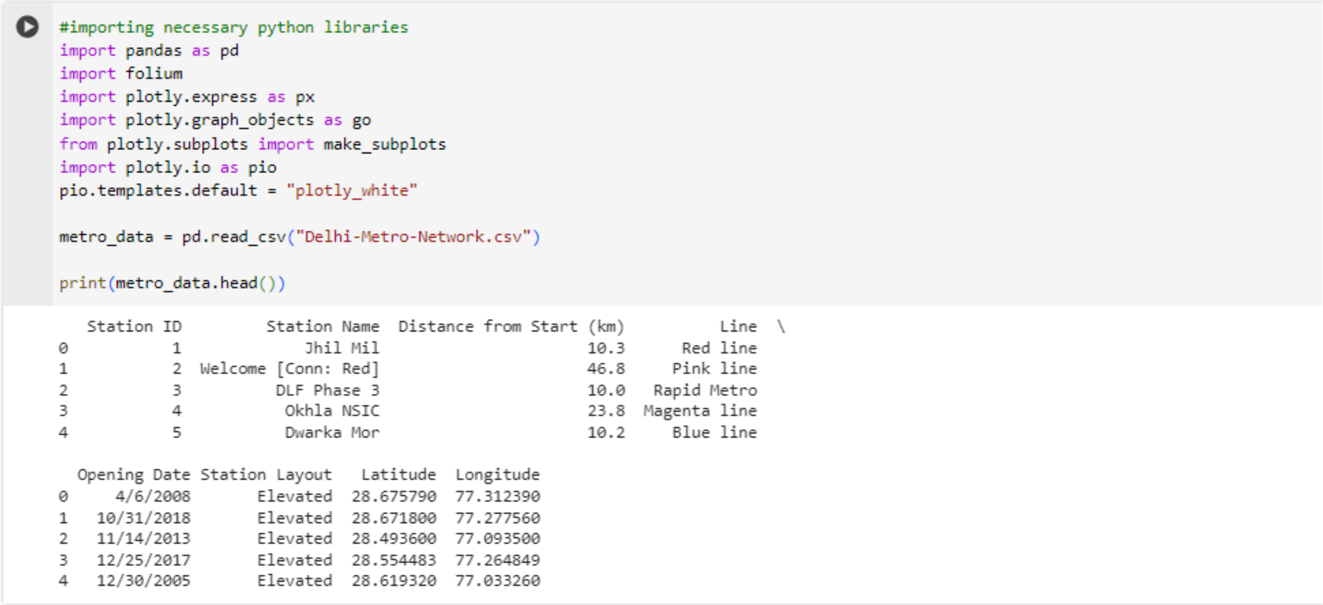
Delhi Metro Analysis: Process We Can Follow

Analyzing the metro network in a city like Delhi helps improve urban transportation infrastructure, leading to better city planning and enhanced commuter experiences. Below is the process we can follow for the task of Metro Network Analysis of Delhi:

- Determine what you want to achieve. It could be optimizing routes, reducing congestion, improving passenger flow, or understanding travel patterns.
- Collect data on metro lines, stations, connections, and transit schedules.
- Clean the data for inconsistencies, missing values, or errors.
- Create visual representations of the network, such as route maps, passenger flow charts, or heat maps of station congestion.
- Analyze how effectively the network handles passenger traffic and meets operational targets.

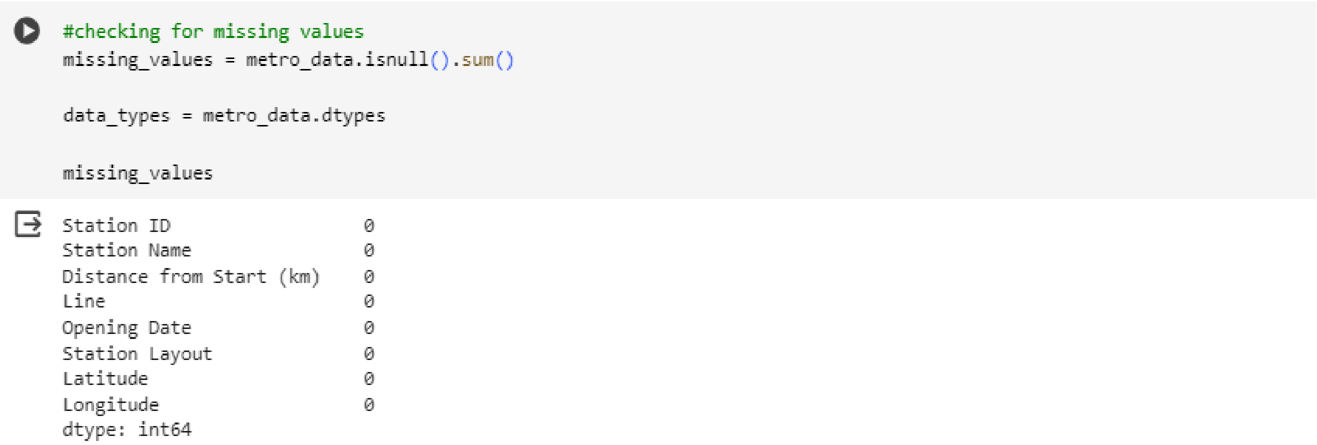
So, for the Delhi Metro Network Analysis task, we need to have a dataset based on all metro lines in Delhi and how they connect.



```python
#importing necessary python libraries
import pandas as pd
import folium
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import plotly.io as pio
pio.templates.default = "plotly_white"

metro_data = pd.read_csv("Delhi-Metro-Network.csv")

print(metro_data.head())
```

```
   Station ID         Station Name  Distance from Start (km)          Line  \
0           1             Jhil Mil                      10.3      Red line   
1           2  Welcome [Conn: Red]                      46.8     Pink line   
2           3          DLF Phase 3                      10.0   Rapid Metro   
3           4           Okhla NSIC                      23.8  Magenta line   
4           5           Dwarka Mor                      10.2     Blue line   

  Opening Date Station Layout   Latitude  Longitude  
0     4/6/2008       Elevated  28.675790  77.312390  
1   10/31/2018       Elevated  28.671800  77.277560  
2   11/14/2013       Elevated  28.493600  77.093500  
3   12/25/2017       Elevated  28.554483  77.264849  
4   12/30/2005       Elevated  28.619320  77.033260  
```

