In [20]:
```
%matplotlib inline
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import plotly.graph_objs as go
from plotly import tools
import seaborn as sns
from plotly.offline import download_plotlyjs, init_notebook_mode, plot
, iplot
init_notebook_mode(connected=True)
```

Imports

In [269]:
```
#df = pd.read_csv('../input/201701-citibike-tripdata.csv')
df = pd.read_csv('/Users/jeffbloom13/Downloads/201703-citibike-tripdat
a.csv')

weatherDf = pd.read_csv('/Users/jeffbloom13/WPI/DS504/week-6/NYCWeathe
rClean.csv')
```

In [3]:
```
df.head(10)
```

Out[3]:

	Trip Duration	Start Time	Stop Time	Start Station ID	Start Station Name	Start Station Latitude	Start Station Longitude	End Station ID	End Station Name
0	1893	2017-03-01 00:00:32	2017-03-01 00:32:06	2009	Catherine St & Monroe St	40.711174	-73.996826	527	E 33 St & 2 Ave
1	223	2017-03-01 00:01:09	2017-03-01 00:04:53	127	Barrow St & Hudson St	40.731724	-74.006744	284	Greenwich Ave & 8 Ave
2	1665	2017-03-01 00:01:27	2017-03-01 00:29:12	174	E 25 St & 1 Ave	40.738177	-73.977387	307	Canal St & Rutgers St
3	100	2017-03-01 00:01:29	2017-03-01 00:03:10	316	Fulton St & William St	40.709560	-74.006536	306	Cliff St & Fulton St
4	1229	2017-03-01 00:01:33	2017-03-01 00:22:02	536	1 Ave & E 30 St	40.741444	-73.975361	259	South St & Whitehall St
5	613	2017-03-01 00:01:57	2017-03-01 00:12:11	259	South St & Whitehall St	40.701221	-74.012342	276	Duane St & Greenwich St
6	157	2017-03-01 00:02:12	2017-03-01 00:04:49	3329	Degraw St & Smith St	40.682915	-73.993182	3384	Smith St & 3 St
7	233	2017-03-01 00:02:15	2017-03-01 00:06:08	3107	Bedford Ave & Nassau Ave	40.723117	-73.952123	3090	N 8 St & Driggs Ave
8	317	2017-03-01 00:02:38	2017-03-01 00:07:55	3328	W 100 St & Manhattan Ave	40.795000	-73.964500	3285	W 87 St & Amsterdam Ave
9	2042	2017-03-01 00:02:54	2017-03-01 00:36:57	128	MacDougal St & Prince St	40.727103	-74.002971	3289	W 90 St & Amsterdam Ave

Clean data

In [22]:
```
df.isna().sum()
```

Out[22]:
```
Trip Duration                0
Start Time                   0
Stop Time                    0
Start Station ID             0
Start Station Name           0
Start Station Latitude       0
Start Station Longitude      0
End Station ID               0
End Station Name             0
End Station Latitude         0
End Station Longitude        0
Bike ID                      0
User Type                 5136
Birth Year               32846
Gender                       0
dtype: int64
```

In [23]:
```
df_wo_na = df.dropna()
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 727665 entries, 0 to 727664
Data columns (total 15 columns):
Trip Duration              727665 non-null int64
Start Time                 727665 non-null object
Stop Time                  727665 non-null object
Start Station ID           727665 non-null int64
Start Station Name         727665 non-null object
Start Station Latitude     727665 non-null float64
Start Station Longitude    727665 non-null float64
End Station ID             727665 non-null int64
End Station Name           727665 non-null object
End Station Latitude       727665 non-null float64
End Station Longitude      727665 non-null float64
Bike ID                    727665 non-null int64
User Type                  722529 non-null object
Birth Year                 694819 non-null float64
Gender                     727665 non-null int64
dtypes: float64(5), int64(5), object(5)
memory usage: 83.3+ MB
None
```

In [24]:
```
df['Birth Year'].apply(lambda y: y + 100 if y < 1918 else y)
df = df.reset_index(drop=True)
```

In [25]:
```
from sklearn import neighbors
knn = neighbors.KNeighborsRegressor(10, weights='distance')
for col in ['Start Time', 'Stop Time']:
    df_wo_na[col] = df_wo_na[col].apply(lambda x: pd.Timestamp(x).valu
e)
for col in ['Start Station Name', 'End Station Name', 'User Type']:
    df_wo_na[col] = pd.Categorical(df_wo_na[col]).codes
knn.fit(df_wo_na.drop(['Birth Year'], axis=1).values, df_wo_na['Birth
Year'].values)
```

Out[25]:
```
KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowsk
i',
                    metric_params=None, n_jobs=None, n_neighbors=10,
p=2,
                    weights='distance')
```

In [26]:
```
df.shape

#df.size
#df.nlargest(5, 'Birth Year')
#df.nsmallest(40,'Birth Year')
```

Out[26]:
```
(727665, 15)
```

In [10]:
```
df.describe()
```

Out[10]:

	Trip Duration	Start Station ID	Start Station Latitude	Start Station Longitude	End Station ID	End Stat Latitu
count	7.276650e+05	727665.000000	727665.000000	727665.000000	727665.000000	727665.000(
mean	7.895136e+02	1209.370475	40.736782	-73.985208	1198.276256	40.736
std	6.318341e+03	1277.615327	0.026649	0.016056	1272.246228	0.054
min	6.100000e+01	72.000000	40.646538	-74.017134	72.000000	0.000(
25%	3.360000e+02	354.000000	40.720196	-73.995481	350.000000	40.720
50%	5.390000e+02	478.000000	40.739017	-73.987520	478.000000	40.738
75%	8.900000e+02	3090.000000	40.754666	-73.976806	3082.000000	40.754
max	2.480190e+06	3456.000000	40.804213	-73.929891	3456.000000	40.804

In [27]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 727665 entries, 0 to 727664
Data columns (total 15 columns):
Trip Duration              727665 non-null int64
Start Time                 727665 non-null object
Stop Time                  727665 non-null object
Start Station ID           727665 non-null int64
Start Station Name         727665 non-null object
Start Station Latitude     727665 non-null float64
Start Station Longitude    727665 non-null float64
End Station ID             727665 non-null int64
End Station Name           727665 non-null object
End Station Latitude       727665 non-null float64
End Station Longitude      727665 non-null float64
Bike ID                    727665 non-null int64
User Type                  722529 non-null object
Birth Year                 694819 non-null float64
Gender                     727665 non-null int64
dtypes: float64(5), int64(5), object(5)
memory usage: 83.3+ MB
```

In [28]:
```
nan_by = df[df['Birth Year'].isnull()]
for col in ['Start Time', 'Stop Time']:
    nan_by[col] = nan_by[col].apply(lambda x: pd.Timestamp(x).value)
for col in ['Start Station Name', 'End Station Name', 'User Type']:
    nan_by[col] = pd.Categorical(nan_by[col]).codes
nan_by = nan_by.drop(['Birth Year'], axis=1)
nan_by['Birth Year'] = knn.predict(nan_by)
df.loc[df['Birth Year'].isnull(), 'Birth Year'] = nan_by['Birth Year']
.astype(int)
df['age'] = df['Birth Year'].apply(lambda y: 2017 - y)
```

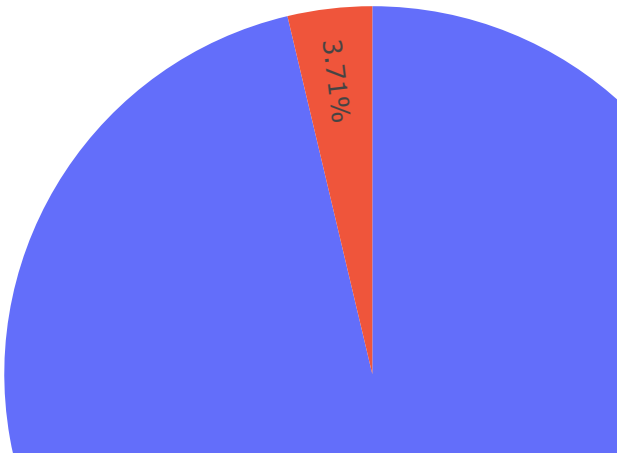
```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:3:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y

