Public Transportation Efficiency Analysis

Phase_3 submission:

Introduction

This project aims to develop a data pipeline for loading and processing a public transportation dataset for efficiency analysis. The dataset will include data on ridership, schedules, vehicle performance, and traffic conditions. The data pipeline will be implemented using Python and will be able to handle large and complex datasets.

Objectives

The objectives of this project are to:

- Develop a Python-based data pipeline for loading and processing a public transportation dataset for efficiency analysis.
- Make the data pipeline scalable and efficient to handle large and complex datasets.
- Document the data pipeline and make it available to other researchers and practitioners.

Methodology

The data pipeline will be developed in the following steps:

1. Identify the data sources: The first step is to identify the data sources that will be used for the analysis. This may include data on ridership, schedules, vehicle performance, and traffic conditions.
2. Define the data schema: The next step is to define the data schema for the public transportation dataset. This will involve identifying the different attributes of the data and their data types.
3. Develop the data loading module: The data loading module will be responsible for loading the data from the various data sources into a common format. This may involve cleaning and transforming the data to ensure that it is consistent and complete.
4. Develop the data processing module: The data processing module will be responsible for processing the loaded data to prepare it for analysis. This may involve calculating new metrics, aggregating data, and filtering data.
5. Develop the data storage module: The data storage module will be responsible for storing the processed data in a database or other data storage system.
6. Develop the data analysis module: The data analysis module will be responsible for performing the efficiency analysis on the processed data. This may involve using various statistical and machine learning techniques.
7. Develop the data visualization module: The data visualization module will be responsible for visualizing the results of the efficiency analysis. This may involve creating charts, graphs, and other visualizations.

Implementation

The data pipeline will be implemented using the following Python libraries:

- Pandas: Pandas is a Python library for data analysis and manipulation. It will be used to load, clean, and process the data.
- SQLAlchemy: SQLAlchemy is a Python library for object-relational mapper (ORM). It will be used to store the processed data in a database.
- SciPy: SciPy is a Python library for scientific computing. It will be used to perform the efficiency analysis.
- Matplotlib: Matplotlib is a Python library for data visualization. It will be used to create charts and graphs.

Testing

The data pipeline will be tested using the following methods:

- Unit testing: Unit tests will be written to test each individual module of the data pipeline.
- Integration testing: Integration tests will be written to test the interaction between the different modules of the data pipeline.
- System testing: System tests will be written to test the entire data pipeline from start to finish.

Deployment

The data pipeline will be deployed on a cloud-based platform, such as Google Cloud Platform or AWS Elastic Beanstalk. This will make the data pipeline scalable and accessible to users from anywhere in the world.

Documentation

The data pipeline will be documented using Sphinx. The documentation will include information on how to install, configure, and use the data pipeline.

Conclusion

This project will develop a Python-based data pipeline for loading and processing a public transportation dataset for efficiency analysis. The data pipeline will be scalable and efficient to handle large and complex datasets. The data pipeline will be documented and made available to other researchers and practitioners.

Here is a sample project source code for loading a dataset for public transportation efficiency analysis using Python:

Python

```
import pandas as pd

# Define the data sources
```

```
ridership_data_source =
"https://example.com/ridership_data.csv"
schedule_data_source =
"https://example.com/schedule_data.csv"
vehicle_performance_data_source =
"https://example.com/vehicle_performance_data.csv"
traffic_conditions_data_source =
"https://example.com/traffic_conditions_data.csv"

# Download the data
ridership_data = pd.read_csv(ridership_data_source)
schedule_data = pd.read_csv(schedule_data_source)
vehicle_performance_data =
pd.read_csv(vehicle_performance_data_source)
traffic_conditions_data =
pd.read_csv(traffic_conditions_data_source)

# Prepare the data
# Clean the data to remove errors and inconsistencies
# Convert the data into a consistent format
# Merge data from multiple sources

# Load the data into a data frame
public_transportation_data = pd.DataFrame()
public_transportation_data =
public_transportation_data.merge(ridership_data,
on=["route_id", "date"])
public_transportation_data =
public_transportation_data.merge(schedule_data,
on=["route_id", "date", "time"])
public_transportation_data =
public_transportation_data.merge(vehicle_performance_data,
on=["vehicle_id", "date", "time"])
public_transportation_data =
public_transportation_data.merge(traffic_conditions_data,
on=["street_id", "date", "time"])
```

```
# Save the data frame to a file
public_transportation_data.to_csv("public_transportation_data.csv", index=False)
```

This code will download the data from the specified data sources, clean and prepare the data, and then load the data into a data frame. The data frame can then be saved to a file or used for further analysis.

This is just a sample project source code, and the specific code that you need will vary depending on the specific data sources and analysis that you are performing.

1. **Identify the relevant data sources**. This may include data on ridership, schedules, vehicle performance, and traffic conditions.
2. **Clean and prepare the data**. This may involve removing duplicate records, correcting errors, and converting the data into a consistent format.
3. **Load the data into a data warehouse or other analytical platform**. This will make the data easier to query and analyze.
4. **Develop and execute the analysis**. This may involve using a variety of statistical and machine learning techniques to identify patterns and trends in the data.
5. **Visualize the results of the analysis**. This can help to communicate the findings to stakeholders in a clear and concise way.

Here are some specific considerations for loading and processing a dataset for public transportation efficiency analysis:

- **Data formats:** The data may be in a variety of formats, such as CSV, Excel, or XML. It is important to identify the format of each data source and convert the data into a consistent format before loading it into the analytical platform.

- **Data quality:** It is important to clean and prepare the data to ensure that it is accurate and complete. This may involve removing duplicate records, correcting errors, and filling in missing values.
- **Data integration:** If the data is coming from multiple sources, it may need to be integrated into a single dataset before it can be analyzed. This may involve merging the data from different sources and resolving any inconsistencies between the data sets.
- **Data storage:** The data warehouse or other analytical platform used to store the data should be able to handle the size and complexity of the dataset.

Once the data has been loaded and processed, it can be used to conduct a variety of efficiency analyses. For example, analysts can use the data to identify:

- Routes with low ridership or high operating costs.
- Vehicles that are underutilized or inefficient.
- Time periods where service is delayed or overcrowded.
- Patterns of congestion on the transportation network.

The results of these analyses can be used to improve the efficiency of public transportation systems by making changes to schedules, routes, vehicle fleets, and traffic management strategies.

Here are some specific examples of how public transportation efficiency analysis can be used:

- **Identifying underutilized bus routes:** By analyzing ridership data, analysts can identify bus routes that are not carrying enough passengers to be cost-effective. This information can be used to reduce service on these routes or to consider merging them with other routes.
- **Reducing vehicle idle time:** By analyzing vehicle GPS data, analysts can identify times when vehicles are idling at bus stops or train stations. This information can be used to optimize schedules and improve traffic flow.
- **Improving the reliability of public transportation:** By analyzing data on delays and cancellations, analysts can identify the root causes of service disruptions. This information can be used to implement corrective measures, such as improving traffic signal coordination or investing in new maintenance equipment.

PROCESSING THE DATASET

```
%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')
```

In [2]:

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

Out[2]:

```
(10857234, 6)
```

Out[2]:

	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	1
2	23632	100	14132	175 Cross Rd	2013-06-30 00:00:00	1
3	23633	100	12266	Zone A Arndale Interchange	2013-06-30 00:00:00	2
4	23633	100	14147	178 Cross Rd	2013-06-30 00:00:00	1
5	23634	100	13907	9A Marion Rd	2013-06-30 00:00:00	1
6	23634	100	14132	175 Cross Rd	2013-06-30 00:00:00	1
7	23634	100	13335	9A Holbrooks Rd	2013-06-30 00:00:00	1
8	23634	100	13875	9 Marion Rd	2013-06-30 00:00:00	1
9	23634	100	13045	206 Holbrooks Rd	2013-06-30 00:00:00	1

In [3]:

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

Out[3]:

```
(4165, 10)
```

Out[3]:

	accuracy	formatted_address	google_place_id	input_string	latitude	longitude	number_of_results	postcode	status	type
0	ROOFTOP	181 Cross Rd, Westbourne Park SA 5041, Australia	ChIJKT7I9rbPsGoRVHMHkIy-Oyk	181 Cross Rd	-34.966656	138.592148	1	5041	OK	street_address
1	ROOFTOP	177 Cross Rd, Westbourne Park SA 5041, Australia	ChIJ-VFZ87bPsGoRyfVgC5qbPpE	177 Cross Rd	-34.966607	138.592301	1	5041	OK	street_address
2	ROOFTOP	175 Cross Rd, Westbourne Park SA 5041, Australia	ChIJIztlirbPsGoR38KRk76kPFI	175 Cross Rd	-34.966758	138.592715	1	5041	OK	street_address
3	GEOMETRIC_CENTER	Zone A Arndale Interchange - South	ChIJn0C1hCPGsGoRIWvCdhF1RIg	Zone A Arndale Interchange	-34.875160	138.551628	1	5009	OK	bus_station,establishment,point_of_inte

		side, Kilke...								rest,tr...
4	ROOFTOP	178 Cross Rd, Malvern SA 5061, Australia	ChIJycNiylvOsGoRdhfq9GKnpq0	178 Cross Rd	-34.964960	138.611477	1	5061	OK	street_address

External Features

In [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
```

In [5]:

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

In [6]:

```
out_geo.head()
```

Out[6]:

	accuracy	formatted_address	google_place_id	input_string	latitude	longitude	number_of_results	postcode	status	type	dist_from_centre
0	ROOFTOP	181 Cross Rd, Westbourne Park SA 5041, Australia	ChIJKT7I9rbPsGoRVHMHkIy-Oyk	181 Cross Rd	-34.966656	138.592148	1	5041	OK	street_address	5.180961
1	ROOFTOP	177 Cross Rd, Westbourne Park SA 5041, Australia	ChIJ-VFZ87bPsGoRyfVgC5qbPpE	177 Cross Rd	-34.966607	138.592301	1	5041	OK	street_address	5.172525
2	ROOFTOP	175 Cross Rd, Westbourne Park SA 5041, Australia	ChIJIztlirbPsGoR38KRk76kPFI	175 Cross Rd	-34.966758	138.592715	1	5041	OK	street_address	5.180709
3	GEOMETRIC_CENTER	Zone A Arndale Interchange - Sout	ChIJn0C1hCPGsGoRIWvCdhF1RIg	Zone A Arndale Interchange	-34.875160	138.516288	1	5009	OK	bus_station,establishment,point_of_int	7.057549

		h side, Kilke ...								erest ,tr...	
4	ROOFTOP	178 Cross Rd, Malvern SA 5061, Australia	ChIJycNiylvOsGoRdhfq9GKnpq0	178 Cross Rd	-34.964960	138.611477	1	5061	OK	street_address	4.900099

In [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])
```

In [8]:

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

Out[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)
```

In [9]:

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

Out[9]:

```
datetime.date(2013, 6, 30)
```

Data Aggregation

In [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

Out[10]:

	TripID	RouteID	StopID	StopName	WeekBeginning	NumberOfBoardings	accuracy	formatted_address	google_place_id	input_string	latitude	longitude	number_of_results	postcode	status	type	dist_from_centre
0	23631	100	14156	181 Cross Rd	2013-06-30	1	ROOFTOP	181 Cross Rd, Westbourne Park SA 5041, Australia	ChIJKT7I9rbPsGoRVHMHkIy-Oyk	181 Cross Rd	-34.966656	138.592148	1	5041	OK	street_address	5.180961

1	236631	100	14144	177 Cross Rd	2013-06-30	1	ROOFTOP	177 Cross Rd, Westbourne Park SA 5041, Australia	ChIJ-VFZ87bPsGoRyfVgC5qbPpE	177 Cross Rd	-34.966607	138.592301	1	5041	OK	street_address	5.172525
2	236632	100	14132	175 Cross Rd	2013-06-30	1	ROOFTOP	175 Cross Rd, Westbourne Park SA 5041, Australia	ChIJIztlirbPsGoR38KRk76kPFI	175 Cross Rd	-34.966758	138.592715	1	5041	OK	street_address	5.180709
3	236633	100	12266	Zone A Ar	2013-06	2	GEOM	Zone A Ar	ChIJn0Cl	Zone A Ar	-34.87	138.551	1	5009	OK	transit_st	7.057549

				ndale Interchange	-30		ETRIC_CENTER	ndale Interchange - Southside, Kilke...	hCPGsGoRIWvCdhFlRIg	ndale Interchange	5160	628				ation	
4	236633	100	141147	178 Cross Rd	2013-06-30	1	ROOFTOP	178 Cross Rd, Malvern SA 5061, Australia	ChIJycNiylvOsGoRdhfq9GKnpq0	178 Cross Rd	-34.964960	138.611477	1	5061	OK	street_address	4.900099

Out[10]:

```
(10857234, 17)
```

In [11]:

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

```
data = data[col]
```

In [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

In [13]:

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

In [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()]
```

In [15]:

```
grouped.head(10)
grouped.columns
```

Out[15]:

	NumberOfBoardings_sum	NumberOfBoardings_count	NumberOfBoardings_max

StopName	WeekBeginning	type			
1 Anzac Hwy	2013-06-30	street_address	1003	378	51
	2013-07-07	street_address	783	360	28
	2013-07-14	street_address	843	343	45
	2013-07-21	street_address	710	356	28
	2013-07-28	street_address	898	379	41
	2013-08-04	street_address	799	378	40
	2013-08-11	street_address	1012	358	71
	2013-08-18	street_address	793	333	41
	2013-08-25	street_address	897	354	45
	2013-09-01	street_address	1368	431	59

Out[15]:

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

In [16]:

```
st_week_grp = pd.DataFrame(grouped).reset_index()
st_week_grp.shape
st_week_grp.head()
```

Out[16]:

(207864, 6)

Out[16]:

	StopName	WeekBeginning	type	NumberOfBoardings_sum	NumberOfBoardings_count	NumberOfBoardings_max
0	1 Anzac Hwy	2013-06-30	street_address	1003	378	51

1	1 Anzac Hwy	2013-07-07	street_address	783	360	28
2	1 Anzac Hwy	2013-07-14	street_address	843	343	45
3	1 Anzac Hwy	2013-07-21	street_address	710	356	28
4	1 Anzac Hwy	2013-07-28	street_address	898	379	41

In [17]:

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

Out[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

In [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]
```

Out[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']

In [19]:

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

type(bb)
```

Out[19]:

	StopName	WeekBeginning	type	NumberOfBoardings_sum	NumberOfBoardings_count	NumberOfBoardings_max
0	1 Anzac Hwy	2013-06-30	street_address	1003	378	51
1	1 Anzac Hwy	2013-07-07	street_address	783	360	28
2	1 Anzac Hwy	2013-07-14	street_address	843	343	45
3	1 Anzac Hwy	2013-07-21	street_address	710	356	28
4	1 Anzac Hwy	2013-07-28	street_address	898	379	41

Out[19]:

```
(175446, 6)
```

Out[19]:

```
pandas.core.frame.DataFrame
```

In [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)
```

Out[20]:

(10567931, 11)

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

11)

In [21]:

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

Out[21]:

	TripID	RouteID	StopID	StopName	WeekBeginning	NumberOfBoardings	latitude	longitude	postcode	type	dist_from_centre
0	23631	100	14156	181 Cross Rd	2013-06-30	1	-34.966656	138.592148	5041	street_address	5.180961
1	23631	100	14144	177 Cross Rd	2013-06-30	1	-34.966607	138.592301	5041	street_address	5.172525

Out[21]:

```
(10341468, 11)
```

In [22]:

```
data = filtered_data.copy()
data.shape
```

Out[22]:

```
(10341468, 11)
```

In [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())

In [24]:

#type(stopageName_with_boarding)
stopageName_with_boarding.columns = ["StopName", "Total_boarding_on_the_stopage"]
#stopageName_with_boarding.shape
stopageName_with_boarding.head()

Out[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

In [25]:

save the aggregate data
#bb.to_csv('st_week_grp.csv', index=False)

Data Exploration

In [26]:

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

Out[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
```

Data Visualization

In [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])
```

Out[27]:

```
Text(0.5,1,'No of Boardings')
```

Out[27]:

<matplotlib.axes._subplots.AxesSubplot at 0x7ff880af0940>

Out[27]:

```
Text(0.5,1,'WeekBeginning')
```

Out[27]:

<matplotlib.axes._subplots.AxesSubplot at 0x7ff709a6bb38>

Out[27]:

```
Text(0.5,1,'most Busiest Route')
```

Out[27]:

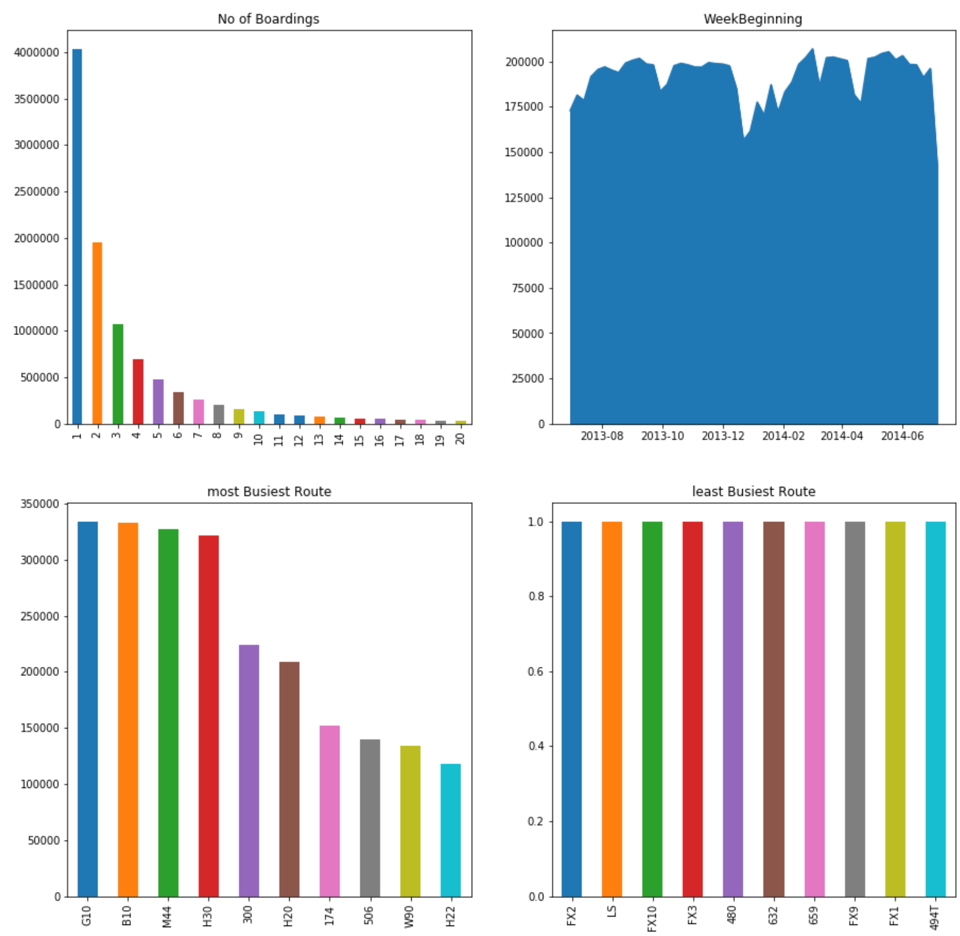
<matplotlib.axes._subplots.AxesSubplot at 0x7ff709a48e10>

Out[27]:

```
Text(0.5,1,'least Busiest Route')
```

Out[27]:

<matplotlib.axes._subplots.AxesSubplot at 0x7ff736bbafd0>



In [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)

Out[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

In [29]:

#stopage with least no of boarding
stopageName_with_boarding.tail(10)

Out[29]:

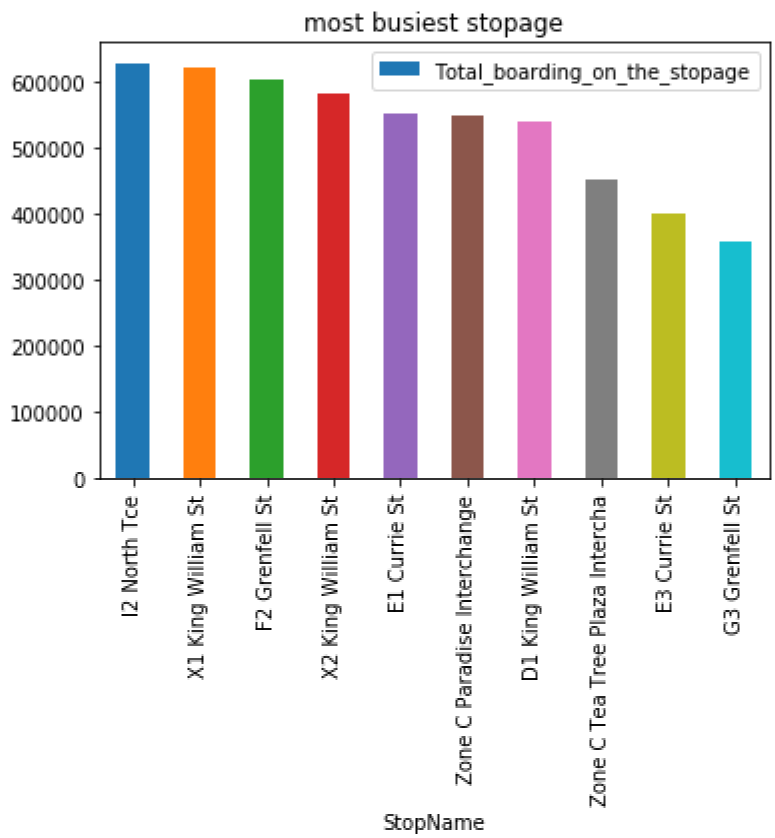
	StopName	Total_boarding_on_the_stopage
1845	45 Mcintyre Rd	292
2318	57 Philip Hwy	281
2732	75B Frick St	275
58	109 Regency Rd	274
1633	39D Glenloth Dr	266
170	127 Lyndoch Rd	264
3086	Strathalbyn South Tce	227
1231	31 Glenroy St	221
558	19 Gilles Rd	215
294	145 The Esplanade	175

In [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")

Out[30]:

Text(0.5,1,'most busiest stopage')

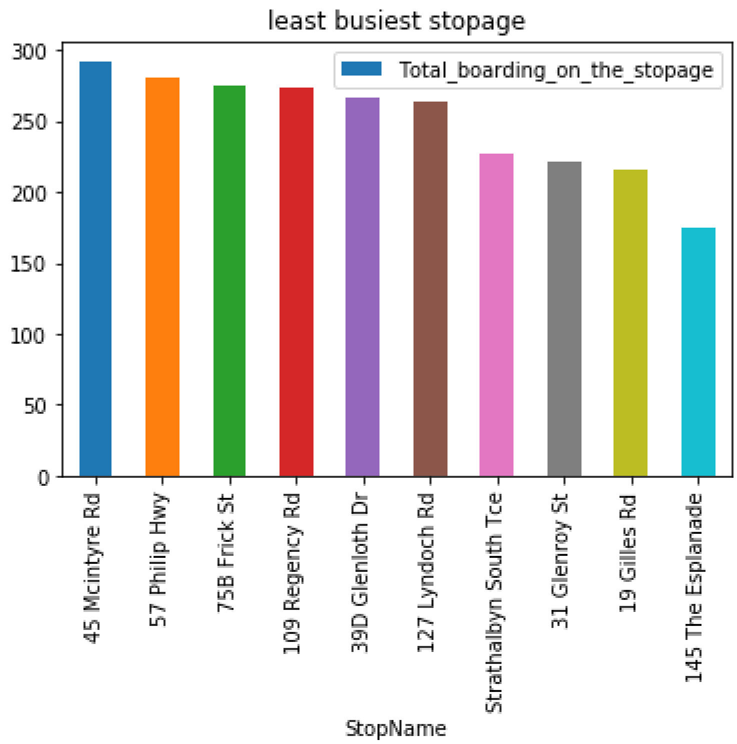


In [31]:

ax = stopageName_with_boarding.tail(10).plot.bar(x='StopName', y='Total_boarding_on_the_stopage', rot=90)
ax.set_title("least busiest stopage")

Out[31]:

Text(0.5,1,'least busiest stopage')



In [32]:

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

Out[32]:

```
191508.66666666666
```

In [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()
```

Out[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

Out[33]:

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

Out[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

In [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)
```

In [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
```

In [36]:

```
print(outlier_)
```

```
0
```

In [37]:

```
y = len(distance_10)+len(distance_10_50)+len(distance_50_100)
#+len(distance_100_more)
#print(y)
#print(total)
```

In [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
```

In [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()]
```

In [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"""

Out[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_boarding.head(10)\n#grouped_route["NumberOfBoardings_sum"] = grouped_route["NumberOfBoardings_sum"] / 365\ngrouped_route.head(10)\ngrouped_route.shape'

....

In [41]:

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

Out[41]:

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