**PHASE 5**

**PROJECT DOCUMENTATION & SUBMISSION**

**Project’s Objective:**

* The objective of a public transportation analysis project is to assess, improve, and optimize the public transportation system within a specific region or city.
* This analysis aims to enhance the efficiency, accessibility, and sustainability of public transportation services while addressing the needs and preferences of the community.
* The specific objectives may vary depending on the project's scope and the challenges faced by the transportation system, but common goals include reducing congestion, improving environmental sustainability, enhancing accessibility, and increasing ridership.

**Design Thinking:**

* Design thinking is a problem-solving approach that can be applied to the analysis and improvement of public transportation systems.
* It involves empathizing with the users (commuters and passengers), defining the problem areas, ideating potential solutions, prototyping, and testing to come up with innovative and user-centric improvements.
* In the context of public transportation analysis, design thinking helps ensure that the solutions are not only technically sound but also cater to the needs and preferences of the passengers.

**Development Phases:**

**Understand the Current State:**

* Data Collection: Gather information on the existing public transportation system, including routes, schedules, ridership data, and operational performance.
* Stakeholder Interviews: Engage with various stakeholders, including transportation authorities, commuters, city planners, and environmental experts to understand their perspectives and needs.

**Define the Problem:**

* Identify Pain Points: Analyze the collected data to identify pain points, such as overcrowding, inefficient routes, long waiting times, and environmental concerns.
* Define Objectives: Set clear and measurable objectives for the project, which may include improving on-time performance, reducing carbon emissions, and increasing ridership.

**Ideation and Solution Generation:**

* Brainstorming: Collaboratively generate ideas and potential solutions to address the identified problems. This can involve technology enhancements, route redesign, infrastructure improvements, and policy changes.
* Prioritization: Evaluate and prioritize the proposed solutions based on their potential impact and feasibility.

**Prototyping:**

* Create prototypes or simulations of the proposed solutions to test their feasibility and effectiveness. This can include creating pilot programs, building digital models, or conducting small-scale trials.

**Testing and Iteration:**

* Implement and test the selected solutions on a limited scale to gather feedback and data on their performance.
* Iterate and refine the solutions based on the feedback and data collected.

**DATA PREPROCESSING:**

1. **Route Optimization:**

Analysis can reveal underutilized or congested routes. Transportation authorities can optimize routes to reduce travel times, decrease fuelconsumption, and enhance overall system efficiency. This can involve adjusting routes, changing schedules, or adding express services to meet passenger demand effectively.

1. **Resource Allocation:**

By analyzing passenger volume and trends, transportation agencies can allocate resources more efficiently. This includes scheduling the right number of buses or trains during peak hours and reducing services during off-peak times to optimize operational costs.

1. **On-Time Performance Improvement:**

Data analysis can track on-time performance metrics and identify factors leading to delays. Insights can support initiatives to improve punctuality through better scheduling, maintenance, and traffic management, ultimately enhancing passenger satisfaction.

1. **Load Balancing:**

Analysis can help identify imbalances in passenger loads across different routes and times. Transportation authorities can redistribute resources to ensure a more equitable distribution of passengers, leading to a more comfortable and efficient ride.

1. **Predictive Maintenance:**

Through data analysis, transportation agencies can predict when vehicles and infrastructure components require maintenance. This proactive approach can reduce downtime, improve vehicle reliability, and ensure efficient service delivery.

**Data Visualization using IBM Cognos:**

* IBM Cognos provides a range of tools for data visualization. The main tool for this purpose is Cognos Analytics. You can create interactive reports and dashboards to visualize your data.
* Use Cognos Report Studio, Cognos Dashboards, or Cognos Workspace to build customized reports and visualizations.
* You can create various chart types, tables, and graphs to present your data in a meaningful way.

**Code Integration :**

* If you have specific requirements that can't be achieved using Cognos' built-in features, you may need to integrate code or scripts. For example, you can use JavaScript to enhance interactivity in Cognos reports.
* You might also need to write SQL queries or use other languages for data transformation or manipulation, which can be integrated into your Cognos environment.