/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/panda
s-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cop
y
```

In [29]:
```
#df.isna().sum()
#df = df.dropna()
#df.isna().sum()

#df['Bike ID'].unique().shape
df.info()
```

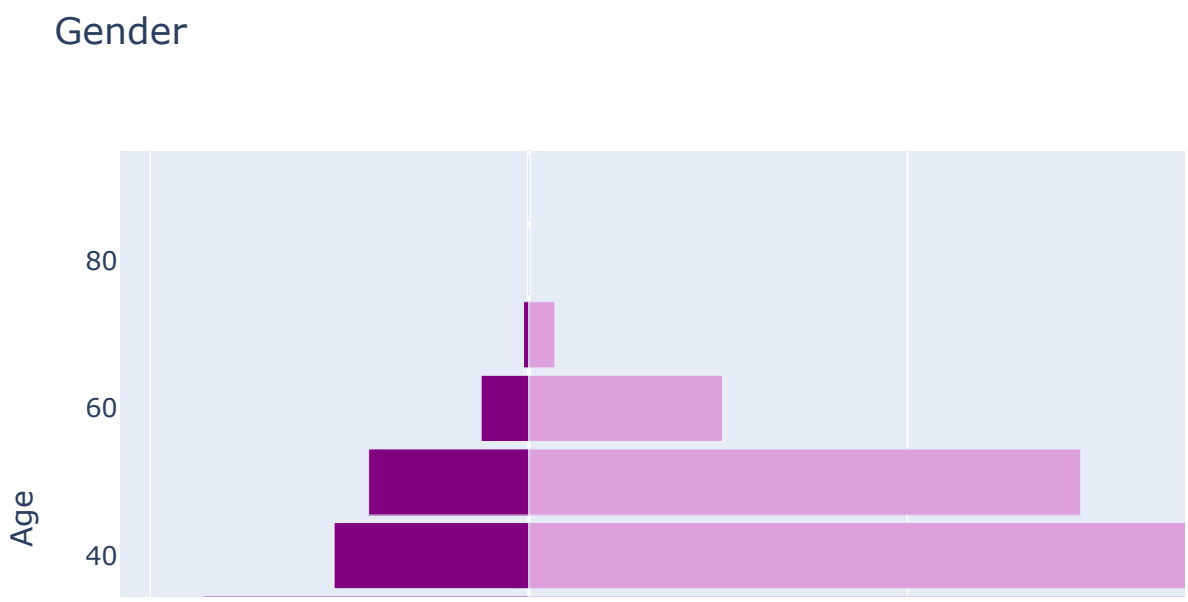
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 727665 entries, 0 to 727664
Data columns (total 16 columns):
Trip Duration              727665 non-null int64
Start Time                 727665 non-null object
Stop Time                  727665 non-null object
Start Station ID           727665 non-null int64
Start Station Name         727665 non-null object
Start Station Latitude     727665 non-null float64
Start Station Longitude    727665 non-null float64
End Station ID             727665 non-null int64
End Station Name           727665 non-null object
End Station Latitude       727665 non-null float64
End Station Longitude      727665 non-null float64
Bike ID                    727665 non-null int64
User Type                  722529 non-null object
Birth Year                 727665 non-null float64
Gender                     727665 non-null int64
age                        727665 non-null float64
dtypes: float64(6), int64(5), object(5)
memory usage: 88.8+ MB
```

In [30]:
```
customer_type_df = pd.DataFrame(data=df['User Type'].value_counts())
customer_type_df = customer_type_df.reset_index()
customer_type_df.rename(columns={'User Type':'count', 'index': 'type'}
, inplace=True)
```

In [122]:
```
layout = go.Layout(
    title='User Type',
)
trace = go.Pie(labels=customer_type_df['type'].values, values=customer
_type_df['count'].values)
fig = go.Figure(data=[trace], layout=layout)
iplot(fig)
```



In [51]:
```
y = list(range(0, 110, 10))
men_bins = []
women_bins = []
for i in range(0, len(y) - 1):
    df_gender = pd.DataFrame(data=df[( df['age'] >  y[i] ) & (df['age'
] < y[i+1]) ]['Gender'].value_counts())
    df_gender = df_gender.reset_index()
    df_gender.rename(columns={'Gender':'count', 'index':'gender'}, inp
lace=True)
    count = df_gender[df_gender['gender'] == 1]['count']
    men_bins.append(0 if len(count) == 0 else count.values[0])
    count2 = df_gender[df_gender['gender'] == 2]['count']
    women_bins.append(0 if len(count2) == 0 else -count2.values[0])

layout = go.Layout(yaxis=go.layout.YAxis(title='Age'),
                   title="Gender",
                    barmode='overlay',
                    bargap=0.1)

data = [
        go.Bar(y=y,
               x=women_bins,
               orientation='h',
               name='Women',
               text=-1 * women_bins,
               hoverinfo='text',
               marker=dict(color='purple')
               ),go.Bar(y=y,
               x=men_bins,
               orientation='h',
               name='Men',
               hoverinfo='x',
               marker=dict(color='plum')
               )]
iplot(dict(data=data, layout=layout))
```


Gender
Age
80
60
40

In [53]:
```
#layout2 = go.Layout(yaxis=go.layout.YAxis(range=[0,80], title='Age'),
 #                   barmode='overlay',
 #                    xaxis=go.layout.XAxis(
 #                        ticvals=[-150, -100, -50, 0, 50, 100, 150],
 #                        tictext=[150, 100, 50, 0, 50, 100, 150],
 #                   title="Gender")

#data2 = [go.Histogram(
#             y=y,
#                x=women_bins,
#                orientation='h',
#                name='Women',
#                text=-1 * women_bins,
#                hoverinfo='skip',
#                marker=dict(color='plum')
#                ),
#         go.Histogram(
#             y=y,
#                x=men_bins,
#                orientation='h',
#                name='Men',
#                hoverinfo='skip',
#                marker=dict(color='purple'),
#                 histfunc="sum"
#                )
#         ]
#iplot(dict(data=data2, layout=layout2))
```

In [54]:
```
#df['Bike ID'].unique().shape
#df['Start Station ID'].unique().shape
print('Unique Station IDs', df['End Station ID'].unique().shape)
print('Unique Bike IDs', df['Bike ID'].unique().shape)
min_minutes = df.loc[(df['Trip Duration'] < 300)][['Trip Duration']].c
ount()
print('Under 5 minutes', min_minutes)
min_minutes = df.loc[(df['Trip Duration'] < 120)][['Trip Duration']].c
ount()
print('Under 2 minutes', min_minutes)
```

```
Unique Station IDs (619,)
Unique Bike IDs (9765,)
Under 5 minutes Trip Duration    145915
dtype: int64
Under 2 minutes Trip Duration    11292
dtype: int64
```

In [112]:
```
import matplotlib as mpl
import matplotlib.pyplot as plt
plt.style.use('ggplot')

# Compute trip counts / day
df.index = df['Start Time'] # Set 'starttime' variable as the index
countsPerDay = df['Start Time'].resample('D', how = ['count'])
#countsPerDay = df['Start Time'].resample('D').sum()

# Plot trip counts
#countsPerDay.plot(kind = 'area', stacked = False, figsize = (15, 5),
#                  color = 'teal', linewidth = 2, legend = False)

#plt.tick_params(axis = 'both', which = 'major', labelsize = 18)
#plt.title('Number of trips per day\n')
#plt.xlabel('')
#plt.ylabel('Number of trips')
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:7:
FutureWarning:

how in .resample() is deprecated
the new syntax is .resample(...)..apply(<func>)
```

In [170]:
```
#countsPerDay['count'].value_counts()
#countsPerDay('T').groupby('time')['val'].value_counts()
df4 = countsPerDay.values
#df5 = pd.DataFrame('count':df4[:,0]})
df5 = pd.DataFrame()
df5['count'] = df4.tolist()

#type(df4)
#df5 = df4.astype(np.int)
df5
#df5.shape
```

Out[170]:

	count
0	[40592]
1	[35739]
2	[31105]
3	[15646]
4	[15919]
5	[32467]

6 [29683]

7 [43469]

