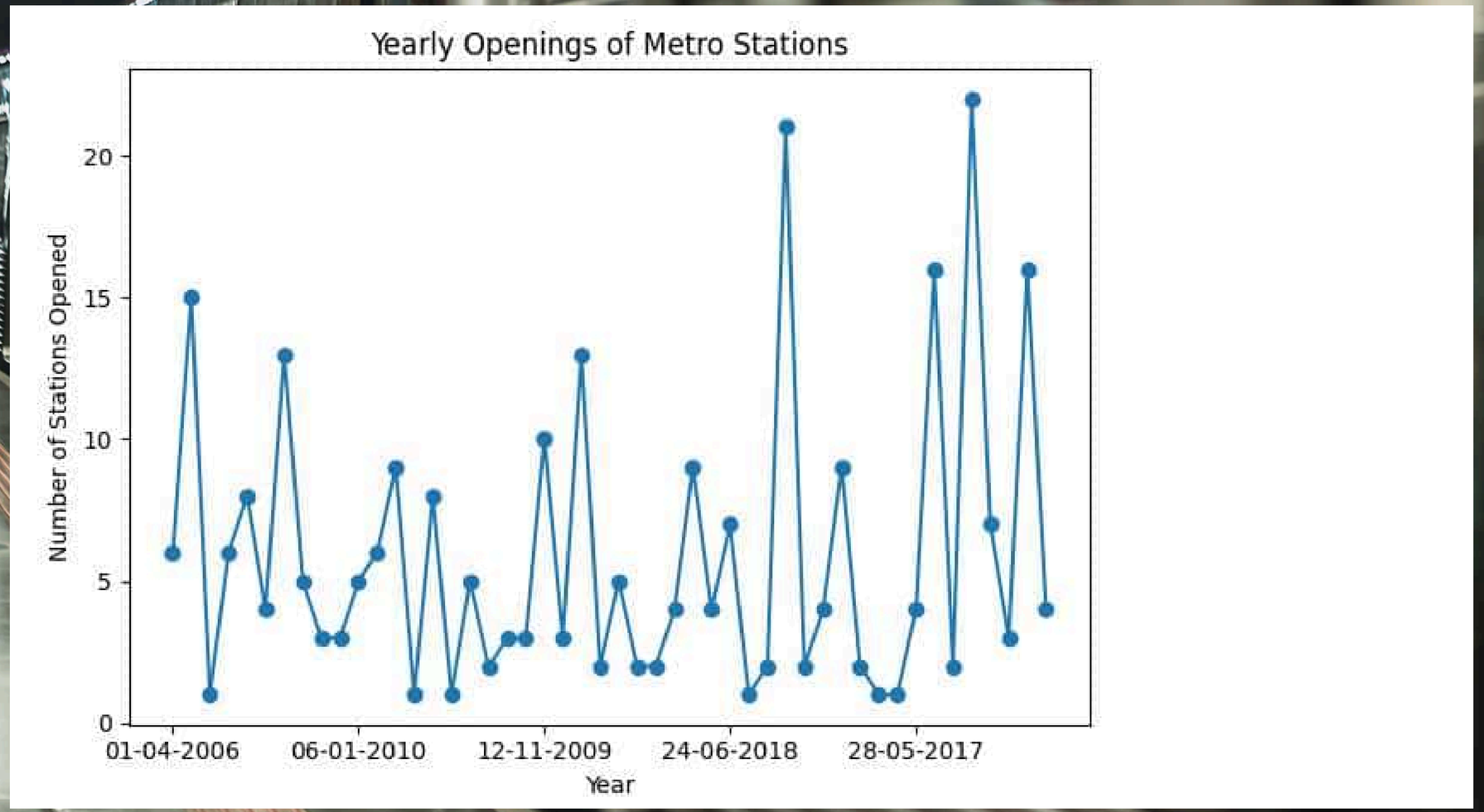
INPUT

```
# 14. DISPLAY LINE PLOT YEARLY OPENINGS

yearly_openings = df.groupby('Opened(Year)').size()
yearly_openings.plot(kind='line', marker='o')
plt.xlabel('Year')
plt.ylabel('Number of Stations Opened')
plt.title('Yearly Openings of Metro Stations')
plt.tight_layout()
plt.show()
```

SUMMARY

The code analyzes the number of metro stations opened each year by grouping the data in the DataFrame by the 'Opened(Year)' column. It then plots this data as a line chart with markers indicating the number of stations opened each year. The x-axis is labeled "Year," the y-axis is labeled "Number of Stations Opened," and the title of the chart is "Yearly Openings of Metro Stations." The layout is adjusted for better fit, and the chart is displayed using plt.show()

OUTPUT


Yearly Openings of Metro Stations
Number of Stations Opened
0
5
10
15
20
01-04-2006
06-01-2010
12-11-2009
24-06-2018
28-05-2017
Year