**# IMPORTING PACKAGES**

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

from sklearn.model\_selection import train\_test\_split

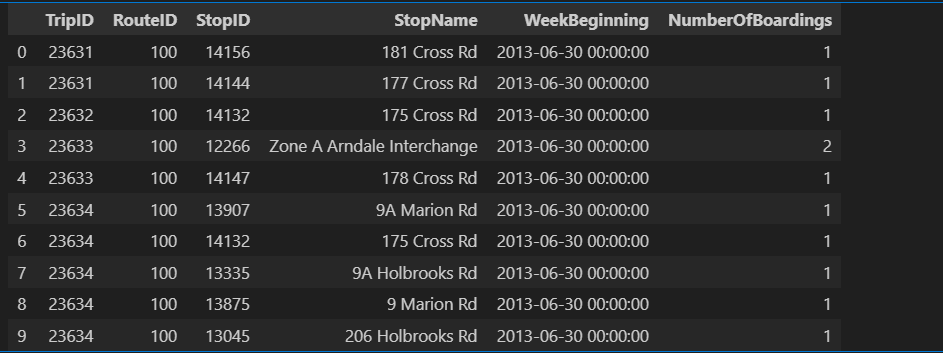
from sklearn.preprocessing import Liner Regression

**#Displaying Datasets**

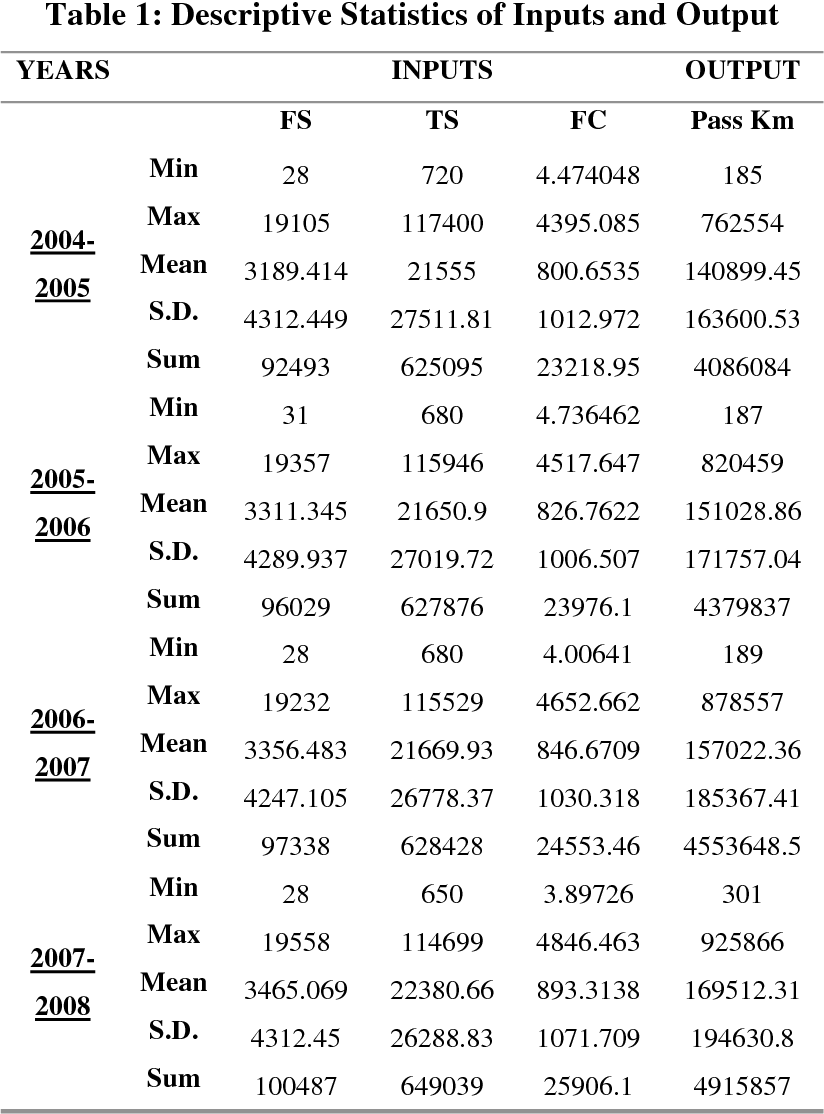
analysis = pd.read\_csv( 'D:/anaconda/archive/20140711.CSV ')

analysis.head()

