## **PUBLIC TRANSPORTATION ANALYSIS**

**PHASE 4**

## External Features

Some Important external data fields calculation

* ****IsHoliday**** Number of public holidays within that week
* ****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

In [8]:

from math import sin, cos, sqrt, atan2, radiansdef 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 [9]:

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

In [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 [11]:

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

Out[11]:

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)

Adding the details regarding the Public holidays from June 2013 to June 2014

In [12]:

*'''Holidays--2013-09-01,Father's Day2013-10-07,Labour day2013-12-25,Christmas day2013-12-26,Proclamation Day2014-01-01,New Year2014-01-27,Australia Day2014-03-10,March Public Holiday2014-04-18,Good Friday2014-04-19,Easter Saturday2014-04-21,Easter Monday2014-04-25,Anzac Day2014-06-09,Queen's Birthday'''*

Out[12]:

"Holidays--\n2013-09-01,Father's Day\n2013-10-07,Labour day\n2013-12-25,Christmas day\n2013-12-26,Proclamation Day\n2014-01-01,New Year\n2014-01-27,Australia Day\n2014-03-10,March Public Holiday\n2014-04-18,Good Friday\n2014-04-19,Easter Saturday\n2014-04-21,Easter Monday\n2014-04-25,Anzac Day\n2014-06-09,Queen's Birthday"

In [13]:

def holiday\_label (row):

if row == datetime.date(2013, 9, 1) :

return '1'

if row == datetime.date(2013, 10, 6) :

return '1'

if row == datetime.date(2013, 12, 22) :

return '2'

if row == datetime.date(2013, 12, 29):

return '1'

if row == datetime.date(2014, 1, 26):

return '1'

if row == datetime.date(2014, 3, 9):

return '1'

if row == datetime.date(2014, 4, 13) :

return '2'

if row == datetime.date(2014, 4, 20):

return '2'

if row == datetime.date(2014, 6, 8):

return '1'

return '0'

In [14]:

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

In [15]:

data['holiday\_label'] = data['WeekBeginning'].apply (lambda row: holiday\_label(row))

## Data Aggregation

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

In [16]:

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

In [17]:

data = pd.merge(data, route, how='left', left\_on = 'RouteID', right\_on = 'route\_id')

Columns to keep for further analysis

In [18]:

col = ['TripID', 'RouteID', 'StopID', 'StopName', 'WeekBeginning','NumberOfBoardings','formatted\_address',

'latitude', 'longitude','postcode','type','route\_desc','dist\_from\_centre','holiday\_label']

In [19]:

data = data[col]

In [20]:

*##saving the final dataset*data.to\_csv('Weekly\_Boarding.csv',index=False)

In [21]:

*## getting the addresses for geolocation api.# Address data['StopName'].unique()# sub = pd.DataFrame({'Address': Address})# sub=sub.reindex(columns=["Address"])# sub.to\_csv('addr.csv')*

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 [22]:

*# 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 [23]:

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

Out[23]:

(207864, 6)

Out[23]:

|  | 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 |

Gathering only the Stop Name which having all 54 weeks of Data

In [24]:

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

In [25]:

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

In [26]:

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

In [27]:

*## save the aggregate data*bb.to\_csv('st\_week\_grp.csv', index=False)