INPUT

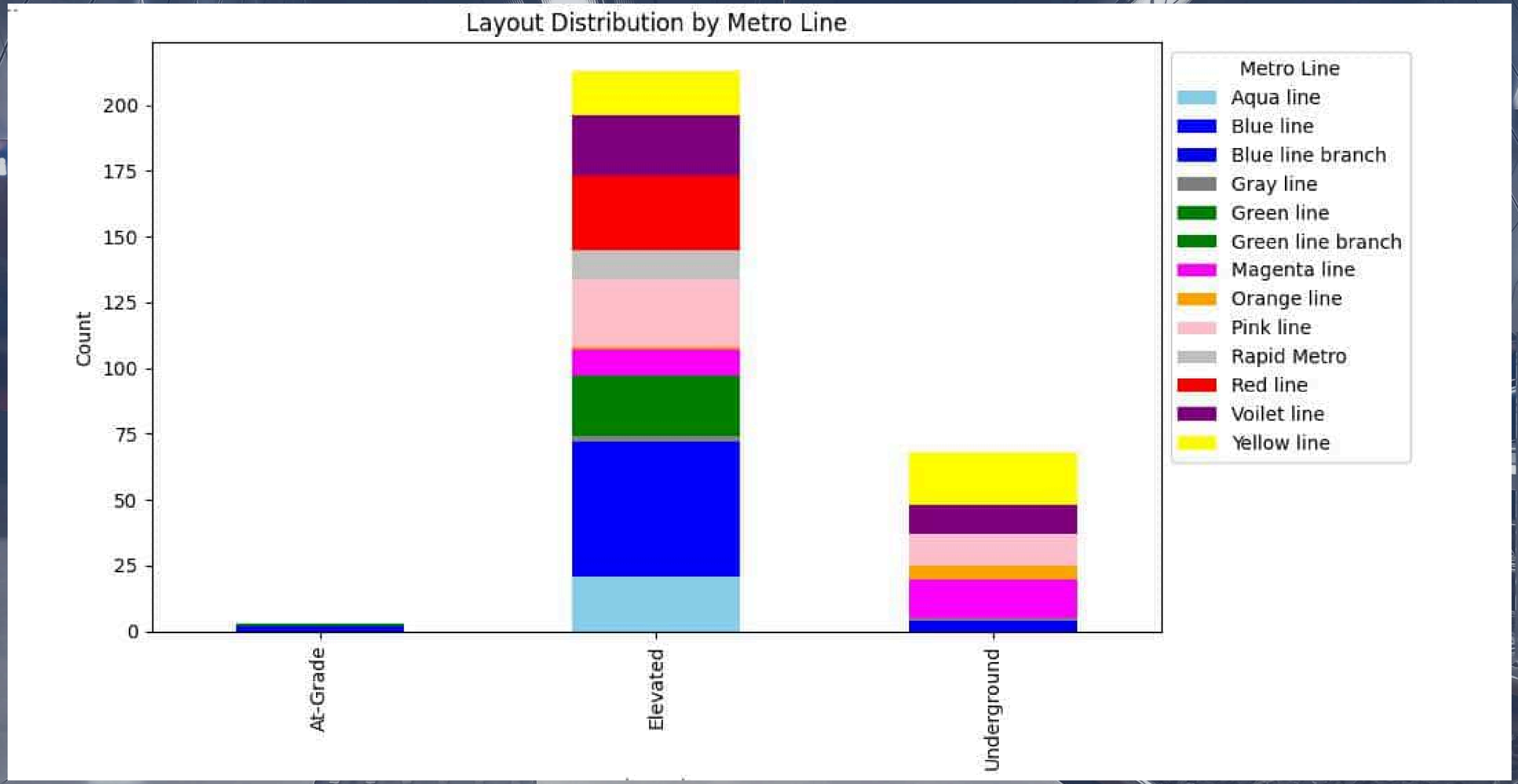
```
# 15. DISPLAY BAR PLOT OF LAYOUT BY METRO LINE

# Create a cross-tabulation of Layouts and Metro Lines
layout_by_line = pd.crosstab(df['Layout'], df['Metro Line'])

# Define line colors according to the pattern
line_colors = {
    'Red line': 'red',
    'Yellow line': 'yellow',
    'Blue line': 'blue',
    'Blue line branch': 'blue',
    'Green line branch': 'green',
    'Green line': 'green',
    'Rapid Metro': 'silver',
    'Voilet line': 'purple',
    'Magenta line': 'magenta',
    'Pink line': 'pink',
    'Aqua line': 'skyblue',
    'Gray line': 'gray',
    'Orange line': 'orange'
}
# Create a list of colors corresponding to each line
colors = [line_colors[line] for line in layout_by_line.columns]
```

```
# Plot a stacked bar plot with specified colors
layout_by_line.plot(kind='bar', stacked=True, figsize=(10, 6), color=colors)
plt.xlabel('Layout')
plt.ylabel('Count')
plt.title('Layout Distribution by Metro Line')
plt.legend(title='Metro Line', loc='upper left', bbox_to_anchor=(1, 1))
plt.tight_layout()
plt.show()
```

SUMMARY

The code creates a cross-tabulation of metro station layouts and metro lines from the DataFrame, summarizing the distribution of different layouts across various metro lines. It then defines a color mapping for each metro line and creates a list of colors corresponding to the lines in the cross-tabulation. A stacked bar plot is generated to visualize the distribution, with each bar section colored according to the respective metro line. The x-axis is labeled "Layout," the y-axis is labeled "Count," and the chart title is "Layout Distribution by Metro Line." The legend is placed outside the plot for clarity, and the layout is adjusted for better fit before displaying the chart.

OUTPUT


Layout Distribution by Metro Line
Count
0
25
50
75
100
125
150
175
200
At-Grade
Elevated
Underground
Metro Line
Aqua line
Blue line
Blue line branch
Gray line
Green line
Green line branch
Magenta line
Orange line
Pink line
Rapid Metro
Red line
Voilet line
Yellow line


Thank
You!