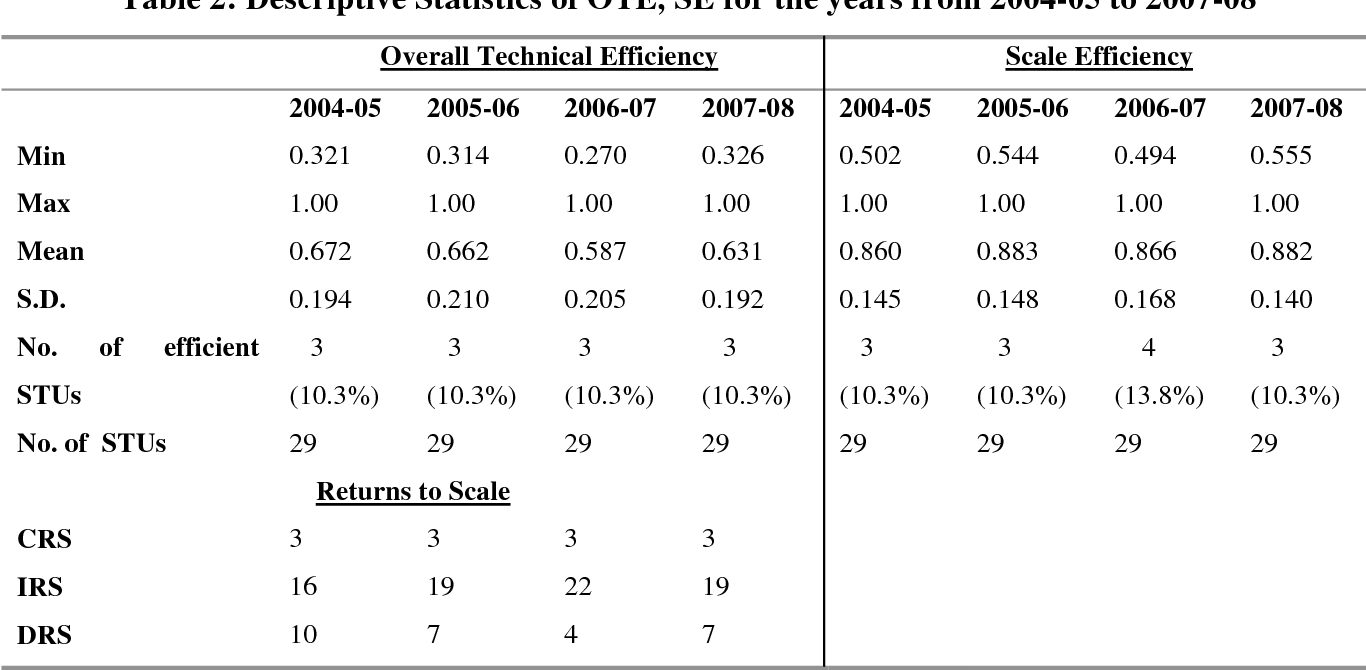
**#Output**

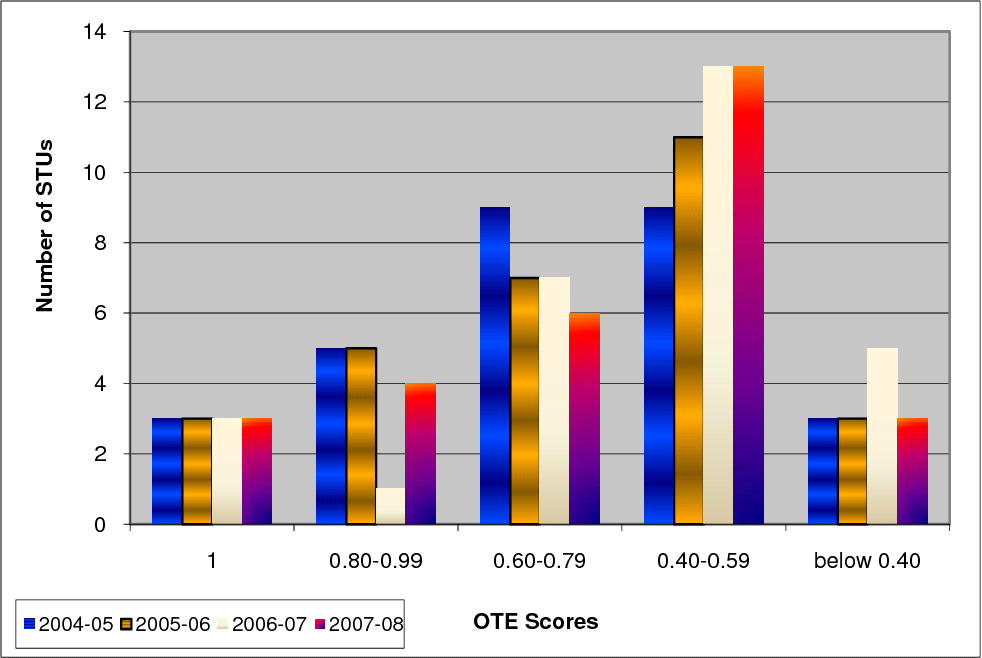


**Data base Tables\_1**

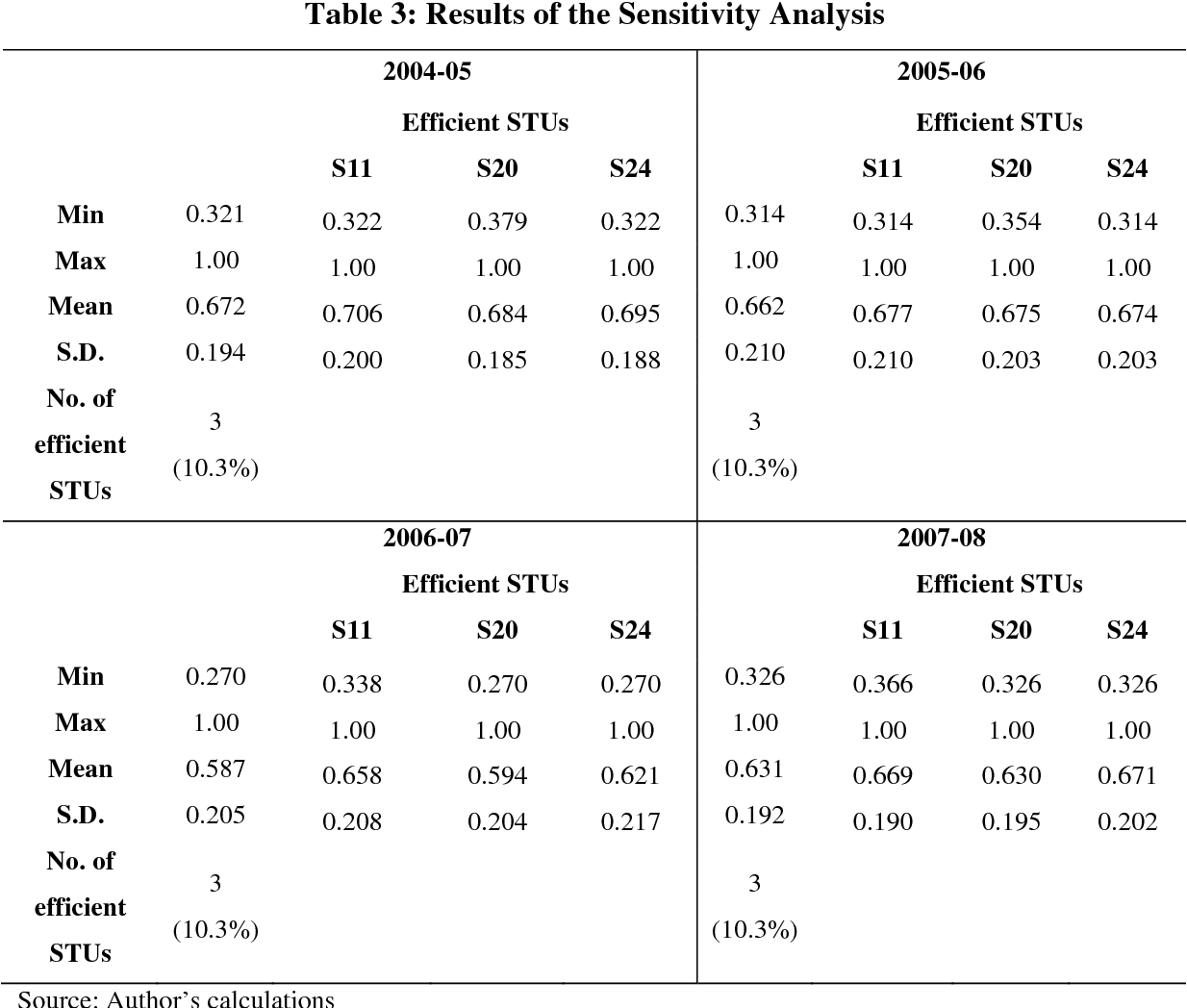


**Data base Tables\_2**





**Results Table\_3**



**VISUALIZATION :**

fig,axrr=plt.subplots(2,2,figsize=(15,15))

ax=axrr[0][0]

ax.set\_title("No of Boardings")

data['NumberOfBoardings'].value\_counts().sort\_index().head(20).plot.bar(ax=axrr[0][0])

ax=axrr[0][1]

ax.set\_title("WeekBeginning")

data['WeekBeginning'].value\_counts().plot.area(ax=axrr[0][1])

ax=axrr[1][0]

ax.set\_title("most Busiest Route")

data['RouteID'].value\_counts().head(10).plot.bar(ax=axrr[1][0])

ax=axrr[1][1]

ax.set\_title("least Busiest Route")

data['RouteID'].value\_counts().tail(10).plot.bar(ax=axrr[1][1])

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 warningsfrom IPython.core.interactiveshell import InteractiveShell

InteractiveShell.ast\_node\_interactivity = "all"

warnings.filterwarnings('ignore')

out\_geo = pd.read\_csv('../input/outgeo/output\_geo.csv')

