**Title: PUBLIC TRANSPORT ANALYSIS**

**PHASE 3:** Development part 1

**INTRODUCTION :**

Public transport analysis is a critical evaluation of transportation systems that serve the general public. It involves assessing various aspects of public transportation, including buses, trains, trams, subways, and other modes of communal mobility. The analysis can encompass a range of factors such as efficiency, accessibility, affordability, safety, environmental impact, and social equity. Researchers and policymakers often conduct these analyses to make informed decisions about improving public transport services, addressing congestion, reducing emissions, and enhancing urban planning. Understanding the strengths and weaknesses of public transportation systems is crucial for creating sustainable and efficient transportation networks that benefit communities and regions.

**GIVEN DATASET :**

TripID RouteID StopID StopName WeekBeginnumberofboarding

23631 100 14156 181 CrossRd6/30/20130:00 1

23631 100 14144 177Cross Rd6/0/20130:00 1

23632 100 14132 175CrossRd6/30/2010:00 1

23633 100 12266 ZoneAArndaleInterchange6/30/201 0:00 2

23634 100 14132 175Cross Rd6/30/2013 0:00 1

23634 100 13335 9AHolbrooksRd6/30/2013 0:00 1

23634 100 13878 9MarionRd6/30/2013 0:00 1

23634 100 13045 206HolbrooksRd6/30/2013 0:00 1

23635 100 13335 9AHolbrooksRd6/30/2013 0:00 1

**Necessary Step to Follow**

**1.Import libraries:**

Start by importing the necessary libraries:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

**Program**

import geopandas as gpd

# Read public transport data from a shapefile or other supported formats

gdf = gpd.read\_file('public\_transport.shp')

# Perform analysis, e.g., plot the data

gdf.plot()

**2.Load the Dataset :**

Load your dataset into a Pandas DataFrame. You can typically find public transport analysis dataset in CSV format,but you can adapt this code to other formats as needed.

**Program :**

Df=pd.read\_csv(‘D:\USA\_Transport.csv’)

Pd.read()

**3.Exploratory Data Analysis**

1 .Data Collection:

- Gather relevant public transport data. Common sources include APIs, CSV files, or databases.

2. Data Cleaning:

- Remove duplicates, handle missing values, and format data types.

3. Data Visualization:

- Import libraries like `matplotlib` or `seaborn` to create visualizations. For example, you can create histograms, box plots, and scatter plots to understand the distribution of data and relationships between variables.

4. Summary Statistics:

- Use libraries like `pandas` to generate descriptive statistics, such as mean, median, standard deviation, etc., to get a better understanding of the data.

5. Geospatial Analysis:

- If you have location data, use libraries like `geopandas` to visualize routes, stops, or geographic patterns.

6. Time Series Analysis:

- For time-based data, you can use libraries like `pandas` to resample data, calculate rolling statistics, and create time series plots.

7*.* Data Insights:

- Use EDA to draw preliminary insights about public transport usage, peak hours, popular routes, and other relevant information.

Here's a Python code template to get you started:

**program**

python

# Import necessary libraries

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load the dataset (replace 'data.csv' with your file)

df = pd.read\_csv('data.csv')

# Data Cleaning (e.g., handle missing values)

df.dropna(inplace=True)

# Data Visualization

plt.figure(figsize=(10, 6))

sns.histplot(df['passenger\_count'], kde=True)

plt.title('Distribution of Passenger Counts')

plt.show()

# Summary Statistics

summary\_stats = df.describe()

# Geospatial Analysis (if applicable)

# Example: Create a scatter plot of bus stop locations

import geopandas as gpd

gdf = gpd.GeoDataFrame(df, geometry=gpd.points\_from\_xy(df.longitude, df.latitude))

world = gpd.read\_file(gpd.datasets.get\_path('naturalearth\_lowres'))

world.plot()

gdf.plot(marker='o', color='red', markersize=5, ax=plt.gca())

# Time Series Analysis (if applicable)

# Example: Resample data to daily frequency

df['date'] = pd.to\_datetime(df['date'])

df.set\_index('date', inplace=True)

daily\_passengers = df['passenger\_count'].resample('D').mean()

**4. Feature Scaling and Encoding:**

If you're using machine learning models, scale numerical features and encode categorical variables.

python

from sklearn.preprocessing import StandardScaler, LabelEncoder

scaler = StandardScaler()

df['ScaledPassengerCount'] = scaler.fit\_transform(df[['PassengerCount']])

encoder = LabelEncoder()

df['DayOfWeekEncoded'] = encoder.fit\_transform(df['DayOfWeek'])

**Preprocessing the Dataset:**

1.Import Libraries:

Import the necessary Python libraries, such as Pandas, NumPy, and Matplotlib for data manipulation and visualization.

python

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

2. Load the Dataset:

Load your public transport dataset into a Pandas DataFrame.

python

df = pd.read\_csv('public\_transport\_data.csv')

3. Data Cleaning:

Clean the data by handling missing values, removing duplicates, and correcting any inconsistencies in the data.

python

# Handling missing values

df.dropna(inplace=True)

# Removing duplicates

df.drop\_duplicates(inplace=True)

4. Data Transformation:

Perform any necessary transformations like converting data types, creating new features, or aggregating data.

python

# Convert date/time strings to datetime objects

df['Timestamp'] = pd.to\_datetime(df['Timestamp'])

# Create new features, e.g., day of the week, hour of the day

df['DayOfWeek'] = df['Timestamp'].dt.day\_name()

df['HourOfDay'] = df['Timestamp'].dt.hour

5. Data Exploration:

Explore the data using descriptive statistics and visualizations to gain insights.

python

# Summary statistics

summary\_stats = df.describe()

# Plotting data

plt.figure(figsize=(10, 6))

plt.hist(df['PassengerCount'], bins=20, color='skyblue')

plt.title('Passenger Count Distribution')

plt.xlabel('Passenger Count')

plt.ylabel('Frequency')

plt.show()

6. Filtering and Subsetting:

Depending on your analysis, you may need to filter and subset the data to focus on specific aspects or time periods.

python

# Filter data for a specific day or time period

filtered\_data = df[df['DayOfWeek'] == 'Monday']

7. Feature Scaling and Encoding:

If you're using machine learning models, scale numerical features and encode categorical variables.

python

from sklearn.preprocessing import StandardScaler, LabelEncoder

scaler = StandardScaler()

df['ScaledPassengerCount'] = scaler.fit\_transform(df[['PassengerCount']])

encoder = LabelEncoder()

df['DayOfWeekEncoded'] = encoder.fit\_transform(df['DayOfWeek'])

8. Save Preprocessed Data:

Save the preprocessed data to a new CSV file for further analysis.

python

df.to\_csv('preprocessed\_transport\_data.csv', index=False)

**Challenges involved in loading and preprocessing dataset**

**1. Data Volume:** Public transport datasets can be massive, containing information on millions of trips, vehicles, and passengers. Handling such large volumes of data can strain computational resources.

**2. Data Quality:** Public transport data may contain errors, missing values, or inconsistencies. Cleaning and validating the data is a crucial step in preprocessing.

**3. Data Sources**: Public transport data can come from various sources, such as ticketing systems, GPS trackers, and manual surveys. Integrating data from different sources can be complex.

**4. Real-Time Updates**: Many public transport systems provide real-time data, which requires continuous updates and synchronization, adding complexity to preprocessing.

**5. Geospatial Data**: Public transport data often involves geospatial information, such as routes and stops. Managing and processing geospatial data requires specialized tools and knowledge.

**6. Data Privacy**: Public transport data may contain sensitive information about passengers. Anonymizing and handling data in compliance with privacy regulations is a concern.

**7. Time-Series Data**: Public transport data is often time-series data, which necessitates handling time-related features, trends, and seasonality.

**8. Feature Engineering**: Creating meaningful features from raw data is essential for analysis. It can be a time-consuming and iterative process.

**9. Data Aggregation**: Aggregating data at different spatial and temporal resolutions to match the analysis objectives can be challenging.

**10. Visualization**: Communicating insights from the data often requires effective visualization tools and techniques.

**11. Scalability**: As public transport systems grow, the preprocessing pipeline must be scalable to handle increasing data volumes.

**12. Performance Optimization**: Optimizing preprocessing steps for efficiency is crucial to reduce processing times.





