Project Title: Public Transportation Efficiency Analysis Documentation

1) Objective:

The objective of the "Public Transportation Efficiency Analysis " project is to evaluate the effectiveness and quality of public transportation services within a specific region.
Through this analysis, we aim to identify areas for improvement, optimize transportation routes, and enhance overall service efficiency.
By collecting and analyzing relevant data, our goal is to provide valuable insights and recommendations that can support the enhancement of public transportation infrastructure, making it more reliable, accessible, and convenient for the community.
This project seeks to contribute to the improvement of public transportation systems, ultimately benefiting both passengers and the environment by encouraging the use of sustainable and efficient transit options.

2) Design Thinking Process:

The "Public Transportation Efficiency Analysis" project employs the Design Thinking process to tackle complex issues and develop innovative solutions that address the needs of both transportation providers and passenger.

Empathize:

In this initial phase, we empathize with the commuters and transportation authorities. We gather feedback, conduct surveys, and engage with stakeholders to understand their challenges, expectations, and aspirations regarding public transportation. problem statement that guides our efforts, such as identifying bottlenecks in the transportation system, optimizing routes, or enhancing passenger experience.

Ideate:

With a well-defined problem statement, we brainstorm potential solutions. We encourage creative thinking and ideation to explore a wide range of ideas. This phase involves generating innovative strategies and considering various approaches to address the identified challenges.

Prototype:

In this phase, we create prototypes or experimental models of our proposed solutions. These could be in the form of data driven simulations, route optimization algorithms, or service enhancement concepts. Prototyping allows us to test and refine our ideas before full-scale implementation.

Test:

We rigorously test our prototypes and solutions, gathering real-world data and feedback. This phase is essential for validating the effectiveness of our proposed improvements. By evaluating the performance of our solutions in practical scenarios, we ensure that they meet the needs of both passengers and transportation authorities.

3) Development Phases:

The "Public Transportation Efficiency Analysis" project is organized into several key phases to systematically address the challenges and goals of our analysis.

Problem Formulation:

The project begins with the formulation of a clear problem statement. We define the specific objectives and challenges, such as improving transportation efficiency, reducing travel time, or enhancing service quality. This phase sets the direction for the entire project.

Data Collection:

In the data collection phase, we gather relevant data from various sources. This includes transportation records, passenger feedback, geographic information, and other datasets that can provide insights into the current state of public transportation in the region. Comprehensive and accurate data is crucial for informed analysis.

Data Preprocessing:

Raw data often requires preprocessing to clean, format, and prepare it for analysis. This phase involves handling missing values, removing outliers, and converting data into a suitable format for further investigation.

Data Analysis:

With the preprocessed data, we conduct in-depth data analysis to uncover patterns, trends, and areas for improvement. Analysis techniques may include route optimization, passenger behaviour analysis, and performance assessment of transportation services.

Data Visualization:

To effectively communicate our findings, we use data visualization tools, such as IBM Cognos, to create informative and insightful visuals. Visualizations help stakeholders easily grasp the results of our analysis and support decision-making.

Recommendations:

Based on the analysis results, we develop practical recommendations for improving public transportation efficiency. These recommendations may include route adjustments, service schedule modifications, or infrastructure enhancements.

Validation:

The project includes a validation phase w here we assess the feasibility and impact of our recommendations. We use real world data and simulations to confirm that our proposed changes will lead to positive outcomes.

Documentation and Reporting:

Finally, we document the entire project, outlining our objectives, data sources, analysis techniques, and recommendations. A comprehensive report is prepared to present our findings and insights to relevant stakeholders.

4) Analysis Objectives:

The primary objectives of the "Public Transportation Efficiency Analysis" project are to assess, optimize, and enhance the efficiency and quality of public transportation services within a specific region. Our analysis aims to achieve the following key goals:

Identify Efficiency Bottlenecks:

One of the central objectives is to pinpoint existing bottlenecks and inefficiencies in the public transportation system. By analyzing data and performance metrics, we seek to uncover areas w here services are underperforming or w here delays and congestion are common. This identification is crucial for targeted improvement.

Optimize Route Planning:

We aim to optimize transportation routes to minimize travel time and improve overall efficiency. This involves evaluating current route structures, considering traffic patterns, and implementing changes that reduce passenger travel time and enhance convenience.

Enhance Service Quality:

Improving passenger experience and service quality is a core objective. We analyze factors such as reliability, on-time performance, and passenger satisfaction. Our goal is to recommend changes that lead to more reliable and comfortable transportation services.

Promote Sustainability:

Sustainability is a key objective, and we assess the environmental impact of public transportation. We aim to recommend measures that reduce emissions and promote eco-friendly transportation solutions.

Cost Efficiency:

Analyzing cost-effectiveness is also part of our objectives. We seek to identify opportunities for cost reduction in the operation and maintenance of public transportation systems without compromising quality.

Accessibility and Inclusivity:

We consider accessibility for all passengers, including individuals with disabilities. Our analysis focuses on making public transportation more inclusive and convenient for a diverse range of commuters.

Data-Driven Decision-Making:

Ultimately, our analysis aims to support data-driven decision making for transportation authorities. We provide insights and recommendations based on factual data and performance metrics, enabling informed and strategic choices for improvement initiatives

5) Data Collection Process:

The data collection process for the "Public Transportation Efficiency Analysis" project is a critical step in gathering the necessary information to support our analysis and recommendations. It involves the following key components:

Data Sources Identification:

We begin by identifying and selecting the relevant data sources. These may include transportation authority records, GPS tracking data from vehicles, passenger surveys, public transit schedules, geographic information systems (GIS) data, and other relevant sources.

Data Retrieval:

Once the sources are identified, we retrieve the data needed for analysis. This may involve accessing databases, requesting data from transportation agencies, or collecting survey responses from

passengers. The goal is to compile comprehensive datasets that cover various aspects of public transportation.

Data Cleaning:

Raw data is often noisy and may contain errors or inconsistencies. The data cleaning process involves removing duplicates, handling missing values, and correcting inaccuracies. Clean data is essential for accurate analysis.

Data Integration:

In cases where data comes from multiple sources, we integrate and consolidate the data. This ensures that all relevant information is available for analysis in a single, coherent dataset.

Data Privacy and Compliance:

Data privacy and compliance with regulations are paramount. We take necessary precautions to anonymize and protect sensitive data, ensuring that all data collection and handling procedures adhere to legal and ethical standards.

Data Quality Assurance:

We perform quality assurance checks to validate the accuracy and consistency of the data. This includes verifying timestamps, geospatial information, and the integrity of passenger feedback.

Data Storage and Management:

Organized data storage is crucial. We use databases or data management systems to store, index, and retrieve data efficiently during the analysis phase.

Continuous Data Updates:

Public transportation data is dynamic. We establish mechanisms for continuous data updates to ensure that our analysis reflects real-time conditions and evolving patterns in public transportation.

6) Data Visualization:

Data visualization is a crucial component of the "Public Transportation Efficiency Analysis" project, allowing us to present complex information in a clear and meaningful way. The data visualization process involves the following key elements:

Tool Selection:

We utilize data visualization tools, such as IBM Cognos, to create visual representations of the collected data. The choice of tools is based on their capabilities to generate informative and interactive visuals.

Visualization Types:

Various types of visualizations are employed to convey different aspects of the analysis. These may include:

Geospatial Maps: Displaying route networks and congestion areas.
Line Charts and Bar Graphs: Showing trends and performance metrics over time.
Heatmaps: Highlighting passenger demand or congestion hotspots.
Pie Charts: Illustrating distribution of transportation modes or service usage.

Interactivity:

We ensure that our visualizations are interactive, allowing users to explore the data, zoom in on specific areas, and filter information based on their needs. Interactive visuals enhance engagement and understanding.

Insightful Presentation:

Visualizations are designed to provide actionable insights. They are accompanied by explanatory labels, legends, and annotations to help stakeholders interpret the data effectively.

Comparative Analysis:

We often create comparative visualizations that allow for the assessment of different performance metrics or changes over time. These visuals aid in decision-making by highlighting areas for improvement.

Accessibility and Inclusivity:

Considerations for accessibility are crucial. We ensure that visualizations are designed to be inclusive and understandable for all users, including those with visual impairments.

Real-time Monitoring:

Some visualizations may be used for real-time monitoring of transportation systems, allowing transportation authorities to respond swiftly to changing conditions.

Report Generation:

Visualizations are incorporated into comprehensive reports, alongside the analysis results, to provide a holistic view of the public transportation system's performance.

7) Code Integration:

The "Public Transportation Efficiency Analysis" project involves the integration of custom code to perform data analysis and generate insights. This process is essential for extracting valuable information from the collected data and supporting the project's objectives. The code integration encompasses several key aspects:

Programming Language Selection:

The choice of a programming language is fundamental to code integration. Depending on the specific requirements of the analysis, we may opt for languages such as Python, R, or others known for their data analysis capabilities.

Data Processing and Analysis:

Custom code is developed to preprocess and analyze the collected data. This includes data cleaning, feature engineering, statistical analysis, and machine learning algorithms if applicable. Code is structured to extract meaningful patterns and trends from the data.

Integration with Data Sources:

The code integrates with the data sources, ensuring that it can access and manipulate the data efficiently. This may involve connecting to databases, APIs, or real-time data streams.

Visualization Code:

In collaboration with the data visualization phase, code is integrated to generate interactive and informative visualizations. Data visualization libraries or tools may be incorporated, allowing for the creation of charts, graphs, and maps that enhance data presentation.

Insight Generation:

The code is designed to generate insights from the analyzed data. This involves the identification of performance metrics, trends, and areas for improvement. The code-generated insights complement the visualizations and support decision making.

Optimization Algorithms:

In cases where route optimization or other performance enhancements are required, custom code implementing optimization algorithms may be integrated. These algorithms aim to find the most efficient solutions for transportation routes, schedules, or resource allocation.

Real-time Monitoring and Alerts:

For real-time monitoring of transportation systems, the code is often configured to provide alerts and notifications when specific conditions or performance thresholds are met. This ensures timely responses to changing situations.

Testing and Validation:

Rigorous testing is an integral part of the code integration process. We verify that the code functions as intended, providing accurate results and insights. Validation ensures that the code aligns with the project's objectives.

Documentation:

Code integration is documented to explain the algorithms, data processing steps, and insights generated. This documentation is crucial for project transparency and know ledge sharing.

Public Transport Effectively Analysis

Analysis phase:

[ ]:

[1]:
```
%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')
```

[2]:
```
data = pd.read_csv('../input/unisys/ptsboardingsummary/20140711.CSV')
data.shape
data.head(10)
```

[2]: (10857234, 6)

[2]:

	TripID	RouteID	…	WeekBeginning	NumberOfBoardings
0	23631	100	…	2013-06-30 00:00:00	1
1	23631	100	…	2013-06-30 00:00:00	1
2	23632	100	…	2013-06-30 00:00:00	1
3	23633	100	…	2013-06-30 00:00:00	2
4	23633	100	…	2013-06-30 00:00:00	1
5	23634	100	…	2013-06-30 00:00:00	1
6	23634	100	…	2013-06-30 00:00:00	1
7	23634	100	…	2013-06-30 00:00:00	1
8	23634	100	…	2013-06-30 00:00:00	1

```
9   23634     100        …          2013-06-30 00:00:00                  1

[10 rows x 6 columns]
```

[3]:
```
out_geo = pd.read_csv('../input/outgeo/output_geo.csv')
out_geo.shape
out_geo.head()
```

[3]:
```
(4165, 10)
```

[3]:
```
           accuracy                          …
type
0           ROOFTOP                          …
street_address
1           ROOFTOP                          …
street_address
2           ROOFTOP                          …
street_address
3  GEOMETRIC_CENTER                          …
bus_station,establishment,point_of_interest,tr…
4           ROOFTOP                          …
street_address

[5 rows x 10 columns]
```

0.1 External Features

[4]:
```
#DistanceFromCentre: Distance measure from the city centre
#For Calculating Distance between centre with other bus stops by using␣
 ↪Longitude and Latitude
#we have used the Haversine formula

from math import sin, cos, sqrt, atan2, radians
def calc_dist(lat1,lon1):
    ## approximate radius of earth in km
    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
```

[5]:
```
out_geo['dist_from_centre'] = out_geo[['latitude','longitude']].apply(lambda x:␣
 ↪calc_dist(*x), axis=1)
```

[6]:
```
out_geo.head()
```

[6]:
```
            accuracy       …        dist_from_centre
0             ROOFTOP      …                5.180961
1             ROOFTOP      …                5.172525
2             ROOFTOP      …                5.180709
3  GEOMETRIC_CENTER        …                7.057549
4             ROOFTOP      …                4.900099

[5 rows x 11 columns]
```

[7]:
```
#exp_data = out_geo.head(10)
##Fill the missing values with mode
out_geo['type'].fillna('street_address',inplace=True)
out_geo['type'] = out_geo['type'].apply(lambda x: str(x).split(',')[-1])
```

[8]:
```
out_geo['type'].unique()
```

[8]:
```
array(['street_address', 'transit_station', 'premise', 'political',
       'school', 'route', 'intersection', 'point_of_interest',
       'subpremise', 'real_estate_agency', 'university', 'travel_agency',
       'restaurant', 'supermarket', 'store', 'post_office'], dtype=object)
```

[9]:
```
data['WeekBeginning'] = pd.to_datetime(data['WeekBeginning']).dt.date
data['WeekBeginning'][1]
```

[9]:
```
datetime.date(2013, 6, 30)
```

0.2 Data Aggregation

[10]:
```
#Combine the Geolocation and main input file to get final Output File.
data= pd.merge(data,out_geo,how='left',left_on = 'StopName',right_on =␣
 ↪'input_string')
data.head(5)
data.shape
```

[10]:
```
   TripID RouteID       …                    type dist_from_centre
0   23631     100       …          street_address         5.180961
1   23631     100       …          street_address         5.172525
2   23632     100       …          street_address         5.180709
3   23633     100       …         transit_station         7.057549
4   23633     100       …          street_address         4.900099

[5 rows x 17 columns]
```

[10]:
```
(10857234, 17)
```

[11]:
```
#Columns to keep for further analysis
col = ['TripID', 'RouteID', 'StopID', 'StopName',␣
 ↪'WeekBeginning','NumberOfBoardings',
      'latitude', 'longitude','postcode','type','dist_from_centre']
data = data[col]
```

[12]:
```
##saving the final dataset
#data.to_csv('Weekly_Boarding.csv',index=False)
```

Aggregate the Data According to Weeks and Stop names * **NumberOfBoardings_sum** Number of Boardings within particular week for each Bus stop * **NumberOfBoardings_count** Number of times data is recorded within week * **NumberOfBoardings_max** Maximum number of boarding done at single time within week

[13]:
```
grouped = data.groupby(['StopName','WeekBeginning','type'])
#grouped.head()
```

[14]:
```
# st_week_grp1 = pd.DataFrame(data.groupby(['StopName','WeekBeginning','type']).
 ↪agg({'NumberOfBoardings': ['sum', 'count']})).reset_index()
grouped = data.groupby(['StopName','WeekBeginning','type']).
 ↪agg({'NumberOfBoardings': ['sum', 'count','max']})
grouped.columns = ["_".join(x) for x in grouped.columns.ravel()]
```

[15]:
```
grouped.head(10)
grouped.columns
```

[15]:
```
                                          NumberOfBoardings_sum          …
NumberOfBoardings_max
StopName    WeekBeginning type                                           …
1 Anzac Hwy 2013-06-30    street_address                   1003          …
51
            2013-07-07    street_address                    783          …
28
            2013-07-14    street_address                    843          …
45
            2013-07-21    street_address                    710          …
28
            2013-07-28    street_address                    898          …
41
            2013-08-04    street_address                    799          …
40
            2013-08-11    street_address                   1012          …
71
            2013-08-18    street_address                    793          …
41
            2013-08-25    street_address                    897          …
45
```

```
             2013-09-01     street_address                    1368          …
59

[10 rows x 3 columns]
```

[15]:
```
Index(['NumberOfBoardings_sum', 'NumberOfBoardings_count',
       'NumberOfBoardings_max'],
      dtype='object')
```

[16]:
```
st_week_grp = pd.DataFrame(grouped).reset_index()
st_week_grp.shape
st_week_grp.head()
```

[16]:
```
(207864, 6)
```

[16]:
```
       StopName          …          NumberOfBoardings_max
0  1 Anzac Hwy           …                              51
1  1 Anzac Hwy           …                              28
2  1 Anzac Hwy           …                              45
3  1 Anzac Hwy           …                              28
4  1 Anzac Hwy           …                              41

[5 rows x 6 columns]
```

[17]:
```
st_week_grp1 = pd.DataFrame(st_week_grp.groupby('StopName')["WeekBeginning"].
↪count()).reset_index()
st_week_grp1.head()
```

[17]:
```
          StopName  WeekBeginning
0     1 Anzac Hwy              54
1    1 Bartels Rd              54
2    1 Botanic Rd              54
3      1 Frome Rd              54
4  1 Fullarton Rd              54
```

[18]:
```
#Gathering only the Stop Name which having all 54 weeks of Dat
aa = list(st_week_grp1[st_week_grp1['WeekBeginning'] == 54]['StopName'])
aa[1:10]
```

[18]:
```
['1 Bartels Rd',
 '1 Botanic Rd',
 '1 Frome Rd',
 '1 Fullarton Rd',
 '1 George St',
 '1 Glen Osmond Rd',
 '1 Goodwood Rd',
 '1 Henley Beach Rd',
```

```
 '1 Kensington Rd']
```

[19]:
```
bb = st_week_grp[st_week_grp['StopName'].isin(aa)]
bb.head()
bb.shape

type(bb)
```

[19]:
```
      StopName          …          NumberOfBoardings_max
0  1 Anzac Hwy          …                             51
1  1 Anzac Hwy          …                             28
2  1 Anzac Hwy          …                             45
3  1 Anzac Hwy          …                             28
4  1 Anzac Hwy          …                             41

[5 rows x 6 columns]
```

[19]:
```
(175446, 6)
```

[19]:
```
pandas.core.frame.DataFrame
```

[20]:
```
#removing the stoppage which are not having the data of whole 54 weeks
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)
```

[20]:
```
(10567931, 11)

data without stopage removing:  (10857234, 11)
data, after removing stoppage not having the data of whole 54 weeks:  (10567931,
11)
```

[21]:
```
new_data.head(2)
filtered_data = new_data[new_data['dist_from_centre'] <= 100]
filtered_data.shape
```

[21]:
```
   TripID RouteID       …                  type dist_from_centre
0   23631     100       …       street_address         5.180961
1   23631     100       …       street_address         5.172525

[2 rows x 11 columns]
```

[21]:
```
(10341468, 11)
```

[22]:
```
data = filtered_data.copy()
data.shape
```

[22]:
```
(10341468, 11)
```

[23]:
```
#No of boarding for each stopage in all weeks
#bb["StopName"].groupby(NumberOfBoardings_sum)
stopageName_with_boarding = bb.groupby(['StopName']).
 ↪agg({'NumberOfBoardings_sum': ['sum']})

#stopageName_with_boarding.columns = ["_".join(x) for x in␣
 ↪stopageName_with_boarding.columns.ravel()]
#stopageName_with_boarding.head()
stopageName_with_boarding = pd.DataFrame(stopageName_with_boarding.
 ↪reset_index())
```

[24]:
```
#type(stopageName_with_boarding)
stopageName_with_boarding.columns = ["StopName",␣
 ↪"Total_boarding_on_the_stopage"]
#stopageName_with_boarding.shape
stopageName_with_boarding.head()
```

[24]:

	StopName	Total_boarding_on_the_stopage
0	1 Anzac Hwy	39429
1	1 Bartels Rd	8412
2	1 Botanic Rd	14868
3	1 Frome Rd	67458
4	1 Fullarton Rd	585

[25]:
```
## save the aggregate data
#bb.to_csv('st_week_grp.csv', index=False)
```

0.3 Data Exploration

[26]:
```
data.nunique()
#data.isnull().sum()
#data['WeekBeginning'].unique()
```

[26]:
```
TripID              39211
RouteID               616
StopID               5838
StopName             3127
WeekBeginning          54
NumberOfBoardings     359
latitude             2393
longitude            2379
```

```
postcode               138
type                     8
dist_from_centre      2397
dtype: int64
```

0.4 Data Visualization

[27]:
```
##can assign the each chart to one axes at a time
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])
```

[27]: Text(0.5,1,'No of Boardings')

[27]: <matplotlib.axes._subplots.AxesSubplot at 0x7c5bd6604ba8>

[27]: Text(0.5,1,'WeekBeginning')

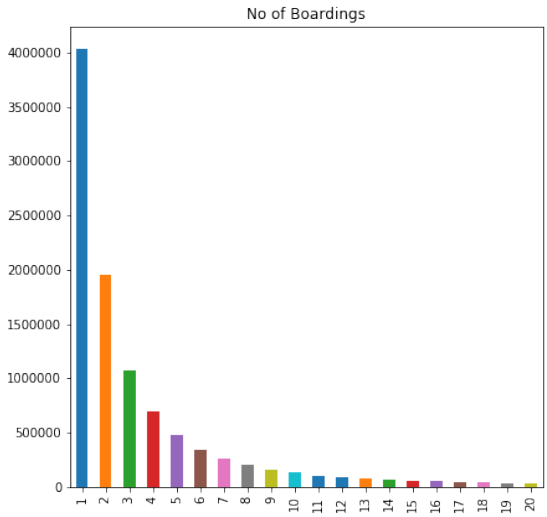
[27]: <matplotlib.axes._subplots.AxesSubplot at 0x7c5a5f56ab00>

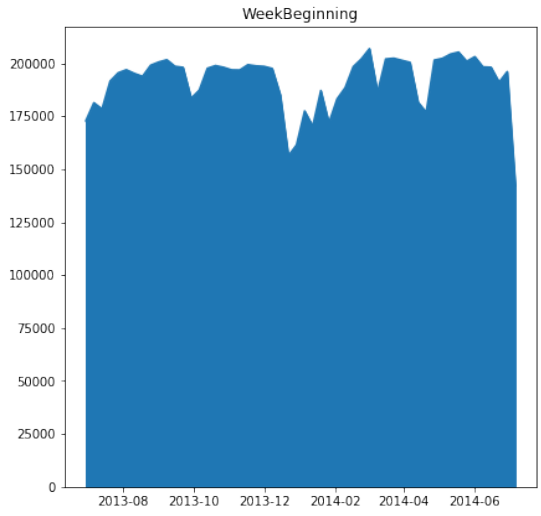
[27]: Text(0.5,1,'most Busiest Route')

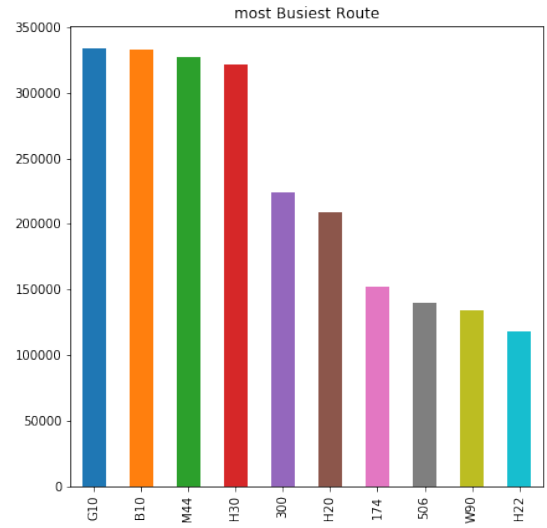
[27]: <matplotlib.axes._subplots.AxesSubplot at 0x7c5a5f548dd8>

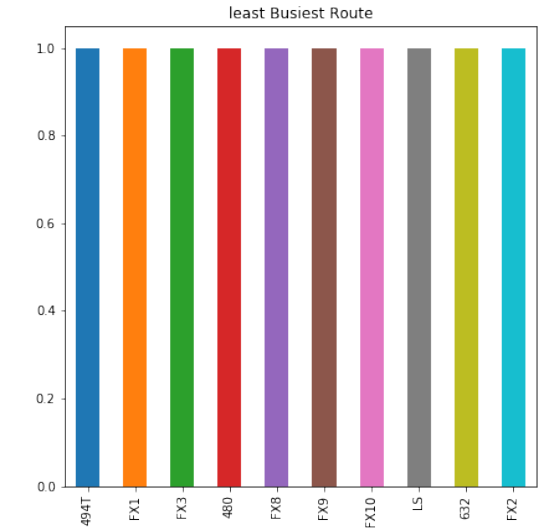
[27]: Text(0.5,1,'least Busiest Route')

[27]: <matplotlib.axes._subplots.AxesSubplot at 0x7c5a8c6a1048>









[28]:
```
stopageName_with_boarding = stopageName_with_boarding.
  ↪sort_values('Total_boarding_on_the_stopage', ascending = False)
#stopage with most no of boarding
stopageName_with_boarding.head(10)
```

[28]:

	StopName	Total_boarding_on_the_stopage
3054	I2 North Tce	628859
3125	X1 King William St	622099
3032	F2 Grenfell St	604149
3130	X2 King William St	583227
3021	E1 Currie St	550396
3207	Zone C Paradise Interchange	547709
3015	D1 King William St	541046
3211	Zone C Tea Tree Plaza Intercha	451960

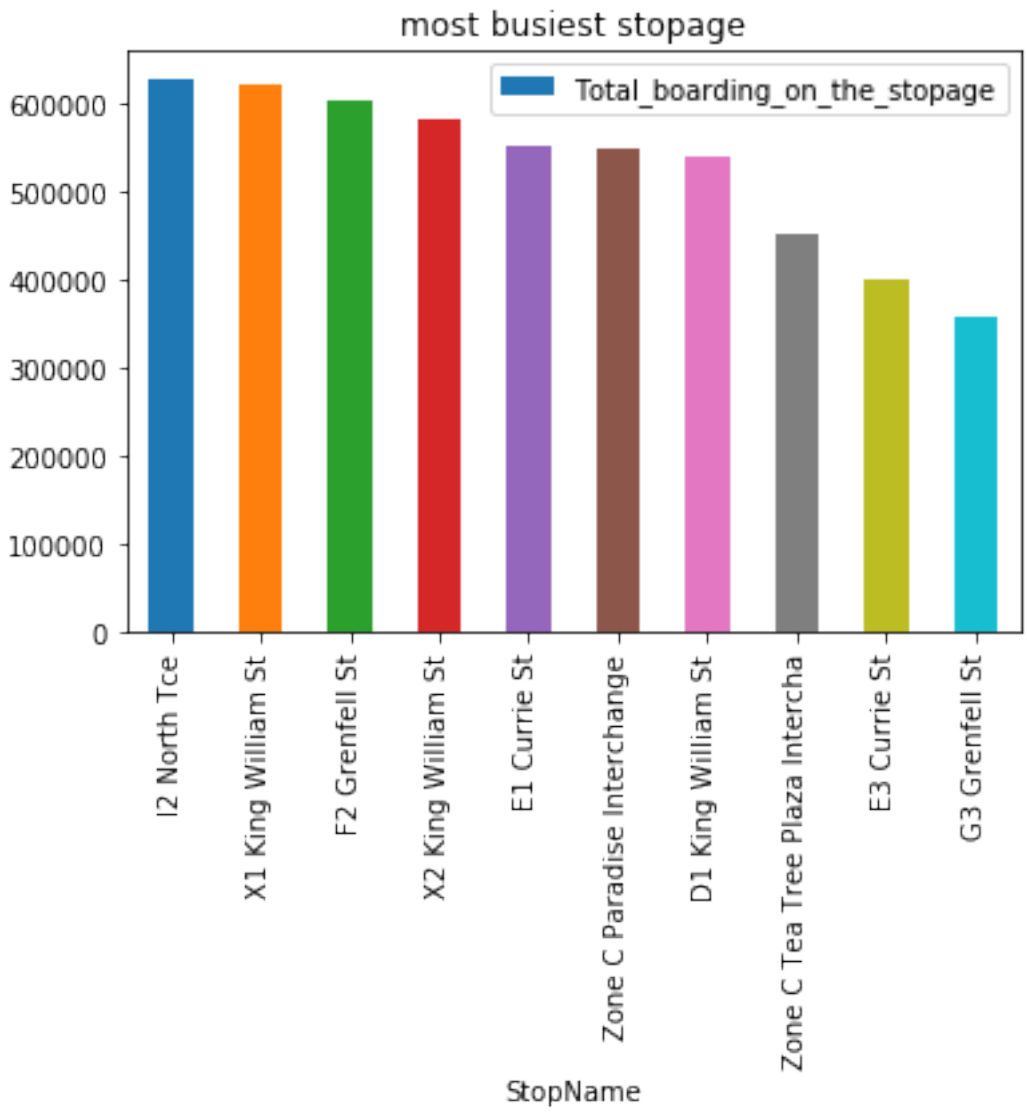
```
3025                    E3 Currie St                          399351
3039                  G3 Grenfell St                          356518
```

[29]:
```
#stopage with least no of boarding
stopageName_with_boarding.tail(10)
```

[29]:
```
                    StopName  Total_boarding_on_the_stopage
1845         45 Mcintyre Rd                             292
2318          57 Philip Hwy                             281
2732           75B Frick St                             275
58            109 Regency Rd                            274
1633         39D Glenloth Dr                            266
170           127 Lyndoch Rd                            264
3086  Strathalbyn South Tce                             227
1231          31 Glenroy St                             221
558             19 Gilles Rd                            215
294        145 The Esplanade                              175
```

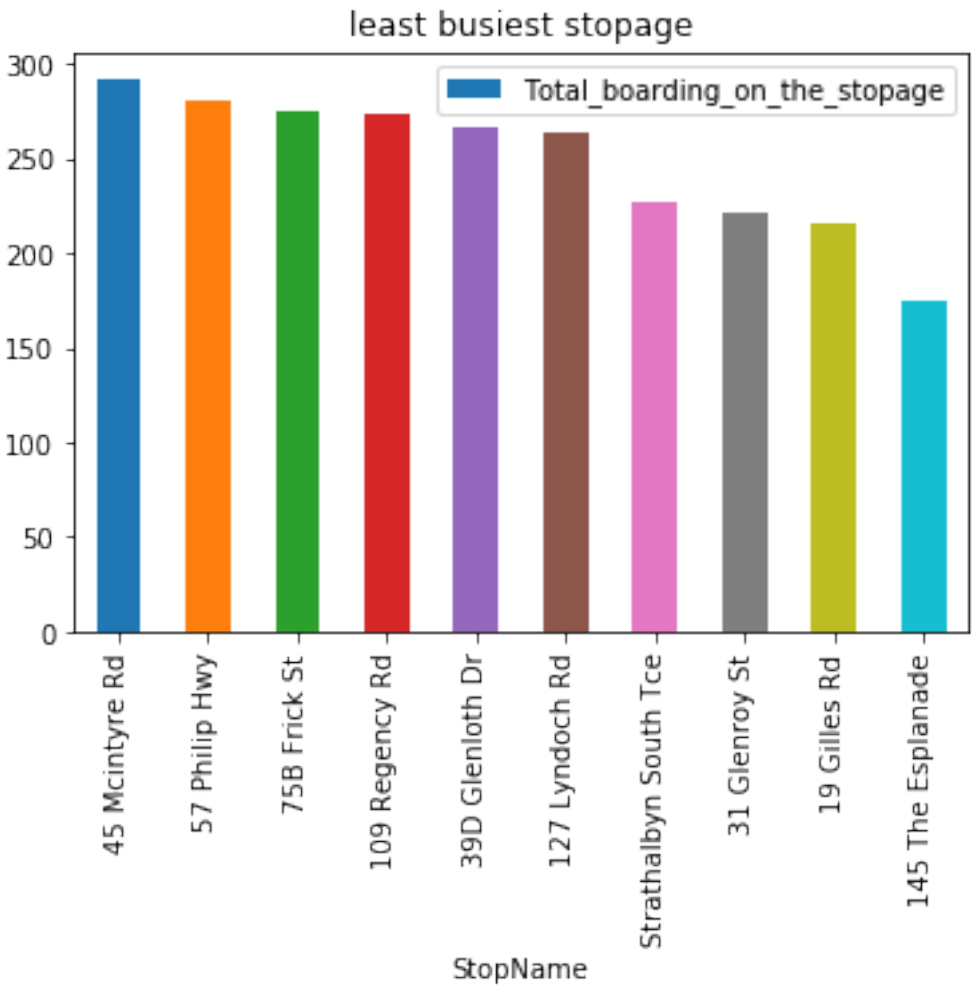
[30]:
```
ax = stopageName_with_boarding.head(10).plot.bar(x='StopName',␣
↪y='Total_boarding_on_the_stopage', rot=90)
ax.set_title("most busiest stopage")
```

[30]:
```
Text(0.5,1,'most busiest stopage')
```



[31]:
```
ax = stopageName_with_boarding.tail(10).plot.bar(x='StopName',␣
↪y='Total_boarding_on_the_stopage', rot=90)
ax.set_title("least busiest stopage")
```

[31]: Text(0.5,1,'least busiest stopage')



[32]:
```
data['WeekBeginning'].value_counts().mean()
```

[32]:
```
191508.66666666666
```

[33]:
```
# data['dist_from_centre'].nunique()
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()
```

[33]:

	dist_from_centre	NumberOfBoardings
0	0.000018	1892435
1	0.131368	167535
2	0.309089	356518

```
3          0.314937            1484824
4          0.326005             120061
```

[33]:
```
Index(['dist_from_centre', 'NumberOfBoardings'], dtype='object')
```

[33]:
```
      dist_from_centre  NumberOfBoardings
2392         86.471064              18905
2393         94.826409                321
2394         99.625655               1101
2395         99.665190               4373
2396         99.748995              21216
```

[34]:
```
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)
```

[35]:
```
#clustering Technique// based on the distance from city centre

x = data["dist_from_centre"]
distance_10 = []
distance_10_50 = []
distance_50_100 = []
#distance_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
    #elif(i>100 and i< 2000):
        #distance_100_more.append(i)
        #total += 1
    #elif(i>2000):
        #outlier.append(i)
        #outlier_ += 1
```

[36]:
```
print(outlier_)
```

```
0
```

[37]:
```
y = len(distance_10)+len(distance_10_50)+len(distance_50_100)
#+len(distance_100_more)
#print(y)
#print(total)
```

[38]:
```
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)
#print("passanger, boarding the buses from the distance of 100Km and more from␣
 ↪the city center = ", (len(distance_100_more)/total)*100)
```

```
10341468
passangers, boarding the buses in the radious of 10Km from the city center =
64.31275521038212
passanger, boarding the buses from the distance of 10Km to 50Km from the city
center =  33.16731241638035
passanger, boarding the buses from the distance of 50Km to 100 from the city
center =  2.5199323732375327
```

[39]:
```
#busiest route on weekly basis
#data.head(10)
# st_week_grp1 = pd.DataFrame(data.groupby(['StopName','WeekBeginning','type']).
 ↪agg({'NumberOfBoardings': ['sum', 'count']})).reset_index()
grouped_route = data.groupby(['RouteID']).agg({'NumberOfBoardings': ['sum',␣
 ↪'max']})
grouped_route.columns = ["_".join(x) for x in grouped_route.columns.ravel()]
```

[40]:
```
"""grouped_route = grouped_route.head().reset_index()
type(grouped_route)
grouped_route = grouped_route.sort_values("NumberOfBoardings_sum", ascending =␣
 ↪True)
```

```
#stopageName_with_boarding = stopageName_with_boarding.
 ↪sort_values('Total_boarding_on_the_stopage', ascending = False)
#stopage with most no of boarding
#stopageName_with_boarding.head(10)
#grouped_route["NumberOfBoardings_sum"] =␣
 ↪grouped_route["NumberOfBoardings_sum"] / 365
grouped_route.head(10)
grouped_route.shape"""
```

[40]:
```
'grouped_route =
grouped_route.head().reset_index()\ntype(grouped_route)\ngrouped_route =
grouped_route.sort_values("NumberOfBoardings_sum", ascending =
True)\n#stopageName_with_boarding =
stopageName_with_boarding.sort_values(\'Total_boarding_on_the_stopage\',
ascending = False)\n#stopage with most no of boarding\n#stopageName_with_boardin
g.head(10)\n#grouped_route["NumberOfBoardings_sum"] =
grouped_route["NumberOfBoardings_sum"] /
365\ngrouped_route.head(10)\ngrouped_route.shape'
```

....

[41]:
```
"""route_data = grouped_route[grouped_route['RouteID'] == "G10"]
route_data.head()"""
```

[41]:
```
'route_data = grouped_route[grouped_route[\'RouteID\'] ==
"G10"]\nroute_data.head()'
```

[ ]:
```

```