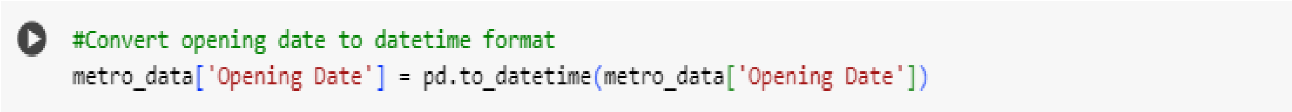
Now, let’s have a look at whether the dataset has any null values or not and then look at the data types:



```
#checking for missing values
missing_values = metro_data.isnull().sum()

data_types = metro_data.dtypes

missing_values
```

```
Station ID                 0
Station Name               0
Distance from Start (km)   0
Line                       0
Opening Date               0
Station Layout             0
Latitude                   0
Longitude                  0
dtype: int64
```

Now, I’ll convert the Opening Date column into a DateTime format for ease of analysis:

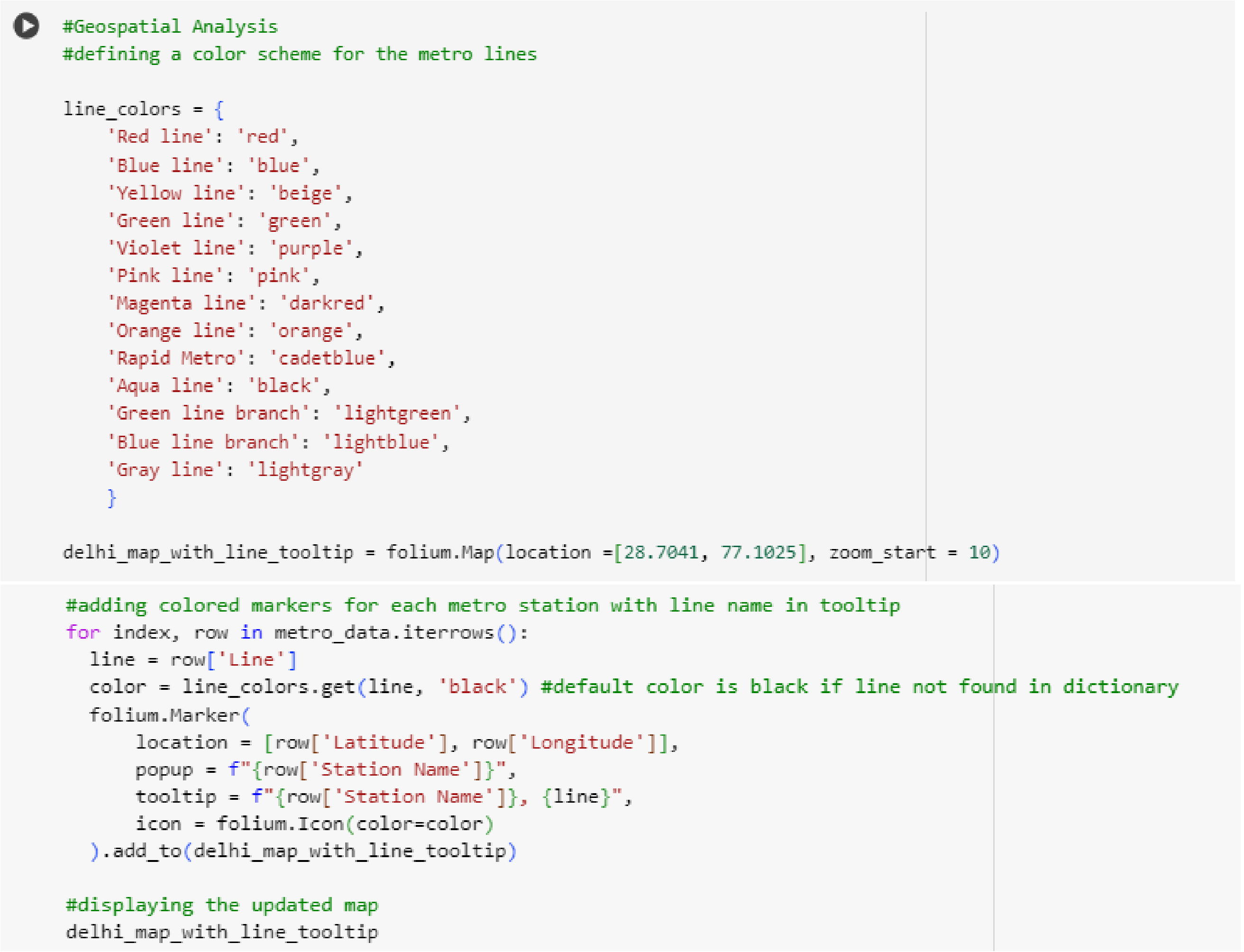


```
#Convert opening date to datetime format
metro_data['Opening Date'] = pd.to_datetime(metro_data['Opening Date'])
```

