Document Info

**Author:** Ankur  
**Version:** 1.0  
**Project Status:** Completed

Delhi Metro Network Analysis - Project Documentation

Project Title

Delhi Metro Network Analysis with Interactive Visualizations

Project Scope

This project analyzes the Delhi Metro Network using a dataset (`Delhi-Metro-Network.csv`) containing station details such as names, lines, coordinates, opening dates, and layouts. It generates an interactive Folium map to visualize station locations, color-coded by metro line, and uses Plotly to create charts for temporal trends, line statistics, and station layout distribution. The analysis is implemented in a Jupyter Notebook (`metro\_analysis.ipynb`) on a Windows 11 environment, leveraging Python libraries (`pandas`, `folium`, `plotly`). The project aims to provide insights into the metro network’s growth, structure, and spatial distribution.

Learning Outcomes

**• Data Processing:** Master data cleaning and preprocessing with `pandas`.  
**• Visualization:** Create interactive maps with `folium` and charts with `plotly`.  
**• Data Analysis:** Analyze temporal and spatial patterns in metro network data.  
**• Troubleshooting:** Resolve data format and visualization issues.  
**• Project Management:** Plan, execute, and document a data analysis project.

Index

1. Document Metadata

2. Project Title

3. Project Scope

4. Planning

- Objectives

- Resources Required

- Key Stakeholders

5. Execution

- Startup Steps

- Progress Tracking

- Timeline

- Deviations from the Plan

6. Code Explanations

7. Troubleshooting Steps

8. Desired Outputs

9. Dependencies and Installations

10. Appendices

- Appendix 1: Dependency Setup

- Appendix 2: Cleanup Steps

- Appendix 3: Performance Enhancements

Planning

Objectives

• Load and preprocess `Delhi-Metro-Network.csv` for analysis.

• Create an interactive Folium map of metro stations with line-specific colors.

• Visualize station openings per year using a Plotly bar chart.

• Analyze metro lines (station counts and average distances) with Plotly subplots.

• Visualize station layout distribution with a Plotly bar chart.

• Document setup, execution, and troubleshooting steps.

Resources Required

**Hardware:** Windows 11 PC, 8GB RAM, 100GB SSD.  
**Software:** • Python 3.9.13 (`metro\_venv`).  
 • Jupyter Notebook.  
**Libraries:** • `pandas`  
 • `folium`  
 • `plotly`  
 • `jupyter`  
**Scripts:** • `metro\_analysis.ipynb`: Main analysis notebook.  
**Files:** • `D:\Metro\_Analysis\Delhi-Metro-Network.csv`: Dataset.  
**Storage:** `D:\Metro\_Analysis\`.

Key Stakeholders

**• Developer:** Writes code, tests visualizations, documents findings.  
**• Mentor:** Guides coding and debugging.  
**• End User:** Analyst or urban planner using visualizations for insights.

Execution

Startup Steps

┌──(metro\_venv)──[D:\]  
│ 1. \*\*Activate Virtual Environment\*\*:  
│ D:\metro\_venv\Scripts\activate  
└─$

┌──(metro\_venv)──[D:\]  
│ 2. \*\*Verify Dataset\*\*:  
│ Ensure `Delhi-Metro-Network.csv` exists in `D:\Metro\_Analysis\`.  
│ dir D:\Metro\_Analysis\Delhi-Metro-Network.csv  
└─$

┌──(metro\_venv)──[D:\]  
│ 3. \*\*Start Jupyter\*\*:  
│ jupyter notebook  
│ Open `metro\_analysis.ipynb` in `D:\Metro\_Analysis\`.  
└─$

┌──(metro\_venv)──[D:\]  
│ 4. \*\*Run Analysis\*\*:  
│ Execute all cells in `metro\_analysis.ipynb` to load data, generate maps, and create plots.  
└─$

Progress Tracking

**Daily Logs:** Record cells run, errors (e.g., `ValueError: Invalid format string`).  
**Metrics:** • Dataset rows: ~300 stations.  
 • Map: All stations plotted with correct colors.  
 • Plots: Bar charts and subplots rendered.  
**Reporting:** • Jupyter console: Dataframe previews, null checks.  
 • Visuals: Folium map, Plotly charts.  
**Checkpoints:** • Day 1: Dataset loaded, map created.  
 • Day 2: Plotly visualizations completed.  
 • Day 3: Documentation finalized.

Timeline

\*\*Day 1 (Setup and Map)\*\*:  
 • Install dependencies, verify dataset.  
 • Develop Folium map with color-coded markers.

\*\*Day 2 (Visualizations)\*\*:  
 • Create Plotly charts for station openings, line analysis, and layouts.

\*\*Day 3 (Documentation)\*\*:  
 • Write `README.md`, `LICENSE`, and `project\_documentation.md`.  
 • Test notebook for reproducibility.

Deviations from the Plan

\*\*Dataset Format Issue\*\*:  
 • \*\*Issue\*\*: Invalid date formats in `Opening Date`.  
 • \*\*Fix\*\*: Used `pd.to\_datetime` with error handling.  
 • \*\*Reason\*\*: Inconsistent date formats in CSV.

\*\*Visualization Errors\*\*:  
 • \*\*Issue\*\*: Plotly bars not rendering due to incorrect column names.  
 • \*\*Fix\*\*: Verified column names with `df.columns`.  
 • \*\*Reason\*\*: Typo in code.

Code Explanations

`metro\_analysis.ipynb` - Data Loading and Preprocessing

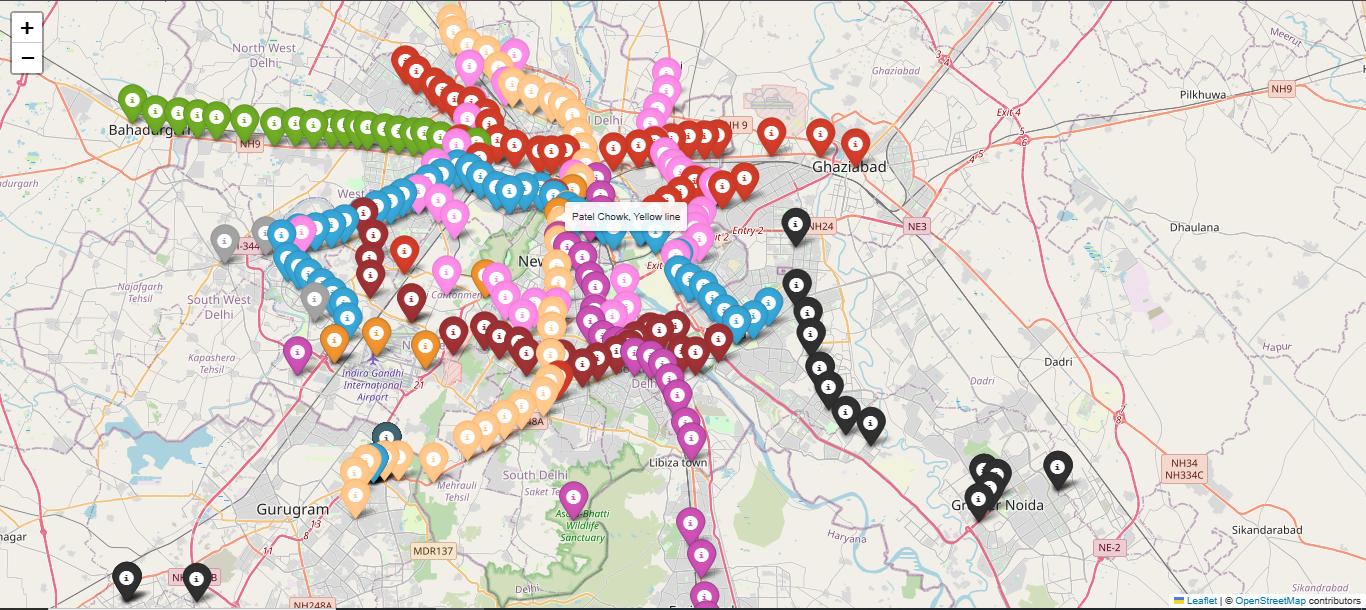
┌──(metro\_venv)──[D:\Metro\_Analysis]  
│ import pandas as pd  
│ import folium  
│ import plotly.express as px  
│ import plotly.graph\_objects as go  
│ from plotly.subplots import make\_subplots  
│ import plotly.io as pio  
│ pio.templates.default = "plotly\_white"  
│   
│ # Load dataset  
│ df = pd.read\_csv("D:/Metro\_Analysis/Delhi-Metro-Network.csv")  
│ print("Dataset loaded, rows:", len(df))  
│   
│ # Verify data for nulls  
│ null\_values = df.isnull().sum()  
│ print("Null values:\n", null\_values)  
│   
│ # Check data types  
│ data\_type = df.dtypes  
│ print("Data types:\n", data\_type)  
│   
│ # Convert dates  
│ df['Opening Date'] = pd.to\_datetime(df['Opening Date'], errors='coerce')  
└─$

**Purpose:** Loads the dataset, checks for null values, verifies data types, and converts `Opening Date` to datetime format for temporal analysis.  
**Console Output:** • Dataset loaded, rows: ~300  
 • Null values: [column-wise counts]  
 • Data types: object, float64, datetime64  
**Key Lines:** • `pd.read\_csv(...)`: Loads CSV from `D:\Metro\_Analysis\`.  
 • `df.isnull().sum()`: Checks for missing data.  
 • `pd.to\_datetime(..., errors='coerce')`: Handles invalid date formats.

`metro\_analysis.ipynb` - Folium Map

┌──(metro\_venv)──[D:\Metro\_Analysis]  
│ 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': 'light green', 'Blue line branch': 'light blue', 'Gray line': 'lightgray'  
│ }  
│   
│ map\_with\_tooltip = folium.Map(location=[28.7041, 77.1025], zoom\_start=11)  
│   
│ for index, row in df.iterrows():  
│ line = row['Line']  
│ color = line\_colors.get(line, 'black')  
│ 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(map\_with\_tooltip)  
│   
│ map\_with\_tooltip  
└─$

**Purpose:** Creates an interactive Folium map centered on Delhi, with markers for each station colored by metro line and tooltips showing station name and line.

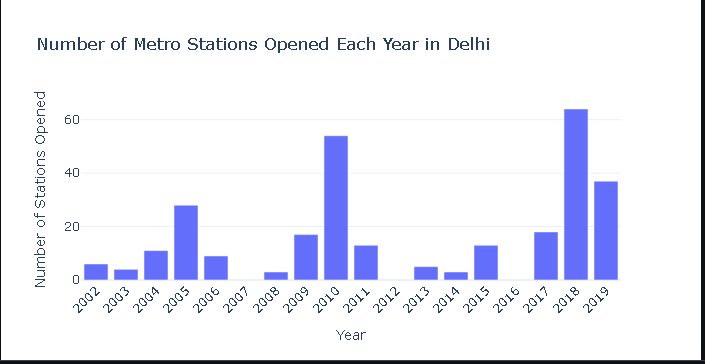
**Console Output:** •   
  
**Key Lines:** • `line\_colors = {...}`: Defines color scheme for metro lines.  
 • `folium.Map(location=[28.7041, 77.1025], ...)`: Initializes map at Delhi’s coordinates.  
 • `folium.Marker(...)`: Adds markers with popups and tooltips.

`metro\_analysis.ipynb` - Station Openings per Year

┌──(metro\_venv)──[D:\Metro\_Analysis]  
│ df['Opening Year'] = df['Opening Date'].dt.year  
│ stations\_per\_year = df['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()  
└─$

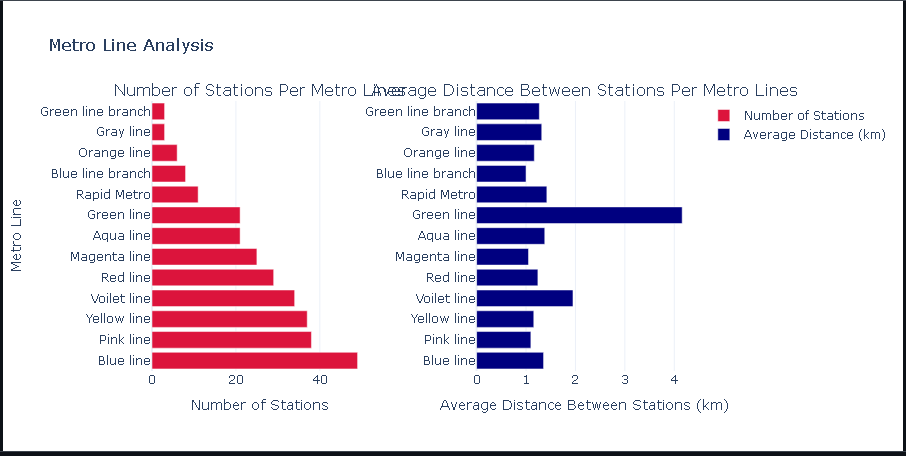
**Purpose:** Visualizes the number of stations opened each year using a Plotly bar chart.

**Console Output:** • [Bar chart displayed in Jupyter]

  
  
**Key Lines:** • `df['Opening Year'] = df['Opening Date'].dt.year`: Extracts year from dates.  
 • `px.bar(...)`: Creates an interactive bar chart.  
 • `fig.update\_layout(...)`: Customizes axes and title.

`metro\_analysis.ipynb` - Line Analysis

┌──(metro\_venv)──[D:\Metro\_Analysis]  
│ stations\_per\_line = df['Line'].value\_counts()  
│ total\_distance\_per\_line = df.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  
│ })  
│ line\_analysis = line\_analysis.sort\_values(by='Number of Stations', ascending=False)  
│ line\_analysis.reset\_index(drop=True, inplace=True)  
│   
│ fig = make\_subplots(rows=1, cols=2, subplot\_titles=('Number of Stations Per Metro Lines',  
│ 'Average Distance Between Stations Per Metro Lines'),  
│ horizontal\_spacing=0.2)  
│ 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)  
│ 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)  
│ 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)  
│ fig.update\_yaxes(title\_text='Metro Line', row=1, col=1)  
│ fig.update\_yaxes(title\_text='', row=1, col=2)  
│ fig.update\_layout(height=450, width=900, title\_text='Metro Line Analysis', template='plotly\_white')  
└─$

**Purpose:** Analyzes metro lines by station count and average distance between stations, visualized in side-by-side horizontal bar charts.  
**Console Output:** • [Subplots displayed in Jupyter]  
  
**Key Lines:** • `total\_distance\_per\_line = df.groupby('Line')['Distance from Start (km)'].max()`: Computes total line length.  
 • `avg\_distance\_per\_line = total\_distance\_per\_line / (stations\_per\_line - 1)`: Calculates average distance.  
 • `make\_subplots(...)`: Creates dual bar charts for comparison.

`metro\_analysis.ipynb` - Station Layout Distribution

┌──(metro\_venv)──[D:\Metro\_Analysis]  
│ layout\_counts = df['Station Layout'].value\_counts()  
│ 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')  
│ fig.update\_layout(xaxis\_title='Station Layout', yaxis\_title='Number of Stations',  
│ coloraxis\_showscale=False, template='plotly\_white')  
└─$

**Purpose:** Visualizes the distribution of station layouts) using a Plotly bar chart.  
**Console Output:** • [Bar chart displayed in Jupyter]  
  
**Key Lines:** • `layout\_counts = df['Station Layout'].value\_counts()`: Counts stations by layout.  
 • `px.bar(...)`: Creates a colored bar chart.  
 • `fig.update\_layout(...)`: Customizes axes and removes color scale.

Troubleshooting Steps

Error: `FileNotFoundError: Delhi-Metro-Network.csv`

\*\*Cause\*\*: Dataset not found in `D:\Metro\_Analysis\`.

\*\*Fix\*\*:  
 ┌──(metro\_venv)──[D:\]  
 │ dir D:\Metro\_Analysis\Delhi-Metro-Network.csv  
 │ Move CSV to correct directory.  
 └─$

\*\*Verify\*\*:  
 ┌──(metro\_venv)──[D:\]  
 │ python -c "import pandas as pd; df = pd.read\_csv('D:/Metro\_Analysis/Delhi-Metro-Network.csv'); print('File loaded')"  
 └─$

Error: `ValueError: Invalid format string` in `pd.to\_datetime`

\*\*Cause\*\*: Inconsistent date formats in `Opening Date`.

\*\*Fix\*\*:  
 ┌──(metro\_venv)──[D:\Metro\_Analysis]  
 │ df['Opening Date'] = pd.to\_datetime(df['Opening Date'], errors='coerce')  
 └─$

\*\*Verify\*\*:  
 ┌──(metro\_venv)──[D:\Metro\_Analysis]  
 │ print(df['Opening Date'].isnull().sum())  
 └─$

Error: Folium Map Not Displaying

\*\*Cause\*\*: Missing `folium` library or Jupyter rendering issue.

\*\*Fix\*\*:  
 ┌──(metro\_venv)──[D:\]  
 │ pip install folium  
 │ Ensure notebook is trusted or restart kernel.  
 └─$

\*\*Verify\*\*:  
 ┌──(metro\_venv)──[D:\]  
 │ python -c "import folium; print(folium.\_\_version\_\_)"  
 └─$

Error: Plotly Charts Not Rendering

\*\*Cause\*\*: Incorrect column names or missing `plotly` library.

\*\*Fix\*\*:  
 ┌──(metro\_venv)──[D:\Metro\_Analysis]  
 │ print(df.columns)  
 └─$  
 ┌──(metro\_venv)──[D:\]  
 │ pip install plotly  
 └─$

\*\*Verify\*\*:  
 ┌──(metro\_venv)──[D:\]  
 │ python -c "import plotly; print(plotly.\_\_version\_\_)"  
 └─$

Desired Outputs

**Folium Map:** • Interactive map centered at [28.7041, 77.1025].  
 • Markers for ~300 stations, color-coded by line (e.g., Red line: red, Blue line: blue).  
 • Tooltips: Station name and line.  
[Screenshot: Folium Map Output]  
**Plotly Visuals:** • Bar chart: Stations opened per year.  
 [Screenshot: Station Openings per Year Bar Chart]  
 • Subplots: Station counts and average distances per line.  
 [Screenshot: Metro Line Analysis Subplots]  
 • Bar chart: Station layout distribution (e.g., Elevated: ~150, Underground: ~100).  
 [Screenshot: Station Layout Distribution Bar Chart]  
**Console:** • Dataset rows: ~300.  
 • Null values: None or minimal.  
 • Data types: Correct (e.g., `Opening Date` as datetime64).

Dependencies and Installations

Dependencies

**Python:** 3.9.13.  
**Libraries:** • `pandas`  
 • `folium`  
 • `plotly`  
 • `jupyter`  
**Scripts:** • `metro\_analysis.ipynb`: Main analysis notebook.  
**Tools:** Jupyter Notebook.

Installation Commands

┌──(metro\_venv)──[D:\]  
│ D:\>python -m venv D:\metro\_venv  
│ D:\>D:\metro\_venv\Scripts\activate  
│ (metro\_venv) D:\>pip install pandas folium plotly jupyter  
└─$

Script Setup

┌──(metro\_venv)──[D:\]  
│ Save Notebook:  
│ Create `metro\_analysis.ipynb` in `D:\Metro\_Analysis\`.  
│ (metro\_venv) D:\>dir D:\Metro\_Analysis\metro\_analysis.ipynb  
│ Test Notebook:  
│ (metro\_venv) D:\>jupyter notebook  
│ Run all cells in `metro\_analysis.ipynb`.  
└─$

Appendices

Appendix 1: Dependency Setup

┌──(metro\_venv)──[D:\]  
│ Virtual Environment:  
│ D:\>python -m venv D:\metro\_venv  
│ D:\>D:\metro\_venv\Scripts\activate  
│ Libraries:  
│ (metro\_venv) D:\>pip install pandas folium plotly jupyter  
│ Dataset:  
│ Place `Delhi-Metro-Network.csv` in `D:\Metro\_Analysis\`.  
│ Notebook:  
│ Save `metro\_analysis.ipynb` to `D:\Metro\_Analysis\`.  
└─$

Appendix 2: Cleanup Steps

┌──(metro\_venv)──[D:\]  
│ Stop Jupyter:  
│ In Jupyter: File > Shut Down.  
│ Deactivate Environment:  
│ (metro\_venv) D:\>deactivate  
│ Verify:  
│ D:\>tasklist | findstr "python jupyter"  
└─$

Appendix 3: Performance Enhancements

┌──(metro\_venv)──[D:\]  
│ Pandas:  
│ df = pd.read\_csv("D:/Metro\_Analysis/Delhi-Metro-Network.csv").cache()  
│ Folium:  
│ map\_with\_tooltip = folium.Map(location=[28.7041, 77.1025], zoom\_start=11, tiles='cartodbpositron')  
│ Plotly:  
│ pio.templates.default = "plotly\_white"  
│ Best Practices:  
│ # Log errors  
│ import logging  
│ logging.basicConfig(filename="metro\_analysis.log")  
│ # Backup dataset  
│ D:\>copy D:\Metro\_Analysis\Delhi-Metro-Network.csv D:\Metro\_Analysis\backup.csv  
└─$