8 [45252]

9 [18412]

10 [15185]

11 [13442]

12 [27417]

13 [0]

14 [0]

15 [0]

16 [7098]

17 [4106]

18 [10550]

19 [27378]

20 [36824]

21 [26927]

22 [29885]

23 [34018]

24 [29646]

25 [19893]

26 [26301]

27 [21589]

28 [42430]

29 [39746]

30 [6946]

In [68]:
```
df2 = pd.read_csv('/Users/jeffbloom13/WPI/DS504/week-7/NYCWeatherCleanTotal.csv')
df2.head()
```

Out[68]:

	Day	AvgTemp	AvgDP	AvgH	AvgW	AvgP	P	weathersit
0	1	59.2	53.1	81.3	8.0	29.7	0.00	0
1	2	48.0	23.3	38.7	10.7	29.8	0.12	1
2	3	34.9	12.8	41.1	6.4	30.3	0.00	0
3	4	26.7	6.2	42.8	5.5	30.5	0.00	0
4	5	28.6	1.2	33.3	4.3	30.6	0.00	0

In [71]:
```
#df2Merged = pd.merge(df2, countsPerDay)
#df2Merged.head()
df2.shape
```

Out[71]:
```
(31, 8)
```

In [124]:
```
#countsPerDay['count']
#countsPerDay.shape
#df2['Start Time'].dt.hour
countsPerDay.shape
#countsPerDay.translate({ord('2017-03-*'): None})
#countsPerDay['count'] = countsPerDay['count'].replace({'2017-03-'} : ''}, regex=True)
#df3 = countsPerDay['count'].str.replace(r'\D', '')
#countsPerDay.columns = countsPerDay.columns.str.replace('\D','')
countsPerDay
```

Out[124]:

	count
Start Time	
2017-03-01	40592
2017-03-02	35739
2017-03-03	31105
2017-03-04	15646
2017-03-05	15919
2017-03-06	32467
2017-03-07	29683
2017-03-08	43469
2017-03-09	45252

2017-03-10	18412
2017-03-11	15185
2017-03-12	13442
2017-03-13	27417
2017-03-14	0
2017-03-15	0
2017-03-16	0
2017-03-17	7098
2017-03-18	4106
2017-03-19	10550
2017-03-20	27378
2017-03-21	36824
2017-03-22	26927
2017-03-23	29885
2017-03-24	34018
2017-03-25	29646
2017-03-26	19893
2017-03-27	26301
2017-03-28	21589
2017-03-29	42430
2017-03-30	39746
2017-03-31	6946

In [271]:
```
#df2Merged = pd.concat([df2,countsPerDay], axis=1, sort=True)
#df2Merged = pd.concat([df2,df5], axis=1)

#df2Merged
#df2Merged.info()
#df2Merged.dtypes

weatherDf
```

Out[271]:

	Day	AvgTemp	AvgDP	AvgH	AvgW	AvgP	P
0	1	59.2	53.1	81.3	8.0	29.7	0.00
1	2	48.0	23.3	38.7	10.7	29.8	0.12

2	3	34.9	12.8	41.1	6.4	30.3	0.00
3	4	26.7	6.2	42.8	5.5	30.5	0.00
4	5	28.6	1.2	33.3	4.3	30.6	0.00
5	6	36.9	17.7	47.0	5.3	30.5	0.00
6	7	47.5	41.7	80.5	4.3	30.2	0.00
7	8	54.7	34.0	49.6	11.5	29.9	0.16
8	9	55.2	22.6	29.2	10.2	29.9	0.00
9	10	37.4	25.8	66.1	7.6	29.9	0.00
10	11	26.2	9.7	50.0	5.8	30.2	0.24
11	12	27.7	9.4	46.7	4.9	30.3	0.00
12	13	30.1	11.8	48.3	6.2	30.4	0.00
13	14	30.8	27.5	88.1	14.1	29.6	0.32
14	15	25.9	15.7	65.3	8.3	29.6	0.71
15	16	33.3	15.0	49.3	8.7	30.0	0.00
16	17	38.1	19.1	47.5	8.1	30.2	0.00
17	18	36.8	27.7	70.9	12.2	30.2	0.00
18	19	41.9	24.3	52.2	9.8	30.2	0.07
19	20	44.4	19.1	38.2	5.2	30.1	0.00
20	21	50.4	29.0	44.5	5.6	29.9	0.00
21	22	38.4	12.2	34.7	4.8	30.1	0.00
22	23	35.0	9.2	37.9	6.4	30.5	0.00
23	24	44.8	28.6	53.9	9.0	30.3	0.00
24	25	50.7	38.3	63.4	7.8	30.2	0.00
25	26	41.1	35.3	79.8	12.9	30.4	0.10
26	27	44.7	41.8	89.7	7.0	30.1	0.02
27	28	44.4	42.5	93.1	9.9	29.9	0.42
28	29	50.3	34.9	60.1	4.7	30.1	0.43
29	30	44.5	27.0	51.0	6.9	30.2	0.00
30	31	40.4	37.8	91.0	14.9	30.0	0.33

In [236]:
```
ndim = countsPerDay.shape[0]
ndim
```

Out[236]: 31

In [282]:
```
#df2Merged = df2Merged.astype({"count: int)
#df2Merged = df2Merged.astype({"count":int})
#df2Merged.count = df2Merged.count.astype('int64')
#df2Merged.count[0]
#df2Merged['count'] = df2Merged['count'].astype(int)

# Now add count
weatherDf['count'] = 0
ndim = countsPerDay.shape[0]
for index,row in weatherDf.iterrows():
    #print('colNum=', i)
    #print('val=', df2.iloc[: , idx].values)
    #print('day=',j['Day'])

    print(row['Day'], row['P'])
    #row['count'] = countsPerDay['count'].values[index]
    if index < ndim:
        print('count=', countsPerDay['count'].values[index])
       # df['count'][index] = countsPerDay['count'].values[index]
        weatherDf.loc[index,'count'] = countsPerDay['count'].values[in
dex]
   # if j['Start Time'].day:
       # j['morning'] += 1
        #print(j['morning'])
    #    df2.at[i,'morning'] += 1
       # print(' Morning =', df2.at[i,'morning'])
```

```
1.0 0.0
count= 40592
2.0 0.12
count= 35739
3.0 0.0
count= 31105
4.0 0.0
count= 15646
5.0 0.0
count= 15919
6.0 0.0
count= 32467
7.0 0.0
count= 29683
8.0 0.16
count= 43469
9.0 0.0
count= 45252
10.0 0.0
count= 18412
11.0 0.24
count= 15185
```

```
12.0 0.0
count= 13442
13.0 0.0
count= 27417
14.0 0.32
count= 0
15.0 0.71
count= 0
16.0 0.0
count= 0
17.0 0.0
count= 7098
18.0 0.0
count= 4106
19.0 0.07
count= 10550
20.0 0.0
count= 27378
21.0 0.0
count= 36824
22.0 0.0
count= 26927
23.0 0.0
count= 29885
24.0 0.0
count= 34018
25.0 0.0
count= 29646
26.0 0.1
count= 19893
27.0 0.02
count= 26301
28.0 0.42
count= 21589
29.0 0.43
count= 42430
30.0 0.0
count= 39746
31.0 0.33
count= 6946
```

In [283]:
```
# Collect all trips shorter than 1 hour
#duration_mins = df.loc[(df['Trip Duration'] / 60 < 60)][['Trip Durati
on']]
#duration_mins = duration_mins / 60 # In minutes

# Plot the distribution of trip durations
#plt.rcParams.update({'font.size': 16})
#duration_mins.hist(figsize = (8,5), bins = 15, alpha = 0.5, color = '
teal')
#plt.tick_params(axis = 'both', which = 'major', labelsize = 18)
#plt.title('Trip duration Histogram\n')
#plt.xlabel('Duration (minutes)')
#plt.ylabel('Trip counts')

weatherDf
```

Out[283]:

	Day	AvgTemp	AvgDP	AvgH	AvgW	AvgP	P	count
0	1	59.2	53.1	81.3	8.0	29.7	0.00	40592
1	2	48.0	23.3	38.7	10.7	29.8	0.12	35739
2	3	34.9	12.8	41.1	6.4	30.3	0.00	31105
3	4	26.7	6.2	42.8	5.5	30.5	0.00	15646
4	5	28.6	1.2	33.3	4.3	30.6	0.00	15919
5	6	36.9	17.7	47.0	5.3	30.5	0.00	32467
6	7	47.5	41.7	80.5	4.3	30.2	0.00	29683
7	8	54.7	34.0	49.6	11.5	29.9	0.16	43469
8	9	55.2	22.6	29.2	10.2	29.9	0.00	45252
9	10	37.4	25.8	66.1	7.6	29.9	0.00	18412
10	11	26.2	9.7	50.0	5.8	30.2	0.24	15185
11	12	27.7	9.4	46.7	4.9	30.3	0.00	13442
12	13	30.1	11.8	48.3	6.2	30.4	0.00	27417
13	14	30.8	27.5	88.1	14.1	29.6	0.32	0
14	15	25.9	15.7	65.3	8.3	29.6	0.71	0
15	16	33.3	15.0	49.3	8.7	30.0	0.00	0
16	17	38.1	19.1	47.5	8.1	30.2	0.00	7098
17	18	36.8	27.7	70.9	12.2	30.2	0.00	4106
18	19	41.9	24.3	52.2	9.8	30.2	0.07	10550
19	20	44.4	19.1	38.2	5.2	30.1	0.00	27378

20	21	50.4	29.0	44.5	5.6	29.9	0.00	36824
21	22	38.4	12.2	34.7	4.8	30.1	0.00	26927
22	23	35.0	9.2	37.9	6.4	30.5	0.00	29885
23	24	44.8	28.6	53.9	9.0	30.3	0.00	34018
24	25	50.7	38.3	63.4	7.8	30.2	0.00	29646
25	26	41.1	35.3	79.8	12.9	30.4	0.10	19893
26	27	44.7	41.8	89.7	7.0	30.1	0.02	26301
27	28	44.4	42.5	93.1	9.9	29.9	0.42	21589
28	29	50.3	34.9	60.1	4.7	30.1	0.43	42430
29	30	44.5	27.0	51.0	6.9	30.2	0.00	39746
30	31	40.4	37.8	91.0	14.9	30.0	0.33	6946

In [227]:
```
df['Start Time'] = df['Start Time'].apply(pd.to_datetime)
def extract_part_of_day(hour):
    if hour < 5:
        return 'early morning'
    if hour < 10:
        return 'morning'
    if hour < 14:
        return 'noon'
    if hour < 18:
        return 'afternoon'
    return 'evening'
df['part_of_day'] = df['Start Time'].apply(lambda t: extract_part_of_d
ay(t.hour))
```

```
---------------------------------------------------------------------
-------
KeyboardInterrupt                         Traceback (most recent cal
l last)
<ipython-input-227-4b8077ea71b4> in <module>
----> 1 df['Start Time'] = df['Start Time'].apply(pd.to_datetime)
      2 def extract_part_of_day(hour):
      3     if hour < 5:
      4         return 'early morning'
      5     if hour < 10:

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/series.py in
apply(self, func, convert_dtype, args, **kwds)
   4040             # passes this meta-data
   4041             kwargs.pop("_axis", None)
-> 4042             kwargs.pop("_level", None)
   4043
   4044             # try a regular apply, this evaluates lambdas
```

```
pandas/_libs/lib.pyx in pandas._libs.lib.map_infer()

/opt/anaconda3/lib/python3.7/site-packages/pandas/util/_decorators.p
y in wrapper(*args, **kwargs)
    206 def _format_argument_list(allow_args: Union[List[str], int])
:
    207     """
--> 208     Convert the allow_args argument (either string or
integer) of
    209     `deprecate_nonkeyword_arguments` function to a string de
scribing
    210     it to be inserted into warning message.

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/tools/datetim
es.py in to_datetime(arg, errors, dayfirst, yearfirst, utc, box, for
mat, exact, unit, infer_datetime_format, origin, cache)
    794             if arg.tz is not None:
    795                 result = result.tz_convert(tz)
--> 796             else:
    797                 result = result.tz_localize(tz)
    798     elif isinstance(arg, ABCSeries):

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/tools/datetim
es.py in _convert_listlike_datetimes(arg, box, format, name, tz, uni
t, errors, infer_datetime_format, dayfirst, yearfirst, exact)
    461             dayfirst=dayfirst,
    462             yearfirst=yearfirst,
--> 463             utc=utc,
    464             errors=errors,
    465             require_iso8601=require_iso8601,

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/arrays/dateti
mes.py in objects_to_datetime64ns(data, dayfirst, yearfirst, utc, er
rors, require_iso8601, allow_object)
   1973             )
   1974             data = data.view(DT64NS_DTYPE)
-> 1975
   1976         assert data.dtype == DT64NS_DTYPE, data.dtype
   1977         result = data

KeyboardInterrupt:
```

In [ ]:
```
df.head()
```

In [57]:
```
df_station_end = df.groupby(['End Station ID', 'End Station Name', 'End Station Latitude', 'End Station Longitude']).count().reset_index()[['End Station ID', 'End Station Name', 'End Station Latitude', 'End Station Longitude', 'age']]
df_station_end.rename(columns={
    'End Station ID': 'id',
    'End Station Name': 'name',
    'End Station Latitude':'lat',
    'End Station Longitude': 'lon'
}, inplace= True)
df_station_start = df.groupby(['Start Station ID', 'Start Station Name', 'Start Station Latitude', 'Start Station Longitude']).count().reset_index()[['Start Station ID', 'Start Station Name', 'Start Station Latitude', 'Start Station Longitude', 'age']]
df_station_start.rename(columns={
    'Start Station ID': 'id',
    'Start Station Name': 'name',
    'Start Station Latitude':'lat',
    'Start Station Longitude': 'lon'
}, inplace= True)

df_paths = df.groupby(['End Station ID', 'End Station Name', 'End Station Latitude', 'End Station Longitude','Start Station ID', 'Start Station Name', 'Start Station Latitude', 'Start Station Longitude']).count().reset_index()
```

In [19]:
```
df_station_end['lat'].values
```

Out[19]:
```
array([40.76727216, 40.71911552, 40.71117416, 40.68382604, 40.741776
03,
       40.69608941, 40.68676793, 40.73172428, 40.72710258, 40.692395
02,
       40.69839895, 40.71625008, 40.7208736 , 40.72210379, 40.714739
93,
       40.75206231, 40.69089272, 40.72917025, 40.75323098, 40.748900
6 ,
       40.73971301, 40.76068327, 40.7381765 , 40.70905623, 40.743349
35,
       40.70037867, 40.70277159, 40.73781509, 40.71146364, 40.741951
38,
       40.7546011 , 40.72743423, 40.69597683, 40.7284186 , 40.730473
09,
       40.7361967 , 40.69196566, 40.68981035, 40.697787  , 40.688226
,
       40.69196035, 40.69327018, 40.73535398, 40.72185379, 40.718709
87,
       40.72317958, 40.73226398, 40.73543934, 40.73532427, 40.646768
,
       40.71939226, 40.68940747, 40.70122128, 40.70365182, 40.694748
```

```
81,
       40.6917823 , 40.70706456, 40.72229346, 40.72368361, 40.750977
11,
       40.71910537, 40.69308257, 40.68691865, 40.68650065, 40.717487
52,
       40.69766564, 40.707873  , 40.73331967, 40.7643971 , 40.707644
94,
       40.73901691, 40.73454567, 40.6845683 , 40.713126  , 40.730206
61,
       40.71406667, 40.71413089, 40.734232  , 40.68683208, 40.722174
44,
       40.72082834, 40.72362738, 40.70463334, 40.76095756, 40.708235
02,
       40.71427487, 40.71307916, 40.7149787 , 40.68926942, 40.717227
4 ,
       40.722055  , 40.69610226, 40.69383   , 40.70355377, 40.709559
58,
       40.72453734, 40.711066  , 40.717571  , 40.69991755, 40.696192
,
       40.69236178, 40.689888  , 40.73624527, 40.72953837, 40.715337
9 ,
       40.72405549, 40.71450451, 40.71173107, 40.71219906, 40.742387
87,
       40.72903917, 40.73047747, 40.7037992 , 40.72580614, 40.712690
42,
       40.71782143, 40.71739973, 40.69794   , 40.6851443 , 40.736494
03,
       40.73652889, 40.728846  , 40.72490985, 40.71850211, 40.715595
09,
       40.70530954, 40.68539567, 40.69363137, 40.71602118, 40.716226
44,
       40.73261787, 40.73291553, 40.75510267, 40.70717936, 40.716058
66,
       40.75172632, 40.70834698, 40.68900443, 40.68223166, 40.693261
,
       40.73038599, 40.73224119, 40.694528  , 40.69331716, 40.708621
44,
       40.72243797, 40.749156  , 40.73401143, 40.73492695, 40.735238
,
       40.683048  , 40.75797322, 40.71494807, 40.71273266, 40.749717
75,
       40.71044554, 40.69221589, 40.69760127, 40.695065  , 40.722992
08,
       40.72521311, 40.68807003, 40.68034242, 40.68415748, 40.691651
83,
       40.68851534, 40.71926081, 40.72019576, 40.7403432 , 40.725028
76,
       40.69512845, 40.700469  , 40.71076228, 40.6906495 , 40.720664
42,
       40.72228087, 40.7158155 , 40.70281858, 40.7047177 , 40.687534
```

```
06,
       40.71291224, 40.70224   , 40.69580705, 40.68764484, 40.695733
98,
       40.770513  , 40.76584941, 40.71754834, 40.701907  , 40.724677
21,
       40.7014851 , 40.72621788, 40.72955361, 40.74317449, 40.741739
69,
       40.68216564, 40.68098339, 40.72779126, 40.7262807 , 40.752554
34,
       40.756014  , 40.746647  , 40.70853074, 40.7423543 , 40.727407
94,
       40.74487634, 40.76370739, 40.75660359, 40.76461837, 40.762272
05,
       40.74475148, 40.75455731, 40.75001986, 40.7597108 , 40.766953
17,
       40.751396  , 40.746745  , 40.71285887, 40.73587678, 40.746919
59,
       40.75513557, 40.74395411, 40.68312489, 40.7652654 , 40.763440
58,
       40.74345335, 40.71286844, 40.7457121 , 40.72110063, 40.745167
7 ,
       40.74394314, 40.75640548, 40.76030096, 40.76019252, 40.766696
71,
       40.71260486, 40.73935542, 40.73223272, 40.75500254, 40.750380
09,
       40.7462009 , 40.73314259, 40.75645824, 40.751551  , 40.740963
74,
       40.75019995, 40.7568001 , 40.74734825, 40.76269882, 40.737261
86,
       40.73704984, 40.74854862, 40.76915505, 40.76228826, 40.744219
,
       40.714215  , 40.73827428, 40.73221853, 40.74901271, 40.739126
01,
       40.76341379, 40.7454973 , 40.72938685, 40.768254  , 40.760875
02,
       40.76009437, 40.75206862, 40.751581  , 40.74780373, 40.751873
,
       40.75992262, 40.75714758, 40.75466591, 40.75527307, 40.755941
59,
       40.74765947, 40.744023  , 40.74290902, 40.7575699 , 40.771522
,
       40.71893904, 40.710451  , 40.75299641, 40.70255065, 40.741443
87,
       40.74025878, 40.71534825, 40.74311555, 40.736502  , 40.744449
21,
       40.70255088, 40.699773  , 40.716887  , 40.73381219, 40.724399
,
       40.70531194, 40.76590936, 40.70569254, 40.71117444, 40.721654
81,
       40.739445  , 40.75929124, 40.759107  , 40.75968085, 40.711512
```

```
,
       40.72036775, 40.646678  , 40.67890679, 40.6794268 , 40.681459
8 ,
       40.6800105 , 40.682601  , 40.6823687 , 40.68402   , 40.68488
,
       40.6851532 , 40.686312  , 40.6900815 , 40.6894932 , 40.688333
7 ,
       40.69072549, 40.69128258, 40.69237074, 40.6933982 , 40.694254
03,
       40.69622937, 40.69539817, 40.69527008, 40.69681963, 40.700295
11,
       40.69957608, 40.7016657 , 40.7031724 , 40.70411791, 40.705109
18,
       40.70538077, 40.70583339, 40.70691254, 40.70767788, 40.707087
01,
       40.70870368, 40.70877084, 40.70925562, 40.70934   , 40.711863
,
       40.71167351, 40.71247661, 40.71469   , 40.715143  , 40.714133
11,
       40.7160751 , 40.71774592, 40.71764   , 40.7190095 , 40.717451
69,
       40.7169811 , 40.71929301, 40.71924   , 40.72481256, 40.720798
21,
       40.72179134, 40.72153267, 40.724055  , 40.72325   , 40.723116
51,
       40.72557   , 40.72606   , 40.72708584, 40.7258483 , 40.72906
,
       40.73026   , 40.73165141, 40.73232194, 40.73266   , 40.73564
,
       40.73555   , 40.74232744, 40.74161   , 40.74524768, 40.744363
29,
       40.74469738, 40.74731   , 40.74708586, 40.74718234, 40.74966
,
       40.75052534, 40.75110165, 40.75325964, 40.7671284 , 40.763505
32,
       40.76312584, 40.77112927, 40.766368  , 40.77282817, 40.771182
88,
       40.77140426, 40.76500525, 40.7612274 , 40.77682863, 40.776777
02,
       40.77862688, 40.77573034, 40.77801203, 40.77565541, 40.775369
05,
       40.7728384 , 40.76873687, 40.76440023, 40.76663814, 40.775186
15,
       40.77163851, 40.77492513, 40.77896784, 40.78018397, 40.783399
81,
       40.7734066 , 40.7770575 , 40.77579377, 40.78057799, 40.779668
09,
       40.78472675, 40.78720869, 40.78499979, 40.78524672, 40.778566
9 ,
       40.77750703, 40.77748046, 40.77452835, 40.7867947 , 40.784144
```

```
72,
       40.698617  , 40.69878   , 40.7127742 , 40.729193  , 40.743
,
       40.75899656, 40.73997354, 40.78275   , 40.7678008 , 40.689621
89,
       40.75724568, 40.75216528, 40.75898481, 40.7739139 , 40.646538
37,
       40.75903008, 40.686203  , 40.69102926, 40.75892386, 40.731437
24,
        0.        , 40.68035608, 40.71690978, 40.75058535, 40.727714
08,
       40.77376867, 40.75018156, 40.74937024, 40.72706363, 40.729236
5 ,
       40.76421007, 40.72456343, 40.71241882, 40.78307   , 40.788221
3 ,
       40.7814107 , 40.78839   , 40.7806284 , 40.778301  , 40.790179
48,
       40.7779453 , 40.7857851 , 40.7921    , 40.7835016 , 40.79127
,
       40.6686627 , 40.686371  , 40.66514682, 40.7919557 , 40.796934
7 ,
       40.6849894 , 40.668127  , 40.7811223 , 40.6662078 , 40.794165
4 ,
       40.6861758 , 40.7859201 , 40.663779  , 40.68763155, 40.781721
2 ,
       40.6663181 , 40.7937704 , 40.6847514 , 40.7989937 , 40.668627
3 ,
       40.7839636 , 40.666287  , 40.793393  , 40.6831164 , 40.668603
,
       40.7981856 , 40.6685455 , 40.7849032 , 40.67434   , 40.787721
4 ,
       40.795     , 40.6829151 , 40.6725058 , 40.8013434 , 40.681990
44,
       40.6747055 , 40.6772744 , 40.787801  , 40.67363551, 40.786258
6 ,
       40.6765304 , 40.6753274 , 40.795346  , 40.6777748 , 40.799756
8 ,
       40.679043  , 40.78948542, 40.67514684, 40.6773429 , 40.677236
,
       40.6729679 , 40.7973721 , 40.7869946 , 40.67267243, 40.674784
4 ,
       40.668132  , 40.76800889, 40.7746671 , 40.8008363 , 40.671197
8 ,
       40.7691572 , 40.7829391 , 40.6740886 , 40.7781314 , 40.790482
8 ,
       40.6751622 , 40.6703837 , 40.8021174 , 40.7922553 , 40.672815
5 ,
       40.7727966 , 40.67461342, 40.7689738 , 40.6750705 , 40.799484
,
       40.7699426 , 40.76471852, 40.6786115 , 40.773763  , 40.790305
```

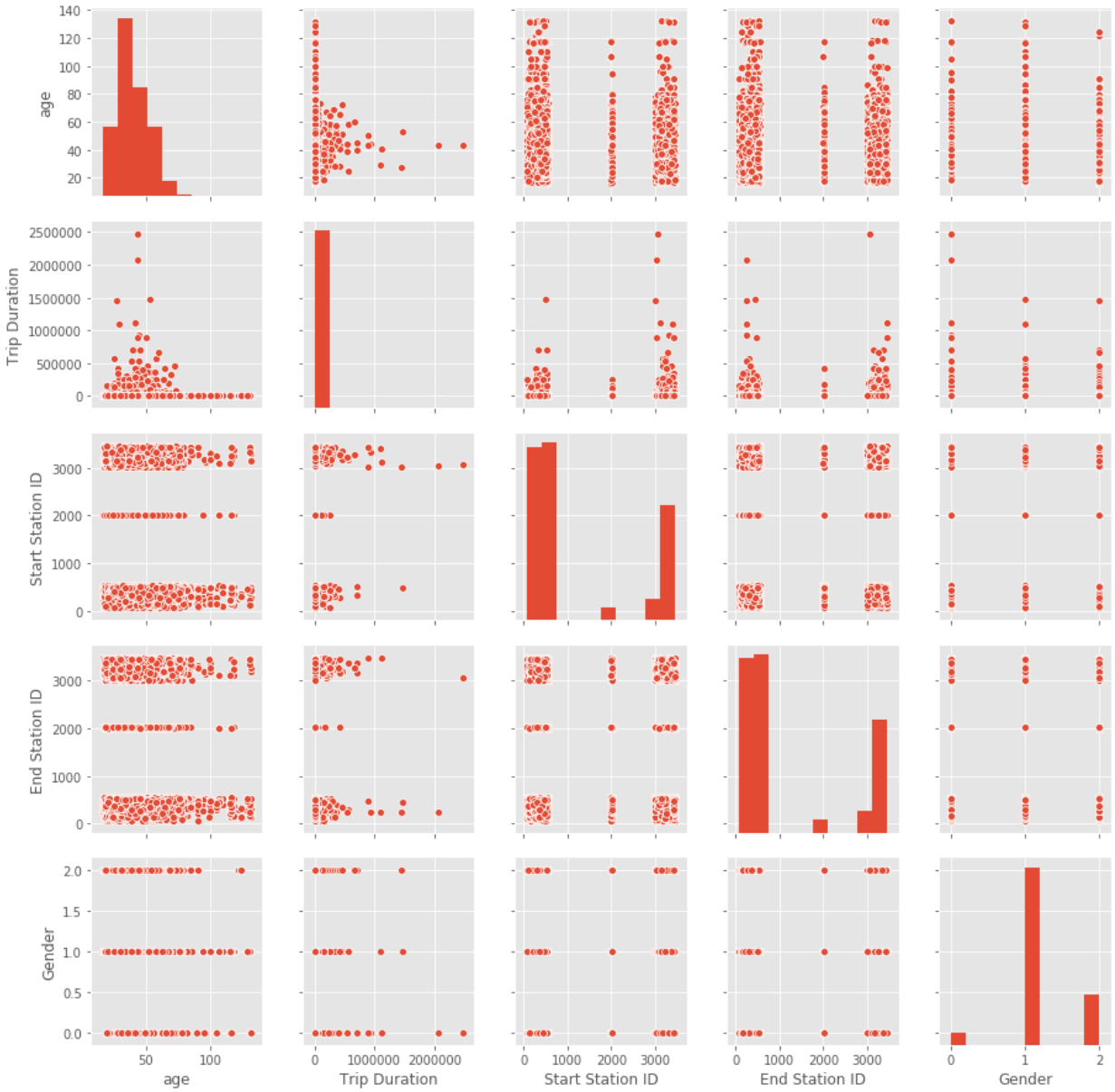
```
1 ,
       40.6777287 , 40.680611  , 40.804213  , 40.6787242 , 40.680959
1 ,
       40.7934337 , 40.6828003 , 40.6830456 , 40.79329668, 40.789252
9 ,
       40.6812117 , 40.6794327 , 40.6769993 , 40.6763744 , 40.678356
3 ,
       40.6763947 , 40.6746957 , 40.67260298, 40.7961535 , 40.672481
1 ,
       40.6902375 , 40.6705135 , 40.6704922 , 40.6704836 , 40.679098
,
       40.6881529 , 40.6867443 , 40.6864442 , 40.6849668 , 40.685376
1 ,
       40.6827549 , 40.68094472, 40.6793307 , 40.6776147 , 40.679576
6 ,
       40.6750207 , 40.6792788 , 40.6802133 , 40.6843549 , 40.685929
6 ,
       40.66106337, 40.791976  , 40.7892105 , 40.72430527, 40.740983
,
       40.68506807, 40.71907891, 40.746524  , 40.79025417, 40.718822
,
       40.721319  , 40.77224854, 40.69241829, 40.752957  , 40.761329
83,
       40.79181172, 40.76703432, 40.72146256, 40.7512836 , 40.719155
72,
       40.71335226, 40.71036854, 40.68680821, 40.71638032])
```

In [58]:
```
#mapbox_access_token = 'pk.eyJ1IjoiYW5keXRyYW4xMTk5NiIsImEiOiJjam9xeXg
2aTMwOWRlM3FvOWk2NDF0N3F4In0.zvrXbjWVMU7dHWHAeLeA4w'
mapbox_access_token = 'pk.eyJ1Ijoiam1ibG9vbSIsImEiOiJja2Y1cm1maHUwYjNn
MnBxN3pqcGZqNjd4In0.7r3hp8Vj1efich43hkUaDQ'
# sk.eyJ1IjoiYW5keXRyYW4xMTk5NiIsImEiOiJjam9yMGFlcW4wOW10M3hucmwwNm83b
TJkIn0.z199ESfbVcWieB3qiOv67A
data = []
data.append(go.Scattermapbox(
        lat=df_station_start['lat'].values,
        lon=df_station_start['lon'].values,
        mode='markers',
        marker=dict(
                size=9
        ),
         text=df_station_start['name'].values
    ))
for i in range(len(df_paths)//2 - 1, len(df_paths)//2-100, -1):
     data.append(go.Scattermapbox(
        lat=[df_paths['Start Station Latitude'][i], df_paths['End Stat
ion Latitude'][i]],
        lon=[ df_paths['Start Station Longitude'][i], df_paths['End St
ation Longitude'][i]],
        mode='lines',
        line = dict(
                width = 1,
                color = 'red',
            ),
    ))
layout = go.Layout(
    autosize=True,
    hovermode='closest',
    mapbox=dict(
        accesstoken=mapbox_access_token,
        bearing=0,
        center=dict(
            lat=40.76,
            lon=-73.99
        ),
        pitch=0,
        zoom=12
    ),
    showlegend = False
)


fig = dict(data=data, layout=layout)
iplot(fig, filename='Multiple Mapbox')
```

(https://www.mapbox.com/)

In [65]:
```
sns.pairplot(df[['age','Trip Duration', 'Start Station ID', 'End Station ID','Gender']])
```

Out[65]: <seaborn.axisgrid.PairGrid at 0x7f886edee0d0>



In [183]:
```
sns.heatmap(df.corr(), annot=True)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-183-6dc1c4c1753e> in <module>
----> 1 sns.heatmap(df.corr(), annot=True)

/opt/anaconda3/lib/python3.7/site-packages/seaborn/matrix.py in heatmap(data, vmin, vmax, cmap, center, robust, annot, fmt, annot_kws, linewidths, linecolor, cbar, cbar_kws, cbar_ax, square, xticklabels, yticklabels, mask, ax, **kwargs)
```

```
    515     plotter = _HeatMapper(data, vmin, vmax, cmap, center, ro
bust, annot, fmt,
    516                           annot_kws, cbar, cbar_kws,
xticklabels,
--> 517                           yticklabels, mask)
    518
    519     # Add the pcolormesh kwargs here

/opt/anaconda3/lib/python3.7/site-packages/seaborn/matrix.py in __in
it__(self, data, vmin, vmax, cmap, center, robust, annot, fmt, annot
_kws, cbar, cbar_kws, xticklabels, yticklabels, mask)
    111
    112         # Validate the mask and convet to DataFrame
--> 113         mask = _matrix_mask(data, mask)
    114
    115         plot_data = np.ma.masked_where(np.asarray(mask),
plot_data)

/opt/anaconda3/lib/python3.7/site-packages/seaborn/matrix.py in _mat
rix_mask(data, mask)
     90     # This works around an issue where `plt.pcolormesh` does
n't represent
     91     # missing data properly
---> 92     mask = mask | pd.isnull(data)
     93
     94     return mask

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/ops/__init__.
py in f(self, other, axis, level, fill_value)

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/frame.py in _
combine_frame(self, other, func, fill_value, level)
   5372             inputs, the key is applied *per level*.
   5373
-> 5374             .. versionadded:: 1.1.0
   5375
   5376         Returns

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/frame.py in _
arith_op(left, right)
   5365
   5366         key : callable, optional
-> 5367             If not None, apply the key function to the index
values
   5368             before sorting. This is similar to the `key`
argument in the
   5369             builtin :meth:`sorted` function, with the notabl
e difference that

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/ops/__init__.
```

```
py in na_op(x, y)

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/computation/e
xpressions.py in evaluate(op, a, b, use_numexpr)
    229     op_str = _op_str_mapping[op]
    230     if op_str is not None:
--> 231         use_numexpr = use_numexpr and _bool_arith_check(op_s
tr, a, b)
    232         if use_numexpr:
    233             return _evaluate(op, op_str, a, b)  # type: igno
re

ValueError: The truth value of an array with more than one element i
s ambiguous. Use a.any() or a.all()
```

In [ ]:
```
#df.head(5)
#sns.clustermap(df[['age','Trip Duration', 'Start Station ID', 'End St
ation ID','Gender']],
  #             figsize=(7,5),
  #              row_cluser=False,
  #              dendrogram_ratio=(.1,.2),
  #              cbar_pos=(0,.2,.03,.4))
```

In [184]:
```
day_parts = ['early morning','morning', 'noon', 'afternoon','evening']
fig = tools.make_subplots(rows=1, cols=5, subplot_titles=day_parts)

df_station_end = df.groupby(['End Station ID', 'End Station Name', 'En
d Station Latitude', 'End Station Longitude','part_of_day']).count().r
eset_index()[['End Station ID', 'End Station Name', 'End Station Latit
ude', 'End Station Longitude', 'age','part_of_day']]
df_station_end.rename(columns={
    'End Station ID': 'id',
    'End Station Name': 'name',
    'End Station Latitude':'lat',
    'End Station Longitude': 'lon'
}, inplace= True)
df_station_start = df.groupby(['Start Station ID', 'Start Station Name
', 'Start Station Latitude', 'Start Station Longitude','part_of_day'])
.count().reset_index()[['Start Station ID', 'Start Station Name', 'Sta
rt Station Latitude', 'Start Station Longitude', 'age','part_of_day']]
df_station_start.rename(columns={
    'Start Station ID': 'id',
    'Start Station Name': 'name',
    'Start Station Latitude':'lat',
    'Start Station Longitude': 'lon'
}, inplace= True)

for idx, daypart in enumerate(day_parts):
    df_start_top10 = df_station_start[df_station_start['part_of_day']
== daypart].sort_values(['age'], ascending=False).head(10)
    trace = go.Bar(
            x=df_start_top10.name,
            y=df_start_top10.age
    )
    fig.append_trace(trace, 1, idx + 1)

fig['layout'].update(title='Top 10 start station')

iplot(fig)
```

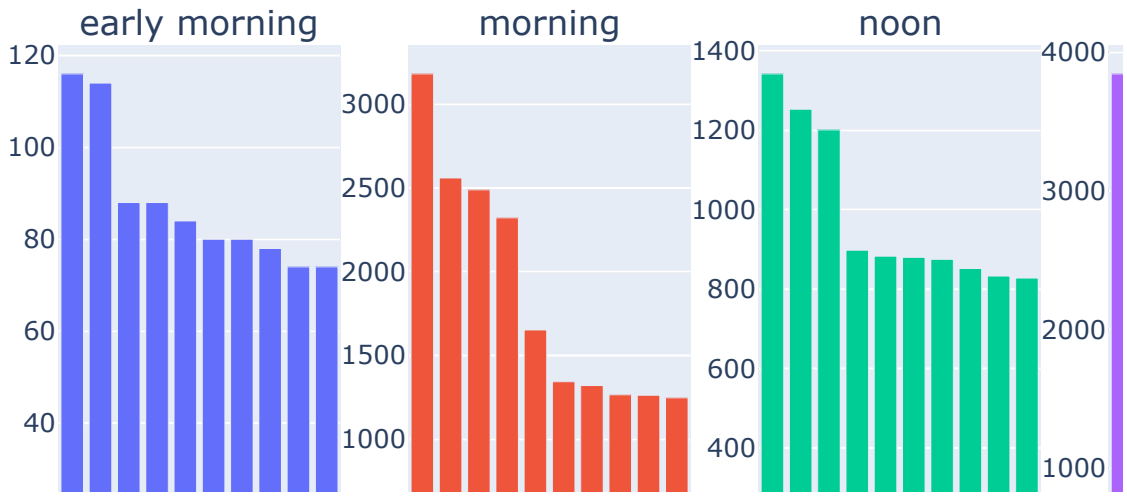
```
/opt/anaconda3/lib/python3.7/site-packages/plotly/tools.py:465: Depr
ecationWarning:

plotly.tools.make_subplots is deprecated, please use plotly.subplots
.make_subplots instead


---------------------------------------------------------------------
-------
KeyError                                  Traceback (most recent cal
l last)
<ipython-input-184-18e4615c2684> in <module>
```

```
      2 fig = tools.make_subplots(rows=1, cols=5, subplot_titles=day
_parts)
      3
----> 4 df_station_end = df.groupby(['End Station ID', 'End Station
Name', 'End Station Latitude', 'End Station Longitude','part_of_day'
]).count().reset_index()[['End Station ID', 'End Station Name', 'End
Station Latitude', 'End Station Longitude', 'age','part_of_day']]
      5 df_station_end.rename(columns={
      6     'End Station ID': 'id',

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/generic.py in
groupby(self, by, axis, level, as_index, sort, group_keys, squeeze,
observed, **kwargs)
   7892
   7893         Resample a year by quarter using 'start' `convention
`. Values are
-> 7894         assigned to the first quarter of the period.
   7895
   7896         >>> s = pd.Series([1, 2], index=pd.period_range('201
2-01-01',

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/groupby/group
by.py in groupby(obj, by, **kwds)
   2520
   2521         # error_msg is "" on an frame/series with no rows or
columns
-> 2522         if len(output) == 0 and error_msg != "":
   2523             raise TypeError(error_msg)
   2524

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/groupby/group
by.py in __init__(self, obj, keys, axis, level, grouper, exclusions,
selection, as_index, sort, group_keys, squeeze, observed, **kwargs)
    389     If the ``'numba'`` engine is chosen, the function must b
e
    390     a user defined function with ``values`` and ``index`` as
the
--> 391     first and second arguments respectively in the function
signature.
    392     Each group's index will be passed to the user defined fu
nction
    393     and optionally available for use.

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/groupby/group
er.py in _get_grouper(obj, key, axis, level, sort, observed, mutated
, validate)
    619     This may be composed of multiple Grouping objects,
indicating
    620     multiple groupers
--> 621
```

```
    622     Groupers are ultimately index mappings. They can origina
te as:
    623     index mappings, keys to columns, functions, or Groupers

KeyError: 'part_of_day'
```