out\_geo.shape

out\_geo.head()

from math import sin, cos, sqrt, atan2, radians

def calc\_dist(lat1,lon1):

R = 6373.0

dlon = radians(138.604801) - radians(lon1)

dlat = radians(-34.921247) - radians(lat1)

a = sin(dlat / 2)\*\*2 + cos(radians(lat1)) \* cos(radians(-34.921247)) \* sin(dlon / 2)\*\*2

c = 2 \* atan2(sqrt(a), sqrt(1 - a))

return R \* c

out\_geo.head()

out\_geo['type'].fillna('street\_address',inplace=True)

out\_geo['type'] = out\_geo['type'].apply(lambda x: str(x).split(',')[-1])

out\_geo['type'].unique()

data['WeekBeginning'] = pd.to\_datetime(data['WeekBeginning']).dt.date

data['WeekBeginning'][1]

data= pd.merge(data,out\_geo,how='left',left\_on = 'StopName',right\_on = 'input\_string')

data.head(5)

data.shape

col = ['TripID', 'RouteID', 'StopID', 'StopName', 'WeekBeginning','NumberOfBoardings',

'latitude', 'longitude','postcode','type','dist\_from\_centre']

data = data[col]

grouped = data.groupby(['StopName','WeekBeginning','type'])

grouped.head(10)

grouped.columns

|  |  |  |
| --- | --- | --- |

st\_week\_grp = pd.DataFrame(grouped).reset\_index()

st\_week\_grp.shape

st\_week\_grp.head()

st\_week\_grp1 = pd.DataFrame(st\_week\_grp.groupby('StopName')["WeekBeginning"].count()).reset\_index()

st\_week\_grp1.head()

aa = list(st\_week\_grp1[st\_week\_grp1['WeekBeginning'] == 54]['StopName'])

aa[1:10]

bb = st\_week\_grp[st\_week\_grp['StopName'].isin(aa)]

bb.head()

bb.shape

type(bb)

new\_data = data[data['StopName'].isin(aa)]

new\_data.shape

print("data without stopage removing: ", data.shape)

print("data, after removing stoppage not having the data of whole 54 weeks: ", new\_data.shape)

new\_data.head(2)

filtered\_data = new\_data[new\_data['dist\_from\_centre'] <= 100]

filtered\_data.shape

data = filtered\_data.copy()

data.shape

stopageName\_with\_boarding = bb.groupby(['StopName']).agg({'NumberOfBoardings\_sum': ['sum']})

stopageName\_with\_boarding = pd.DataFrame(stopageName\_with\_boarding.reset\_index())

stopageName\_with\_boarding.columns = ["StopName", "Total\_boarding\_on\_the\_stopage"]

stopageName\_with\_boarding.head()

data.nunique()

fig,axrr=plt.subplots(2,2,figsize=(15,15))

ax=axrr[0][0]

ax.set\_title("No of Boardings")

data['NumberOfBoardings'].value\_counts().sort\_index().head(20).plot.bar(ax=axrr[0][0])

ax=axrr[0][1]

ax.set\_title("WeekBeginning")

data['WeekBeginning'].value\_counts().plot.area(ax=axrr[0][1])

ax=axrr[1][0]

ax.set\_title("most Busiest Route")

data['RouteID'].value\_counts().head(10).plot.bar(ax=axrr[1][0])

ax=axrr[1][1]

ax.set\_title("least Busiest Route")

data['RouteID'].value\_counts().tail(10).plot.bar(ax=axrr[1][1])

stopageName\_with\_boarding = stopageName\_with\_boarding.sort\_values('Total\_boarding\_on\_the\_stopage', ascending = False)

stopageName\_with\_boarding.head(10)

stopageName\_with\_boarding.tail(10)

ax = stopageName\_with\_boarding.head(10).plot.bar(x='StopName', y='Total\_boarding\_on\_the\_stopage', rot=90)

ax.set\_title("most busiest stopage")

ax = stopageName\_with\_boarding.tail(10).plot.bar(x='StopName', y='Total\_boarding\_on\_the\_stopage', rot=90)

ax.set\_title("least busiest stopage")

data['WeekBeginning'].value\_counts().mean()

bb\_grp = data.groupby(['dist\_from\_centre']).agg({'NumberOfBoardings': ['sum']}).reset\_index()

bb\_grp.columns = bb\_grp.columns.get\_level\_values(0)

bb\_grp.head()

bb\_grp.columns

bb\_grp.tail()

import plotly.graph\_objs as go

from plotly.offline import iplot

trace0 = go.Scatter(

x = bb\_grp['dist\_from\_centre'],

y = bb\_grp['NumberOfBoardings'],mode = 'lines+markers',name = 'X2 King William St')

data1 = [trace0]

layout = dict(title = 'Distance Vs Number of boarding',

xaxis = dict(title = 'Distance from centre'),

yaxis = dict(title = 'Number of Boardings'))

fig = dict(data=data1, layout=layout)

iplot(fig)

x = data["dist\_from\_centre"]

distance\_10 = []

distance\_10\_50 = []

distance\_50\_100 = []

distance\_100\_more = []

total = 0

outlier = []

outlier\_ = 0

for i in x:

if(i<=10):

distance\_10.append(i)

total += 1

elif(i<=50):

distance\_10\_50.append(i)

total += 1

elif(i<=100):

distance\_50\_100.append(i)

total += 1

print(outlier\_)

y = len(distance\_10)+len(distance\_10\_50)+len(distance\_50\_100)

print(total)

print("passangers, boarding the buses in the radious of 10Km from the city center = ", (len(distance\_10)/total)\*100)

print("passanger, boarding the buses from the distance of 10Km to 50Km from the city center = ", (len(distance\_10\_50)/total)\*100)

print("passanger, boarding the buses from the distance of 50Km to 100 from the city center = ", (len(distance\_50\_100)/total)\*100)

grouped\_route = data.groupby(['RouteID']).agg({'NumberOfBoardings': ['sum', 'max']})

grouped\_route.columns = ["\_".join(x) for x in grouped\_route.columns.ravel()]

grouped.head(10)

grouped.columns

|  |  |
| --- | --- |

st\_week\_grp = pd.DataFrame(grouped).reset\_index()

st\_week\_grp.shape

st\_week\_grp.head()

st\_week\_grp1 = pd.DataFrame(st\_week\_grp.groupby('StopName')["WeekBeginning"].count()).reset\_index()

st\_week\_grp1.head()

aa = list(st\_week\_grp1[st\_week\_grp1['WeekBeginning'] == 54]['StopName'])

aa[1:10]

bb = st\_week\_grp[st\_week\_grp['StopName'].isin(aa)]

bb.head()

bb.shape

type(bb)

new\_data = data[data['StopName'].isin(aa)]

new\_data.shape

print("data without stopage removing: ", data.shape)

print("data, after removing stoppage not having the data of whole 54 weeks: ", new\_data.shape)

new\_data.head(2)

filtered\_data = new\_data[new\_data['dist\_from\_centre'] <= 100]

filtered\_data.shape

data = filtered\_data.copy()

data.shape

stopageName\_with\_boarding = bb.groupby(['StopName']).agg({'NumberOfBoardings\_sum': ['sum']})

stopageName\_with\_boarding = pd.DataFrame(stopageName\_with\_boarding.reset\_index())

stopageName\_with\_boarding.columns = ["StopName", "Total\_boarding\_on\_the\_stopage"]

stopageName\_with\_boarding.head()

data.nunique()

fig,axrr=plt.subplots(2,2,figsize=(15,15))

ax=axrr[0][0]

ax.set\_title("No of Boardings")

data['NumberOfBoardings'].value\_counts().sort\_index().head(20).plot.bar(ax=axrr[0][0])

ax=axrr[0][1]

ax.set\_title("WeekBeginning")

data['WeekBeginning'].value\_counts().plot.area(ax=axrr[0][1])

ax=axrr[1][0]

ax.set\_title("most Busiest Route")

data['RouteID'].value\_counts().head(10).plot.bar(ax=axrr[1][0])

ax=axrr[1][1]

ax.set\_title("least Busiest Route")

data['RouteID'].value\_counts().tail(10).plot.bar(ax=axrr[1][1])

stopageName\_with\_boarding = stopageName\_with\_boarding.sort\_values('Total\_boarding\_on\_the\_stopage', ascending = False)

stopageName\_with\_boarding.head(10)

stopageName\_with\_boarding.tail(10)

ax = stopageName\_with\_boarding.head(10).plot.bar(x='StopName', y='Total\_boarding\_on\_the\_stopage', rot=90)

ax.set\_title("most busiest stopage")

ax = stopageName\_with\_boarding.tail(10).plot.bar(x='StopName', y='Total\_boarding\_on\_the\_stopage', rot=90)

ax.set\_title("least busiest stopage")

data['WeekBeginning'].value\_counts().mean()

bb\_grp = data.groupby(['dist\_from\_centre']).agg({'NumberOfBoardings': ['sum']}).reset\_index()

bb\_grp.columns = bb\_grp.columns.get\_level\_values(0)

bb\_grp.head()

bb\_grp.columns

bb\_grp.tail()

Index(['dist\_from\_centre', 'NumberOfBoardings'], dtype='object')

import plotly.graph\_objs as go

from plotly.offline import iplot

trace0 = go.Scatter(

x = bb\_grp['dist\_from\_centre'],

y = bb\_grp['NumberOfBoardings'],mode = 'lines+markers',name = 'X2 King William St')

data1 = [trace0]

layout = dict(title = 'Distance Vs Number of boarding',

xaxis = dict(title = 'Distance from centre'),

yaxis = dict(title = 'Number of Boardings'))

fig = dict(data=data1, layout=layout)

iplot(fig)

x = data["dist\_from\_centre"]

distance\_10 = []

distance\_10\_50 = []

distance\_50\_100 = []

distance\_100\_more = []

total = 0

outlier = []

outlier\_ = 0

for i in x:

if(i<=10):

distance\_10.append(i)

total += 1

elif(i<=50):

distance\_10\_50.append(i)

total += 1

elif(i<=100):

distance\_50\_100.append(i)

total += 1

print(outlier\_)

y = len(distance\_10)+len(distance\_10\_50)+len(distance\_50\_100)

print(total)

print("passangers, boarding the buses in the radious of 10Km from the city center = ", (len(distance\_10)/total)\*100)

print("passanger, boarding the buses from the distance of 10Km to 50Km from the city center = ", (len(distance\_10\_50)/total)\*100)

print("passanger, boarding the buses from the distance of 50Km to 100 from the city center = ", (len(distance\_50\_100)/total)\*100)

grouped\_route = data.groupby(['RouteID']).agg({'NumberOfBoardings': ['sum', 'max']})

grouped\_route.columns = ["\_".join(x) for x in grouped\_route.columns.ravel()]

**EXPLANATION:**

Step-1 This line create the new figure for plotting

Step-2 This line create the histogram of the data

Step-3 This line sets the title of the plot

Step-4 Finally, this line displays the plot

**OUTPUT:**

