Untitled

June 2, 2018

In [23]:
```
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sb
import os
import sys
import time
import requests
import datetime
```

In [15]:
```
import glob
import csv

# Change delimiter to the appropriate one in CSV
delimiter = '/'
csvs = glob.glob('*.csv')
f = open('merged_data.csv', 'w')
master_csv = csv.writer(f)
#Take the header of the first CSV and make it the master header
first_csv = open(csvs[0], 'r')
headers = first_csv.readline().strip().split(delimiter)
master_csv.writerow(headers)
# Write remaining rows
for line in first_csv:
    master_csv.writerow(line.strip().split(delimiter))
# Read remaining CSVs and skip the first row
for file in csvs[1:]:
    current_csv = open(file, 'r')
    for line_num, line in enumerate(current_csv):
        if line_num > 0:
            master_csv.writerow(line.strip().split(delimiter))
f.close()

# This takes the 12 files and merges them.
```

In [46]:
```
tripData = pd.read_csv("C:/Users/Dell/Desktop/nycTaxi/trip/trip_data_1.csv", nrows=1
fareData = pd.read_csv("C:/Users/Dell/Desktop/nycTaxi/fare/trip_fare_1.csv", nrows=1
```

```
merge1 = pd.merge(tripData, fareData, on = 'medallion', how='inner')
fareTripMerge = merge1.to_csv('C:/Users/Dell/Desktop/nycTaxi/fare/merged.csv')

# Merging fare & trip data
```

```
In [58]: fareTripMerge = pd.read_csv('C:/Users/Dell/Desktop/nycTaxi/fare/merged.csv', nrows=100
         fareTripMerge.head()
         fareTripMerge.isnull().sum()
```

```
Out[58]: Unnamed: 0                0
         medallion                 0
         hack_license              0
         vendor_id                 0
         rate_code                 0
         store_and_fwd_flag    31221
         pickup_datetime           0
         dropoff_datetime          0
         passenger_count           0
         trip_time_in_secs         0
         trip_distance             0
         pickup_longitude          0
         pickup_latitude           0
         dropoff_longitude         0
         dropoff_latitude          0
          hack_license             0
          vendor_id                0
          pickup_datetime          0
          payment_type             0
          fare_amount              0
          surcharge                0
          mta_tax                  0
          tip_amount               0
          tolls_amount             0
          total_amount             0
         dtype: int64
```

```
In [9]: tripData = pd.read_csv("C:/Users/Dell/Desktop/nycTaxi/trip/trip_data_1.csv", nrows=1000
        tripData.head()
        tripData.isnull().sum()

        # This checks for null values in our data. And since none of the columns we use have a
```

```
Out[9]: medallion                 0
        hack_license              0
        vendor_id                 0
        rate_code                 0
        store_and_fwd_flag    996465
        pickup_datetime           0
```

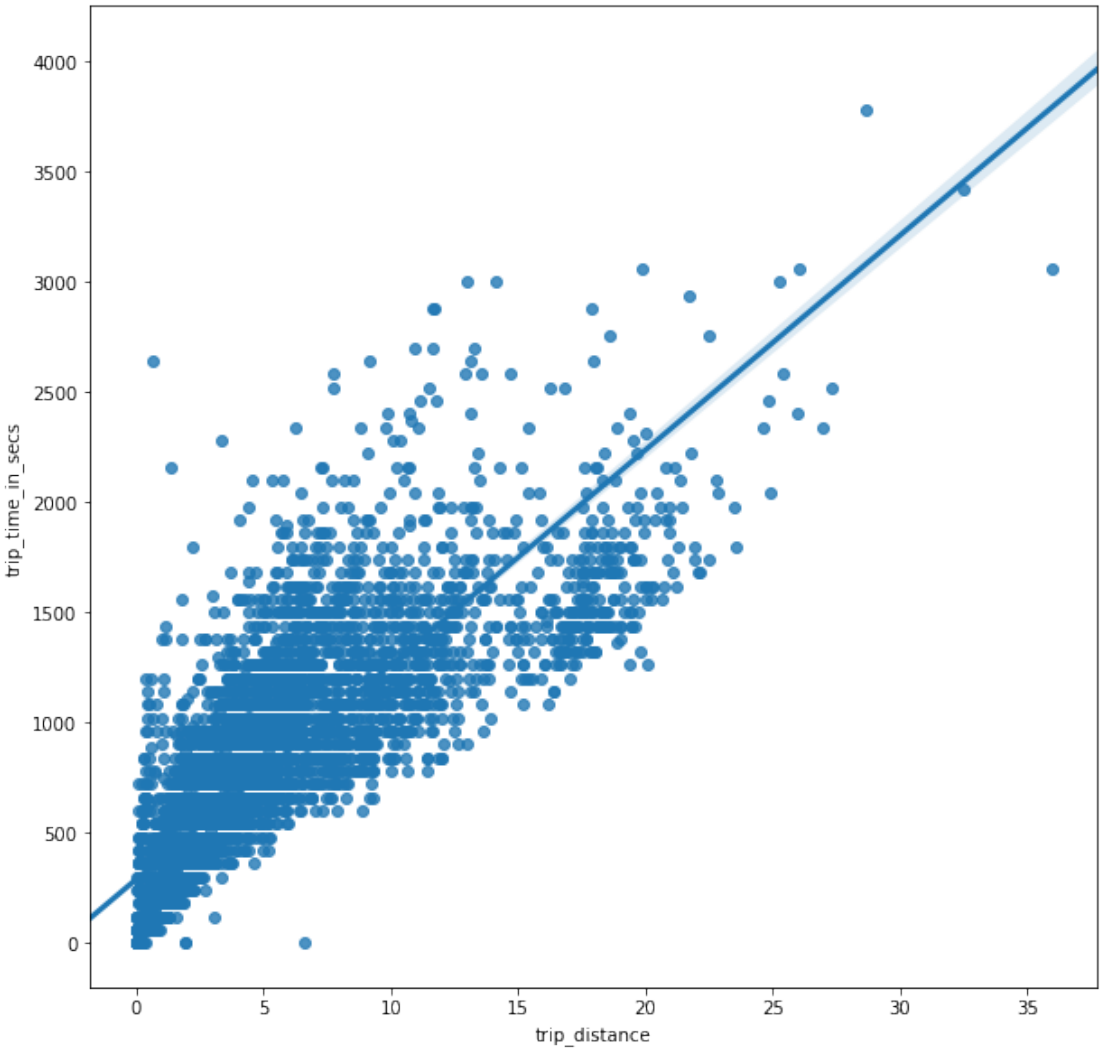
```
dropoff_datetime          0
passenger_count           0
trip_time_in_secs         0
trip_distance             0
pickup_longitude          0
pickup_latitude           0
dropoff_longitude         0
dropoff_latitude          0
dtype: int64
```

```
In [37]: tripData["passenger_count"].describe()
```

```
Out[37]: count    1000000.000000
         mean            2.129268
         std             1.724241
         min             0.000000
         25%             1.000000
         50%             1.000000
         75%             3.000000
         max             6.000000
         Name: passenger_count, dtype: float64
```

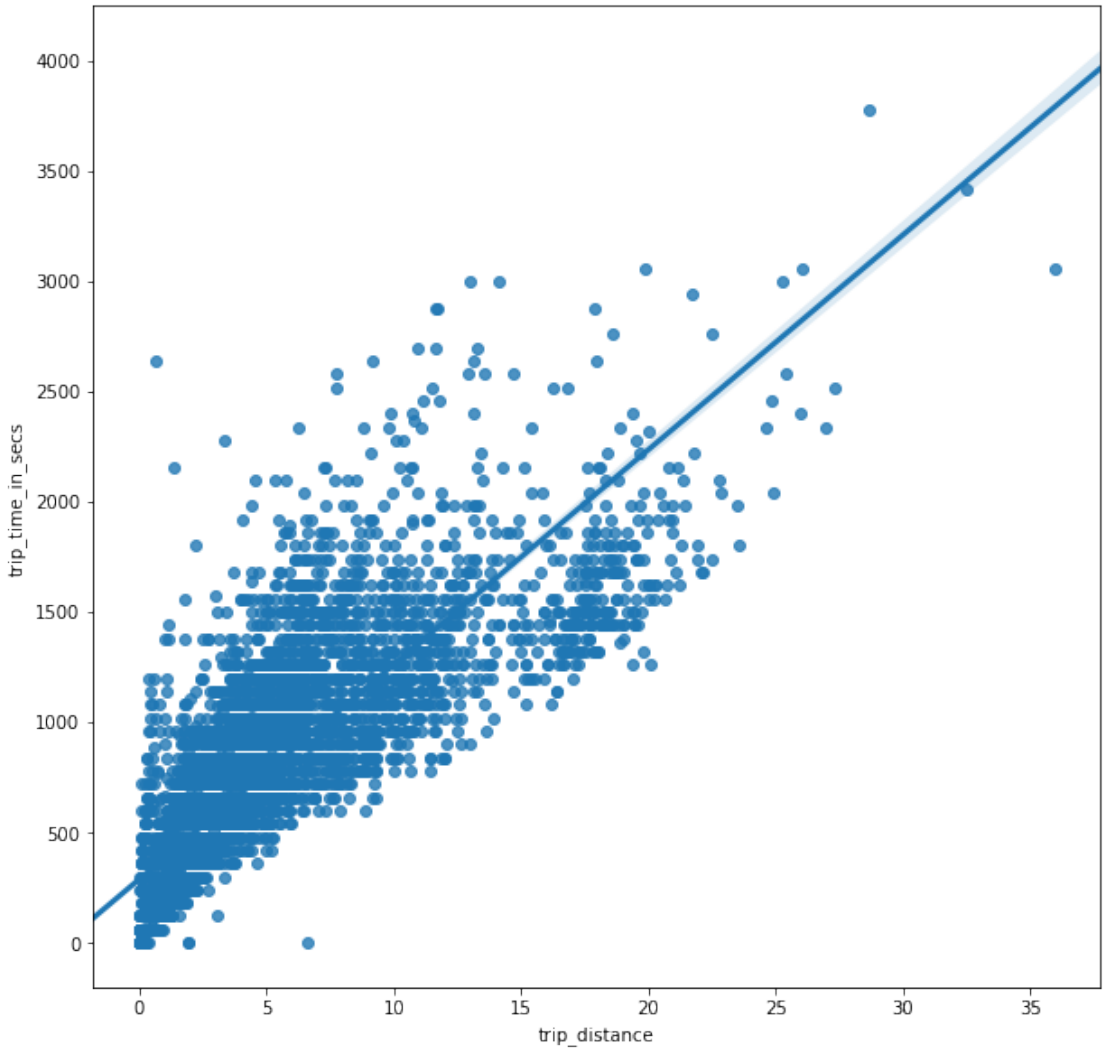
```
In [62]: plt.figure(figsize=(10,10))
         sb.regplot(x="trip_distance", y="trip_time_in_secs", data=tripData)

         # From the plot below there are very few outliers in terms of trip distance and in te
```

```
Out[62]: <matplotlib.axes._subplots.AxesSubplot at 0x2ab8cb74978>
```



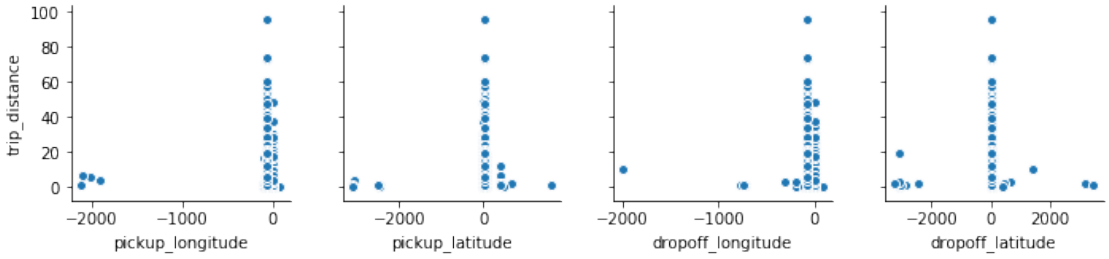

```
In [63]: plt.figure(figsize=(10,10))
         sb.regplot(x="trip_distance", y="trip_time_in_secs", data=tripData)
```

```
Out[63]: <matplotlib.axes._subplots.AxesSubplot at 0x2ab94bae550>
```

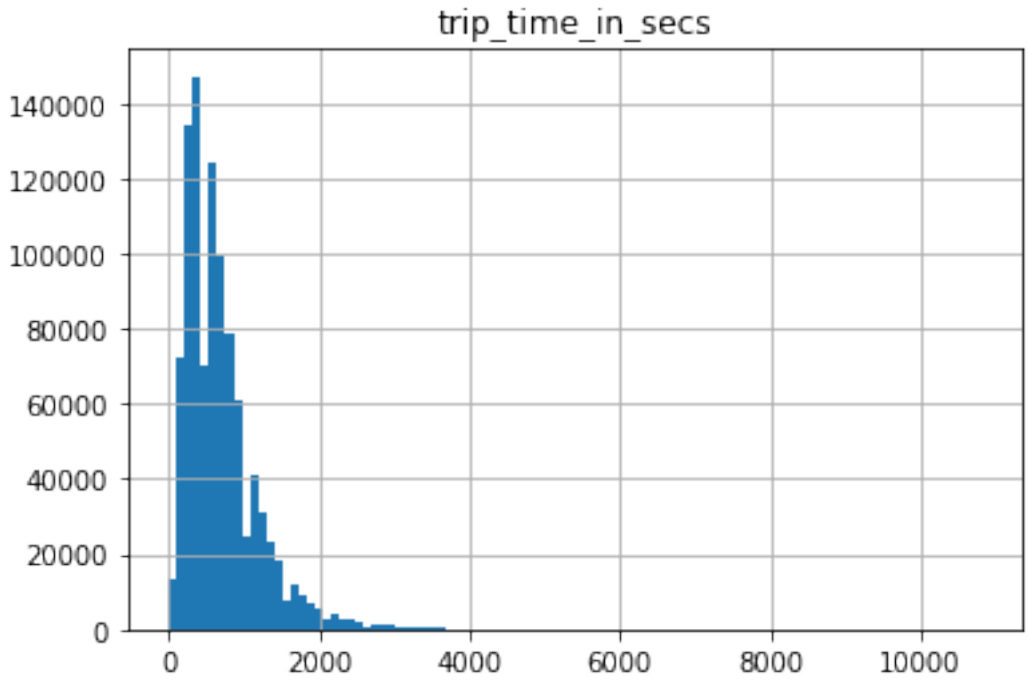



```
In [40]: g = sb.pairplot(tripData,
                         x_vars=["pickup_longitude","pickup_latitude","dropoff_longitude","dr
                         y_vars=["trip_distance"])
```

```
# Those latitude longitude has outliers which are clearly wrong values and have to be
```

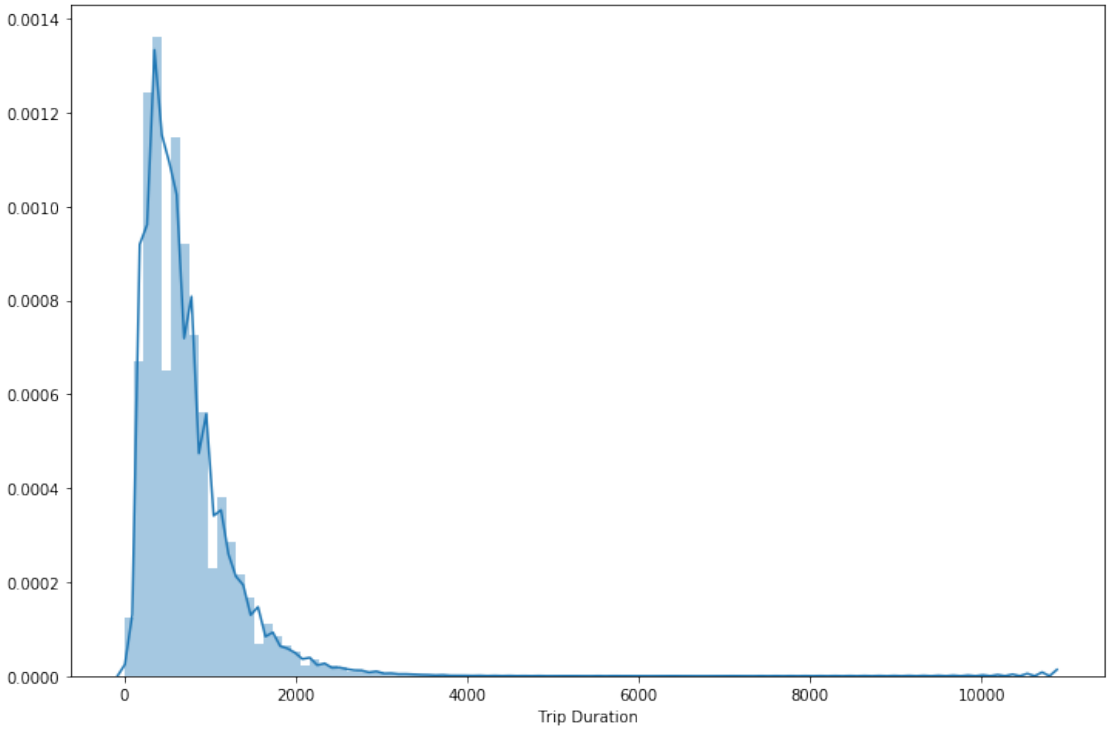


```
In [10]: tripData["trip_time_in_secs"].describe()
         # This gives the summary statistics for the trip duration

Out[10]: count    1000000.000000
         mean         691.155787
         std          492.380086
         min            0.000000
         25%          360.000000
         50%          540.000000
         75%          900.000000
         max        10800.000000
         Name: trip_time_in_secs, dtype: float64

In [15]: tripData.hist(column='trip_time_in_secs', bins = 100)

Out[15]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000002AB8B65CB70>]],
               dtype=object)
```



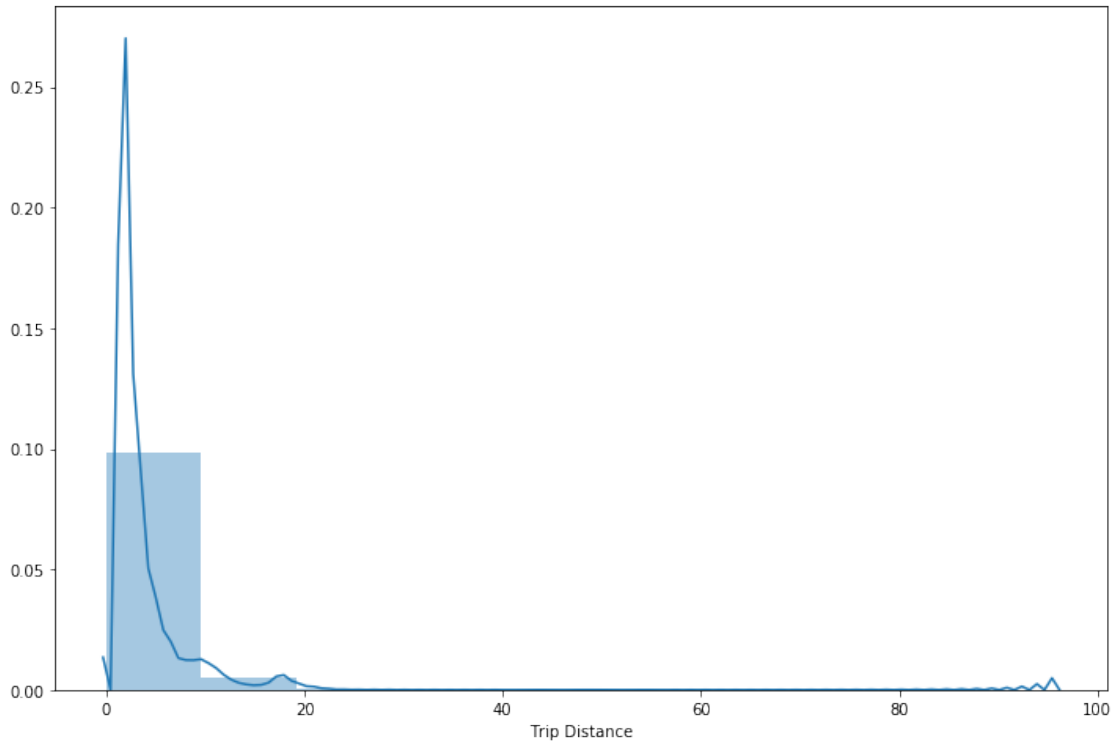

```
In [24]: plt.figure(figsize=(12,8))
         plt.xlabel('Trip Duration')
         sb.distplot(tripData.trip_time_in_secs.values, bins=100)
         plt.show()

         # This plot shows most of the trips having an avergae time of less than 1000 seconds.
```

```
C:\Users\Dell\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462: UserWarning: The 'norm
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```



In [26]:
```
plt.figure(figsize=(12,8))
plt.xlabel('Trip Distance')
sb.distplot(tripData.trip_distance.values, bins=10)
plt.show()
```

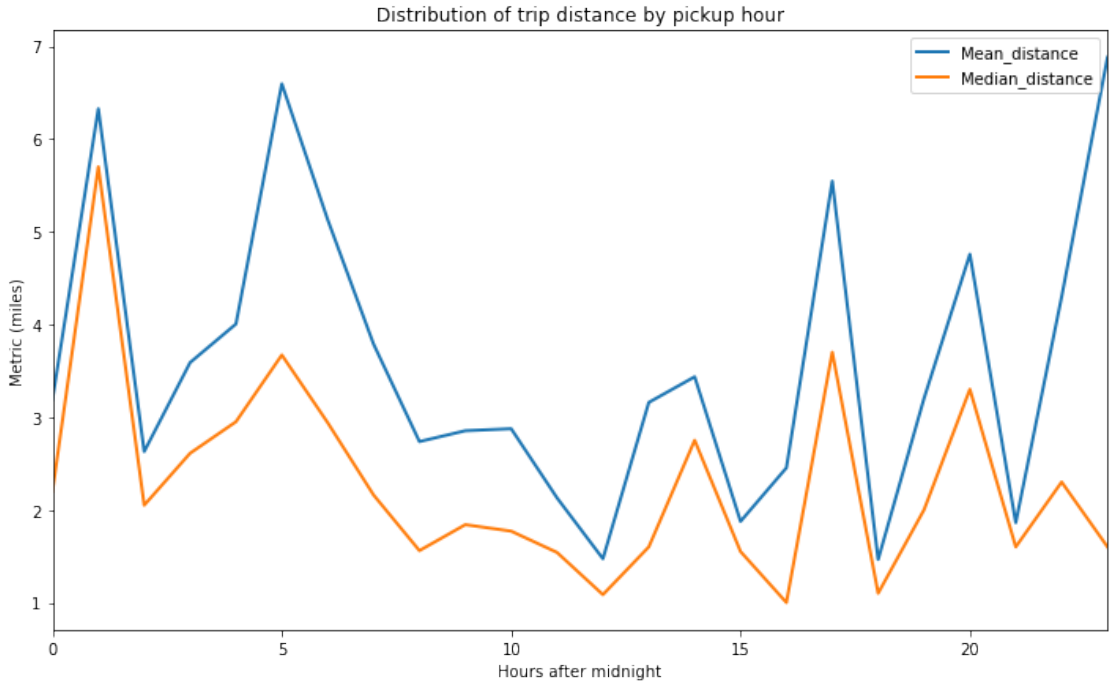
```
C:\Users\Dell\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462: UserWarning: The 'norm
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```



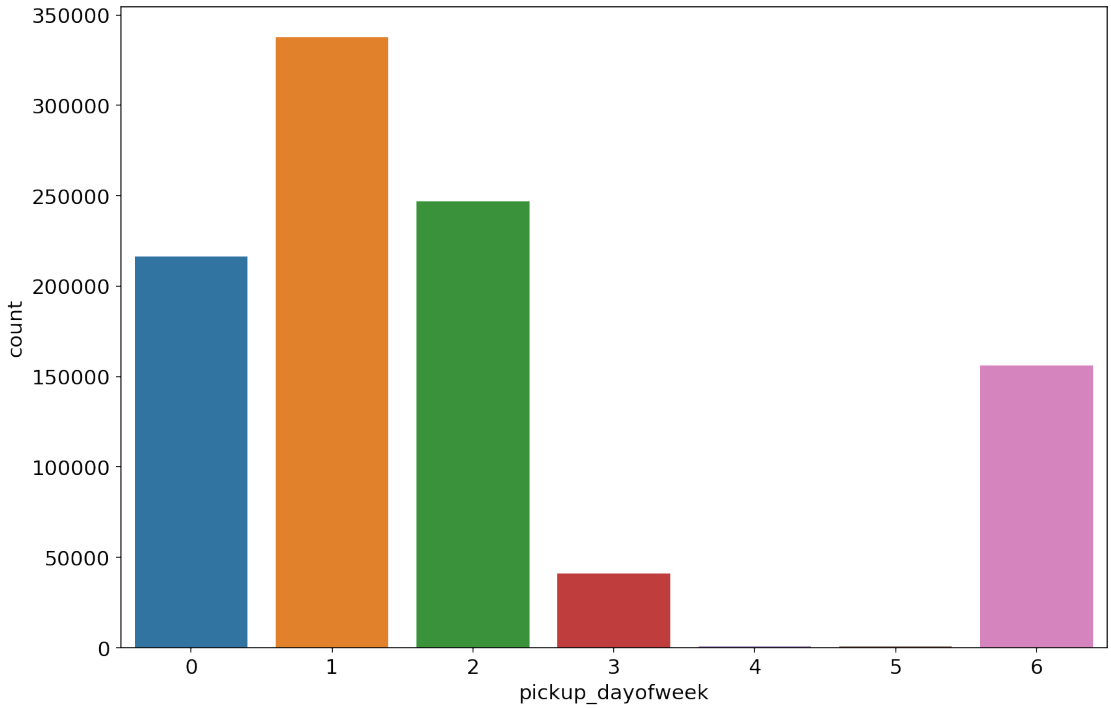
In [70]:
```
# #String to Datetime conversion
tripData["pickup_datetime"] = pd.to_datetime(tripData["pickup_datetime"])
tripData["dropoff_datetime"] = pd.to_datetime(tripData["dropoff_datetime"])
tripData["pickup_datetime"] = pd.to_datetime(tripData["pickup_datetime"])
tripData["pickup_dayofweek"] = tripData.pickup_datetime.dt.dayofweek
tripData["pickup_weekday_name"] = tripData.pickup_datetime.dt.weekday_name
tripData["pickup_hour"] = tripData.pickup_datetime.dt.hour
tripData["pickup_month"] = tripData.pickup_datetime.dt.month
```

In [72]:
```
# Distribution of trip distance by pickup hour

# Q: does time of the day affect the taxi ridership?


fix, axis = plt.subplots(1,1,figsize=(12,7))

#aggregate trip_distance by hour for plotting

tab = tripData.pivot_table(index='pickup_hour', values='trip_distance', aggfunc=('mea

tab.columns = ['Hour','Mean_distance','Median_distance']

tab[['Mean_distance','Median_distance']].plot(ax=axis)
```

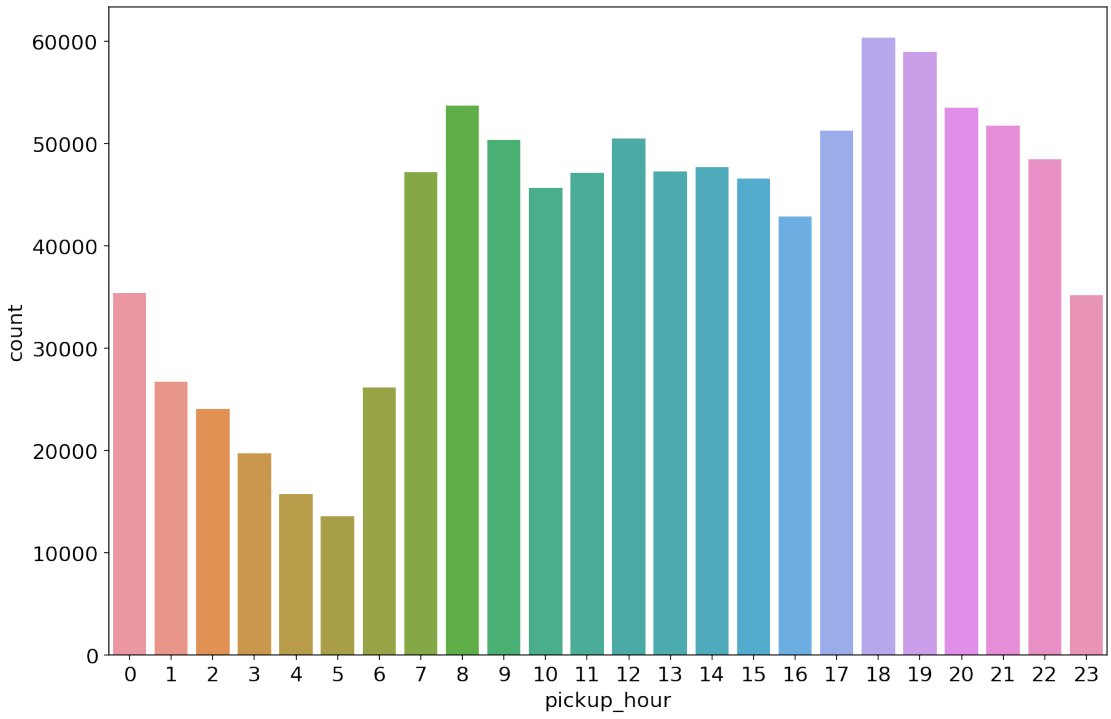
```
plt.ylabel('Metric (miles)')

plt.xlabel('Hours after midnight')

plt.title('Distribution of trip distance by pickup hour')

plt.xlim([0,23])

plt.show()
```



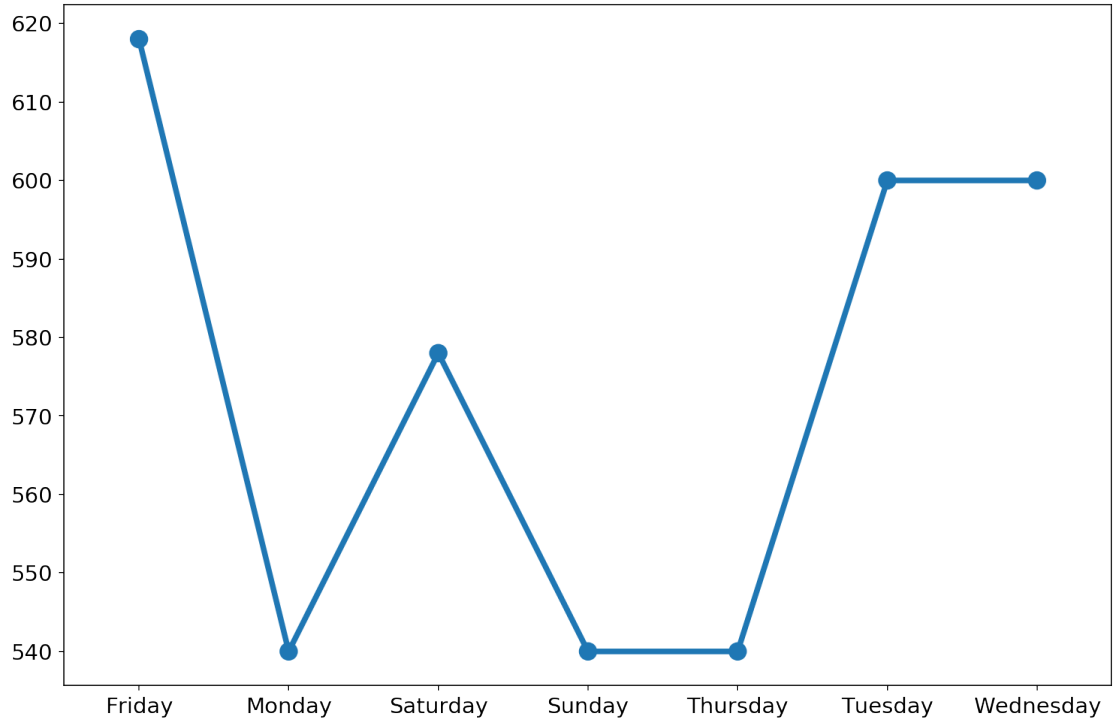
In [150]:
```
plt.figure(figsize=(12,8))
sb.countplot(x="pickup_dayofweek", data=tripData)
plt.show()
```



In [149]:
```
plt.figure(figsize=(12,8))
sns.countplot(x="pickup_hour", data=tripData)
plt.show()
```



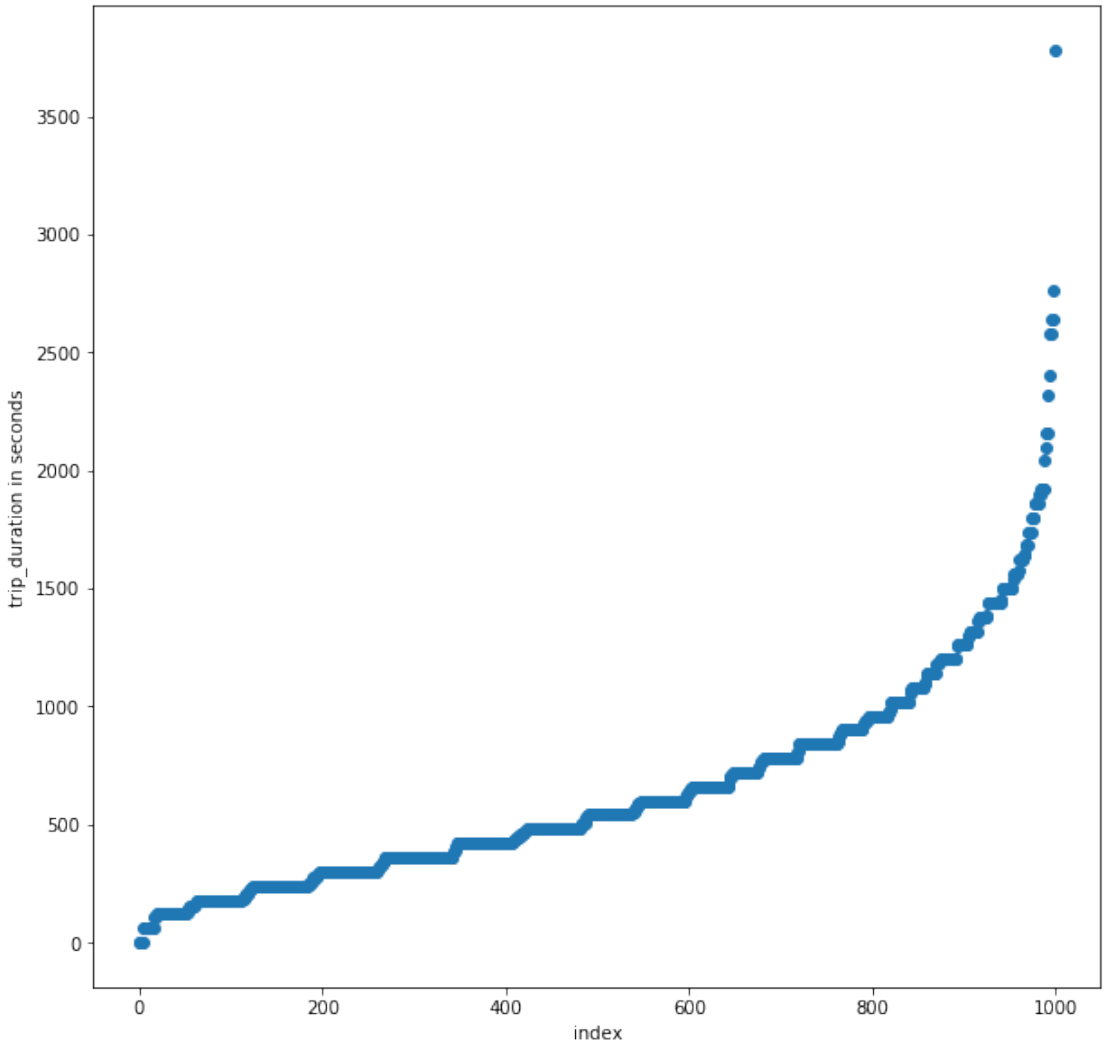
In [153]:
```
df_train_agg = tripData.groupby('pickup_weekday_name')['trip_time_in_secs'].aggregat

plt.figure(figsize=(12,8))
sns.pointplot(df_train_agg.pickup_weekday_name.values, df_train_agg.trip_time_in_secs
plt.show()
```



In [63]:
```
(tripData["dropoff_datetime"] - tripData["pickup_datetime"]).describe()
```

Out[63]:
```
count                      1000
mean     0 days 00:10:43.618000
std      0 days 00:07:34.634255
min             0 days 00:00:00
25%             0 days 00:05:00
50%             0 days 00:09:00
75%             0 days 00:14:00
max             0 days 01:03:00
dtype: object
```

In [65]:
```
plt.figure(figsize=(10,10))
plt.scatter(range(len(tripData["trip_time_in_secs"])), np.sort(tripData["trip_time_in
plt.xlabel('index')
plt.ylabel('trip_duration in seconds')
plt.show()
```

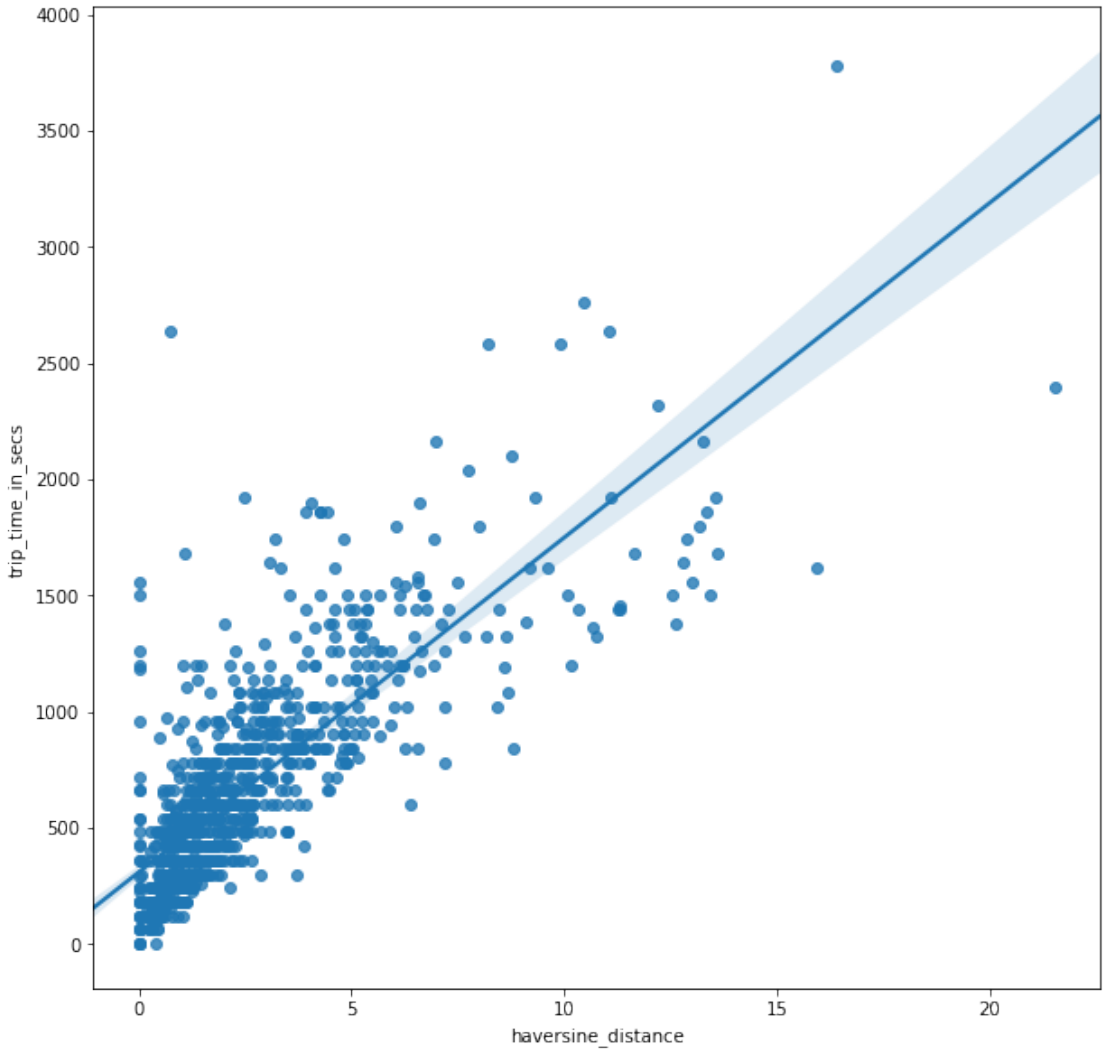



```
In [67]: #Removing the outliers in the dataset
         tripData = tripData[tripData["trip_time_in_secs"] < 500000]

In [69]: from haversine import haversine

         def calculate_haversine_distance(var_row):
             return haversine((var_row["pickup_latitude"], var_row["pickup_longitude"]),
                              (var_row["dropoff_latitude"], var_row["dropoff_longitude"]), mil

         # #Calculating the Haversine Distance
         # #The haversine formula determines the great-circle distance between two points on a
         # #given their longitudes and latitudes.
         tripData["haversine_distance"] = tripData.apply(lambda row: calculate_haversine_dista

In [70]: tripData["haversine_distance"].describe()
```

```
Out[70]: count    1000.000000
         mean        2.326663
         std         2.515960
         min         0.000000
         25%         0.780945
         50%         1.448235
         75%         2.918228
         max        21.543723
         Name: haversine_distance, dtype: float64
```

```
In [71]: plt.figure(figsize=(10,10))
         sb.regplot(x="haversine_distance", y="trip_time_in_secs", data=tripData)
```

```
Out[71]: <matplotlib.axes._subplots.AxesSubplot at 0x14f8bfd0240>
```

