**PHASE 3 SUBMISSION**

**PUBLIC TRANSPORTATION EFFICIENCY ANALYSIS**

**IMPLEMENTATION :**

**LOADING AND PREPROCESSING THE DATA :**

%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

data = pd.read\_csv('../input/unisys/ptsboardingsummary/20140711.CSV')

data.shape

data.head(10)

**OUTPUT :**

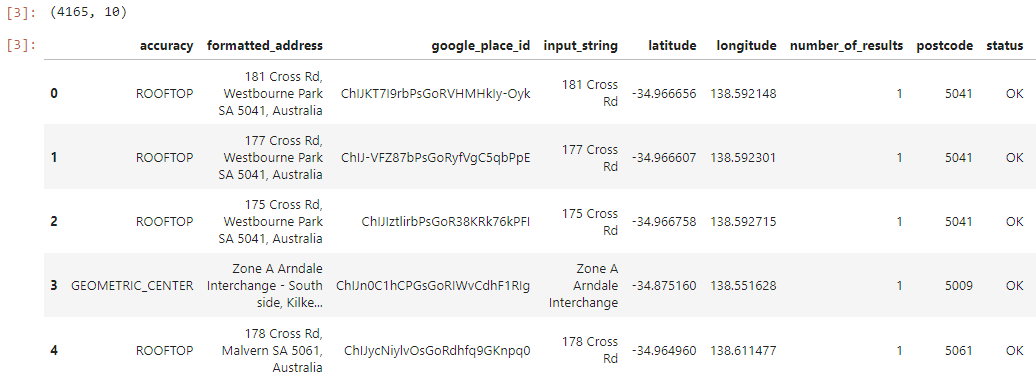


out\_geo = pd.read\_csv('../input/outgeo/output\_geo.csv')

out\_geo.shape

out\_geo.head()

**OUTPUT :**



#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)

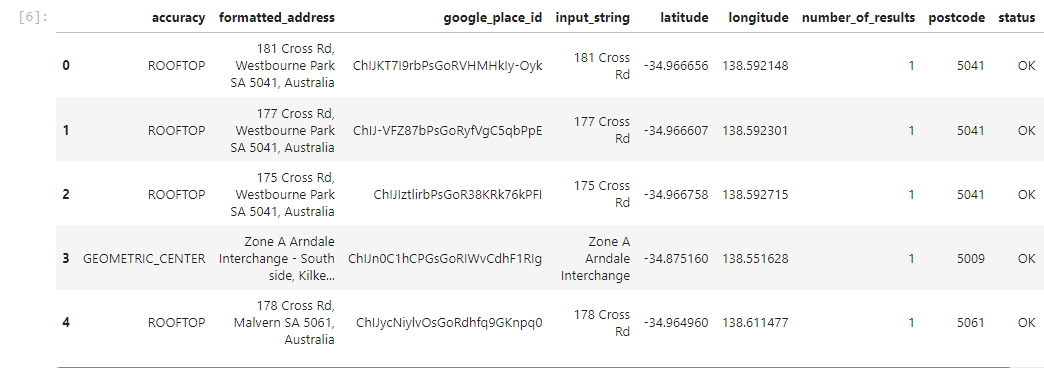
a = sin(dlat / 2)\*\*2 + cos(radians(lat1)) \* cos(radians(-34.921247)) \* sin(dlon / 2)\*\*2

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

return R \* c

out\_geo['dist\_from\_centre'] = out\_geo[['latitude','longitude']].apply(lambda x: calc\_dist(\*x), axis=1)

**OUTPUT :**



#exp\_data = out\_geo.head(10)

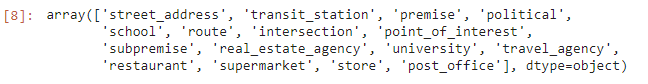
##Fill the missing values with mode

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

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

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

**OUTPUT :**



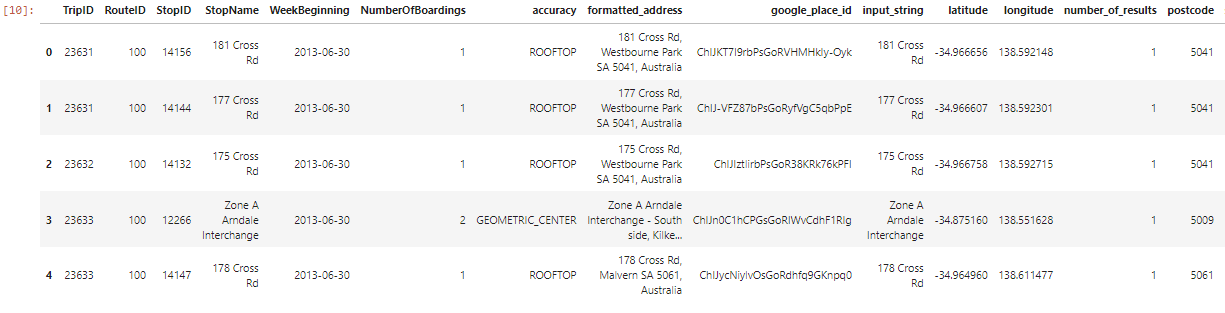
#Combine the Geolocation and main input file to get final Output File.

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

data.head(5)

data.shape

**OUTPUT :**





#Columns to keep for further analysis

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()

# st\_week\_grp1 = pd.DataFrame(data.groupby(['StopName','WeekBeginning','type']).agg({'NumberOfBoardings': ['sum', 'count']})).reset\_index()

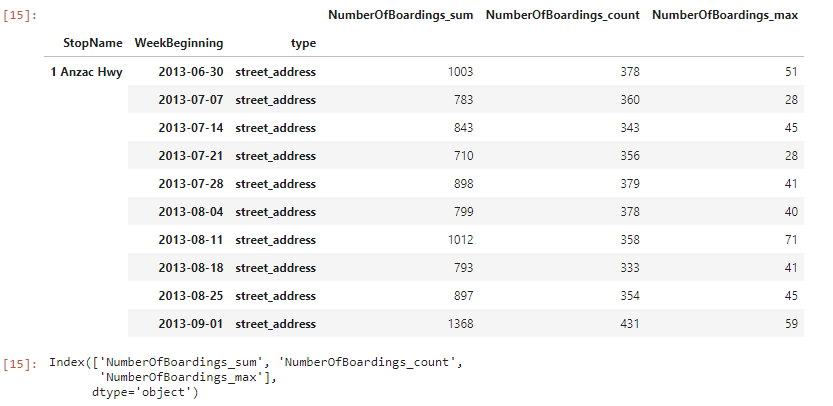
grouped = data.groupby(['StopName','WeekBeginning','type']).agg({'NumberOfBoardings': ['sum', 'count','max']})

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

grouped.head(10)

grouped.columns

**OUTPUT :**

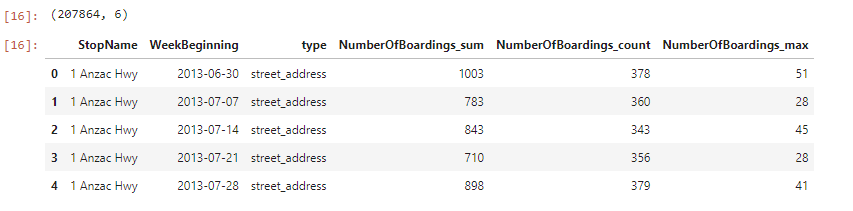


st\_week\_grp = pd.DataFrame(grouped).reset\_index()

st\_week\_grp.shape

st\_week\_grp.head()

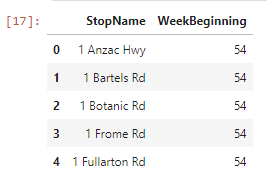
**OUTPUT :**



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

st\_week\_grp1.head()

**OUTPUT :**

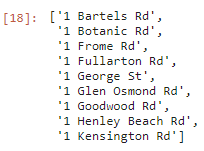


#Gathering only the Stop Name which having all 54 weeks of Dat

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

Aa[1:10]

**OUTPUT :**

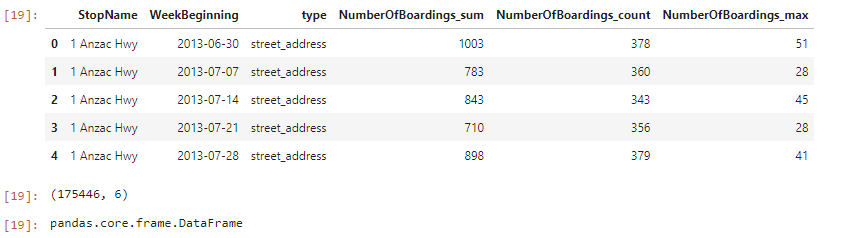


bb = st\_week\_grp[st\_week\_grp['StopName'].isin(aa)]

bb.head()

bb.shape

**OUTPUT :**



#removing the stoppage which are not having the data of whole 54 weeks

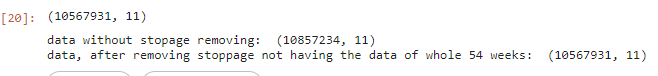
new\_data = data[data['StopName'].isin(aa)]

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)

**OUTPUT :**



new\_data.head(2)

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

filtered\_data.shape

**OUTPUT :**



data = filtered\_data.copy()

data.shape

**OUTPUT :**



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

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

#type(stopageName\_with\_boarding)

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

#stopageName\_with\_boarding.shape

stopageName\_with\_boarding.head()

**OUTPUT :**

