Project Final Report (project group 8)

-Lijun Yang, Ruoyi Li, Yuheng Zhong

Project Objectives:

Our project is to analyze status of operation for Taxi in New York City from different aspects of inspection. First, our goal is to explore whether some external factors, like weather, time span, season, and different areas, may affect the demand of Taxi in NYC. Second, our team is going to find the change of number of taxi ride in NYC along the day according to different areas. Finally, we will analyze the distribution of demand and weekly revenue.

By doing this analysis, we will have a clearly taxi market demand distribution map in different time period, weekday, holidays, different seasons, and different weather. For taxi driver's perspective, we hope that we can provide them a reference that where and when they should go can pick up more customers and make more revenue. It also provides customers a reference about rush hour duration and which area might has a heavy traffic during the rush hours.

Data Description

In this project, we are going to employ two datasets related to NYC taxi analysis.

The first one is NYC taxi trip descriptive dataset in 2016 with 16 million records published at NYC open data official website(https://data.cityofnewyork.us/Transportation/2016-Green-Taxi-Trip-Data/hvrh-b6nb (https://data.cityofnewyork.us/Transportation/2016-Green-Taxi-Trip-Data/hvrh-b6nb)). There are 23 fields available such as pickup longitude/latitude, drop off longitude/latitude, pickup datetime, drop off datetime, total_amount_fee, passenger cout and trip distance in the dataset. We want to figure out the relationship between NYC taxi demand and other factors the dataset provides.

For the second dataset, we scraped 2016 NYC weather data From w2.weather.gov. This data set has 7 fields including date, max/min/avg temperature, precipitation, snow fall, snow depth and average wind speed. We assume that various weather factors may affect the transportation situation and the demand of taxi in NYC.

Progress Summary:

1. Clean taxi data
    - find and select any outliers and drop them
    - find any missing data
    - change string columns to numerical
    - drop any meaningless columns
2. Clean weather data
    - drop any meaningless columns
    - change string columns to numerical
    - extract useful infomation from weather description
3. Merge data
    - group weather data by day
    - group weather data by hour
    - group taxi data by day
    - group taxi data by hour
    - merge taxi and weather data
4. Data visualization
    - create bar chart for demand by hours
    - create bar chart for distance by hours
    - create scatter plot for demand by days
    - create scatter plot for demand by weathers
    - create scatter plot for demand by seasons
    - explain each graph
5. Demand Distrubution Analysis
    - create scatter plot for demand distrubution analysis
    - create geographic map for pickup & dropoff area analysis
6. Weekly Revenue Analysis
    - find any factors that influence revenue
7. Conclusion

Question:

1. Does weather make effects on taxi demand?
2. How taxi revenue changes seasonally?
3. How demand and average distance of trips change hourly?
4. Which area shows a high demand in different time period?
5. Which area is the most popular destination?
6. What is the trend of weekly revenue in 2016?

Direction of Analysis:

In our project, all research and analysis should focus on how taxi demand changes in NYC.

Firstly, we will focus on analyzing the relationship between taxi demand and other external factors such as temperature, precipitation, snow fall. Several seaborn visualization graphs will be built to support the analysis.

Secondly, we will build serveral charts to analyze the relationship between demand in specific area and different time period, so that we can know which area has high taxi demand during the communting time, lunch time, dinner time, and midnight time. What is more, we are going to build barplots to find the impilict relationship between demand, average distance and different time in a day.

The last but not the least, we will focus on analysis distrubution difference between working day and holiday, and geographic distrubution of pickup and dropoff records.

Coding

In [1]:
```
import pandas as pd
import datetime
import numpy as np
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt
```

Read 2016 NYC taxi dataset

In [2]:
```
#read taxi data
taxi_raw = pd.read_csv('/Users/reggieyang/Desktop/Python/Python Project/
Dataset/2016_Green_Taxi_Trip_Data.csv')
```

In [3]:
```
#taxi data shape
taxi_raw.shape
```

Out[3]:
```
(16385532, 23)
```

In [4]:
```
#create copy file, check head
taxi = taxi_raw.copy()
taxi.head()
```

Out[4]:

	VendorID	lpep_pickup_datetime	Lpep_dropoff_datetime	Store_and_fwd_flag	RateCodeID	Pick
0	2	03/24/2016 11:14:54 AM	03/24/2016 11:20:54 AM	N	1	
1	2	03/24/2016 11:08:43 AM	03/24/2016 11:14:18 AM	N	1	
2	2	03/24/2016 11:26:11 AM	03/24/2016 11:42:36 AM	N	1	
3	2	03/24/2016 11:45:05 AM	03/24/2016 11:55:36 AM	N	1	
4	2	03/24/2016 11:01:44 AM	03/24/2016 11:09:00 AM	N	1	

5 rows × 23 columns

1. Clean taxi data

In [5]:
```
#describe column data, check if there are some outliers
taxi.describe()
```

Out[5]:

	VendorID	RateCodeID	Pickup_longitude	Pickup_latitude	Dropoff_longitude	Dropoff_
count	1.638553e+07	1.638553e+07	9.018030e+06	9.018030e+06	9.018030e+06	9.018(
mean	1.790590e+00	1.094279e+00	-7.381650e+01	4.067972e+01	-7.383075e+01	4.068(
std	4.068873e-01	7.891883e-01	3.013607e+00	1.655605e+00	2.815466e+00	1.546(
min	1.000000e+00	1.000000e+00	-1.152825e+02	0.000000e+00	-1.153322e+02	0.000(
25%	2.000000e+00	1.000000e+00	-7.396101e+01	4.069426e+01	-7.396825e+01	4.069(
50%	2.000000e+00	1.000000e+00	-7.394657e+01	4.074608e+01	-7.394551e+01	4.074(
75%	2.000000e+00	1.000000e+00	-7.391886e+01	4.080197e+01	-7.391222e+01	4.079(
max	2.000000e+00	9.900000e+01	0.000000e+00	4.316801e+01	0.000000e+00	4.811(

In [6]:
```
#describe column data, check if there are some outliers
taxi.isnull().sum()
```

Out[6]:
```
VendorID                       0
lpep_pickup_datetime           0
Lpep_dropoff_datetime          0
Store_and_fwd_flag             0
RateCodeID                     0
Pickup_longitude         7367502
Pickup_latitude          7367502
Dropoff_longitude        7367502
Dropoff_latitude         7367502
Passenger_count                0
Trip_distance                  0
Fare_amount                    0
Extra                          0
MTA_tax                        0
Tip_amount                     0
Tolls_amount                   0
Ehail_fee               16385532
improvement_surcharge          0
Total_amount                   0
Payment_type                   0
Trip_type                    472
PULocationID             9018030
DOLocationID             9018030
dtype: int64
```

In [7]:
```
#1. latitude and longitude of 7 million rows are NA, but we should keep
 it for demand analysis
#drop rows that are not in NYC
#checking NYC surround longitude and latitude on https://gps-coordinate
s.org/
#(-75, -72)
#(40,41)

xlim = (40,41)
ylim = (-75,-72)
taxi = taxi[(taxi['Pickup_latitude'].between(xlim[0],xlim[1])) | (taxi[
'Pickup_latitude'].isnull())]
taxi = taxi[(taxi['Pickup_longitude'].between(ylim[0],ylim[1])) | (taxi[
'Pickup_longitude'].isnull())]
taxi = taxi[(taxi['Dropoff_latitude'].between(xlim[0],xlim[1])) | (taxi[
'Dropoff_latitude'].isnull())]
taxi = taxi[(taxi['Dropoff_longitude'].between(ylim[0],ylim[1])) | (taxi
['Dropoff_longitude'].isnull())]
```

In [8]:
```
#2. min value of Total_amout,Tolls_amount,Tip_amount,Fare_amount is nega
tive. Clearly, there are some outliers.
taxi = taxi[taxi['Total_amount']>0]
taxi = taxi[taxi['Tolls_amount']>=0]
taxi = taxi[taxi['Tip_amount']>=0]
taxi = taxi[taxi['Fare_amount']>=0]
```

In [9]:
```
#3. Trip_distance should greater than 0
taxi = taxi[taxi['Trip_distance']>0]
```

In [10]:
```
#4. passengers should less or equal to 5 (including if such passenger is under the age of seven)
taxi = taxi[taxi['Passenger_count']<=5]
```

In [11]:
```
#drop useless columns
taxi.drop(['PULocationID','DOLocationID','Ehail_fee'],axis=1,inplace=True)
```

2. Extract and clean 2016 NYC hourly and daily weather data

Extract hourly weather data from i-weather.com

In [12]:
```
#extract 2016 NYC daily weather data and merge them into yearly weather
 data.
begin = datetime.date(2016,1,1)
end = datetime.date(2016,12,31)

d = begin
delta = datetime.timedelta(days=1)
year_weather = pd.DataFrame()
while d <= end:
    weather = pd.read_html('https://i-weather.com/weather/new-york/histo
ry/daily-history/?gid=5128581&station=19438&date={}&language=english&cou
ntry=us-united-states'
                          .format(d.strftime("%Y-%m-%d")))[6]
    weather['Date']=d.strftime("%Y-%m-%d")
    year_weather = year_weather.append(weather)
    d += delta
year_weather[:10]
```

Out[12]:

