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(dl
on / 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 modeout_geo['type'].fillna('street_address',inpl
ace=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_la
bel(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 = 'inp
ut_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','NumberOf
Boardings','formatted_address',
    'latitude', 'longitude','postcode','type','route_desc','dist_from_centre','holid
ay_label']
```

In [19]:

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

In [20]:

```
##saving the final datasetdata.to_csv('Weekly_Boarding.csv',index=False)
```

In [21]:

```
## getting the addresses for geolocation api.# Address data['StopName'].uniq
ue()# 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({'NumberOfBoardi
ngs': ['sum', 'count','max']})grouped.columns = ["_".join(x) for x in grouped.c
olumns.ravel()]
```

In [23]:

```
st_week_grp = pd.DataFrame(grouped).reset_index()st_week_grp.shapest_w
eek_grp.head()
```

Out[23]:

(207864, 6)

Out[23]:

	Stop Name	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	2013-07-21	street_address	710	356	28

	Stop Name	WeekBeginning	type	NumberOfBoardings_sum	NumberOfBoardings_count	NumberOfBoardings_max
	Hwy					
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 databb.to_csv('st_week_grp.csv', index=False)
```