Geospatial Analysis

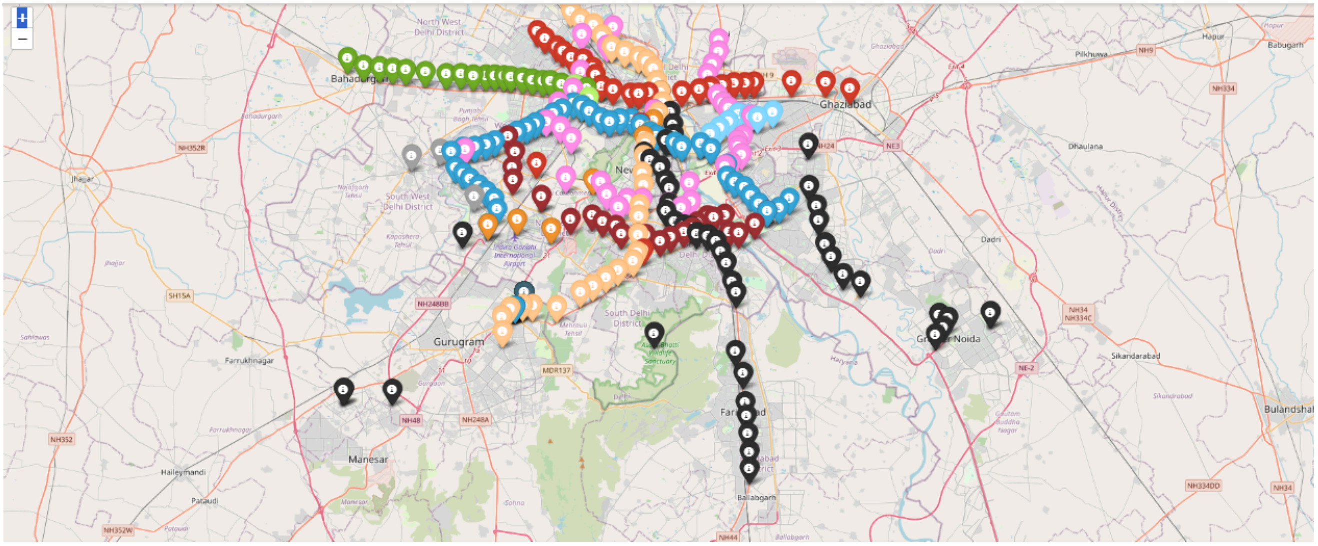
Now, I’ll start by visualizing the locations of the metro stations on a map. It will give us an insight into the geographical distribution of the stations across Delhi. We will use the latitude and longitude data to plot each station.

For this, I’ll create a map with markers for each metro station. Each marker will represent a station, and we can analyze aspects like station density and geographic spread. Let’s proceed with this visualization:



```
#Geospatial Analysis
#defining a color scheme for the metro lines

line_colors = {
    'Red line': 'red',
    'Blue line': 'blue',
    'Yellow line': 'beige',
    'Green line': 'green',
    'Violet line': 'purple',
    'Pink line': 'pink',
    'Magenta line': 'darkred',
    'Orange line': 'orange',
    'Rapid Metro': 'cadetblue',
    'Aqua line': 'black',
    'Green line branch': 'lightgreen',
    'Blue line branch': 'lightblue',
    'Gray line': 'lightgray'
    }

delhi_map_with_line_tooltip = folium.Map(location =[28.7041, 77.1025], zoom_start = 10)
```

```
#adding colored markers for each metro station with line name in tooltip
for index, row in metro_data.iterrows():
  line = row['Line']
  color = line_colors.get(line, 'black') #default color is black if line not found in dictionary
  folium.Marker(
      location = [row['Latitude'], row['Longitude']],
      popup = f"{row['Station Name']}",
      tooltip = f"{row['Station Name']}, {line}",
      icon = folium.Icon(color=color)
  ).add_to(delhi_map_with_line_tooltip)

#displaying the updated map
delhi_map_with_line_tooltip
```