	Time	Temperature	Relative Temperature	Wind	Wind Gust	Rel. humidity	Dew Point	Pressure	Icon	
0	00:51	6°C	4°C	Variable at 7 Km/h	NaN	49%	-4°C	1018.0mb	NaN	document.w
1	01:51	5°C	4°C	Variable at 6 Km/h	NaN	53%	-4°C	1018.0mb	NaN	document.w
2	02:51	5°C	3°C	280°7 Km/h	NaN	57%	-3°C	1018.0mb	NaN	document.w
3	03:51	5°C	2°C	280°15 Km/h	NaN	57%	-3°C	1018.0mb	NaN	document.w
4	04:51	4°C	0°C	270°17 Km/h	NaN	61%	-3°C	1017.0mb	NaN	document.w
5	05:51	4°C	1°C	290°11 Km/h	NaN	61%	-3°C	1018.0mb	NaN	document.w
6	06:51	4°C	4°C	Calm	NaN	61%	-3°C	1018.0mb	NaN	document.w
7	07:51	4°C	1°C	280°11 Km/h	NaN	56%	-4°C	1018.0mb	NaN	document.w
8	08:51	4°C	1°C	260°11 Km/h	NaN	56%	-4°C	1018.0mb	NaN	document.w
9	09:51	4°C	2°C	290°7 Km/h	NaN	56%	-4°C	1018.0mb	NaN	document.w

Clean weather data

In [13]:

```
#reset index and drop useless columns
year_weather.reset_index(level=0,drop=True,inplace=True)
year_weather.drop('Icon',axis=1,inplace=True)
year_weather.drop('Wind Gust',axis=1,inplace=True)
hour_weather = year_weather.copy()
hour_weather.head()
```

Out[13]:

	Time	Temperature	Relative Temperature	Wind	Rel. humidity	Dew Point	Pressure	D
0	00:51	6°C	4°C	Variable at 7 Km/h	49%	-4°C	1018.0mb	document.write(Icons.GetS
1	01:51	5°C	4°C	Variable at 6 Km/h	53%	-4°C	1018.0mb	document.write(Icons.GetS
2	02:51	5°C	3°C	280°7 Km/h	57%	-3°C	1018.0mb	document.write(Icons.GetS
3	03:51	5°C	2°C	280°15 Km/h	57%	-3°C	1018.0mb	document.write(Icons.GetS
4	04:51	4°C	0°C	270°17 Km/h	61%	-3°C	1017.0mb	document.write(Icons.GetS

In [14]:

```
#change string columns to numerical columns
hour_weather['hour']=hour_weather['Time'].str.extract('(\d\d):\d\d')

hour_weather.head()
hour_weather['Wind_km/h']=hour_weather['Wind'].str.extract('\d°(\d*)\sKm/
h')
hour_weather['Wind_km/h'] = hour_weather['Wind_km/h'].fillna(0).astype(
'int64')

hour_weather['Humidity_percentage'] = hour_weather['Rel. humidity'].str.
extract('(\d+)%').astype('int64')

hour_weather['Temperature_c'] = hour_weather['Temperature'].str.extract(
'(\d+)°C').astype('int64')
hour_weather['Relative Temperature_c'] = hour_weather['Relative Temperat
ure'].str.extract('(\d+)°C').astype('int64')
```

In [15]:

```
#extract weather description, and map each description to each row.
hour_weather['weather'] = hour_weather['DescriptionDetails'].str.extract
("GetShortDescription\((\d+),")
hour_weather[hour_weather['weather']=='94']
hour_weather['weather'].value_counts()
weather_dict = {'1':'Clear','2':'Partly cloudy','3':'Few clouds','4':'Cl
oudy','7':'Rain','10':'Thunder','26':'Snow','94':'Freezing fog'}
hour_weather['weather'] = hour_weather['weather'].map(weather_dict)
```

In [16]:
```
#drop useless columns
hour_weather.drop('Dew Point',axis=1,inplace=True)
hour_weather.drop('Pressure',axis=1,inplace=True)
hour_weather.drop('DescriptionDetails',axis=1,inplace=True)
hour_weather.drop('Relative Temperature',axis=1,inplace=True)
hour_weather.drop('Temperature',axis=1,inplace=True)
hour_weather.drop('Wind',axis=1,inplace=True)
hour_weather.drop('Time',axis=1,inplace=True)
hour_weather.drop('Rel. humidity',axis=1,inplace=True)
```

In [17]:
```
#drop duplicate rows
hour_weather['merge']= hour_weather['Date']+' '+hour_weather['hour']
hour_weather['merge'].nunique()
hour_weather.drop_duplicates('merge',keep='first', inplace=True)
hour_weather['merge'] = pd.to_datetime(hour_weather['merge']).dt.strftim
e('%m/%d/%Y %H')
hour_weather.head()
```

Out[17]:

	Date	hour	Wind_km/h	Humidity_percentage	Temperature_c	Relative Temperature_c	weather	n
0	2016-01-01	00	0	49	6	4	Cloudy	01/01.
1	2016-01-01	01	0	53	5	4	Cloudy	01/01.
2	2016-01-01	02	7	57	5	3	Cloudy	01/01.
3	2016-01-01	03	15	57	5	2	Cloudy	01/01.
4	2016-01-01	04	17	61	4	0	Cloudy	01/01.

2. Extract daily weather data

In [18]:

```
#extract 2016 NYC daily weather data
daily_weather = pd.read_csv('/Users/reggieyang/Desktop/Python/Python Pro
ject/Dataset/weather_data_nyc_centralpark_2016.csv')
daily_weather.head(10)
```

Out[18]:

	date	maximum temperature	minimum temperature	average temperature	precipitation	snow fall	snow depth
0	1-1-2016	42	34	38.0	0.00	0.0	0
1	2-1-2016	40	32	36.0	0.00	0.0	0
2	3-1-2016	45	35	40.0	0.00	0.0	0
3	4-1-2016	36	14	25.0	0.00	0.0	0
4	5-1-2016	29	11	20.0	0.00	0.0	0
5	6-1-2016	41	25	33.0	0.00	0.0	0
6	7-1-2016	46	31	38.5	0.00	0.0	0
7	8-1-2016	46	31	38.5	0.00	0.0	0
8	9-1-2016	47	40	43.5	T	0.0	0
9	10-1-2016	59	40	49.5	1.80	0.0	0

Clean daily weather data

In [19]:

```
#processing daily weather data
#change T to 0, T stands for trace amouts of precipitation, and change d
ata type to float
daily_weather['precipitation'] = daily_weather['precipitation'].str.repl
ace('T','0.00')
daily_weather['snow fall'] = daily_weather['snow fall'].str.replace('T',
'0.00')
daily_weather['snow depth'] = daily_weather['snow depth'].str.replace(
'T','0.00')

daily_weather[['precipitation','snow depth','snow fall','maximum tempera
ture', 'minimum temperature','average temperature']]=daily_weather[['pre
cipitation','snow depth','snow fall','maximum temperature', 'minimum tem
perature','average temperature']].astype(float)
daily_weather.info()
daily_weather['date'] = pd.to_datetime(daily_weather['date']).dt.strftim
e('%m/%d/%Y')
daily_weather.rename({'date':'Date'},axis=1,inplace=True)
daily_weather.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 366 entries, 0 to 365
Data columns (total 7 columns):
date                   366 non-null object
maximum temperature    366 non-null float64
minimum temperature    366 non-null float64
average temperature    366 non-null float64
precipitation          366 non-null float64
snow fall              366 non-null float64
snow depth             366 non-null float64
dtypes: float64(6), object(1)
memory usage: 20.1+ KB
```

Out[19]:

	Date	maximum temperature	minimum temperature	average temperature	precipitation	snow fall	snow depth
0	01/01/2016	42.0	34.0	38.0	0.0	0.0	0.0
1	02/01/2016	40.0	32.0	36.0	0.0	0.0	0.0
2	03/01/2016	45.0	35.0	40.0	0.0	0.0	0.0
3	04/01/2016	36.0	14.0	25.0	0.0	0.0	0.0
4	05/01/2016	29.0	11.0	20.0	0.0	0.0	0.0

3. Merge data

merge taxi data with daily weather data

In [42]:
```
#processing taxi data to seasonal vs demand
taxi_daily = taxi[['lpep_pickup_datetime','Trip_distance','Total_amount'
]]
taxi_daily.head(20)

#extract date
taxi_daily.rename({'lpep_pickup_datetime':'Date'},axis=1,inplace=True)
taxi_daily['Date'] = taxi_daily['Date'].str.extract('(\d\d/\d\d/\d\d\d
\d)\s\d\d:')
#group by Date
taxi_a = taxi_daily.groupby("Date",as_index=False).agg('count')['Trip_di
stance']
taxi_demand = taxi_daily.groupby("Date",as_index=False).agg({'Trip_dista
nce':np.sum,'Total_amount':np.sum})
taxi_demand.loc['count'] = taxi_a
taxi_demand.head()
```

```
/Users/reggieyang/anaconda3/lib/python3.6/site-packages/pandas/core/fra
me.py:4025: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy
  return super(DataFrame, self).rename(**kwargs)
/Users/reggieyang/anaconda3/lib/python3.6/site-packages/ipykernel_launc
her.py:7: 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/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy
  import sys
```

Out[42]:

	Date	Trip_distance	Total_amount
0	01/01/2016	203348.17	945350.980001
1	01/02/2016	131438.25	634015.870000
2	01/03/2016	127470.38	607421.020000
3	01/04/2016	109689.71	573916.140000
4	01/05/2016	108167.97	570210.080000

In [21]:
```
#merge weather & taxi_demand
taxi_weather = taxi_demand.merge(daily_weather,on='Date',how = 'inner')
taxi_weather.head()
```

Out[21]:

