PUBLIC TRANSPORTATION ANALYSIS

TEAM MEMBER

NAME : **ARIGELA THRINESH**

ROLL NO : **211521243020**

TEAM MEMBERS

1. **NARA UTTEJ**
2. **AVULA DEVIVARA PRASAD**
3. **CHANGALA SIDDAIAH VARAPRASAD**
4. **DASARI BHARATH**

Phase 3: **Development Part 1**

PROBLEMS DEFINITION:

The project involves analyzing public transportation data to assess service efficiency, on time performance, and passenger feedback. The objective is to provide insights that support transportation improvement initiatives and enhance the overall public transportation experience. This project includes defining analysis objectives, collecting transportation data, designing relevant visualizations in IBM Cognos, and using code for data analysis.

public-transport-analysis

[15]:
```
%matplotlib inline
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import datetime
import os
from math import sqrt
import warnings

## For Multiple Output in single cell
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
warnings.filterwarnings('ignore')
```

[14]:
```
# Load the Dataset

print("Load the dataset")
import pandas as pd
data = pd.read_csv('20140711.CSV', low_memory=False)
data.shape
data.head(10)
```

```
Load the dataset
```

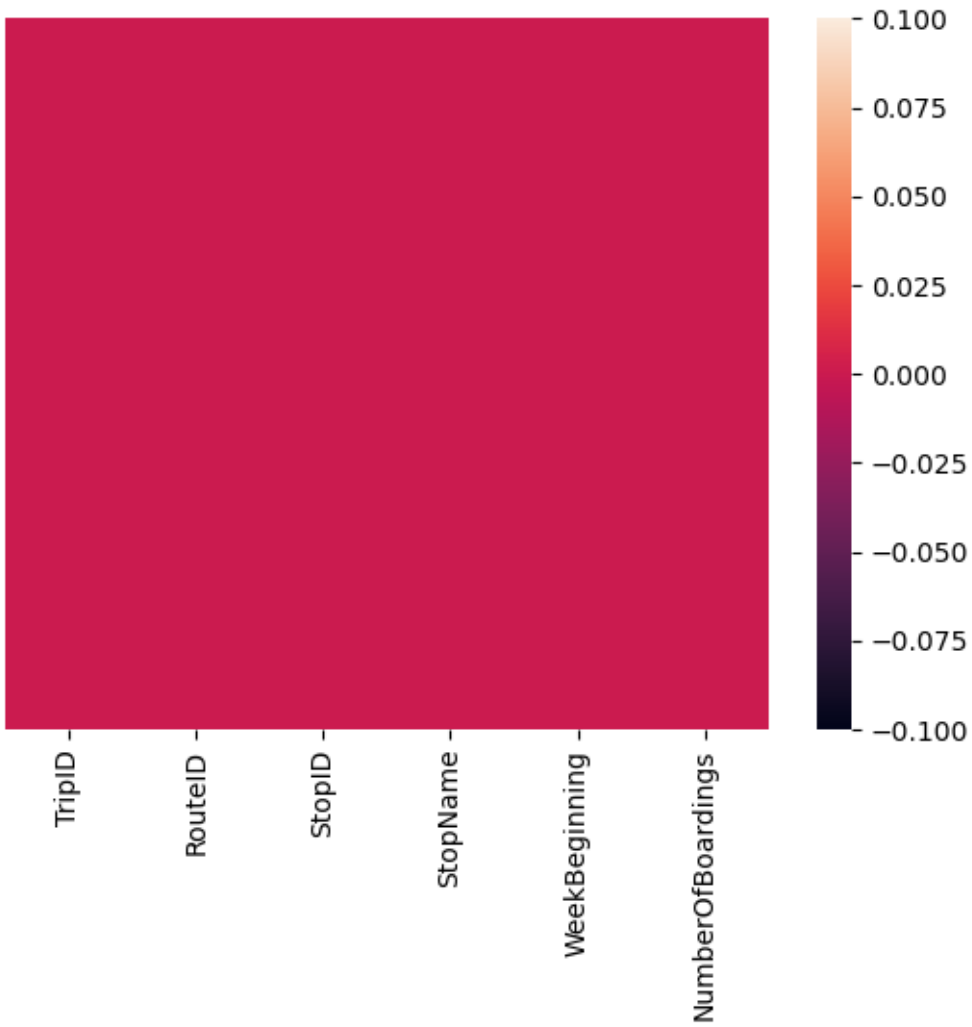
[14]:

	TripID	RouteID	StopID	StopName	WeekBeginning	NumberOfBoardings
0	23631	100	14156	181 Cross Rd	2013-06-30 00:00:00	1
1	23631	100	14144	177 Cross Rd	2013-06-30 00:00:00	
2	23632	100	14132	175 Cross Rd	2013-06-30 00:00:00	
3	23633	100	12266	Zone A Arndale Interchange	2013-06-30 00:00:00	
4	23633	100	14147	178 Cross Rd	2013-06-30 00:00:00	
5	23634	100	13907	9A Marion Rd	2013-06-30 00:00:00	
6	23634	100	14132	175 Cross Rd	2013-06-30 00:00:00	
7	23634	100	13335	9A Holbrooks Rd	2013-06-30 00:00:00	
8	23634	100	13875	9 Marion Rd	2013-06-30 00:00:00	
9	23634	100	13045	206 Holbrooks Rd	2013-06-30 00:00:00	

```
1                   1
2                   1
3                   2
4                   1
5                   1
6                   1
7                   1
8                   1
9                   1
```

[15]:
```
#check for duplicates
data = data.drop_duplicates()
import seaborn as sns
sns.heatmap(data.isnull(),yticklabels= False)
print("\nCheck data types of columns")
print(data.dtypes)
```

```
Check data types of columns
TripID                int64
RouteID              object
StopID                int64
StopName             object
WeekBeginning        object
NumberOfBoardings     int64
dtype: object
```



[5]:
```
#check all the datatypes
data['RouteID'] = pd.to_numeric(data['RouteID'], errors='coerce')
print("Handle mixed data types")
print(data.dtypes)
```

```
Handle mixed data types
TripID                 int64
RouteID              float64
StopID                 int64
StopName              object
WeekBeginning         object
NumberOfBoardings      int64
dtype: object
```

[20]:
```
data.tail()
```

[20]:
```
          TripID RouteID  StopID        StopName        WeekBeginning  \
10857229   13346    W91C   14629    21 Cashel St  2014-07-06 00:00:00
10857230   13346    W91C   14708    22 Cashel St  2014-07-06 00:00:00
10857231   13346    W91C   13709  2 Greenhill Rd  2014-07-06 00:00:00
10857232   13346    W91C   14029      10 East Av  2014-07-06 00:00:00
10857233   13346    W91C   13824     6 Leader St  2014-07-06 00:00:00

          NumberOfBoardings
10857229                  1
10857230                  3
10857231                  1
10857232                  1
10857233                  1
```

[18]:
```
data.columns
data
```

[18]:
```
          TripID RouteID  StopID                    StopName  \
0          23631     100   14156                181 Cross Rd
1          23631     100   14144                177 Cross Rd
2          23632     100   14132                175 Cross Rd
3          23633     100   12266  Zone A Arndale Interchange
4          23633     100   14147                178 Cross Rd
…            …      …      …                          …
10857229   13346    W91C   14629                21 Cashel St
10857230   13346    W91C   14708                22 Cashel St
10857231   13346    W91C   13709              2 Greenhill Rd
10857232   13346    W91C   14029                  10 East Av
10857233   13346    W91C   13824                 6 Leader St

                WeekBeginning  NumberOfBoardings
0         2013-06-30 00:00:00                  1
1         2013-06-30 00:00:00                  1
2         2013-06-30 00:00:00                  1
3         2013-06-30 00:00:00                  2
4         2013-06-30 00:00:00                  1
…                         …                  …
10857229  2014-07-06 00:00:00                  1
10857230  2014-07-06 00:00:00                  3
10857231  2014-07-06 00:00:00                  1
10857232  2014-07-06 00:00:00                  1
10857233  2014-07-06 00:00:00                  1

[10857234 rows x 6 columns]
```

[19]:
```
data.describe()
```

[19]:

	TripID	StopID	NumberOfBoardings
count	1.085723e+07	1.085723e+07	1.085723e+07
mean	2.952100e+04	1.366132e+04	4.743737e+00
std	1.960938e+04	1.971760e+03	9.382286e+00
min	7.900000e+01	1.000100e+04	1.000000e+00
25%	1.191700e+04	1.231100e+04	1.000000e+00
50%	2.747900e+04	1.334600e+04	2.000000e+00
75%	4.885800e+04	1.491600e+04	4.000000e+00
max	6.553500e+04	1.871500e+04	9.770000e+02

[6]:
```
#drop missing values
data = data.dropna()
print("\nHandle missing values")
print(data.shape)
```

```
Handle missing values
(6414906, 6)
```

[7]:
```
#converting column into datetime format
data['WeekBeginning'] = pd.to_datetime(data['WeekBeginning'], errors='coerce')
print("\nConvert 'WeekBeginning' column to datetime format")
print(data['WeekBeginning'].head())
```

```
Convert 'WeekBeginning' column to datetime format
0   2013-06-30
1   2013-06-30
2   2013-06-30
3   2013-06-30
4   2013-06-30
Name: WeekBeginning, dtype: datetime64[ns]
```

[8]:
```
#cleaning data
data['StopName'] = data['StopName'].str.strip()
print("\nClean 'StopName' column")
print(data['StopName'].head())
```

```
Clean 'StopName' column
0                  181 Cross Rd
1                  177 Cross Rd
2                  175 Cross Rd
3    Zone A Arndale Interchange
4                  178 Cross Rd
Name: StopName, dtype: object
```

[9]:
```
#print unique values
print(data.nunique())
```

```
TripID               23926
RouteID                323
StopID                6718
StopName              3840
WeekBeginning           54
NumberOfBoardings      381
dtype: int64
```

[10]:
```
data.shape
data.columns
data.head(3)
```

[10]:

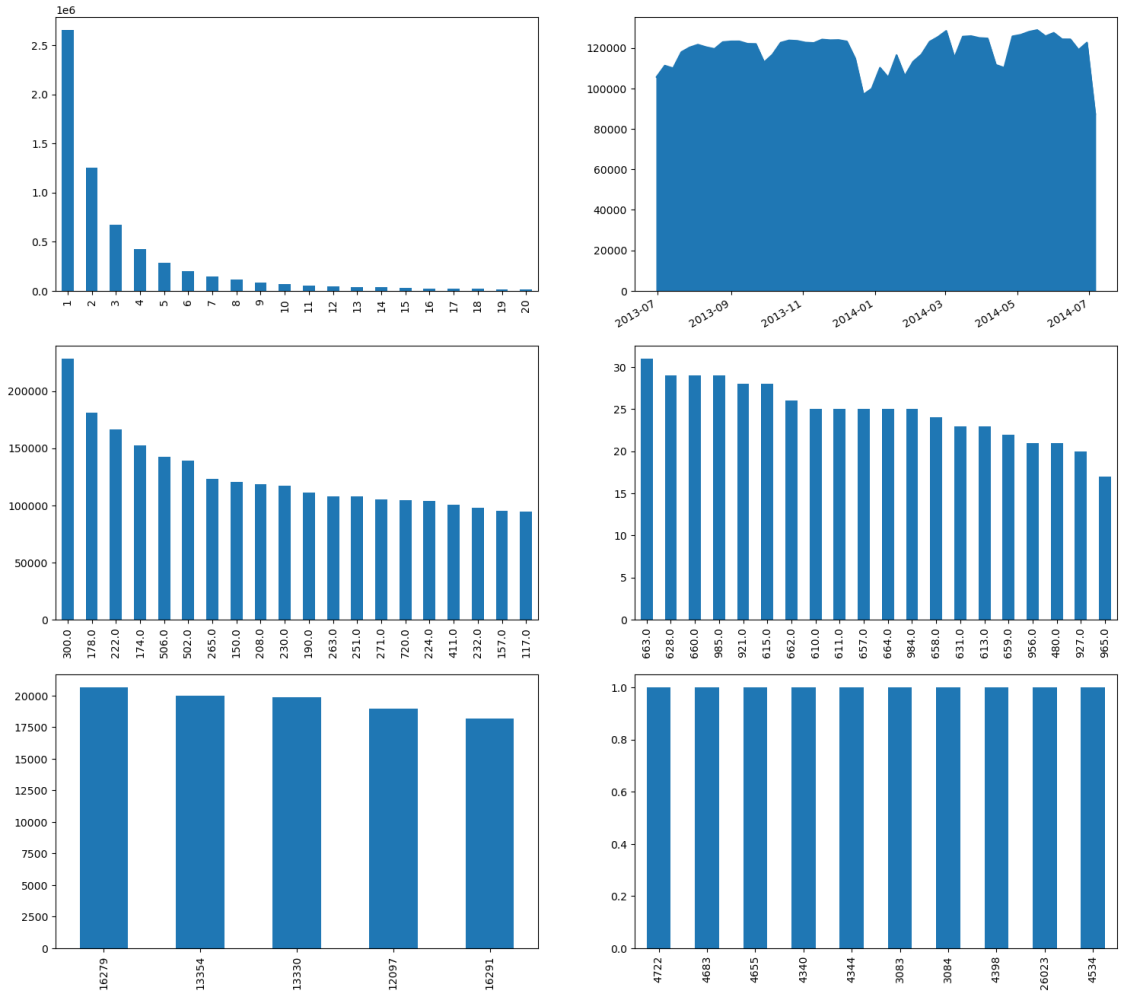
	TripID	RouteID	StopID	StopName	WeekBeginning	NumberOfBoardings
0	23631	100.0	14156	181 Cross Rd	2013-06-30	1
1	23631	100.0	14144	177 Cross Rd	2013-06-30	1
2	23632	100.0	14132	175 Cross Rd	2013-06-30	1

[11]:
```
# checking for null values
data.isnull().sum()
```

[11]:
```
TripID               0
RouteID              0
StopID               0
StopName             0
WeekBeginning        0
NumberOfBoardings    0
dtype: int64
```

[12]:
```
#ploting the columns and rows
import matplotlib.pyplot as plt
fig,axrr=plt.subplots(3,2,figsize=(18,18))
data['NumberOfBoardings'].value_counts().sort_index().head(20).plot.
 ↪bar(ax=axrr[0][0])
data['WeekBeginning'].value_counts().plot.area(ax=axrr[0][1])
data['RouteID'].value_counts().head(20).plot.bar(ax=axrr[1][0])
data['RouteID'].value_counts().tail(20).plot.bar(ax=axrr[1][1])
data['StopID'].value_counts().head(5).plot.bar(ax=axrr[2][0])
data['TripID'].value_counts().tail(10).plot.bar(ax=axrr[2][1])
```

[12]:
```
<Axes: >
```



[1]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split
```

[7]:
```
from sklearn.preprocessing import LabelEncoder

# Initialize the LabelEncoder
label_encoder = LabelEncoder()

# Encode the 'RouteID' column
data['RouteID'] = label_encoder.fit_transform(data['RouteID'])

# Continue with the standardization and scaling steps
# ...
```

[9]:
```
numerical_features = data[['RouteID', 'StopID', 'NumberOfBoardings']]

# Split the dataset into training and testing sets
X_train, X_test = train_test_split(numerical_features, test_size=0.2,␣
 ↪random_state=42)

# Standardization (Z-score scaling)
scaler_standardization = StandardScaler()
X_train_standardized = scaler_standardization.fit_transform(X_train)
X_test_standardized = scaler_standardization.transform(X_test)

# Min-Max Scaling (Normalization)
scaler_min_max = MinMaxScaler()
X_train_normalized = scaler_min_max.fit_transform(X_train)
X_test_normalized = scaler_min_max.transform(X_test)
```

[10]:
```
# Display the standardized data for the training set
print("Standardized Training Data:")
print(X_train_standardized)

# Display the standardized data for the testing set
print("\nStandardized Testing Data:")
print(X_test_standardized)
```

```
Standardized Training Data:
[[-1.11167996  0.17444278 -0.18583549]
 [-0.77476505  0.33777024 -0.39882989]
 [ 0.23597968  1.18940627 -0.39882989]
 …
 [ 1.51427448  1.74938613  0.13365611]
 [ 0.20625189  0.10089471 -0.29233269]
 [ 0.06256759  0.80644904 -0.39882989]]

Standardized Testing Data:
[[ 1.43004575  1.86401969 -0.39882989]
 [ 0.72648815  0.13437176 -0.39882989]
 [ 1.75209677  0.55435665 -0.39882989]
 …
 [-1.02745123  0.13842959 -0.39882989]
 [-1.07699754 -0.19380484 -0.07933829]
 [ 1.44986428 -0.29372878 -0.39882989]]
```

[ ]:
```

```