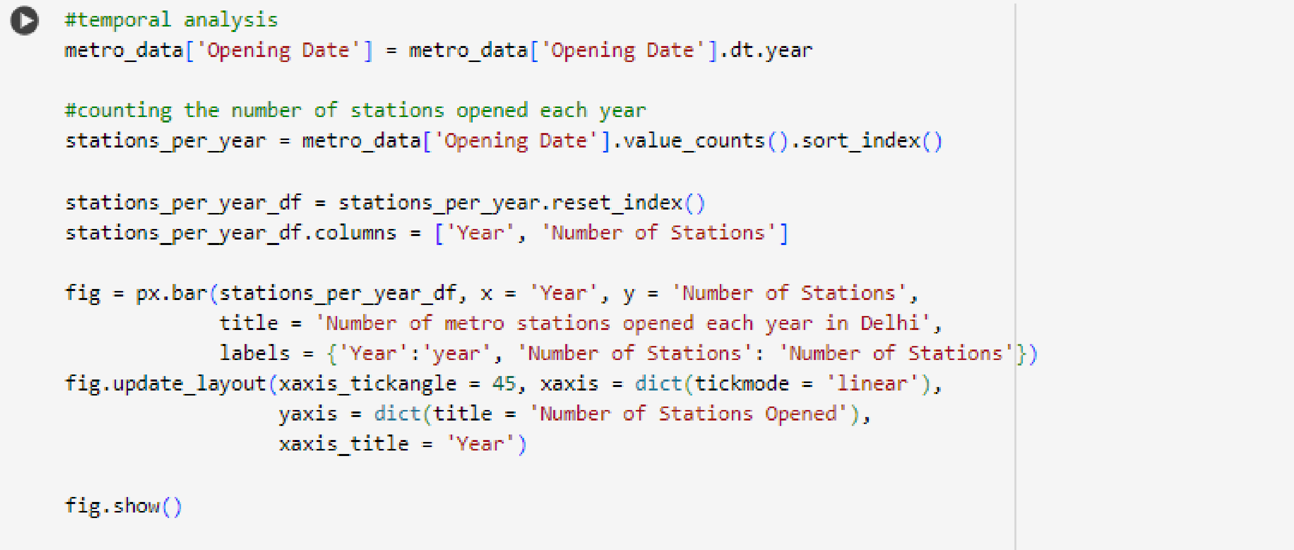


Here is the map showing the **geographical distribution** of Delhi Metro stations. Each marker represents a metro station; you can hover over or click on the markers to see the station name and the metro line it belongs to. This map provides a visual understanding of how the metro stations are spread across Delhi.

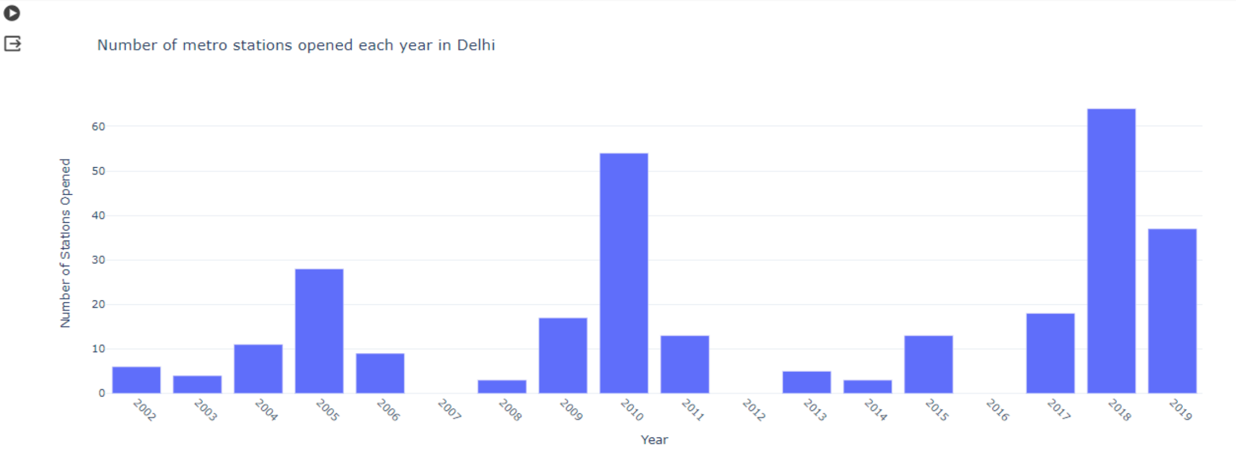
Temporal Analysis

Now, I will analyze the growth of the Delhi Metro network over time. I'll look at how many stations were opened each year and visualize this growth. It can provide insights into the pace of metro network expansion and its development phases.

I'll start by extracting the year from the Opening Date and then count the number of stations opened each year. Following this, I'll visualize this information in a bar plot. Let's proceed with this analysis:



```python
#temporal analysis
metro_data['Opening Date'] = metro_data['Opening Date'].dt.year

#counting the number of stations opened each year
stations_per_year = metro_data['Opening Date'].value_counts().sort_index()

stations_per_year_df = stations_per_year.reset_index()
stations_per_year_df.columns = ['Year', 'Number of Stations']

fig = px.bar(stations_per_year_df, x = 'Year', y = 'Number of Stations',
             title = 'Number of metro stations opened each year in Delhi',
             labels = {'Year':'year', 'Number of Stations': 'Number of Stations'})
fig.update_layout(xaxis_tickangle = 45, xaxis = dict(tickmode = 'linear'),
                  yaxis = dict(title = 'Number of Stations Opened'),
                  xaxis_title = 'Year')

fig.show()
```



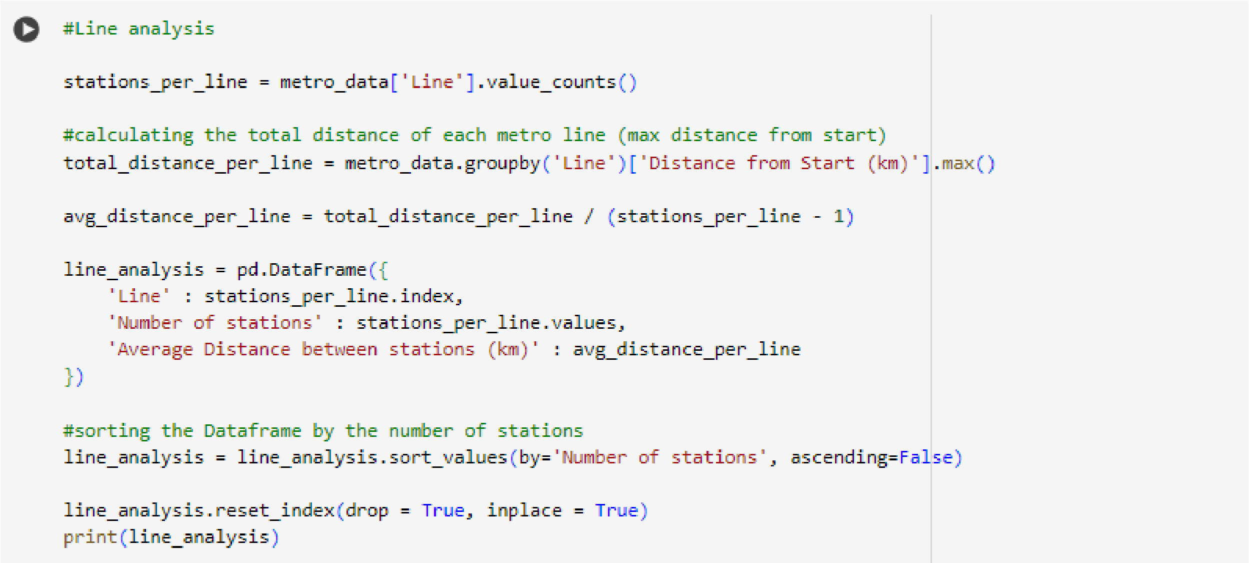
The bar chart illustrates the number of Delhi Metro stations opened each year. This visualization helps us understand the temporal development of the metro network. Some key observations include:

- Some years show a significant number of new station openings, indicating phases of rapid network expansion.
- Conversely, there are years with few or no new stations, which could be due to various factors like planning, funding, or construction challenges.

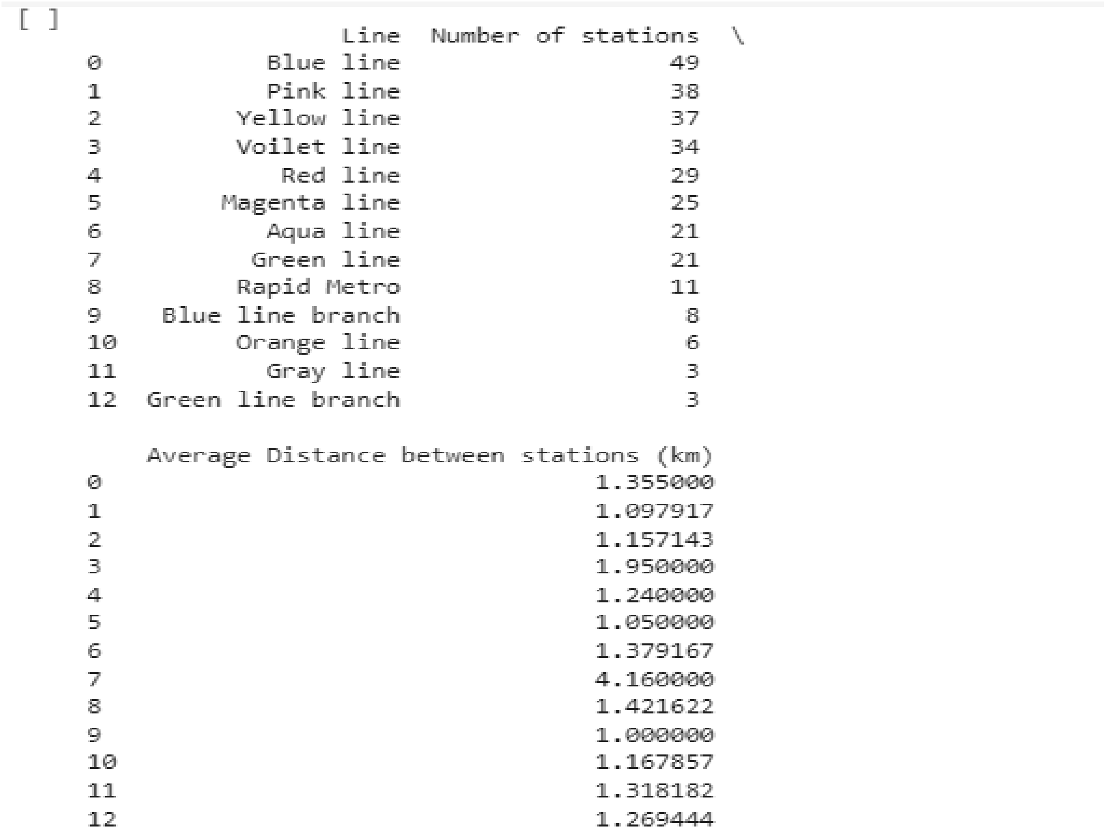
Line Analysis

Now, I'll analyze the various metro lines in terms of the number of stations they have and the average distance between stations. It will give us insights into the characteristics of each metro line, such as which lines are more extensive or denser.

I'll calculate the number of stations per line and the average distance between stations on each line. I'll then visualize these metrics to better understand the differences between the lines. Let's start with these calculations:



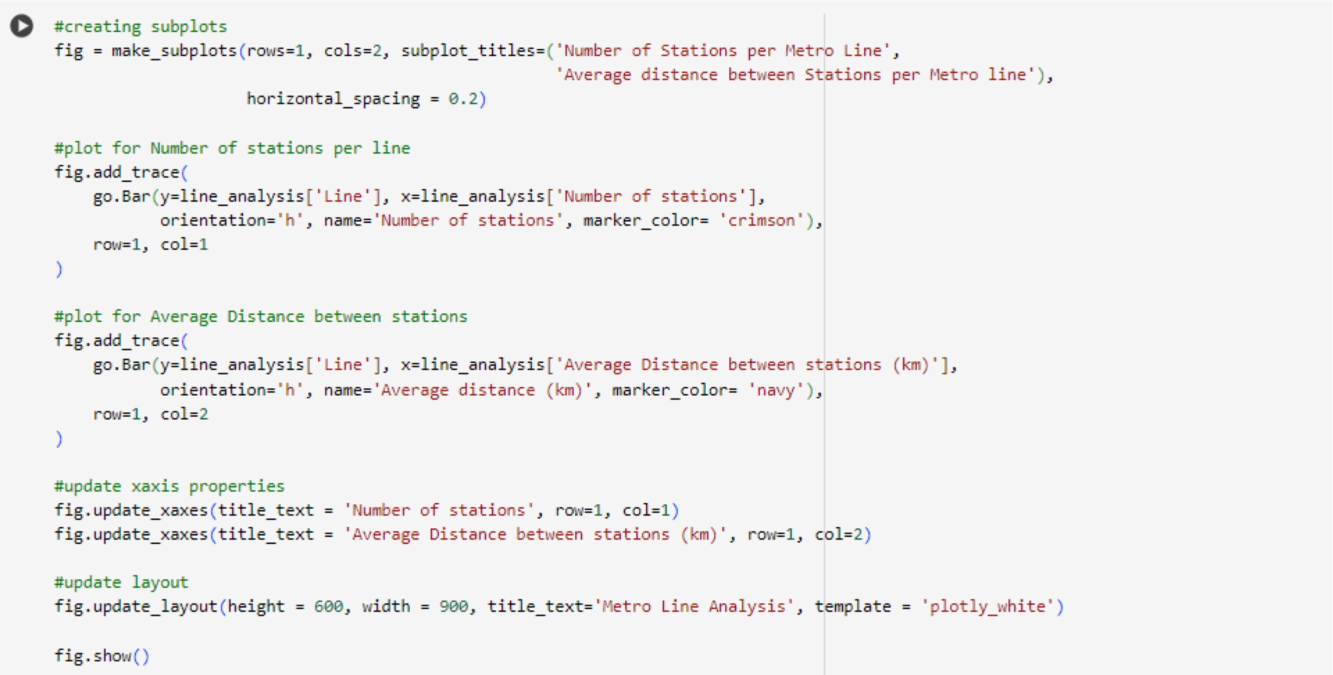
```
#Line analysis

stations_per_line = metro_data['Line'].value_counts()

#calculating the total distance of each metro line (max distance from start)
total_distance_per_line = metro_data.groupby('Line')['Distance from Start (km)'].max()

avg_distance_per_line = total_distance_per_line / (stations_per_line - 1)

line_analysis = pd.DataFrame({
    'Line' : stations_per_line.index,
    'Number of stations' : stations_per_line.values,
    'Average Distance between stations (km)' : avg_distance_per_line
})

#sorting the Dataframe by the number of stations
line_analysis = line_analysis.sort_values(by='Number of stations', ascending=False)

line_analysis.reset_index(drop = True, inplace = True)
print(line_analysis)
```



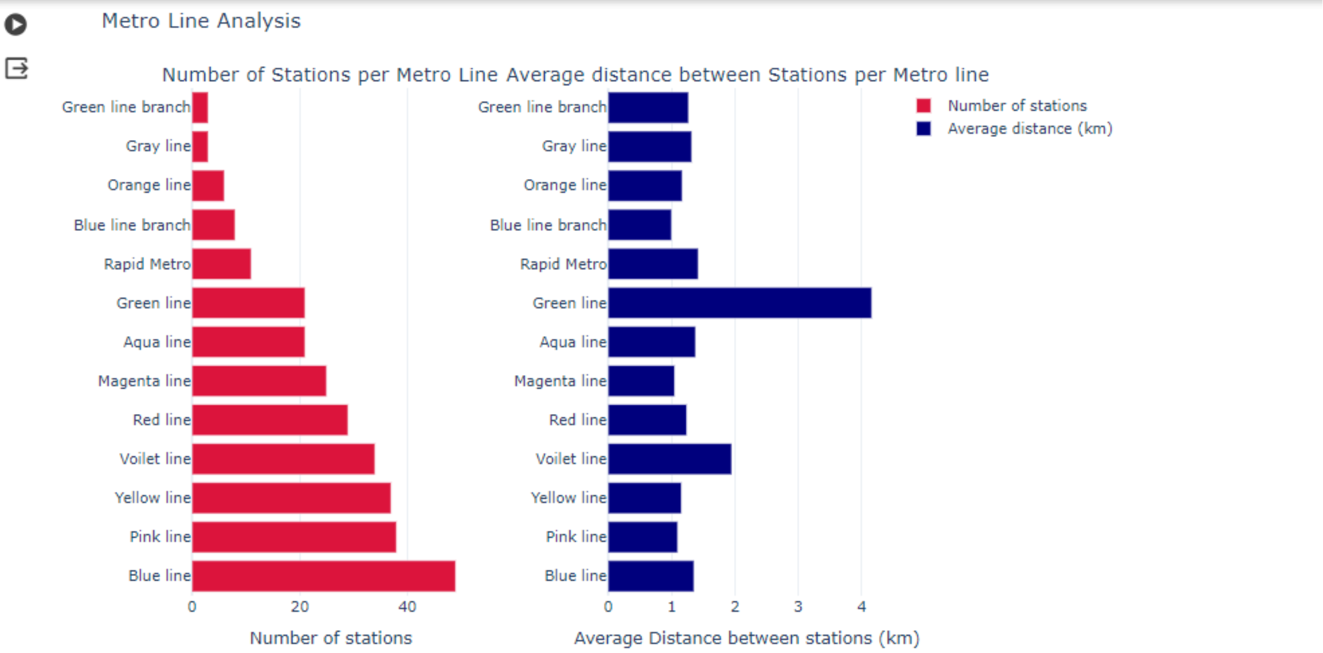
```
[ ]
                  Line  Number of stations  \
0            Blue line                  49
1            Pink line                  38
2          Yellow line                  37
3          Voilet line                  34
4             Red line                  29
5         Magenta line                  25
6            Aqua line                  21
7           Green line                  21
8          Rapid Metro                  11
9     Blue line branch                   8
10         Orange line                   6
11           Gray line                   3
12   Green line branch                   3

    Average Distance between stations (km)
0                                 1.355000
1                                 1.097917
2                                 1.157143
3                                 1.950000
4                                 1.240000
5                                 1.050000
6                                 1.379167
7                                 4.160000
8                                 1.421622
9                                 1.000000
10                                1.167857
11                                1.318182
12                                1.269444
```

The table presents a detailed analysis of the Delhi Metro lines, including the number of stations on each line and the average distance between stations.

To better understand these metrics, let’s visualize them. I’ll create two plots: one for the number of stations per line and another for the average distance between stations. It will provide a comparative view of the metro lines:



```
#creating subplots
fig = make_subplots(rows=1, cols=2, subplot_titles=('Number of Stations per Metro Line',
                                                    'Average distance between Stations per Metro line'),
                    horizontal_spacing = 0.2)

#plot for Number of stations per line
fig.add_trace(
    go.Bar(y=line_analysis['Line'], x=line_analysis['Number of stations'],
           orientation='h', name='Number of stations', marker_color= 'crimson'),
    row=1, col=1
)

#plot for Average Distance between stations
fig.add_trace(
    go.Bar(y=line_analysis['Line'], x=line_analysis['Average Distance between stations (km)'],
           orientation='h', name='Average distance (km)', marker_color= 'navy'),
    row=1, col=2
)

#update xaxis properties
fig.update_xaxes(title_text = 'Number of stations', row=1, col=1)
fig.update_xaxes(title_text = 'Average Distance between stations (km)', row=1, col=2)

#update layout
fig.update_layout(height = 600, width = 900, title_text='Metro Line Analysis', template = 'plotly_white')

fig.show()
```

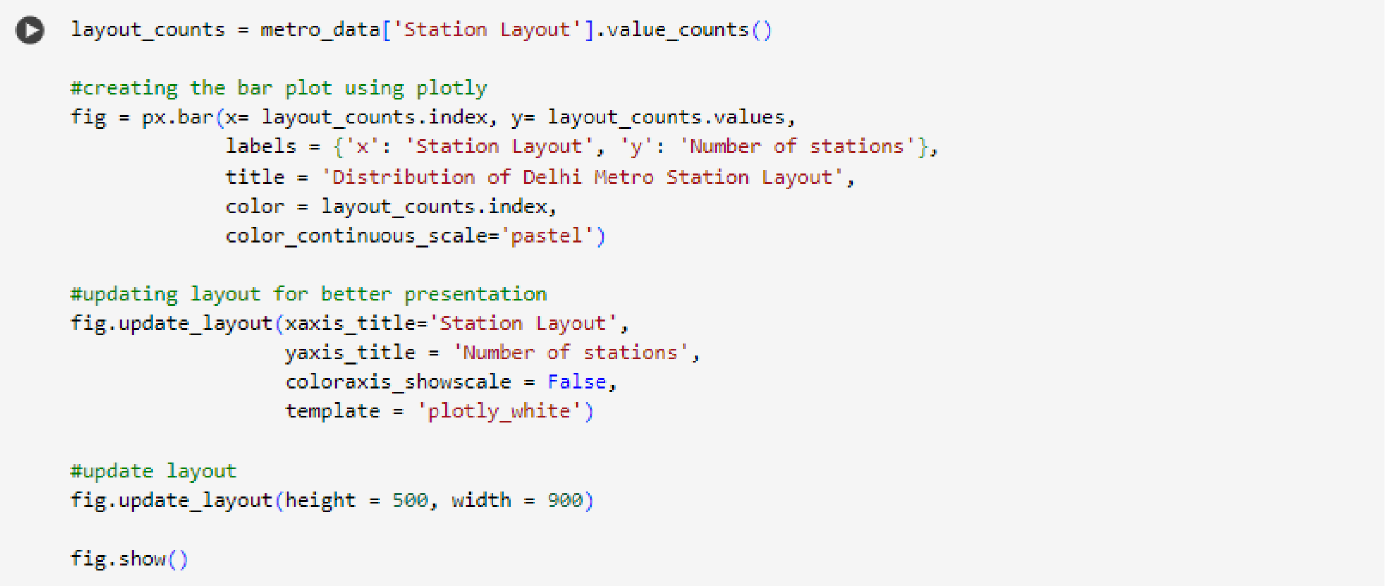


Metro Line Analysis

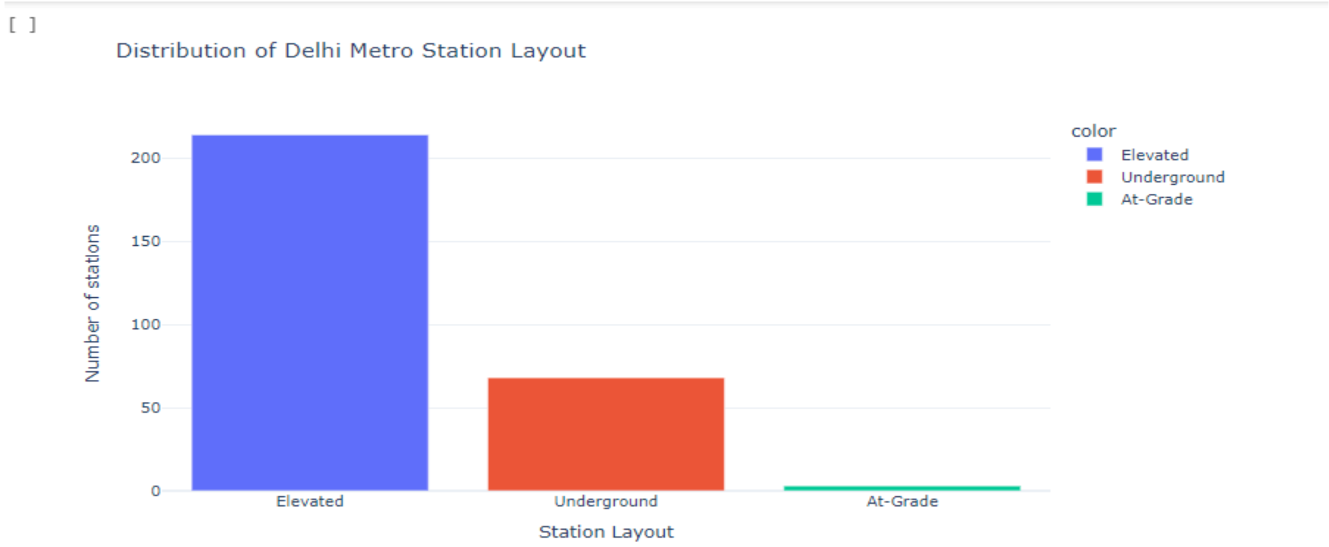
Station Layout Analysis

Next, I'll explore the station layouts (Elevated, Ground Level, Underground). I'll analyze the distribution of these layouts across the network and see if there are any patterns or trends, such as certain lines favoring a particular layout.

I'll calculate the frequency of each layout type and then visualize these frequencies to get a clearer picture of the layout distribution. Let's proceed with this:



```
layout_counts = metro_data['Station Layout'].value_counts()

#creating the bar plot using plotly
fig = px.bar(x= layout_counts.index, y= layout_counts.values,
             labels = {'x': 'Station Layout', 'y': 'Number of stations'},
             title = 'Distribution of Delhi Metro Station Layout',
             color = layout_counts.index,
             color_continuous_scale='pastel')

#updating layout for better presentation
fig.update_layout(xaxis_title='Station Layout',
                  yaxis_title = 'Number of stations',
                  coloraxis_showscale = False,
                  template = 'plotly_white')

#update layout
fig.update_layout(height = 500, width = 900)

fig.show()
```



The bar chart and the counts show the distribution of different station layouts in the Delhi Metro network.

Observations:

Elevated Stations: The majority of the stations are Elevated. It is a common design choice in urban areas to save space and reduce land acquisition issues.
Underground Stations: The Underground stations are fewer compared to elevated ones. These are likely in densely populated or central areas where above-ground construction is less feasible.
At-Grade Stations: There are only a few At-Grade (ground level) stations, suggesting they are less common in the network, possibly due to land and traffic considerations.

Summary

So, this is how you can perform Delhi Metro Network Analysis using Python. Metro Network Analysis involves examining the network of metro systems to understand their structure, efficiency, and effectiveness. It typically includes analyzing routes, stations, traffic, connectivity, and other operational aspects.