	Date	Trip_distance	Total_amount	count	maximum temperature	minimum temperature	average temperature	preci
0	01/01/2016	203348.17	945350.980001	62243	42.0	34.0	38.0	
1	01/02/2016	131438.25	634015.870000	45351	59.0	44.0	51.5	
2	01/03/2016	127470.38	607421.020000	42678	52.0	39.0	45.5	
3	01/04/2016	109689.71	573916.140000	42072	79.0	61.0	70.0	
4	01/05/2016	108167.97	570210.080000	40832	51.0	45.0	48.0	

merge taxi data with hourly weather data

In [22]:
```
#change datetime format in taxi data
taxi_hour = taxi.copy()
taxi_hour = taxi_hour[['lpep_pickup_datetime','Trip_distance','Total_amo
unt']]
taxi_hour['lpep_pickup_datetime'] = pd.to_datetime(taxi_hour['lpep_picku
p_datetime']).dt.strftime('%m/%d/%Y %H')
#group by hour
taxi_b = taxi_hour.groupby("lpep_pickup_datetime",as_index=False).agg('c
ount')['Trip_distance']
taxi_hour_demand = taxi_hour.groupby("lpep_pickup_datetime",as_index=Fal
se).agg({'Trip_distance':np.sum,'Total_amount':np.sum})
taxi_hour_demand['count'] = taxi_b
```

In [23]:
```
#merge taxi_hour and weather hour data
taxi_hour_demand.rename({'lpep_pickup_datetime':'merge'},axis=1,inplace=
True)
taxi_hour_demand = taxi_hour_demand.merge(hour_weather,on='merge',how =
'inner')
taxi_hour_demand.drop('merge',axis=1,inplace=True)
taxi_hour_demand = taxi_hour_demand[['Date','hour',   'Wind_km/h',
       'Humidity_percentage', 'Temperature_c', 'Relative Temperature_c',
       'weather','Trip_distance', 'Total_amount', 'count']]
taxi_hour_demand.head()
```

Out[23]:

	Date	hour	Wind_km/h	Humidity_percentage	Temperature_c	Relative Temperature_c	weather	Trip_d
0	2016-01-01	00	0	49	6	4	Cloudy	2
1	2016-01-01	01	0	53	5	4	Cloudy	2
2	2016-01-01	02	7	57	5	3	Cloudy	2
3	2016-01-01	03	15	57	5	2	Cloudy	1
4	2016-01-01	04	17	61	4	0	Cloudy	1

In [24]:
```
#export hourly weather and daily weather

#taxi_hour_demand.to_csv('/Users/reggieyang/Desktop/Python/Python Projec
t/Dataset/taxi_hour_demand_weather.csv')
#taxi_weather.to_csv('/Users/reggieyang/Desktop/Python/Python Project/Da
taset/taxi_daily_demand_weather.csv')
```

4. Data Visualization

Daily Data

How weather impacts taxi demand?

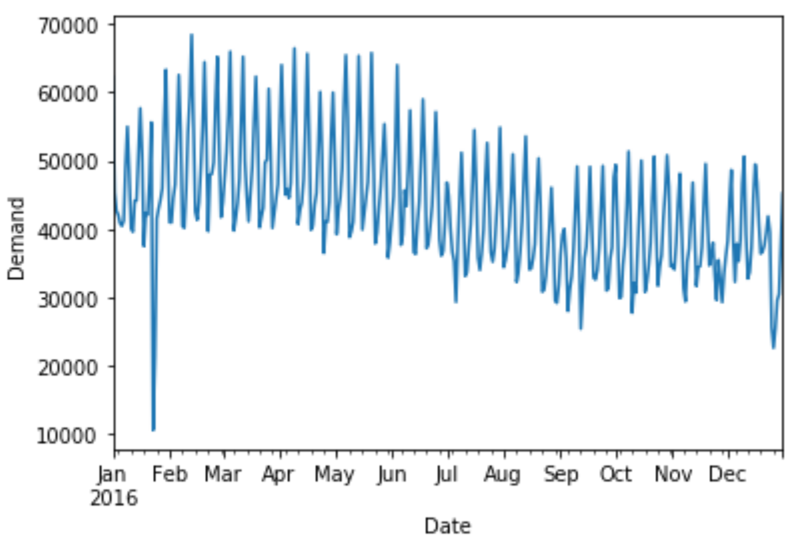
In [25]:
```
df=pd.read_csv('/Users/reggieyang/Desktop/Python/Python Project/Dataset/final dataset/taxi_daily_demand_weather.csv',index_col=[0])
df.head()
```

Out[25]:

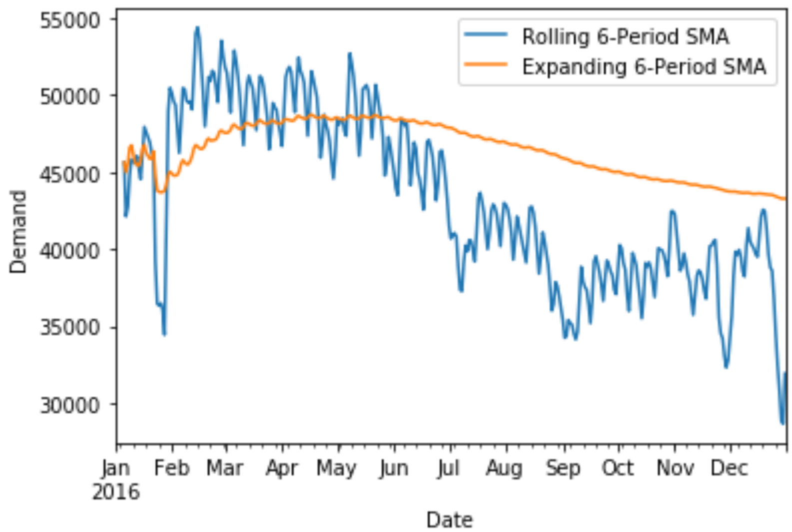
	Date	Trip_distance	Total_amount	count	maximum temperature	minimum temperature	average temperature	preci
0	01/01/2016	203348.17	945350.980001	62243	42.0	34.0	38.0	
1	01/02/2016	131438.25	634015.870000	45351	59.0	44.0	51.5	
2	01/03/2016	127470.38	607421.020000	42678	52.0	39.0	45.5	
3	01/04/2016	109689.71	573916.140000	42072	79.0	61.0	70.0	
4	01/05/2016	108167.97	570210.080000	40832	51.0	45.0	48.0	

In [26]:
```
#set datetime as index of the dataframe
df['Date']=pd.to_datetime(df['Date'],format='%m/%d/%Y')
df.set_index('Date', inplace=True)
```

In [27]:
```
# demand vs date
df['count'].plot()
plt.ylabel('Demand')
plt.show()
```

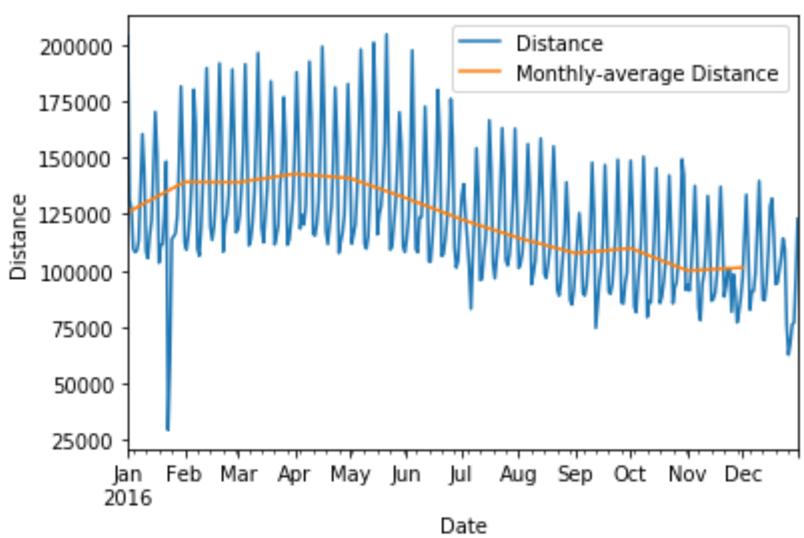


In [28]:
```
df['count'].rolling(6).mean().plot()
df['count'].expanding(6).mean().plot()
plt.legend(('Rolling 6-Period SMA','Expanding 6-Period SMA'))
plt.ylabel('Demand')
plt.show()
```



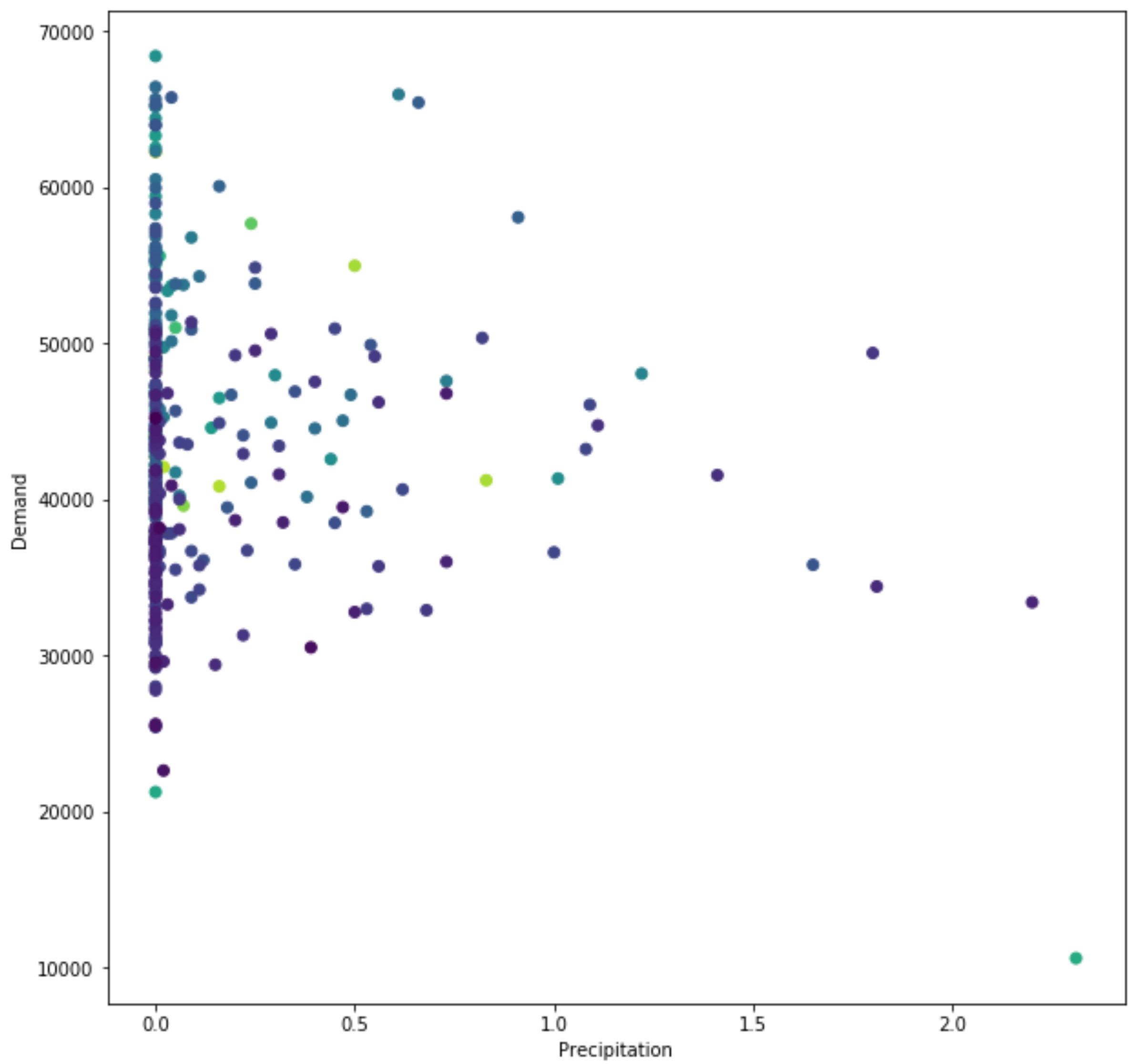
From the Smoothening plot above, we could find that there is a relationship between season and taxi demand. Taxi demand is relatively high in spring. It shows a continous decreasing trend during the summer and keep lower during Fall and Winter.

In [29]:
```
#Season Vs average distance
df['Trip_distance'].plot()
df['Trip_distance'].resample('M').mean().plot()
plt.legend(('Distance','Monthly-average Distance'))
plt.ylabel('Distance')
plt.show()
```



From the graph above, we see that the average distance of trips is continuously decreasing from spring to winter, which is in accordance with the demand change pattern.

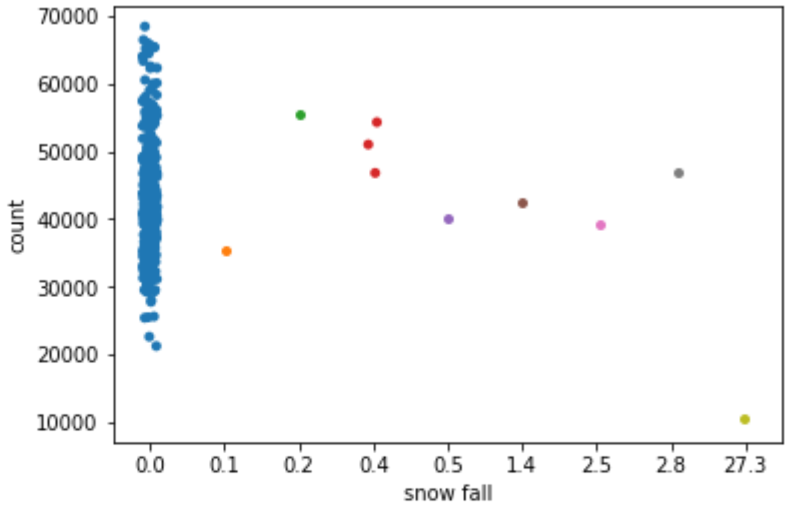
In [30]:
```
# demand vs rain
z = list(reversed([i**3 for i in sorted(df['count'])]))
cmap = matplotlib.cm.get_cmap('viridis')
normalize = matplotlib.colors.Normalize(vmin=min(z), vmax=max(z))
colors = [cmap(normalize(value)) for value in z]
fig, ax = plt.subplots(figsize=(10,10))
plt.scatter(x=df['precipitation'],y=df['count'],color=colors)
plt.xlabel('Precipitation')
plt.ylabel('Demand')
plt.show()
```



From the scatter plot, raining doesn'affect the taxi demand very obviously.

In [31]:
```
# demand vs snow
sns.stripplot(x=df['snow fall'],y=df['count'])
```

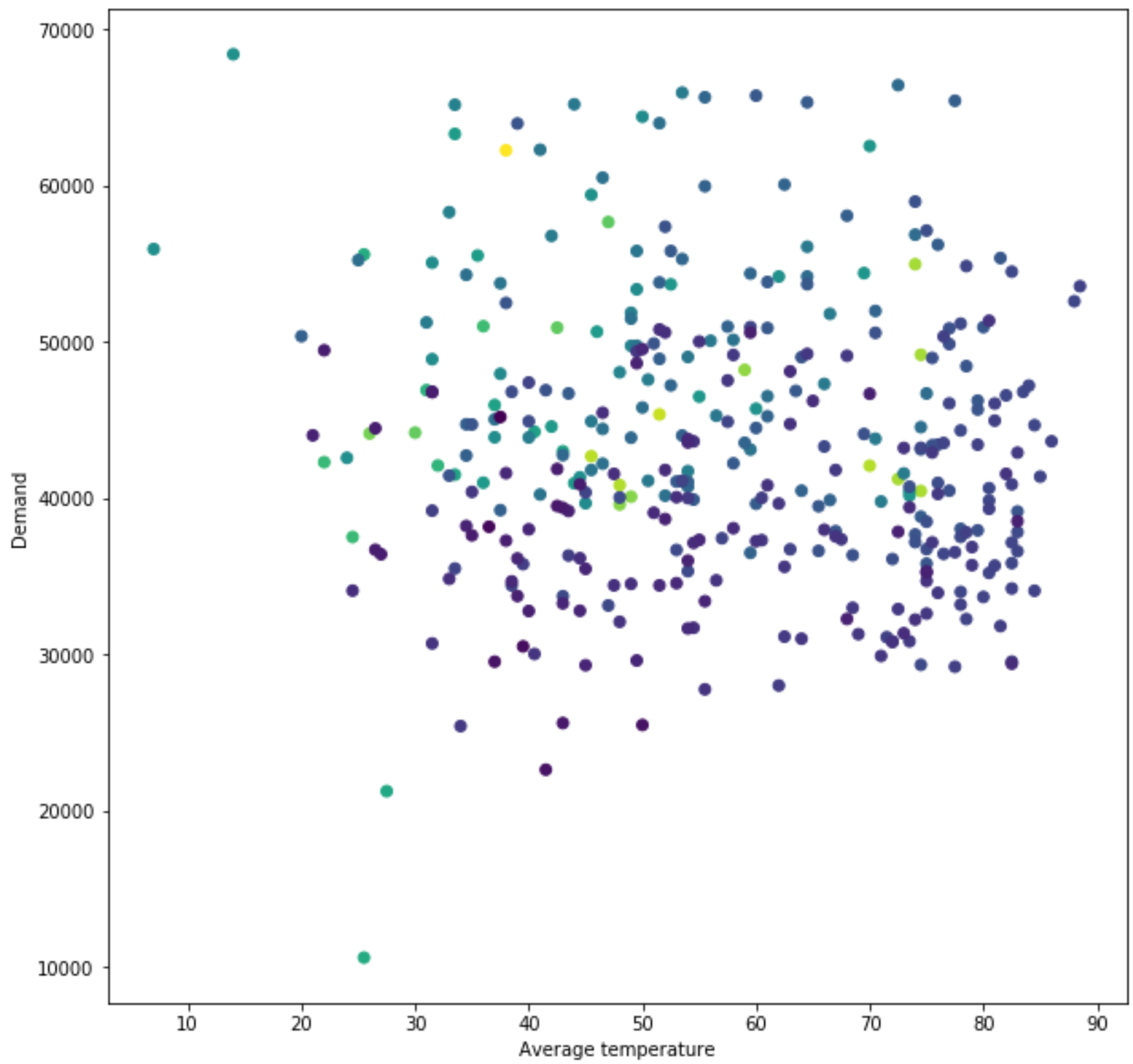
Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x1a2d581828>



There are only a few snowing day in the dataset. I can still find that if the snow is heavy, the taxi demand will decrease dramatically.

In [32]:

```
# Demand vs avg temperature
z = list(reversed([i**3 for i in sorted(df['count'])]))
cmap = matplotlib.cm.get_cmap('viridis')
normalize = matplotlib.colors.Normalize(vmin=min(z), vmax=max(z))
colors = [cmap(normalize(value)) for value in z]
fig, ax = plt.subplots(figsize=(10,10))
plt.scatter(x=df['average temperature'],y=df['count'],color=colors)
plt.xlabel('Average temperature')
plt.ylabel('Demand')
plt.show()
```



No obvious relationship can be found between average temperature and taxi demand.

Hourly Data

How demand and average distance of trips change hourly?

In [33]:
```
df=pd.read_csv('/Users/reggieyang/Desktop/Python/Python Project/Dataset/final dataset/taxi_hour_demand_weather.csv',index_col=[0])
df.head()
```

Out[33]:

	Date	hour	Wind_km/h	Humidity_percentage	Temperature_c	Relative Temperature_c	weather	Trip_c
0	2016-01-01	0	0	49	6	4	Cloudy	2
1	2016-01-01	1	0	53	5	4	Cloudy	2
2	2016-01-01	2	7	57	5	3	Cloudy	2
3	2016-01-01	3	15	57	5	2	Cloudy	1
4	2016-01-01	4	17	61	4	0	Cloudy	1

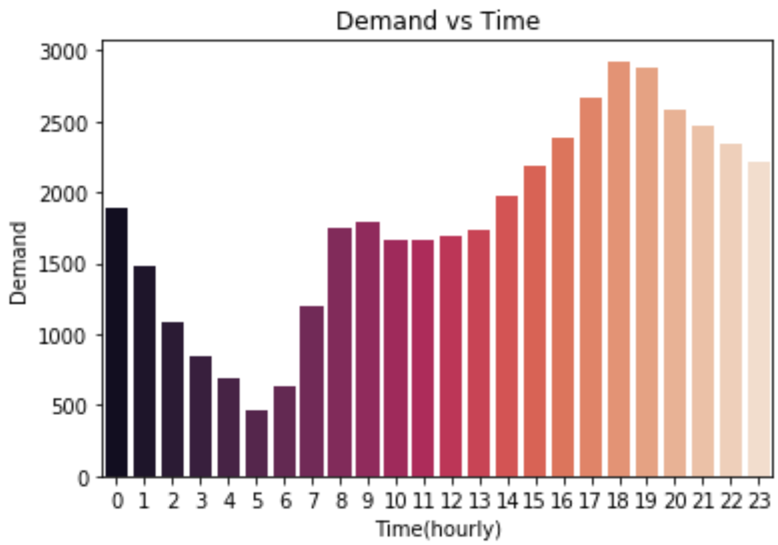
Calculate mean values of trip distance and demand

In [34]:
```
average_trip = df['Trip_distance']/df['count']
df['average_trip_distance_hour'] = average_trip
df_hourdemand = df[['hour','count']]
hour_count = df_hourdemand.groupby("hour",as_index=False).agg({'count':np.average})
```

Construct a barplot for the hourly damand change pattern

In [53]:
```
x = hour_count['hour'].tolist()
y = hour_count['count'].tolist()
fig = sns.barplot(x=x,y=y, palette = 'rocket')
fig.set(xlabel='Time(hourly)', ylabel='Demand',title='Demand vs Time')
```

Out[53]:
```
[Text(0, 0.5, 'Demand'),
 Text(0.5, 0, 'Time(hourly)'),
 Text(0.5, 1.0, 'Demand vs Time')]
```



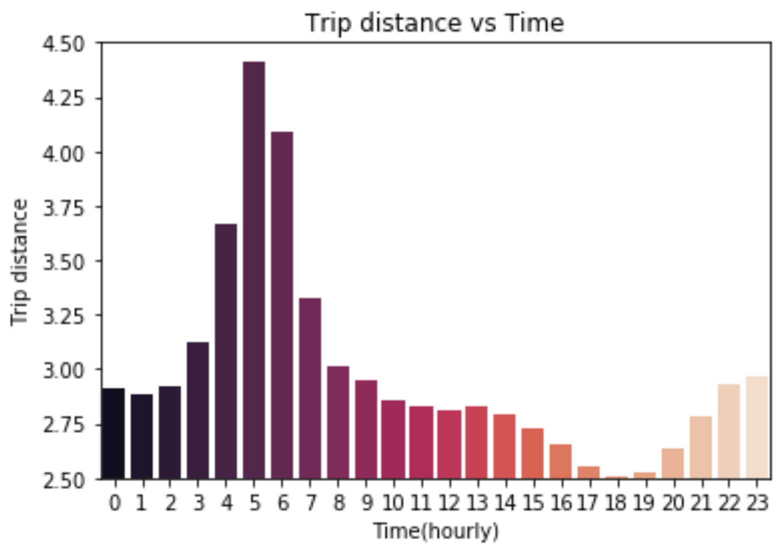
From graph, we can see that there is a huge drop of trip number between 3 am to 6am. The reason behind this phenomenon is that 3am to 5am is NYC taxi drivers' first change shift. Therefore, they often reluctant to pick up passengers in this period.

Second, the average number of taxi at evening rush hour is about twice the average number at morning rush hour. In the morning rush hour, taxi demand is basically commuting to work, while in the evening rush hour, taxi demand includes not only commuting, but also going out to have dinner and entertain with friends after work hours. Therefore, the demand of taxi shows a huge difference.

Construct another barplot for the average trip distance pattern

In [52]:
```
hour_distance = df.groupby("hour",as_index=False).agg({'average_trip_distance_hour':np.average})
x1 = hour_distance['hour'].tolist()
y1 = hour_distance['average_trip_distance_hour'].tolist()
fig2 = sns.barplot(x=x1,y=y1, palette = 'rocket')
plt.ylim(2.5,4.5)
fig2.set(xlabel='Time(hourly)', ylabel='Trip distance',title='Trip distance vs Time')
```

Out[52]:
```
[Text(0, 0.5, 'Trip distance'),
 Text(0.5, 0, 'Time(hourly)'),
 Text(0.5, 1.0, 'Trip distance vs Time')]
```



From graph, the average distance per trip from 4am to 5am is much longer than other time period. Because the change shift of taxi drivers, taxi drivers would prefer to pickup passengers who are in for a long ride - it is more profitable for them. Therefore a higher distance covered in the shift hours is shown.

Second, the average distance per trip at evening rush hour is much shorter than trips at morning rush hour. We believe that different purposes of taxi result in the difference. In the morning rush hour, passengers always use taxi to commute and catch up early flight. The commute routes are relatively long trips, and going to the airport is also a long trip. In the evening rush hour, many taxi passengers go to dinner or entertainment, instead of returing home. Therefore, the average distance is much shorter than in the morning.

5. Demand distribution

Which area shows a high demand in different time period?

In [37]:
```
taxi_location = pd.read_csv('/Users/reggieyang/Desktop/Python/Python Project/Dataset/clean_location_taxi.csv')
taxi_location.info()
taxi_location = taxi_location[['lpep_pickup_datetime','Lpep_dropoff_datetime','Pickup_longitude','Pickup_latitude','Dropoff_longitude', 'Dropoff_latitude']]
taxi_location['lpep_pickup_datetime']=pd.to_datetime(taxi_location['lpep_pickup_datetime'],format='%m/%d/%Y %I:%M:%S %p')
taxi_location.set_index('lpep_pickup_datetime',inplace=True)
taxi_location.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8712122 entries, 0 to 8712121
Data columns (total 21 columns):
Unnamed: 0                int64
VendorID                  int64
lpep_pickup_datetime      object
Lpep_dropoff_datetime     object
Store_and_fwd_flag        object
RateCodeID                int64
Pickup_longitude          float64
Pickup_latitude           float64
Dropoff_longitude         float64
Dropoff_latitude          float64
Passenger_count           int64
Trip_distance             float64
Fare_amount               float64
Extra                     float64
MTA_tax                   float64
Tip_amount                float64
Tolls_amount              float64
improvement_surcharge     float64
Total_amount              float64
Payment_type              int64
Trip_type                 float64
dtypes: float64(13), int64(5), object(3)
memory usage: 1.4+ GB
```

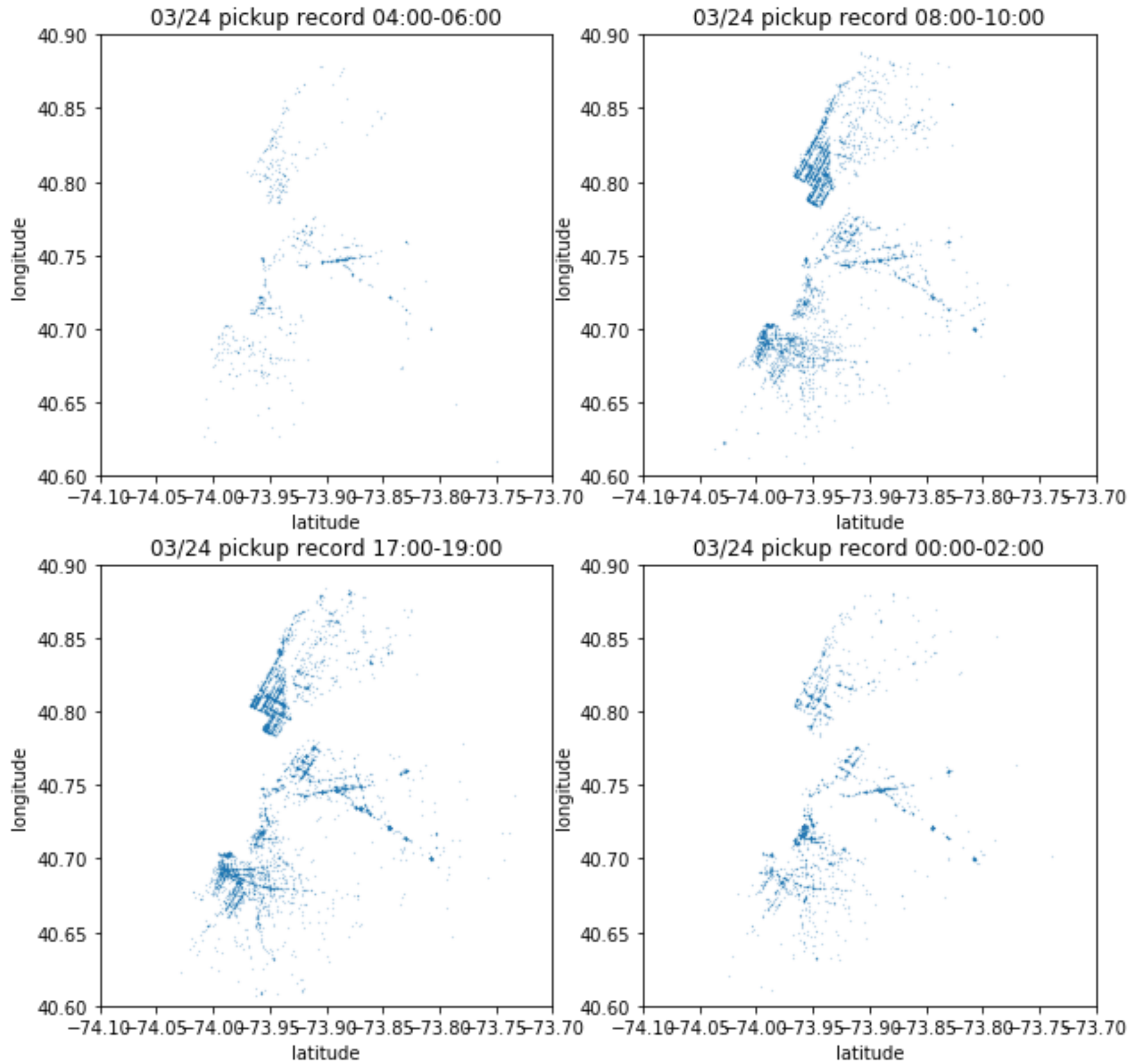
Out[37]:

lpep_pickup_datetime	Lpep_dropoff_datetime	Pickup_longitude	Pickup_latitude	Dropoff_longitude
2016-03-24 11:14:54	03/24/2016 11:20:54 AM	-73.953606	40.822086	-73.960876
2016-03-24 11:08:43	03/24/2016 11:14:18 AM	-73.966606	40.804276	-73.974014
2016-03-24 11:26:11	03/24/2016 11:42:36 AM	-73.966530	40.804249	-73.939430
2016-03-24 11:45:05	03/24/2016 11:55:36 AM	-73.936951	40.804123	-73.872467
2016-03-24 11:01:44	03/24/2016 11:09:00 AM	-73.938347	40.796387	-73.949059

5.1 Weekday Demand Distribution

In [38]:
```
#analysis working day pickup distribution(2016/03/24)
taxi_location_324 = taxi_location['2016-03-24']
#split the datetime to 4 parts: midnight (23:00-5:00) morning (6:00-11:00) afternoon(12:00-17:00) evening(17:00-23:00)
taxi_location_324_midnight=taxi_location_324['2016-03-24 04':'2016-03-24 05']
taxi_location_324_morning=taxi_location_324['2016-03-24 08':'2016-03-24 09']
taxi_location_324_afternoon=taxi_location_324['2016-03-24 17':'2016-03-24 18']
taxi_location_324_evening=taxi_location_324['2016-03-24 00':'2016-03-24 01']

#visulizing data
#extract longitude and latitude
import matplotlib.pyplot as plt
longitude1 = list(taxi_location_324_midnight.Pickup_longitude)
latitude1 = list(taxi_location_324_midnight.Pickup_latitude)
longitude2 = list(taxi_location_324_morning.Pickup_longitude)
latitude2 = list(taxi_location_324_morning.Pickup_latitude)
longitude3 = list(taxi_location_324_afternoon.Pickup_longitude)
latitude3 = list(taxi_location_324_afternoon.Pickup_latitude)
longitude4 = list(taxi_location_324_evening.Pickup_longitude)
latitude4 = list(taxi_location_324_evening.Pickup_latitude)
plt.figure(figsize = (10,10))
#subplot graph
plt.subplot(2,2,1)
plt.xlim(-74.1,-73.7)
plt.ylim(40.6,40.9)
plt.xlabel('latitude')
plt.ylabel('longitude')
plt.title('03/24 pickup record 04:00-06:00')
plt.plot(longitude1,latitude1,'.', alpha = 1, markersize = 0.2)

plt.subplot(2,2,2)
plt.xlim(-74.1,-73.7)
plt.ylim(40.6,40.9)
plt.xlabel('latitude')
plt.ylabel('longitude')
plt.title('03/24 pickup record 08:00-10:00')
plt.plot(longitude2,latitude2,'.', alpha = 1, markersize = 0.2)

plt.subplot(2,2,3)
plt.xlim(-74.1,-73.7)
plt.ylim(40.6,40.9)
plt.xlabel('latitude')
plt.ylabel('longitude')
plt.title('03/24 pickup record 17:00-19:00')
plt.plot(longitude3,latitude3,'.', alpha = 1, markersize = 0.2)

plt.subplot(2,2,4)
plt.xlim(-74.1,-73.7)
plt.ylim(40.6,40.9)
plt.xlabel('latitude')
plt.ylabel('longitude')
plt.title('03/24 pickup record 00:00-02:00')
plt.plot(longitude4,latitude4,'.', alpha = 1, markersize = 0.2)
```

```
plt.show()
```



Weekday pickup:

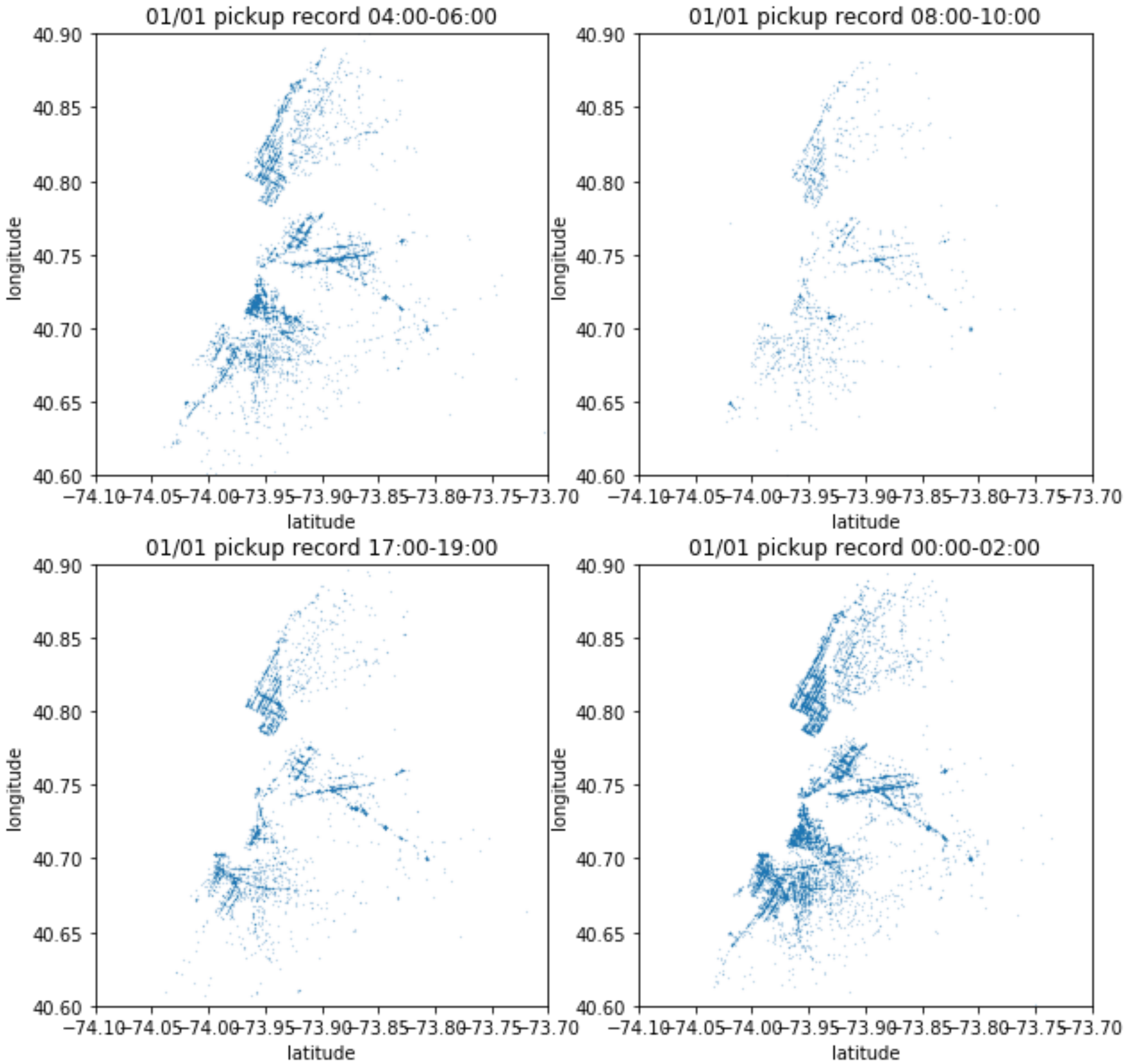
The above graph shows the NYC taxi pickup distribution at midnight, early morning, commute time in morning and afternoon. We could find that Bronx area shows highest pick up at all four time period. The green cabs cannot pick up passengers at Manhattan, so we have no data in the location for analysis. For driver perspective, if they want to make money during the night time, they would better go to Bronx area. Although Bronx area shows very high market demand at the morning and night commute time, some drivers may avoid this area since they don't like heavy traffic. They can go Queen and Brooklyn area, which has relatively high market demand but not such heavy traffic. For customer perspective, if they attend events that last very late, these graphs give them an idea about where they can find a taxi.

Another interesting thing we could find from these graphs is that the market demand is always higher at night commute time than the demand at morning commute time. By analyzing customer behavior and reading some interesting materials we could find several reasons for this phenomenon. Firstly, not everybody goes to office on time during the morning time, but they will leave office on time together at night commute time. Secondly, the most of morning pickup is going to work or school. Night time is different, most of pick up is going to dinner, event, theater, and parties.

5.2 Holiday Demand Distribution

In [39]:
```
#analysis new year pickup distribution(2016/01/01)
taxi_location_0101 = taxi_location['2016-01-01']
taxi_location_0101['2016-01-01'].shape
#split the datetime to 4 parts: midnight (23:00-5:00) morning (6:00-11:00) afternoon(12:00-17:00) evening(17:00-23:00)
taxi_location_0101_midnight=taxi_location_0101['2016-01-01 04':'2016-01-01 05']
taxi_location_0101_morning=taxi_location_0101['2016-01-01 08':'2016-01-01 09']
taxi_location_0101_afternoon=taxi_location_0101['2016-01-01 17':'2016-01-01 18']
taxi_location_0101_evening=taxi_location_0101['2016-01-01 00':'2016-01-01 01']

#extract longitude and latitude
import matplotlib.pyplot as plt
longitude1 = list(taxi_location_0101_midnight.Pickup_longitude)
latitude1 = list(taxi_location_0101_midnight.Pickup_latitude)
longitude2 = list(taxi_location_0101_morning.Pickup_longitude)
latitude2 = list(taxi_location_0101_morning.Pickup_latitude)
longitude3 = list(taxi_location_0101_afternoon.Pickup_longitude)
latitude3 = list(taxi_location_0101_afternoon.Pickup_latitude)
longitude4 = list(taxi_location_0101_evening.Pickup_longitude)
latitude4 = list(taxi_location_0101_evening.Pickup_latitude)
plt.figure(figsize = (10,10))
#subplot graph
plt.subplot(2,2,1)
plt.xlim(-74.1,-73.7)
plt.ylim(40.6,40.9)
plt.xlabel('latitude')
plt.ylabel('longitude')
plt.title('01/01 pickup record 04:00-06:00')
plt.plot(longitude1,latitude1,'.', alpha = 1, markersize = 0.2)

plt.subplot(2,2,2)
plt.xlim(-74.1,-73.7)
plt.ylim(40.6,40.9)
plt.xlabel('latitude')
plt.ylabel('longitude')
plt.title('01/01 pickup record 08:00-10:00')
plt.plot(longitude2,latitude2,'.', alpha = 1, markersize = 0.2)

plt.subplot(2,2,3)
plt.xlim(-74.1,-73.7)
plt.ylim(40.6,40.9)
plt.xlabel('latitude')
plt.ylabel('longitude')
plt.title('01/01 pickup record 17:00-19:00')
plt.plot(longitude3,latitude3,'.', alpha = 1, markersize = 0.2)

plt.subplot(2,2,4)
plt.xlim(-74.1,-73.7)
plt.ylim(40.6,40.9)
plt.xlabel('latitude')
plt.ylabel('longitude')
plt.title('01/01 pickup record 00:00-02:00')
plt.plot(longitude4,latitude4,'.', alpha = 1, markersize = 0.2)
```

```
plt.show()
```



Holiday graph:

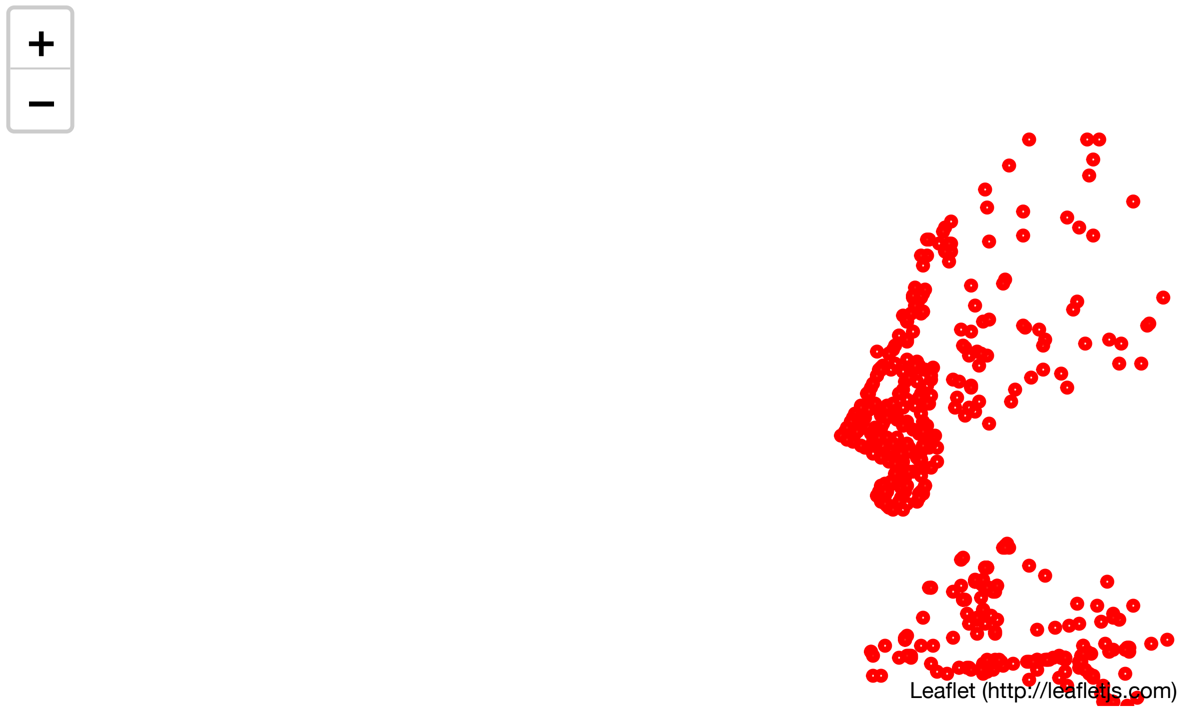
In addition to the weekday pickup distribution, we also analyzed the pickup distribution on New Year's Day as an example for holiday case. Unlike the weekday distribution, the market demand still shows a relatively high level at midnight and early morning time on New Year's Day. The market demand is lowest at the morning commute time. For driver perspective, these graphs can give them an idea about holiday pickup distribution, which can help them a lot to decide working hours and pickup area.

5.3 Geographic map (pickup & dropoff analysis)

5.3.1 pickup geographic map

In [6]:
```python
import folium
#set default map
map_1 = folium.Map(location=[40.767937,-73.922155 ],tiles='OpenStreetMa
p',
 zoom_start=11)
#show first 1000 records
for each in taxi_raw[:1000].iterrows():
    folium.CircleMarker([each[1]['Pickup_latitude'],each[1]['Pickup_long
itude']],
                        radius=2,
                        color='red',
                        popup=str(each[1]['Pickup_latitude'])+','+str(ea
ch[1]['Pickup_longitude']),
                        fill_color='#FD8A6C'
                        ).add_to(map_1)
map_1
```

Out[6]:

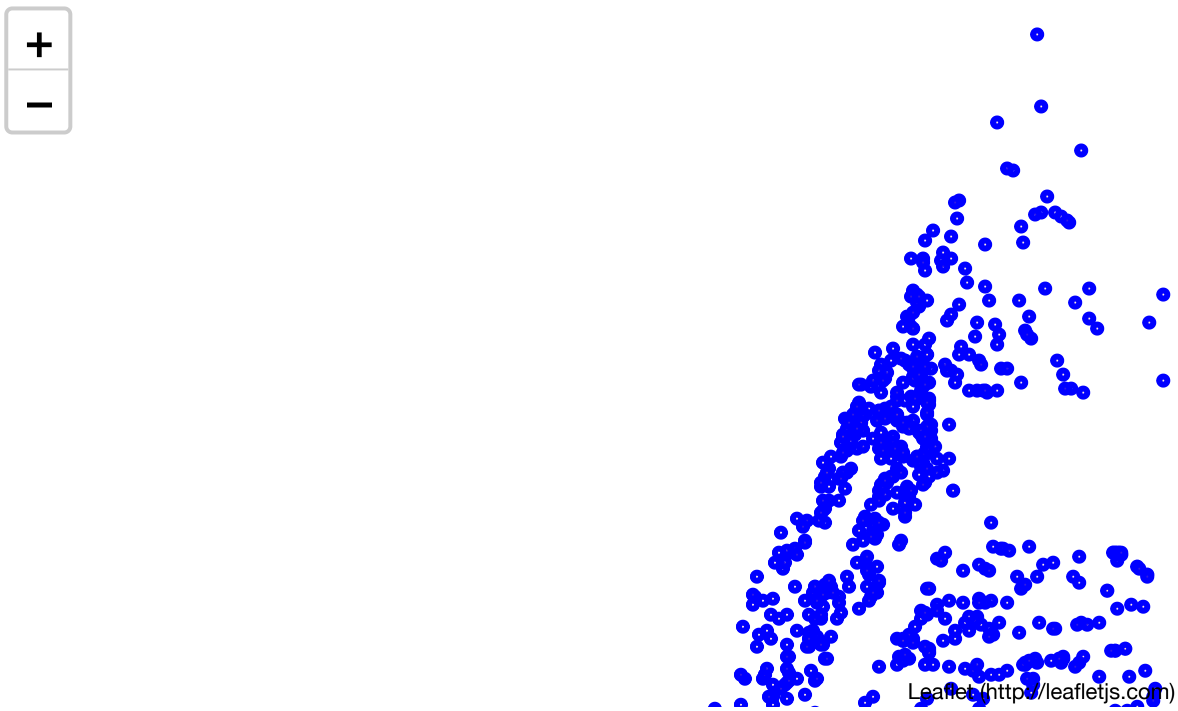


Which area is the most popular destination?

5.3.2 dropoff geographic map

In [7]:
```
#set default map
map_2 = folium.Map(location=[40.767937,-73.922155 ],tiles='OpenStreetMa
p',
 zoom_start=11)
#iterate first 1000 records
for each in taxi_raw[:1000].iterrows():
    folium.CircleMarker([each[1]['Dropoff_latitude'],each[1]['Dropoff_lo
ngitude']],
                        radius=2,
                        color='blue',
                        popup=str(each[1]['Dropoff_latitude'])+','+str(e
ach[1]['Dropoff_longitude']),
                        fill_color='#FD8A6C'
                        ).add_to(map_2)
map_2
```

Out[7]:



The two graphs above only contains first 1000 rows in the dataset. The red datapoints show the pickup records; the blue datapoints show the dropoff records.

There is no doubt that Manhattan has no datapoint in pickup graph because of the limitation of green cab in NYC. Therefore, we cannot find direct evidence of demand of Manhattan. However, from drop-off graph, we can see that Manhattan has one of the densest datapoint in NYC, which means a lot of passengers' destination is Manhattan. Thus, the demand of Manhattan and nearby district should be high too.

6. Weekly Revenue Analysis

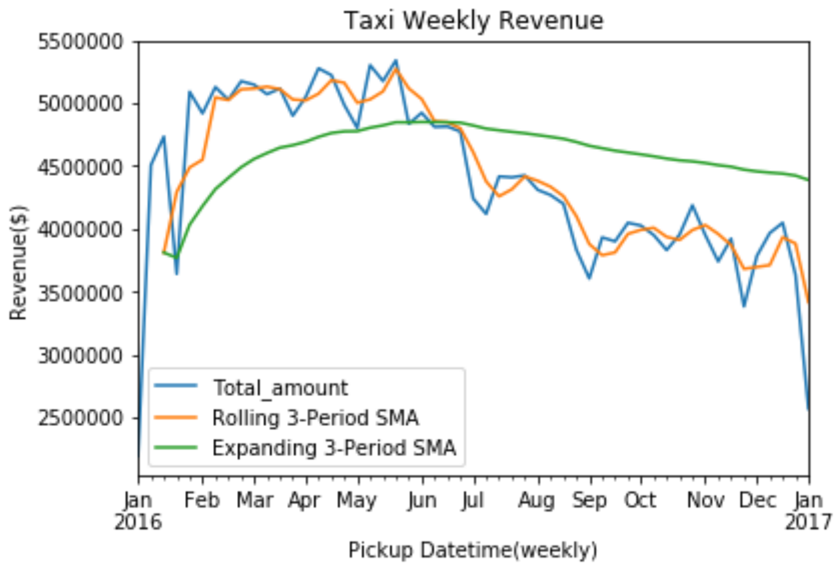
What is the trend of weekly revenue in 2016?

In [44]:
```
#read taxi_fare data
taxi_fare = pd.read_csv('/Users/reggieyang/Desktop/Python/Python Projec
t/Dataset/2016_Green_Taxi_Trip_Data.csv')
#drop any ourliers
taxi_fare = taxi_fare[taxi_fare['Total_amount']>0]
taxi_fare = taxi_fare[taxi_fare['Tolls_amount']>=0]
taxi_fare = taxi_fare[taxi_fare['Tip_amount']>=0]
taxi_fare = taxi_fare[taxi_fare['Fare_amount']>=0]
taxi_fare = taxi_fare[taxi_fare['Trip_distance']>0]
taxi_fare = taxi_fare[taxi_fare['Passenger_count']<=5]
#slice the data
taxi_fare=taxi_fare[['lpep_pickup_datetime','Total_amount']]
#set datetime to index
taxi_fare['lpep_pickup_datetime']=pd.to_datetime(taxi_fare['lpep_pickup_
datetime'],format='%m/%d/%Y %I:%M:%S %p')
taxi_fare.set_index('lpep_pickup_datetime',inplace=True)
taxi_fare.head()
```

Out[44]:

	Total_amount
lpep_pickup_datetime	
2016-03-24 11:14:54	8.16
2016-03-24 11:08:43	6.80
2016-03-24 11:26:11	14.80
2016-03-24 11:45:05	24.84
2016-03-24 11:01:44	7.30

In [45]:
```
#resample pickup datetime to weekly, and plot the data
taxi_fare['Total_amount'].resample('W').sum().plot()
taxi_fare['Total_amount'].resample('W').sum().rolling(3).mean().plot()
taxi_fare['Total_amount'].resample('W').sum().expanding(3).mean().plot()
plt.legend(('Total_amount','Rolling 3-Period SMA','Expanding 3-Period SM
A'))
plt.title('Taxi Weekly Revenue')
plt.xlabel('Pickup Datetime(weekly)')
plt.ylabel('Revenue($)')
plt.show()
```



Timeline:

Above timeline graph shows us how the total taxi revenue changing over one year. The blue line shows the total revenue number counted by daily. The rolling SMA line can show us the trend of how total revenue changed. The NYC total revenue shows a lowest amount in Winter, it shows an increasing trend in Spring and remain the highest level through the summer. After that, the total revenue shows a decreasing trend in the fall. By doing researches about tourist peak season and off-season ([https://santorinidave.com/best-time-to-visit-nyc (https://santorinidave.com/best-time-to-visit-nyc)](https://santorinidave.com/best-time-to-visit-nyc)), we find that there is strong relationship between total taxi revenue and tourist peak season. For taxi driver perspective, this timeline graph will give them an idea about when will they earn the most money during the year. They can also make use of the pickup distribution map to find out where they can make money during tourist off-season.

7. Conclusion

1. How weather impacts taxi demand?

- NYC taxi demand has a continuous-slow decreasing trend from spring to winter, where average distance of trips has the same trend. Due to limitation of the dataset, we cannot find persuasive influences of precipitation, snow-depth and average temperature on the NYC taxi demand pattern. ##### Q1 Takeaway:
    - Both taxi drivers and customers may worry about the bad influence caused by inclement weather on their travel and business. Although there is not an obvious relationship between raining factors and taxi demand, taxi drivers should still pay attention to heavy snow day. The snow day observations are limited in our dataset, but we could find that the taxi demand dropped dramatically in a heavy snow day. People who want to go outside by taxi should also pay attention to this, because they might not find any taxi available to pick up them in such a heavy snow day.

2. How demand and average distance of trips change hourly?

- For hourly data, we have some interesting findings. First of all, owing to NYC taxi drivers' change shift being 3am to 5am, taxi drivers always are reluctant to take passengers in such time period, so there is a huge drop in the graph. Second, the demand of evening rush hour is about twice the demand of morning hour. ##### Q2 Takeaway:
    - This analysis result could give taxi drivers an idea that which time period they will make more money due to the high market demand. Get ready for the highest demand period and relax themselves during the low market demand period. High market demand also means that customers might be hard to get in an available taxi because every driver is very busy during this period. This analysis can give those customers an idea that when should they call a taxi if they want to attend any import event on time by taxi.

3. Which area shows a high demand in different time period?

- For average distance, which is in accordance with demand, it has longer trip from 3am to 5am. Drivers like to pickup long-trip passengers in the change shift time. Moreover, the average distance of evening is shorter than one of morning, because of various purpose of taxi after working. ##### Q3 Takeaway
    - For taxi drivers, if they want to make money during the midnight and early morning time, they could stay in Brooklyn and Queen to pick up customers. For commute time period, although the Bronx shows the highest market demand, those taxi drivers who don't want to suffer from traffic should avoid Bronx. Because the highest taxi demand sometime means the very heavy traffic. They can go to Brooklyn or Queen instead of Bronx. For customers' perspective, this distribution map can give them an idea where they should go for taxi if they stay outside very late.

4. Which area is the most popular destination?

- From drop-off geographic map, we can see that Manhattan has one of the densest dropoff datapoint in NYC, which means a lot of passengers' destination is Manhattan. Thus, Manhattan is the most popular destination. ##### Q4 Takeaway
    - After analyzing, we know that Manhattan is one of the most popular destination of green cabs, but theys cannot pick up passengers because of regulation. Therefore, a lot of green cabs in Manhattan must go to other districts to begin next trip, which results in a lot of green cabs without passengers departing from Manhattan and causing severe traffic jam in Manhattan. To relieve traffic pressure, the authority may consider another more reasonable regulation to restrict green cabs' routes and improve their current living situation.

5. What is the trend of weekly revenue in 2016?

- By analyzing taxi weekly revenue, we found that there is strong relationship between total taxi revenue and tourist peak season. For taxi driver perspective, this timeline graph will give them an idea about when will they earn the most money during the year. They can also make use of the pickup distribution map to find out where they can make money during tourist off-season.
- There is strong relationship between tourist peak/ off-season and the total taxi revenue. The taxi revenue will reach to the highest point in summer. Pickup distribution map will give taxi drivers a reference that where should they go for more customer. Bronx always shows highest market demand at any time. ##### Q5 Takeaway
  - This timeline graph can help taxi drivers make sense which season they can earn more money and get ready for that. They can arrange their vocation during the off-peak season.

In [ ]: