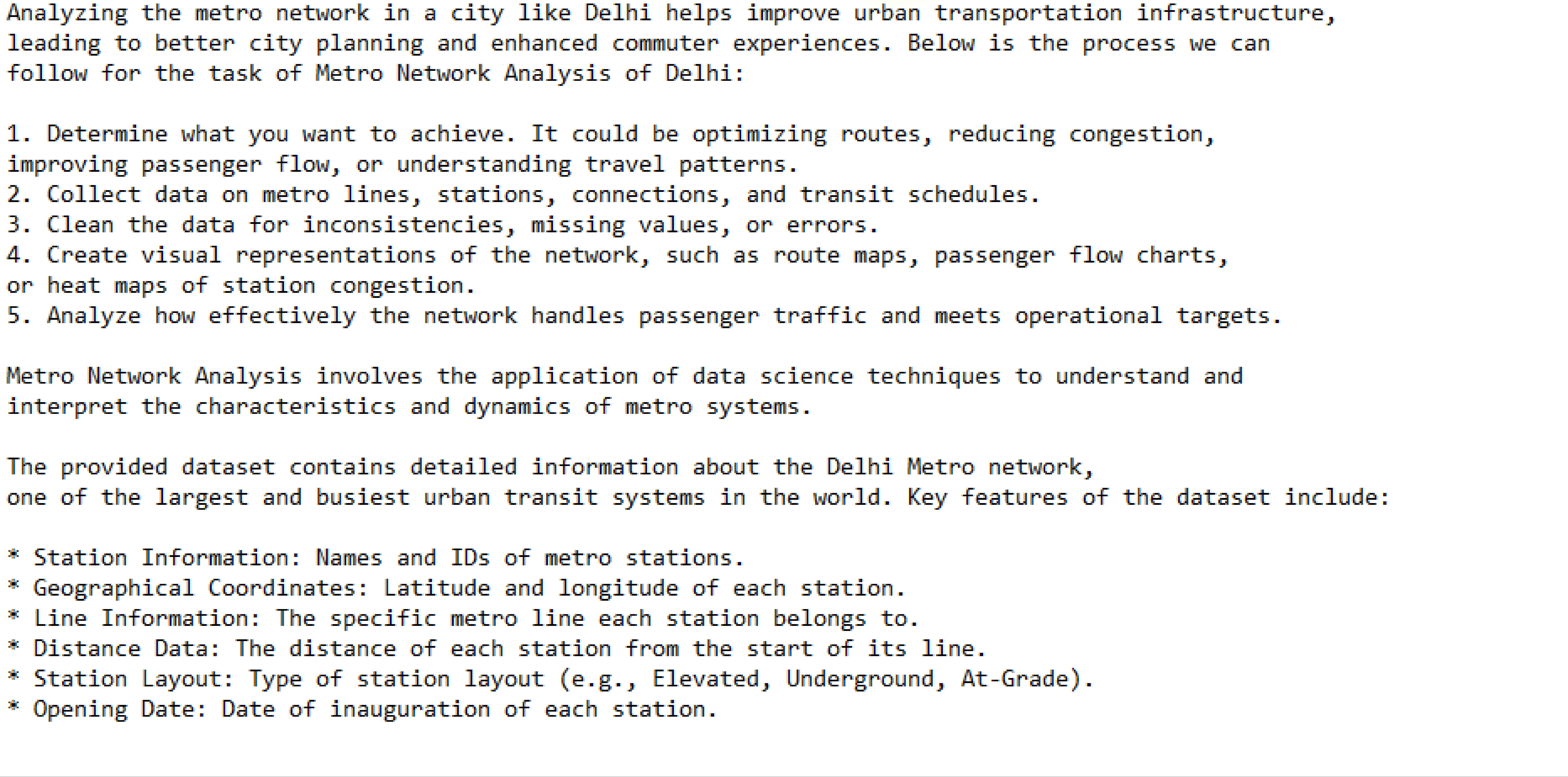
# DELHI

## METRO ANALYSIS PROJECT

UJJAWAL VALIYA

# About Project



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:

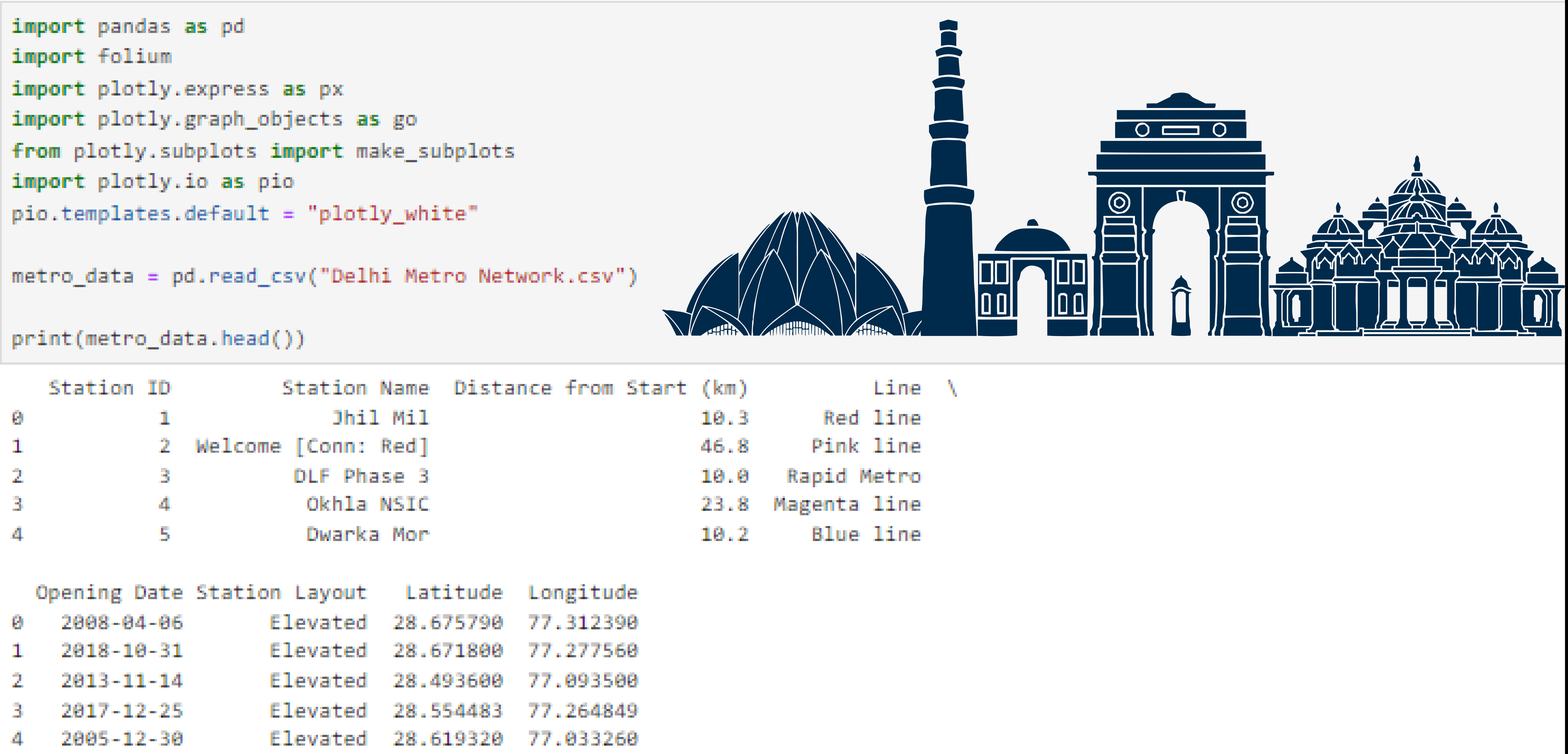
1. Determine what you want to achieve. It could be optimizing routes, reducing congestion, improving passenger flow, or understanding travel patterns.
2. Collect data on metro lines, stations, connections, and transit schedules.
3. Clean the data for inconsistencies, missing values, or errors.
4. Create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

Metro Network Analysis involves the application of data science techniques to understand and interpret the characteristics and dynamics of metro systems.

The provided dataset contains detailed information about the Delhi Metro network, one of the largest and busiest urban transit systems in the world. Key features of the dataset include:

* Station Information: Names and IDs of metro stations.
* Geographical Coordinates: Latitude and longitude of each station.
* Line Information: The specific metro line each station belongs to.
* Distance Data: The distance of each station from the start of its line.
* Station Layout: Type of station layout (e.g., Elevated, Underground, At-Grade).
* Opening Date: Date of inauguration of each station.

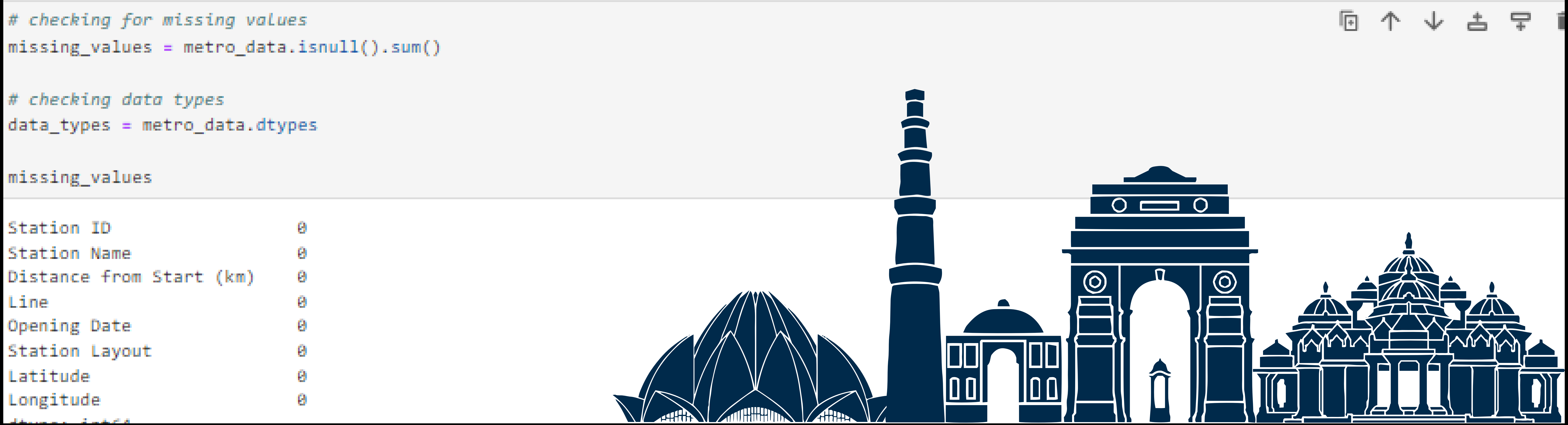
# Delhi Metro Analysis Project



```python
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())
```

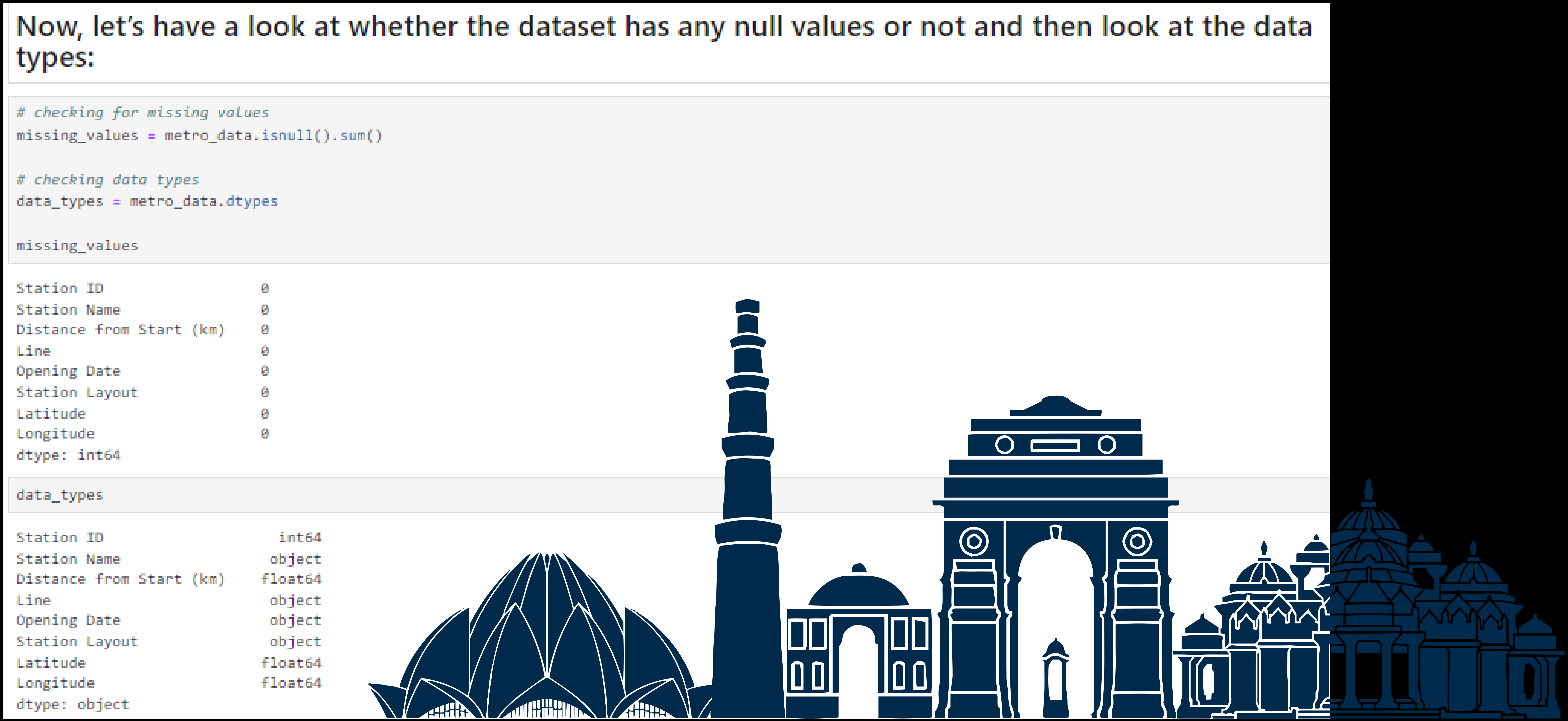
```
   Station ID          Station Name  Distance from Start (km)          Line  \
0           1               Jhil Mil                      10.3      Red line   
1           2  Welcome [Conn: Red]                      46.8     Pink line   
2           3            DLF Phase 3                      10.0   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   2008-04-06       Elevated  28.675790  77.312390  
1   2018-10-31       Elevated  28.671800  77.277560  
2   2013-11-14       Elevated  28.493600  77.093500  
3   2017-12-25       Elevated  28.554483  77.264849  
4   2005-12-30       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()

# checking data types
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
```



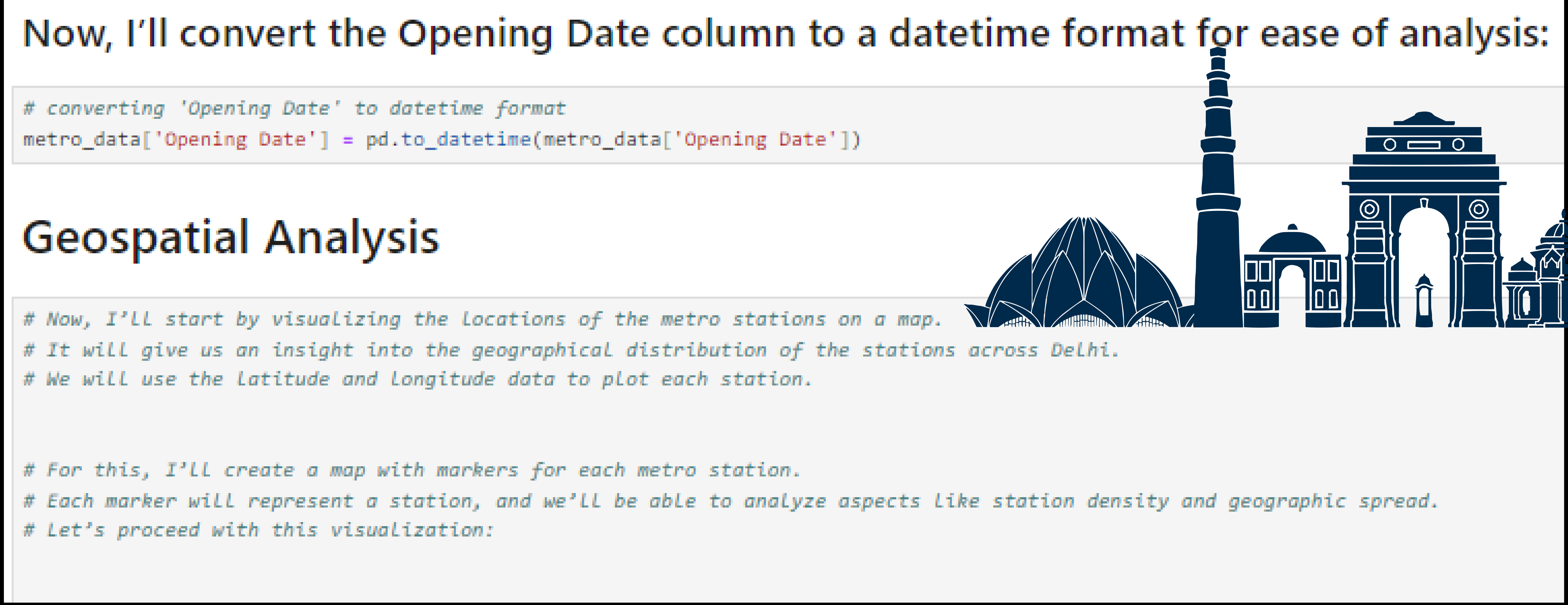
Now, let's have a look at whether the dataset has any null values or not and then look at the data types:

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

# checking data types
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
```

```
data_types
```

```
Station ID                   int64
Station Name                object
Distance from Start (km)   float64
Line                        object
Opening Date                object
Station Layout              object
Latitude                   float64
Longitude                  float64
dtype: object
```

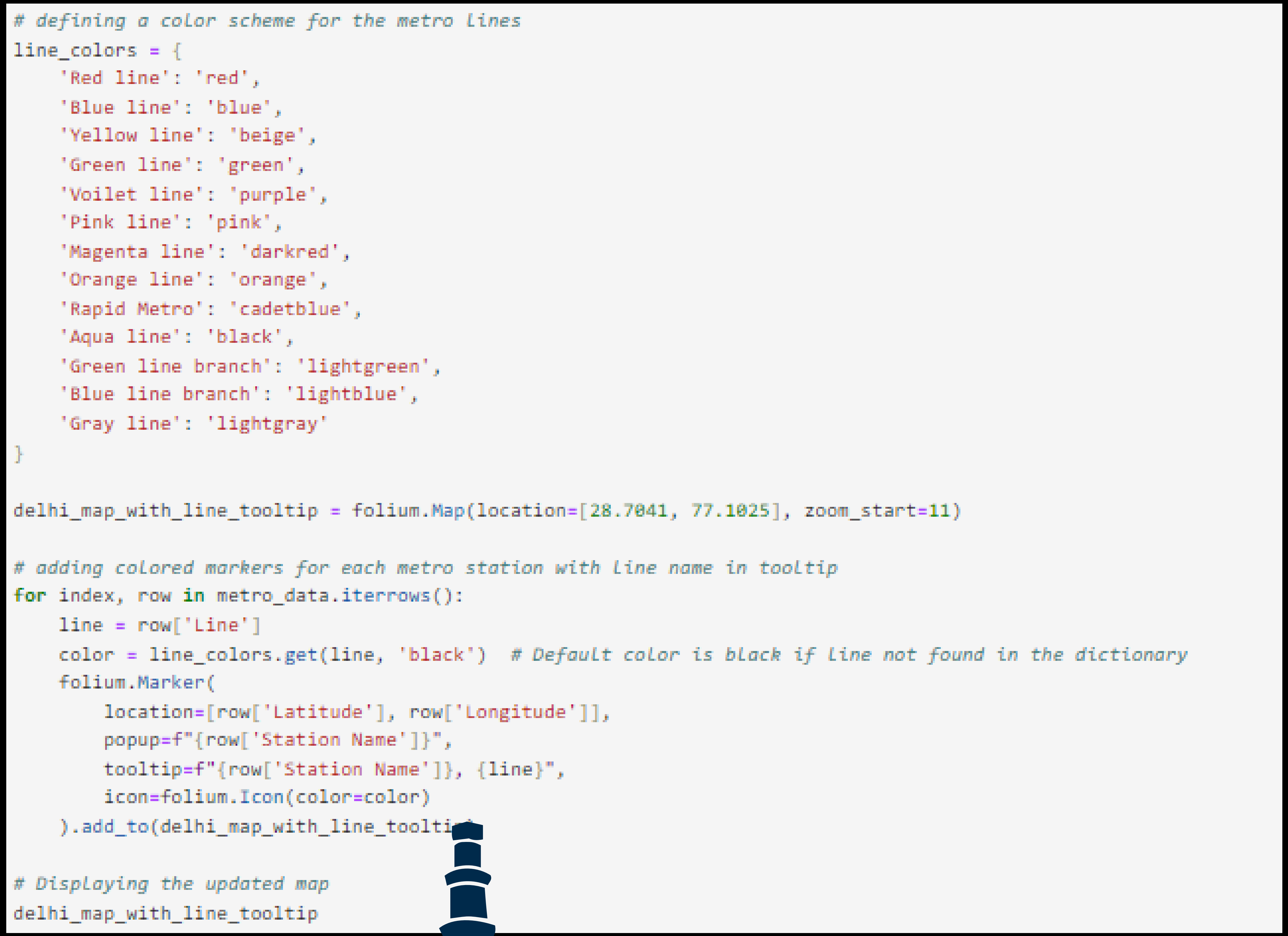


Now, I'll convert the Opening Date column to a datetime format for ease of analysis:

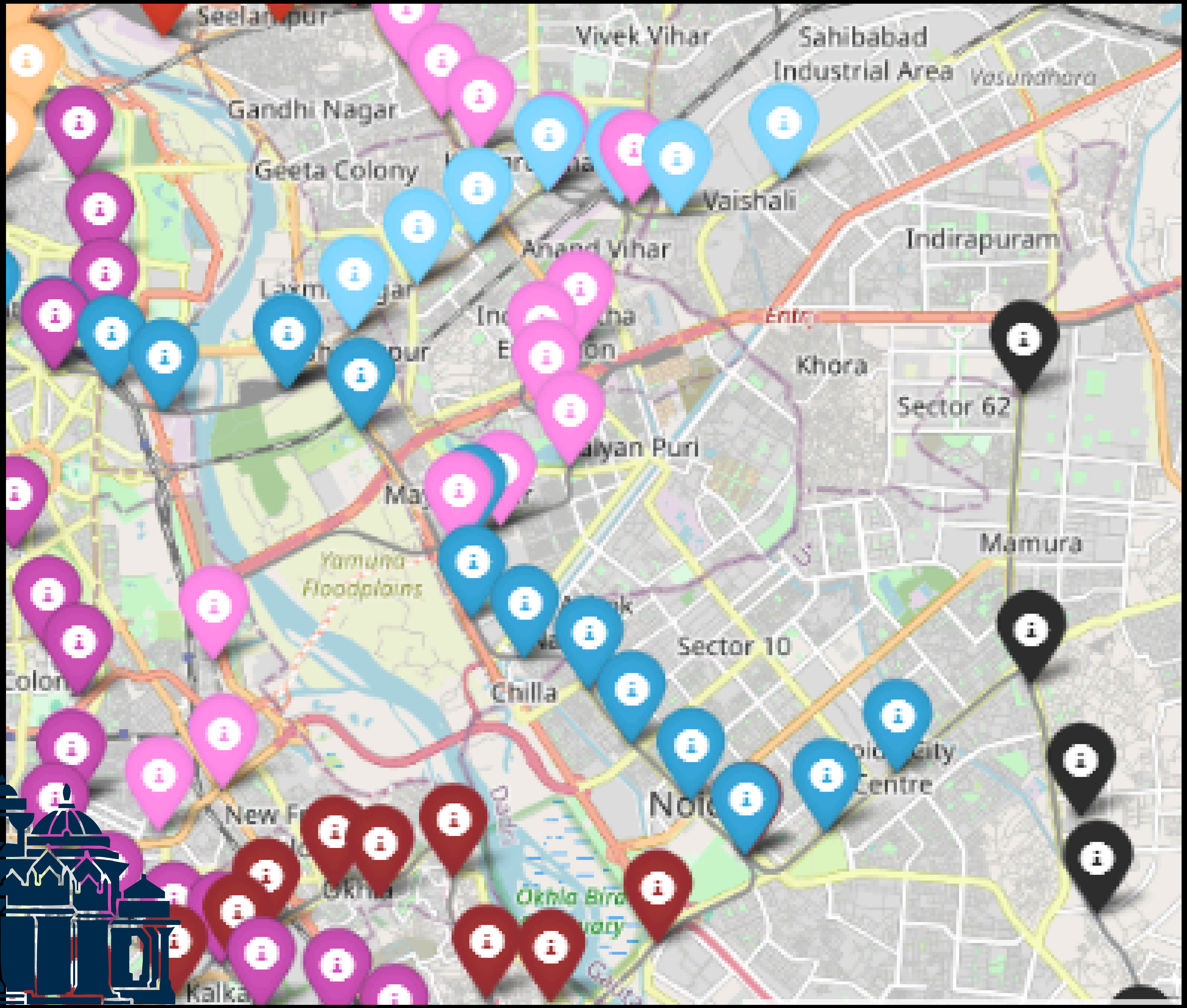
```
# converting 'Opening Date' to datetime format
metro_data['Opening Date'] = pd.to_datetime(metro_data['Opening Date'])
```

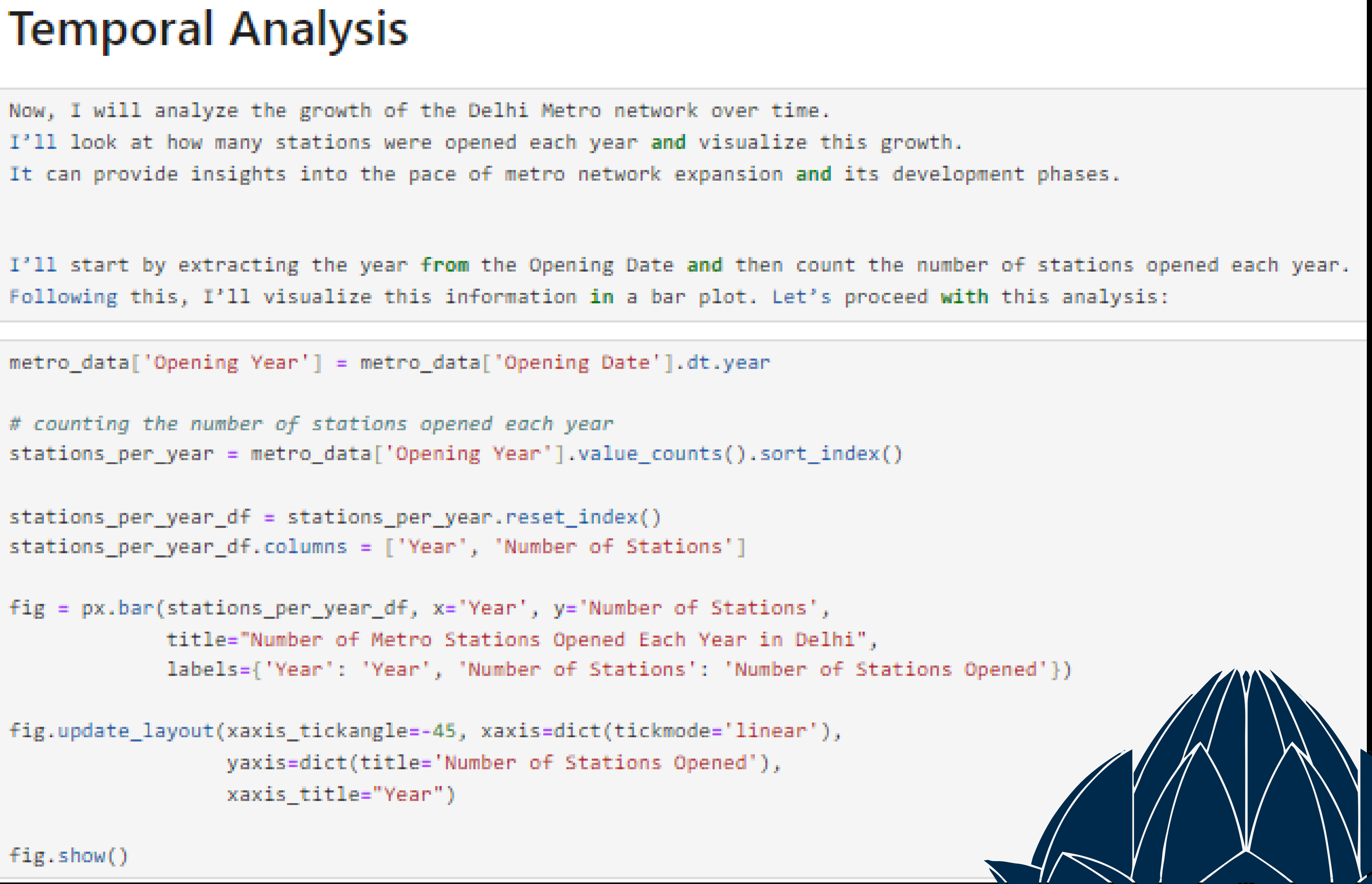
## Geospatial Analysis

```
# 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'll be able to analyze aspects like station density and geographic spread.
# Let's proceed with this visualization:
```



```python
# defining a color scheme for the metro lines
line_colors = {
    'Red line': 'red',
    'Blue line': 'blue',
    'Yellow line': 'beige',
    'Green line': 'green',
    'Voilet 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=11)

# 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 the 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
```

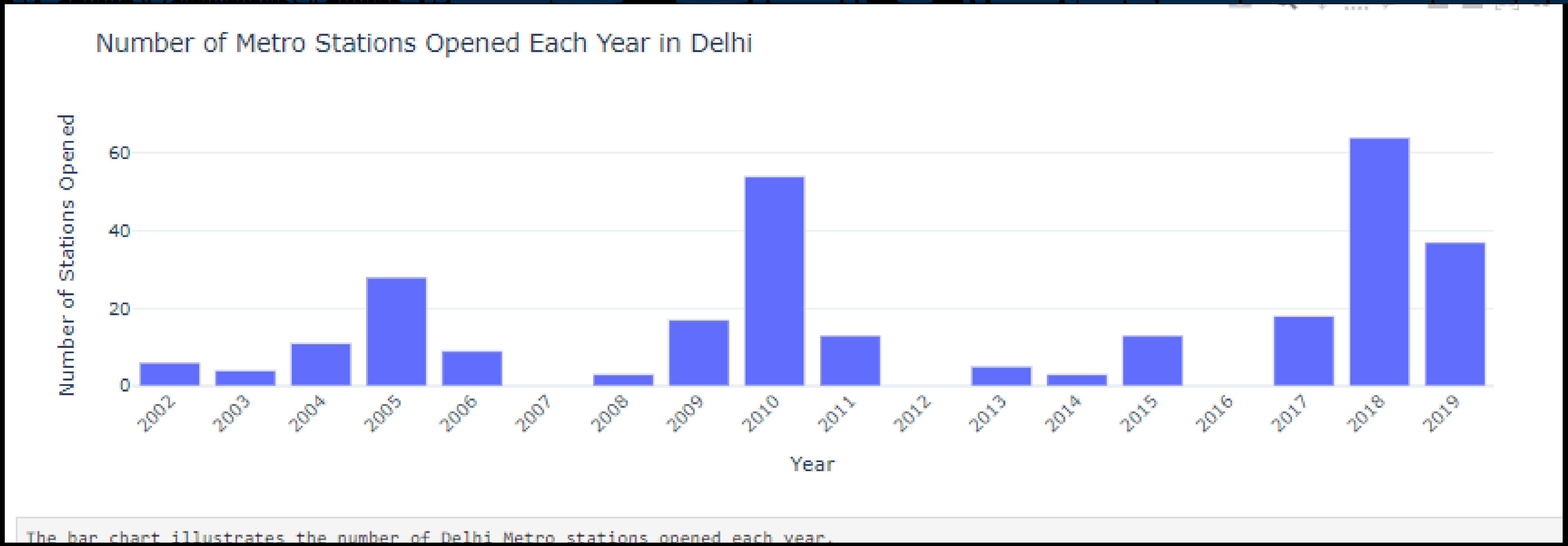




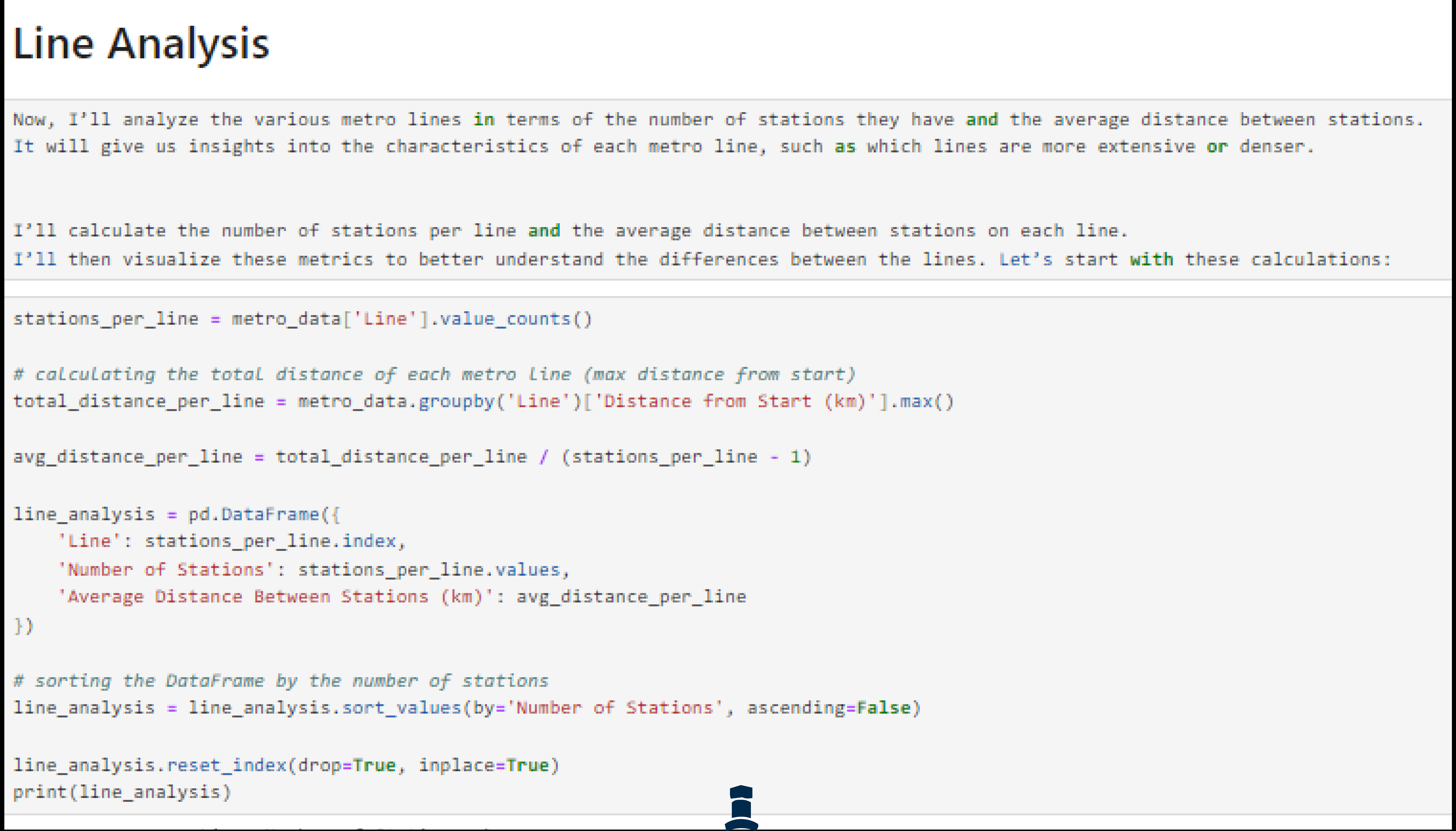
# Temporal Analysis

```
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
metro_data['Opening Year'] = metro_data['Opening Date'].dt.year

# counting the number of stations opened each year
stations_per_year = metro_data['Opening Year'].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 Opened'})

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.

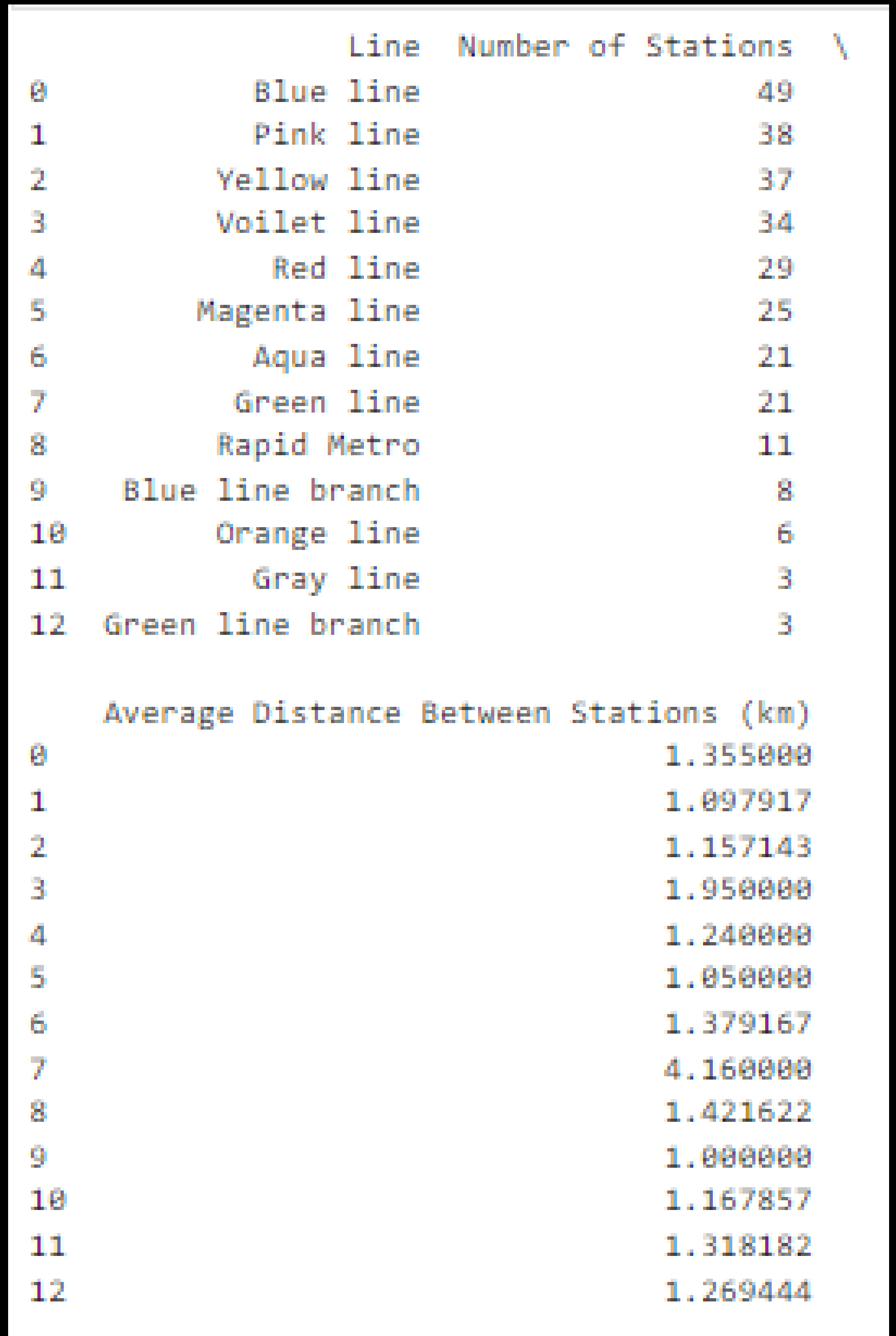


# Line Analysis

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.

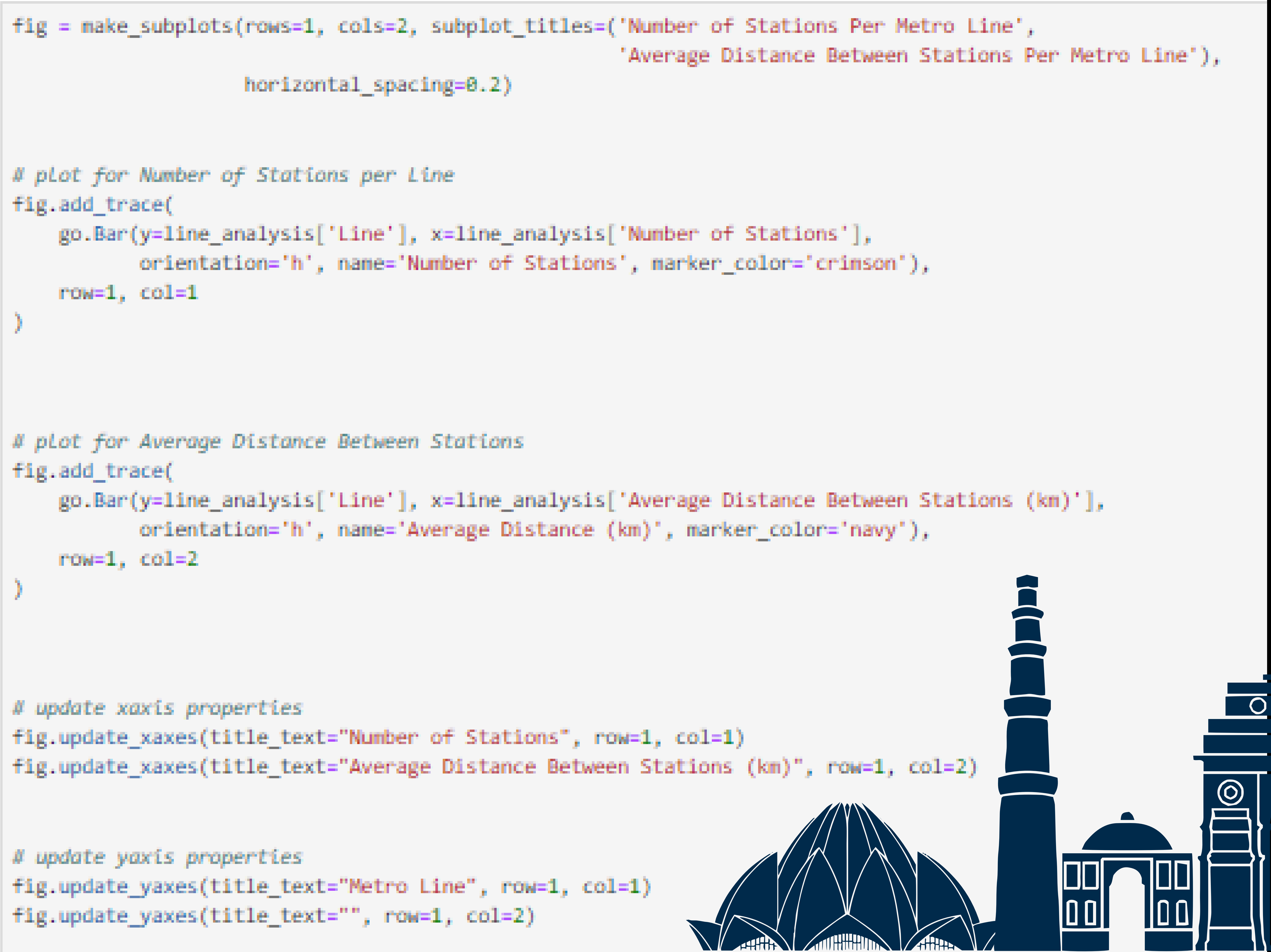
I’ll calculate the number of stations per line and the average distance between stations on each line. I’ll then visualize these metrics to better understand the differences between the lines. Let’s start with these calculations:

```
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)
```

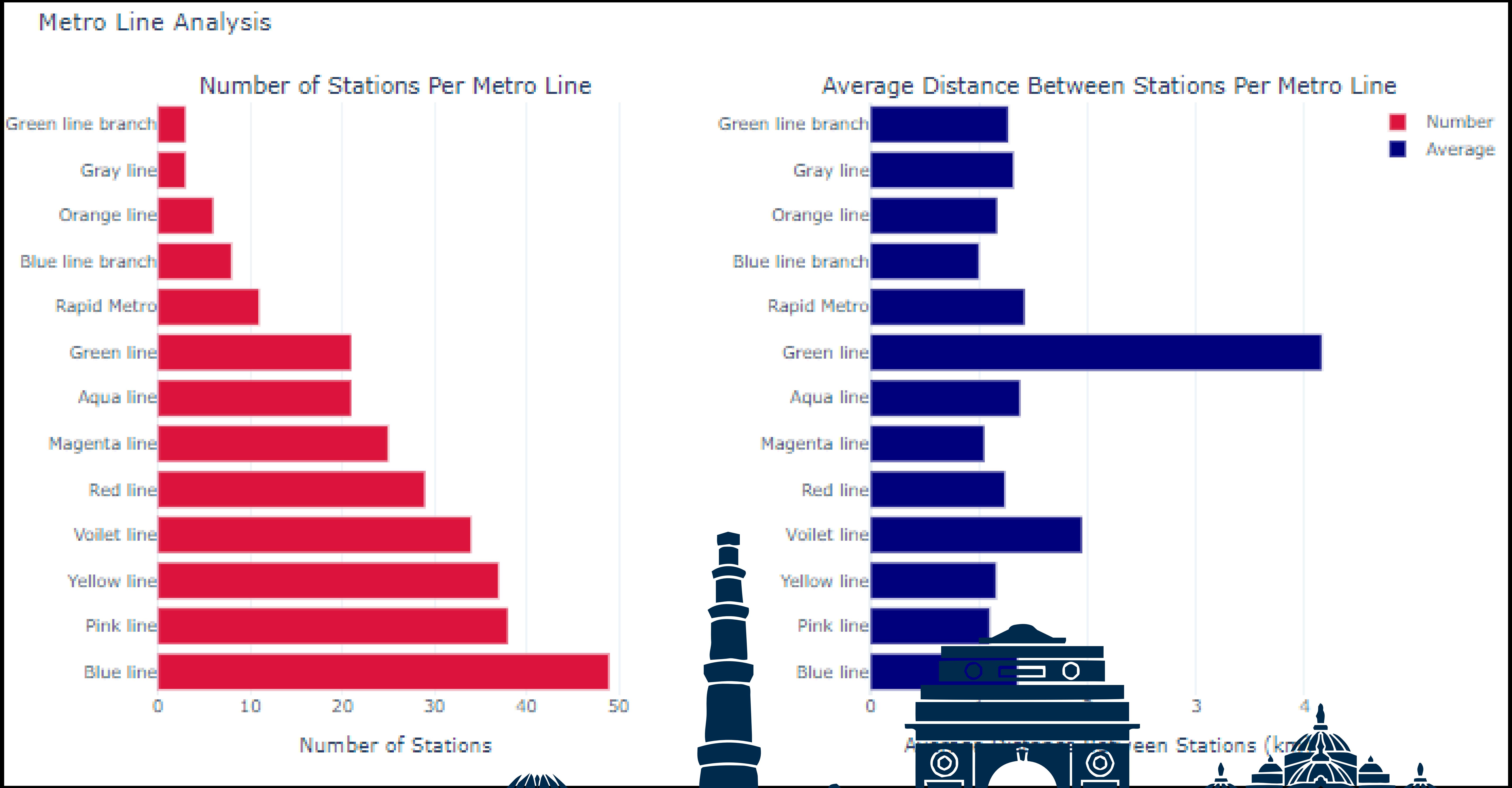


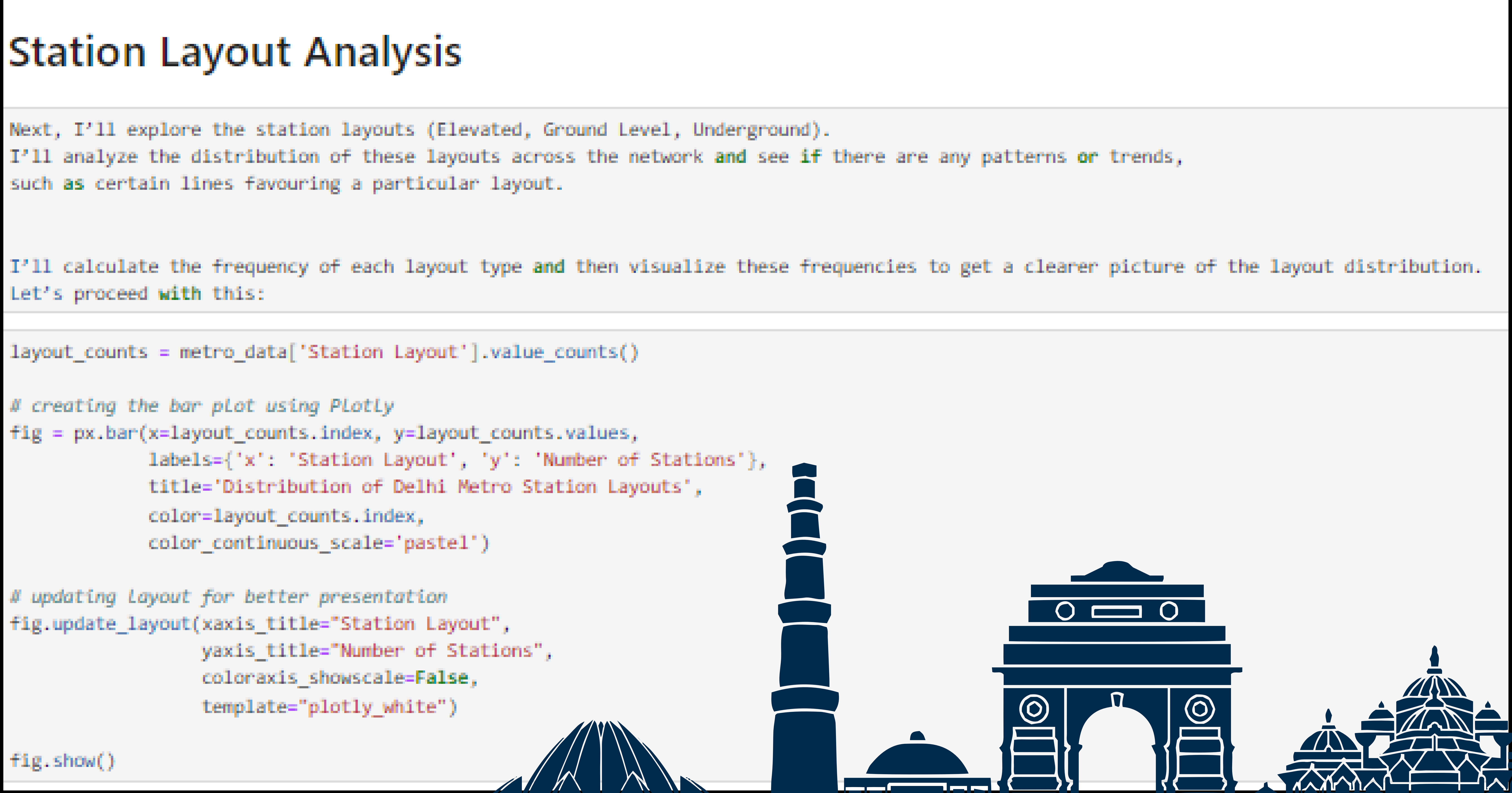
```
                 Line  Number of Stations  \
0           Blue line                  49
1           Pink line                  38
2         Yellow line                  37
3         Voilet line                  34
4            Red line                  29
5        Magenta line                  25
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
```

# 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 yaxis properties
fig.update_yaxes(title_text="Metro Line", row=1, col=1)
fig.update_yaxes(title_text="", row=1, col=2)
```

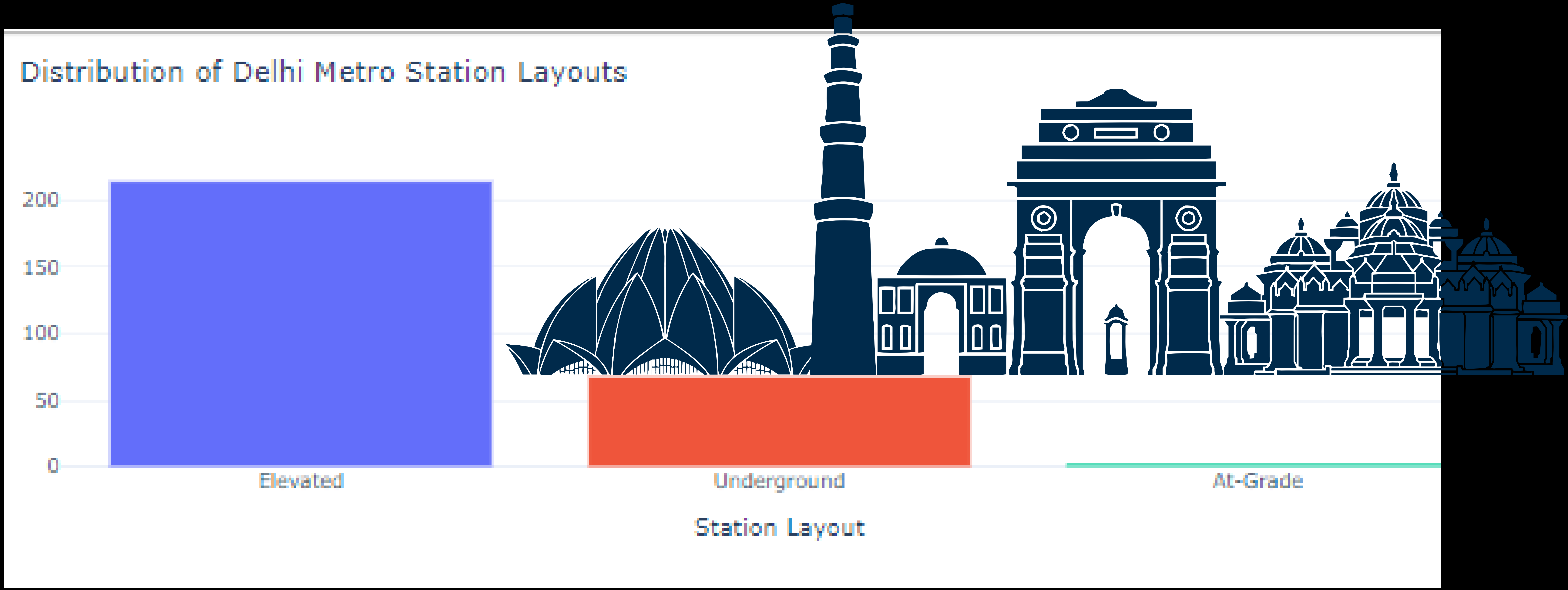

Metro Line Analysis
Number of Stations Per Metro Line
Green line branch
Gray line
Orange line
Blue line branch
Rapid Metro
Green line
Aqua line
Magenta line
Red line
Voilet line
Yellow line
Pink line
Blue line
0
10
20
30
40
50
Number of Stations
Average Distance Between Stations Per Metro Line
Green line branch
Gray line
Orange line
Blue line branch
Rapid Metro
Green line
Aqua line
Magenta line
Red line
Voilet line
Yellow line
Pink line
Blue line
0
3
4
een Stations (km
🟥 Number
🟦 Average



# 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 favouring 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:
```

```python
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 Layouts',
             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")

fig.show()
```


Distribution of Delhi Metro Station Layouts
200
150
100
50
0
Elevated
Underground
At-Grade
Station Layout

# THANK YOU