In [62]:
```
fig2 = tools.make_subplots(rows=1, cols=5, subplot_titles=day_parts)
for idx, daypart in enumerate(day_parts):
    df_start_top10 = df_station_end[df_station_end['part_of_day'] == d
aypart].sort_values(['age'], ascending=False).head(10)
    trace = go.Bar(
            x=df_start_top10.name,
            y=df_start_top10.age
    )
    fig2.append_trace(trace, 1, idx + 1)

fig2['layout'].update(title='Top 10 end station')

iplot(fig2)
```

Top 10 end station



In [63]:
```
#df.index = df['Start Time'] # Set 'starttime' variable as the index

#countsPerDay = df['Start Time'].resample('D', how = ['count'])
df_station_end.head(10)
```

Out[63]:

	id	name	lat	lon	age	part_of_day
0	72	W 52 St & 11 Ave	40.767272	-73.993929	596	afternoon
1	72	W 52 St & 11 Ave	40.767272	-73.993929	31	early morning
2	72	W 52 St & 11 Ave	40.767272	-73.993929	479	evening
3	72	W 52 St & 11 Ave	40.767272	-73.993929	411	morning
4	72	W 52 St & 11 Ave	40.767272	-73.993929	415	noon
5	79	Franklin St & W Broadway	40.719116	-74.006667	368	afternoon
6	79	Franklin St & W Broadway	40.719116	-74.006667	16	early morning
7	79	Franklin St & W Broadway	40.719116	-74.006667	231	evening
8	79	Franklin St & W Broadway	40.719116	-74.006667	331	morning
9	79	Franklin St & W Broadway	40.719116	-74.006667	313	noon

In [284]:
```
# drop day from the list
#weatherDf2 = weatherDf
#weatherDf2.drop(['Day'], axis=1)
#weatherDf.drop(['Day', 'count'], axis=1)
#weatherDf.drop('Day', inplace=True, axis=1)
#weatherDf.drop('count', inplace=True, axis=1)
weatherDf
```

Out[284]:

	Day	AvgTemp	AvgDP	AvgH	AvgW	AvgP	P	count
0	1	59.2	53.1	81.3	8.0	29.7	0.00	40592
1	2	48.0	23.3	38.7	10.7	29.8	0.12	35739
2	3	34.9	12.8	41.1	6.4	30.3	0.00	31105
3	4	26.7	6.2	42.8	5.5	30.5	0.00	15646
4	5	28.6	1.2	33.3	4.3	30.6	0.00	15919
5	6	36.9	17.7	47.0	5.3	30.5	0.00	32467
6	7	47.5	41.7	80.5	4.3	30.2	0.00	29683
7	8	54.7	34.0	49.6	11.5	29.9	0.16	43469
8	9	55.2	22.6	29.2	10.2	29.9	0.00	45252
9	10	37.4	25.8	66.1	7.6	29.9	0.00	18412

10	11	26.2	9.7	50.0	5.8	30.2	0.24	15185
11	12	27.7	9.4	46.7	4.9	30.3	0.00	13442
12	13	30.1	11.8	48.3	6.2	30.4	0.00	27417
13	14	30.8	27.5	88.1	14.1	29.6	0.32	0
14	15	25.9	15.7	65.3	8.3	29.6	0.71	0
15	16	33.3	15.0	49.3	8.7	30.0	0.00	0
16	17	38.1	19.1	47.5	8.1	30.2	0.00	7098
17	18	36.8	27.7	70.9	12.2	30.2	0.00	4106
18	19	41.9	24.3	52.2	9.8	30.2	0.07	10550
19	20	44.4	19.1	38.2	5.2	30.1	0.00	27378
20	21	50.4	29.0	44.5	5.6	29.9	0.00	36824
21	22	38.4	12.2	34.7	4.8	30.1	0.00	26927
22	23	35.0	9.2	37.9	6.4	30.5	0.00	29885
23	24	44.8	28.6	53.9	9.0	30.3	0.00	34018
24	25	50.7	38.3	63.4	7.8	30.2	0.00	29646
25	26	41.1	35.3	79.8	12.9	30.4	0.10	19893
26	27	44.7	41.8	89.7	7.0	30.1	0.02	26301
27	28	44.4	42.5	93.1	9.9	29.9	0.42	21589
28	29	50.3	34.9	60.1	4.7	30.1	0.43	42430
29	30	44.5	27.0	51.0	6.9	30.2	0.00	39746
30	31	40.4	37.8	91.0	14.9	30.0	0.33	6946

In [295]:
```
# Now let's setup a test and train for the month of March 2017
from sklearn.model_selection import train_test_split
#X_train, X_test, y_train, y_test = train_test_split(weatherDf.drop(co
lumns = ['count'], axis=1), weatherDf['count'], test_size=0.3)

#X = np.array(weatherDf.drop(weatherDf[['Day', 'count']], 1))
X = np.array(weatherDf.drop(weatherDf[['Day']], 1))

y = np.array(weatherDf['count'])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.2)
```

In [ ]:
```

```

In [296]:
```
def rmsle(y, y_):
    log1 = np.nan_to_num(np.array([np.log(v + 1) for v in y]))
    log2 = np.nan_to_num(np.array([np.log(v + 1) for v in y_]))
    calc = (log1 - log2) ** 2
    return np.sqrt(np.mean(calc))
```

In [297]:
```
from sklearn.ensemble import RandomForestRegressor

clf = RandomForestRegressor(n_estimators= 200, max_depth=8)
clf.fit(X_train, y_train)
print('Random Forest accuracy => ', clf.score(X_test, y_test), "\nRMSL
E => ", rmsle(y_test, clf.predict(X_test)))
```

```
Random Forest accuracy =>  0.9499141836255015
RMSLE =>  0.17547251631163066
```

In [290]:
```
clf.feature_importances_
```

Out[290]:
```
array([0.06088715, 0.01630025, 0.01173946, 0.02007936, 0.02571686,
       0.01032499, 0.85495193])
```

In [291]:
```
weatherDf.columns
```

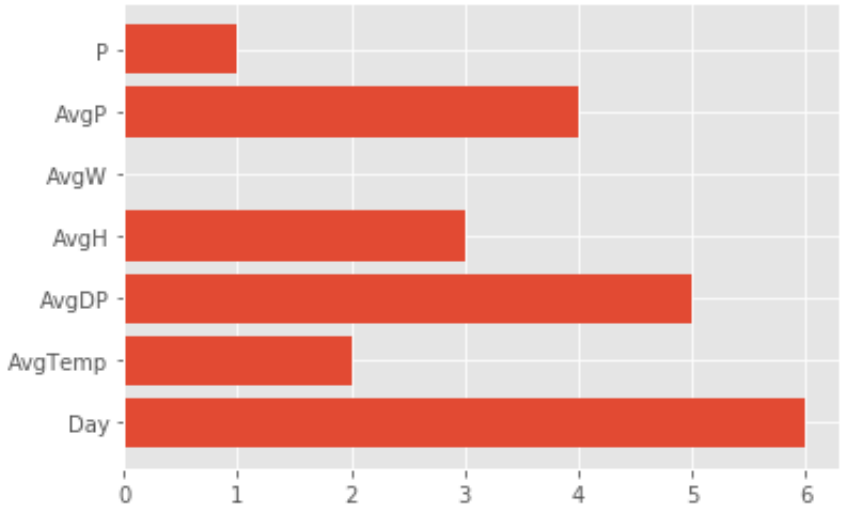
Out[291]:
```
Index(['Day', 'AvgTemp', 'AvgDP', 'AvgH', 'AvgW', 'AvgP', 'P', 'coun
t'], dtype='object')
```

In [280]:
```
dfTest = weatherDf
dfTest.drop(['count'], axis=1, inplace=True)
dfTest.columns
```

Out[280]:
```
Index(['Day', 'AvgTemp', 'AvgDP', 'AvgH', 'AvgW', 'AvgP', 'P'], dtyp
e='object')
```

In [201]:
```
plt.barh(dfTest.columns, clf.feature_importances_.argsort())
```

Out[201]: <BarContainer object of 7 artists>



In [205]:
```
#y_pred=clf.predict(X_test)
```

In [206]:
```
#from sklearn import metrics
#print("Accuracy:",metrics.accuracy_score(y_test,y_pred))
```

In [ ]: