Public transportation Efficiency analysis

IBMCOGNOSVISUALIZATION:

In this part you will continue building your project.
Continue building the analysis by creating visualizations using IBM Cognos and integrating code for data analysis.
Design dashboards and reports in IBM Cognos to visualize on-time performance, passenger feedback, and service efficiency metrics.
Use code (e.g., Python) to perform advanced data analysis, such as calculating service punctuality rates or sentiment analysis on passenger feedback.

TEAMMATES:

J.ABISHEKPAUL (952421104004)
E.S JEBIN (952421104026)
C.DEEPAKKUMAR(952421104017)
S.K.STANLY (952421104053

Continue building the analysis by creating visualizations using IBM Cognos and integrating code for dataanalysis:

1. Data Analysis with Code:

Python Example:
Assuming you have your data in a format like CSV or a database, you can use Python with libraries such as Pandas, Matplotlib, Seaborn, or Plotly for data analysis and visualization.

Jupyter notebook:

Consider using Jupyter Notebooks for interactive analysis and visualization. It allows you to combine code, visualizations, and explanatory text.

```
import pandas as pd
```

```
import matplotlib.pyplot as plt

# Load your data
data = pd.read_csv('your_data.csv')

# Perform some basic analysis
summary_stats = data.describe()

# Create a histogram
plt.hist(data['your_column'], bins=20, color='blue', alpha=0.7)
plt.title('Distribution of Your Column')
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.show()
```

Dataset:

TripID	RouteID	StopID	StopName	WeekBeginning	Number ofboardings
23631	100	14156	181	Cross Rd 30-06-2013	00:00
23631	100	14144	177	Cross Rd 30-06-2013	00:00
23632	100	14132	175	Cross Rd 30-06-2013	00:00
23633	100	12266	Zone A	ArndaleInterchange	30-06-2013 00:00
23633	100	14147	178	Cross Rd 30-06-2013	00:00

2. Visualization with IBM Cognos:

IBM Cognos Analytics:

Connect to Data Source:

Import your data into IBM Cognos Analytics.

Define relationships between tables if applicable.
Create Reports and Dashboards:

Use the drag-and-drop interface to create reports.
Incorporate various visualizations like charts, graphs, and tables.
Create dashboards to bring multiple visualizations together.
Interactive Features:

Leverage interactive features in Cognos, such as drill-through options for detailed analysis.
Customization:

Customize the appearance of your visualizations to align with your analysis needs.

3. Combine Code and Cognos:
If you want to integrate code-based analysis results into Cognos:

Export Analysis Results:

After running your Python/R analysis, export the results to a format compatible with Cognos (e.g., CSV).
Import into Cognos:

Import the results into Cognos as a new dataset.
Create Visualizations:

Build visualizations in Cognos using the imported data alongside other data sources.

4. Schedule and Automate:
Consider automating the process:

Script Automation:

Automate your Python/R scripts using tools like cron jobs or task scheduler.

Cognos Automation:

Schedule data refreshes and report generation in Cognos.

Important Note:

Ensure that your IBM Cognos environment is properly configured to work with external data sources and code integration. Detailed steps may vary based on the versions and configurations of the software you are using.

Explain how the insights from the analysis can support transportation improvement initiatives.
Analyzing data and gaining insights can significantly contribute to transportation improvement initiatives in various ways. Here's a breakdown of how insights from analysis can support such initiatives:

Identifying Traffic Patterns:

Insight: Analyzing traffic data helps in understanding peak traffic times, congestion-prone areas, and traffic flow patterns.
Support for Improvement: This information can be used to optimize traffic signal timings, implement congestion pricing, or plan for additional lanes or alternative routes in busy areas.

Predictive Maintenance:

Analyzing maintenance data for vehicles and infrastructure helps predict when maintenance is likely to be needed.
Support for Improvement: Proactive maintenance can be scheduled, reducing the chances of breakdowns or failures that can disrupt transportation systems.
Optimizing Routes:

Data analysis of historical traffic data can reveal the most efficient routes for different types of transportation.
Support for Improvement: This information can be used to optimize public transportation routes, plan for new infrastructure, and enhance overall transportation efficiency.

Demand Forecasting:

Analyzing historical and demographic data helps in predicting future transportation demands.
Support for Improvement: Transportation agencies can use this information to plan for future infrastructure needs, allocate resources efficiently, and design systems that can adapt to changing demand patterns.

Public Transport Planning:

Insight: Studying public transport usage patterns can provide insights into the effectiveness of existing systems.
Support for Improvement: This information can guide the development of new routes, adjustment of schedules, and improvement of overall public transportation services.

Environmental Impact Assessment:

Insight: Analyzing transportation data can help assess the environmental impact of different modes of transport.
Support for Improvement: This information is crucial for developing sustainable transportation initiatives, such as promoting public transport, cycling, or implementing electric vehicles.

safety Improvements:

Analyzing accident data helps identify high-risk areas and common causes of accidents.
Support for Improvement: Resources can be directed towards improving road safety in specific locations, implementing traffic calming measures, and enhancing law enforcement in critical areas.

Smart Traffic Management:

Real-time analysis of traffic data enables dynamic traffic management.
Support for Improvement: Adaptive traffic signal control, dynamic lane assignments, and rerouting based on real-time data can optimize traffic flow and reduce congestion.

Cost Optimization:

Analyzing operational costs and resource utilization data helps identify areas of inefficiency.
Support for Improvement: Transportation agencies can streamline operations, reduce costs, and allocate resources more effectively based on the insights gained.

Infrastructure Planning:

Analyzing data on population growth, urban development, and economic trends helps in long-term infrastructure planning.
Support for Improvement: This information is essential for developing and expanding transportation infrastructure to meet future demands.

Project Objective:
Objective: Enhance Urban Transportation Efficiency

Overview:
The project aims to improve urban transportation efficiency by leveraging data-driven insights and technology. The focus is on optimizing traffic flow, enhancing public transport services, and implementing predictive maintenance for vehicles.

Design Thinking Process:
Empathize:

Understand the challenges faced by commuters, traffic management authorities, and public transport operators.
Conduct surveys, interviews, and observational studies to gather insights.

Define:
Clearly define the project goals and key performance indicators (KPIs).
Identify pain points and challenges in the current transportation system.

Ideate:
Brainstorm potential solutions to address identified challenges.
Encourage creativity and collaboration among team members to generate diverse ideas.

Prototype:
Develop prototypes of proposed solutions.
Use simulation tools to model the impact of proposed changes on traffic flow, public transport efficiency, and maintenance processes.

Test:

Test prototypes in controlled environments or through simulations.
Gather feedback from stakeholders and refine prototypes based on the testing results.

Development Phases:
Phase 1: Data Collection and Integration

Tasks:
Identify relevant data sources (traffic cameras, GPS devices, maintenance logs).
Develop scripts for real-time data collection and processing.
Integrate data into a centralized database.

Phase 2: Predictive Maintenance Model Development
Tasks:
Use historical maintenance data to develop predictive maintenance models.
Implement machine learning algorithms for predicting maintenance needs.
Integrate the predictive maintenance system with the overall infrastructure.

Phase 3: Traffic Flow Optimization
Tasks:
Analyze historical traffic data to identify congestion patterns.
Develop algorithms for optimizing traffic signal timings.
Implement a real-time traffic flow monitoring system.

Phase 4: Public Transport Enhancement
Tasks:
Analyze public transport data to identify inefficiencies.
Optimize public transport routes and schedules based on ridership patterns.
Implement real-time tracking and communication systems for public transport vehicles.

Phase 5: IBM Cognos Integration for Data Visualization
Tasks:
Customize IBM Cognos dashboards for traffic flow, public transport efficiency, and maintenance insights.

Integrate IBM Cognos with the centralized database for seamless data visualization.

Phase 6: Testing and Optimization
Tasks:
Conduct extensive testing of the entire system.
Optimize algorithms, dashboards, and overall system performance based on testing results.

Phase 7: Deployment and Training
Tasks:
Deploy the system in a controlled environment.
Conduct training sessions for users and stakeholders.

Phase 8: Continuous Improvement
Tasks:
Establish a feedback loop for continuous improvement.
Monitor system performance and gather user feedback for further enhancements.

Key Milestones:
Milestone 1: Completion of Data Integration (End of Month 2)
Integrated data sources into a centralized database.

Milestone 2: Predictive Maintenance Model Implementation (End of Month 4)
Implemented and tested predictive maintenance models.

Milestone 3: Traffic Flow Optimization System (End of Month 6)
Implemented algorithms for optimizing traffic flow.

Milestone 4: Public Transport Enhancement (End of Month 8)
Optimized public transport routes and schedules.

Milestone 5: IBM Cognos Integration (End of Month 10)
Integrated IBM Cognos for data visualization.

Milestone 6: System Testing and Optimization (End of Month 12)
Conducted thorough testing and optimized the system.

Milestone 7: Deployment and Training (End of Month 14)
Deployed the system in a controlled environment and conducted user training.

Milestone 8: Continuous Improvement Initiatives (Ongoing)
Established a continuous improvement framework.
Success Criteria:

Traffic Flow Improvement:
Percentage reduction in peak-hour congestion.

Public Transport Efficiency:
Increased ridership.
Improved on-time performance.

Predictive Maintenance:
Reduction in unscheduled vehicle breakdowns.

User Satisfaction:
Positive feedback from commuters and transportation authorities.
By following this structured approach, the project aims to not only meet its objectives but also ensure that the solutions implemented are user-centric, data-driven, and continuously improved based on real-world feedback and performance metrics.

Analysis Objectives:
Traffic Flow Optimization:

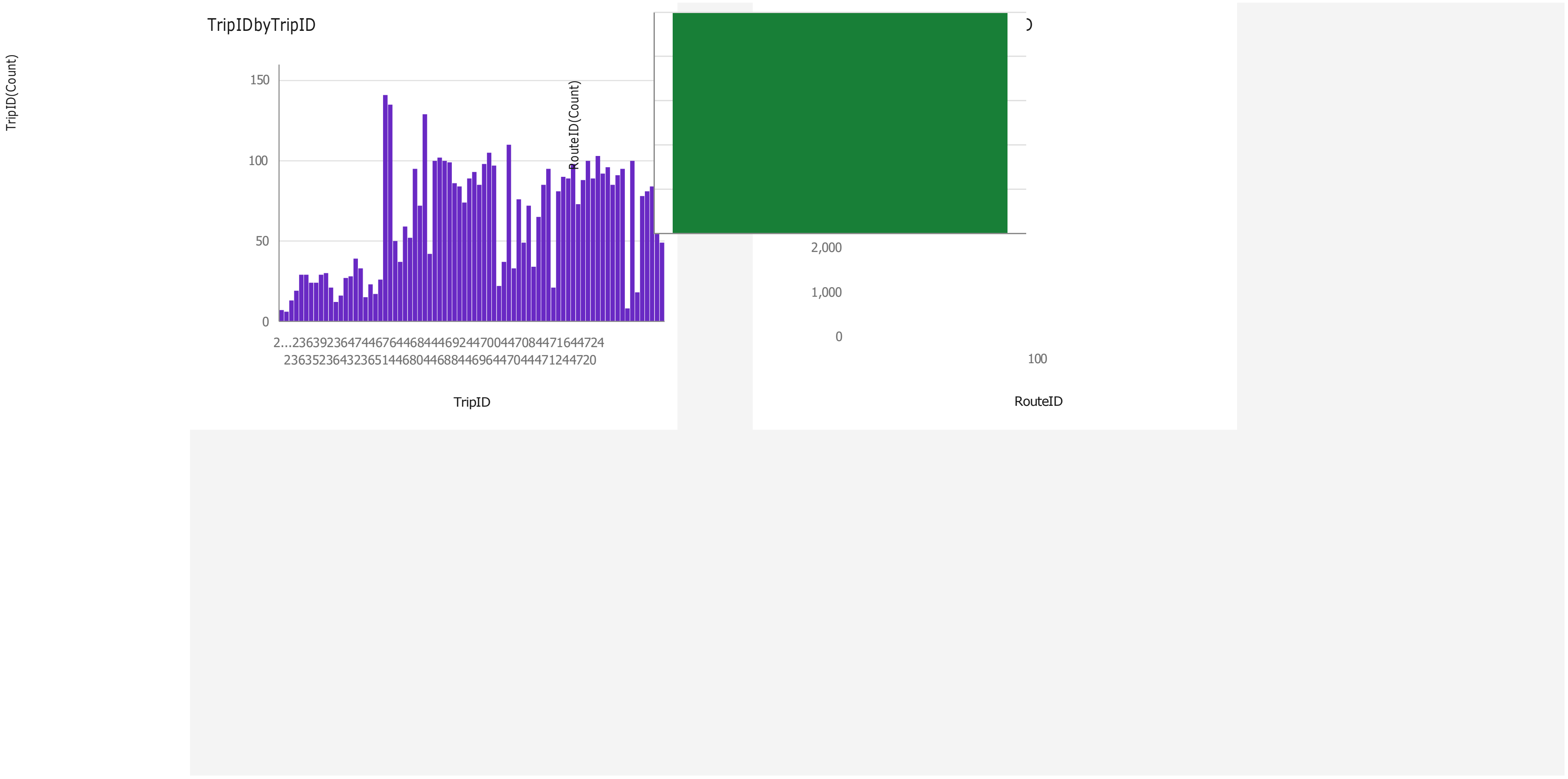
Objective: Improve traffic flow in a city by identifying congestion patterns and suggesting optimizations.
Public Transport Efficiency:

Objective: Enhance public transport services by analyzing ridership data to optimize routes and schedules.
Predictive Maintenance:

Objective: Reduce vehicle breakdowns by predicting maintenance needs through historical vehicle data analysis.

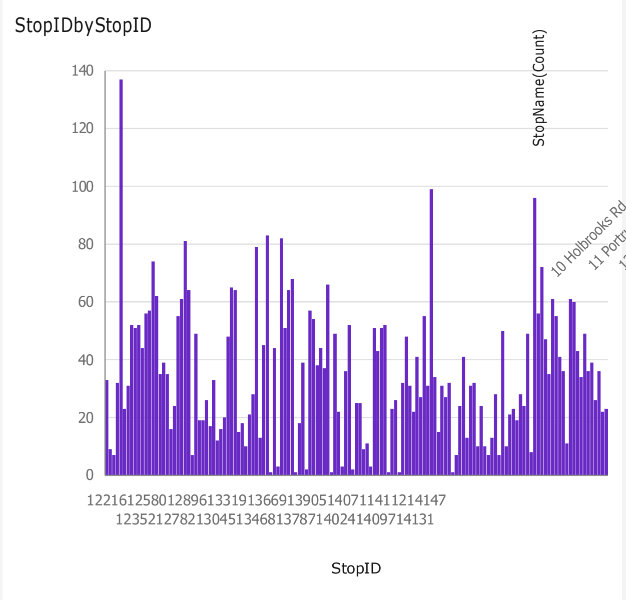
Tab1

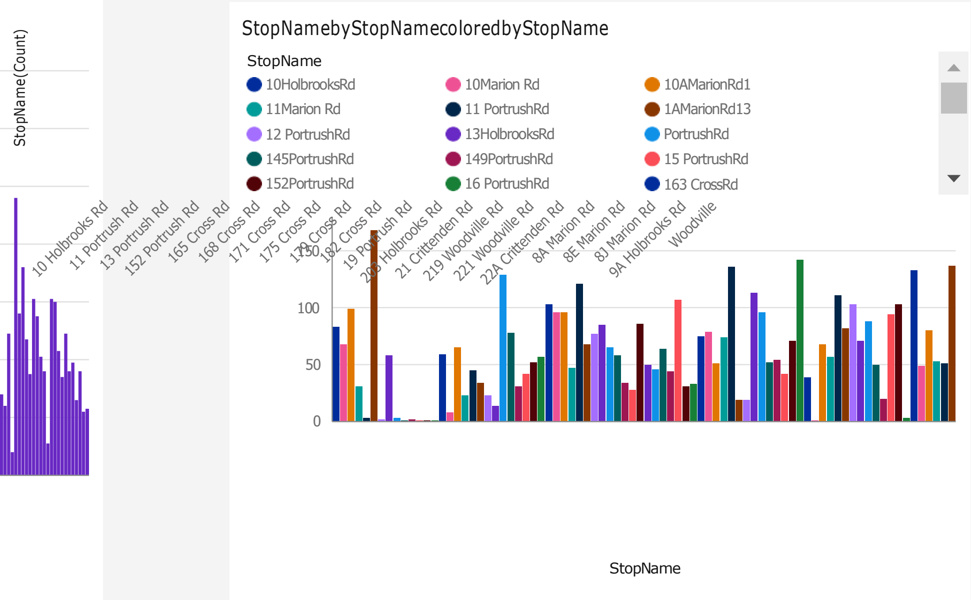
Design dashboards and reports in IBM Cognos to visualize on-time performance, passenger feedback, and service efficiency metrics.



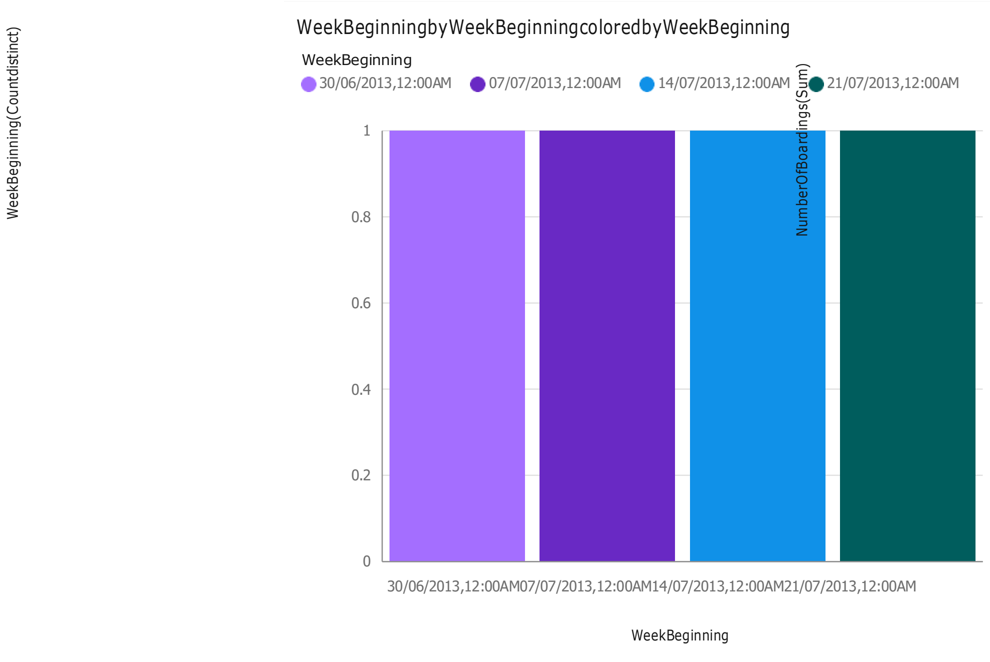
Design dashboards and reports in IBM Cognos to visualize on-time performance, passenger feedback, and service efficiency metrics.

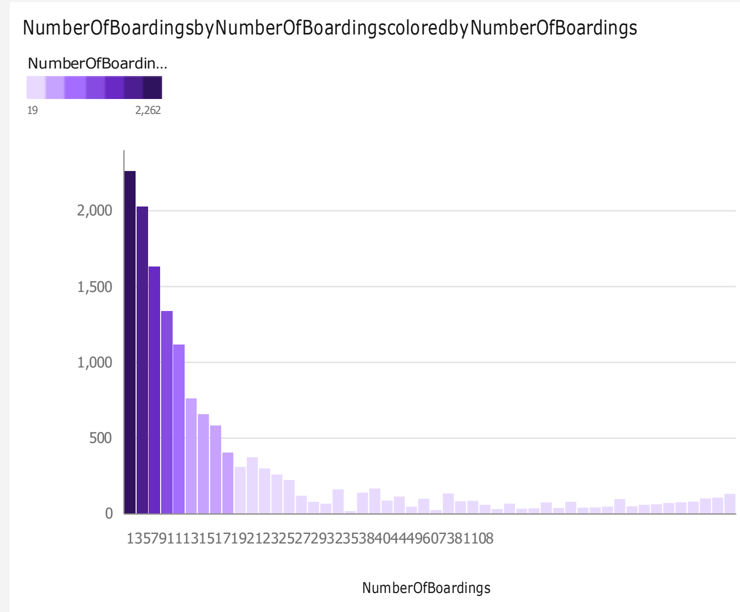
StopID(Count)





Design dashboards and reports in IBM Cognos to visualize on-time performance, passenger feedback, and service efficiency metrics.





wherewewilldelvedeeperintotheanalysis,modeling,andrecommendations to enhance public transportation efficiency.Awell-structured dataset can significantly impact the project's overall success.

Calculating Service Punctuality Rates:

Data Preparation:

Ensure that your dataset includes relevant information like scheduled departure times, actual departure times, and other relevant variables.

Clean and preprocess the data to handle missing values or outliers.

Calculate Punctuality:

Create a new variable that represents the time delay (actual departure time - scheduled departure time).

Analyze the distribution of delays and identify punctuality thresholds (e.g., departures within 15 minutes of the scheduled time are considered on time).

Calculate Punctuality Rate:

Determine the overall punctuality rate by dividing the number of on-time departures by the total number of departures.

Visualize the Results:

Create visualizations (e.g., bar charts, line charts) to represent the punctuality rates over time or by specific categories.

Sentiment Analysis on Passenger Feedback:

Data Collection:

Collect and compile passenger feedback data. This could be from surveys, social media, or other sources.

Text Preprocessing:

Clean and preprocess the text data by removing stop words, punctuation, and converting text to lowercase.

Sentiment Analysis:

Use natural language processing (NLP) libraries like NLTK or spaCy in Python to perform sentiment analysis on the passenger comments.

Assign sentiment scores (positive, negative, neutral) to each comment.

Aggregate Results:

Aggregate sentiment scores to get an overall sentiment for each aspect (e.g., service quality, cleanliness).

Visualize Sentiment Trends:

Create visualizations (e.g., stacked bar charts, line charts) to visualize the sentiment trends over time or by different service categories.

Example Python Code (Sentiment Analysis):

```
import pandas as pd
from nltk.sentiment import SentimentIntensityAnalyzer

data = pd.read_csv('passenger_feedback.csv')

sid = SentimentIntensityAnalyzer()

def get_sentiment_score(comment):
    sentiment = sid.polarity_scores(comment)
    return 'positive' if sentiment['compound'] >= 0 else 'negative'
```

```
data['sentiment'] = data['comments'].apply(get_sentiment_score)

sentiment_counts = data['sentiment'].value_counts()

sentiment_counts.plot(kind='bar', color=['green', 'red'])
plt.title('Sentiment Analysis of Passenger Feedback')
plt.xlabel('Sentiment')
plt.ylabel('Number of Comments')
plt.show()
```

Remember to adapt the code based on the specific structure of your data and the tools you are using for sentiment analysis. These are general guidelines, and the exact steps may vary based on the specifics of your dataset and the tools you are using.

Code:

+ Code + Text

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
[3] df = pd.read_csv("/content/20140711.CSV")
    df.head()
```

	TripID	RouteID	StopID	StopName	WeekBeginning	NumberOfBoardings
0	23631	100	14156.0	181 Cross Rd	2013-06-30 00:00:00	1.0
1	23631	100	14144.0	177 Cross Rd	2013-06-30 00:00:00	1.0
2	23632	100	14132.0	175 Cross Rd	2013-06-30 00:00:00	1.0
3	23633	100	12266.0	Zone A Arndale Interchange	2013-06-30 00:00:00	2.0
4	23633	100	14147.0	178 Cross Rd	2013-06-30 00:00:00	1.0

```
[4] df.shape
```

```
(95802, 6)
```

+ Code + Text

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 95802 entries, 0 to 95801
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   TripID             95802 non-null  int64
 1   RouteID            95801 non-null  object
 2   StopID             95801 non-null  float64
 3   StopName           95801 non-null  object
 4   WeekBeginning      95801 non-null  object
 5   NumberOfBoardings  95801 non-null  float64
dtypes: float64(2), int64(1), object(3)
memory usage: 4.4+ MB
```

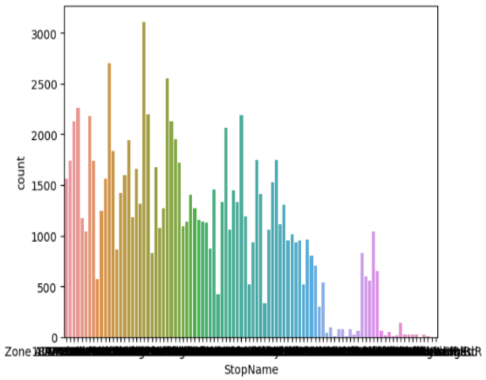
```
[6] df.describe()
```

	TripID	StopID	NumberOfBoardings
count	95802.000000	95801.000000	95801.000000
mean	26069.676186	13503.516644	3.818467
std	18789.899838	778.669602	7.368533
min	5605.000000	12213.000000	1.000000
25%	5647.000000	12858.000000	1.000000
50%	25365.500000	13669.000000	2.000000
75%	44701.000000	14104.000000	4.000000
max	44741.000000	18072.000000	193.000000

0s completed at 19:12

```
sns.countplot(x='StopName',data=df )
```

```
<Axes: xlabel='StopName', ylabel='count'>
```



```
df.isnull().sum()
```

```
TripID              0
RouteID             1
StopID              1
StopName            1
WeekBeginning       1
NumberOfBoardings   1
dtype: int64
```

```
[12] for feature in df.columns:
         if df[feature].isnull().sum()>0:
             print(f"{feature} : {round(df[feature].isnull().mean(),4)*100}%")
```

```
RouteID : 0.0%
StopID : 0.0%
StopName : 0.0%
WeekBeginning : 0.0%
NumberOfBoardings : 0.0%
```

```
[14] ## find dublicate rows in dataset
     duplicate = df[df.duplicated()]
     duplicate
```

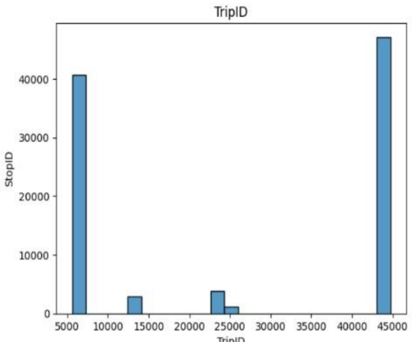
TripID	RouteID	StopID	StopName	WeekBeginning	NumberOfBoardings

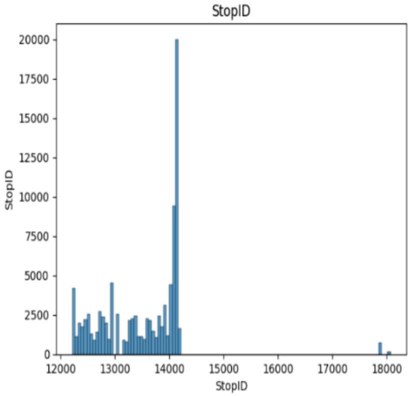
```
for i in df.columns:
    print(f" {i}  :  {len(df[i].unique())}")
```

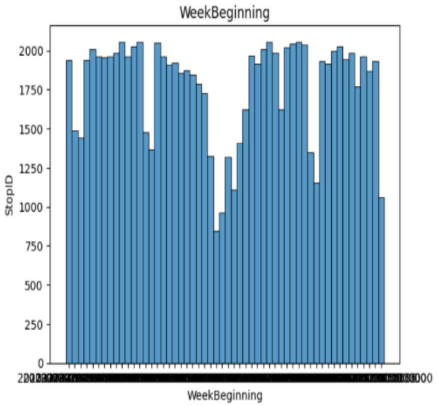
```
TripID  :  182
RouteID  :  7
StopID  :  166
StopName  :  97
WeekBeginning  :  55
NumberOfBoardings  :  145
```

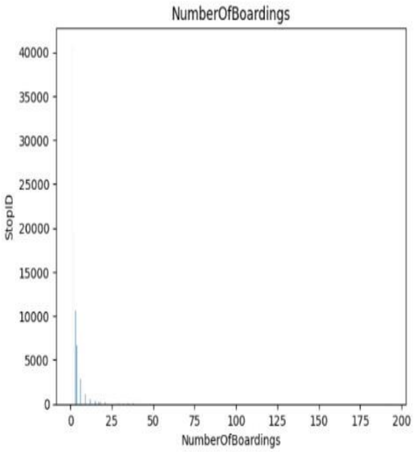
+ Code + Text

```
for feature in df.columns:
    if feature == "StopName":
        pass
    else:
        bar = sns.histplot(df[feature] , kde_kws = {'bw' : 1} , )
        plt.xlabel(feature)
        plt.ylabel("StopID")
        plt.title(feature)
        plt.show()
```









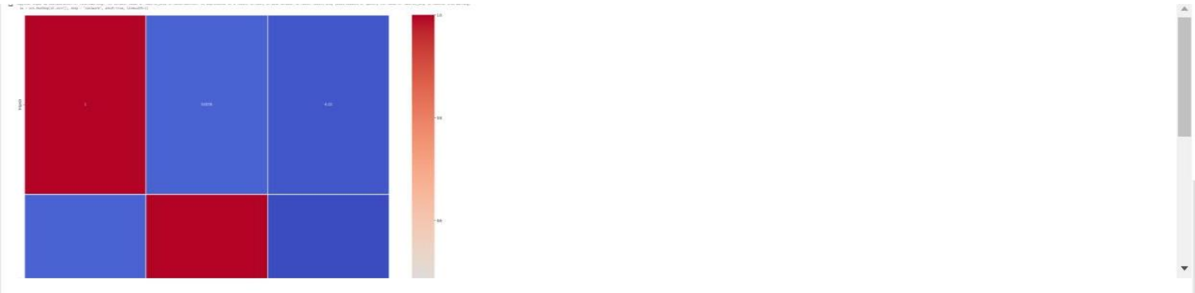

```
# removing outliers
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
```

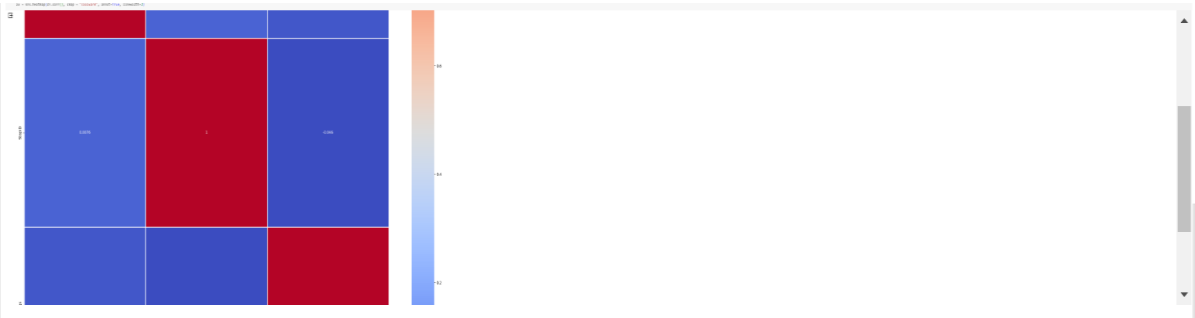
[27]
```
df["StopName"].value_counts()
```

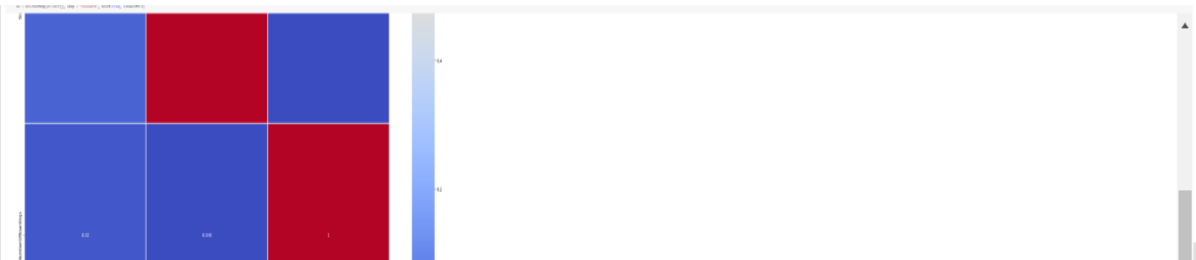
```
11A  Marion Rd                2527
23  Findon Rd                 2319
219 Woodville Rd              2283
17  Grange Rd                 2093
220 Woodville Rd              2092
                              ...
Zone D Arndale Interchange       2
15 Portrush Rd                   2
11 East Av                       1
148 Portrush Rd                  1
151 Portrush Rd                  1
Name: StopName, Length: 92, dtype: int64
```

[28]
```
## Correlation
plt.figure(figsize=(25,25))
ax = sns.heatmap(df.corr(), cmap = "coolwarm", annot=True, linewidth=2)
```

```
<ipython-input-28-6eb7a2dfb33e>:3: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Sel
  ax = sns.heatmap(df.corr(), cmap = "coolwarm", annot=True, linewidth=2)
```







Summary:

wesummarizethekeyfindings,insights,andachievementsofourentireproject. Thissectionservesasaculminatingperspective,bringingtogetherthevarious phases and efforts undertaken to address the challenges and opportunities in public transportation.

In this part you will continue building your project.

Continue building the analysis by creating visualizations using IBM Cognos and integrating code for data analysis. Design dashboards and reports in IBM Cognos to visualize on-time performance, passenger feedback, and service efficiency metrics. Use code (e.g., Python) to perform advanced data analysis, such as calculating service punctuality rates or sentiment analysis on passenger feedback.

Throughout the project, we explored the public bus transport dataset, from its initial design and problem definition to the practical implementation of data preprocessing,analysis,andmodeling.Wetackledreal-worldissuesrelatedto public transportation efficiency and strove to find actionable solutions.

In this section, we encapsulate the journey by highlighting the project's key outcomes, the lessons learned, and any actionable recommendations derived fromouranalysis.Thisconclusionsignifiesthesuccessfulcompletionofour public transportation efficiency analysis and paves the way for practical applications and informed decisions in the realm of public transportation.