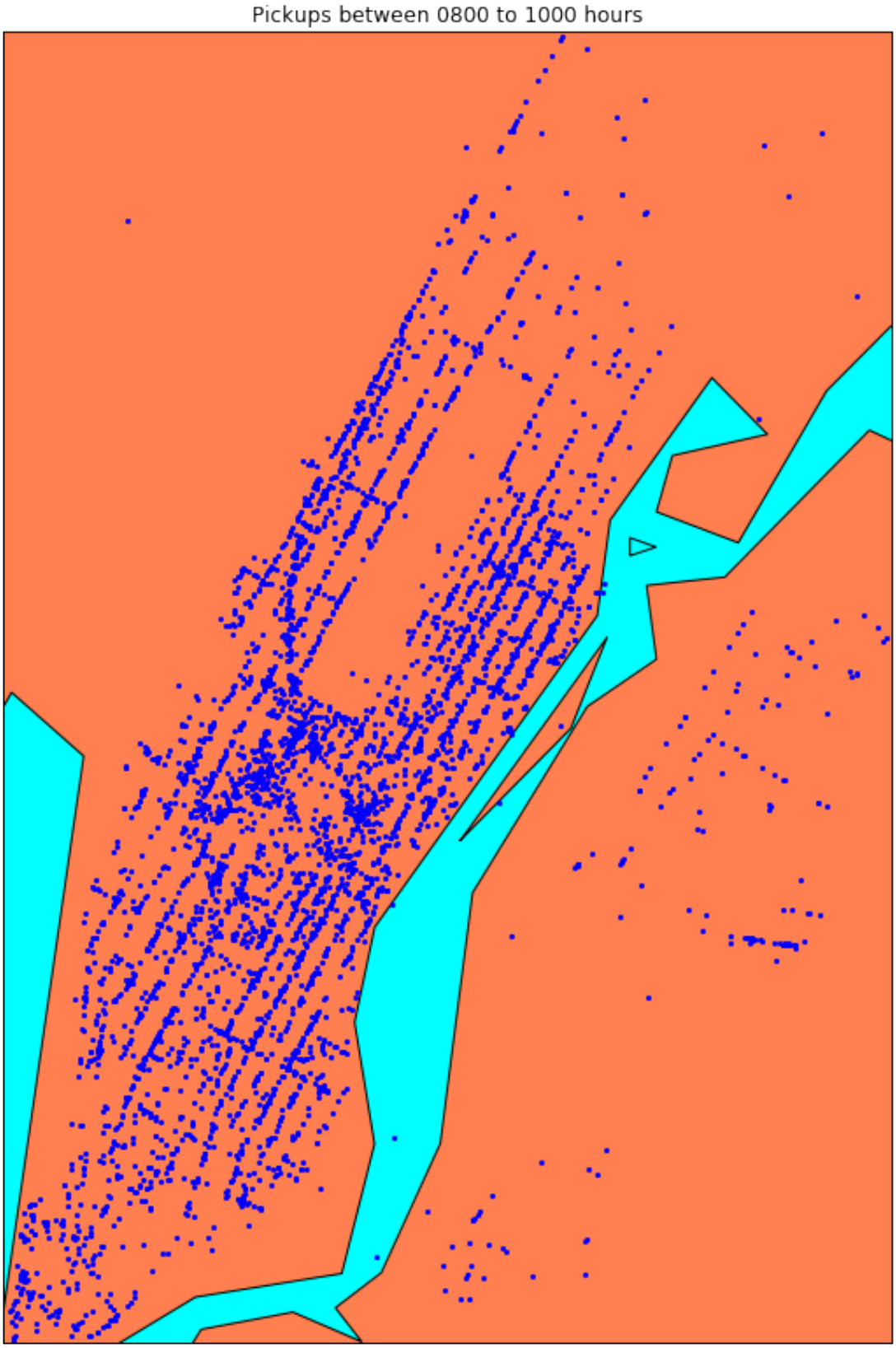


In [33]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import pandas as pd
import numpy as np
import csv
from mpl_toolkits.basemap import Basemap

# set some nicer defaults for matplotlib
from matplotlib import rcParams

#these colors come from colorbrewer2.org. Each is an RGB triplet

#rcParams['figure.figsize'] = (10, 6)
#rcParams['figure.dpi'] = 150
#rcParams['lines.linewidth'] = 2
#rcParams['axes.grid'] = False
#rcParams['axes.facecolor'] = 'white'
#rcParams['font.size'] = 14
#rcParams['patch.edgecolor'] = 'none'
```

In [34]:
```
def draw_pickup_dropoff_maps(hour_start, hour_end):
    fig = plt.figure(figsize=(20,20))


    map = Basemap(projection='merc', lat_0=40.7128, lon_0=-74.0059,
    resolution = 'h', area_thresh = 0.001,
    llcrnrlon=-74.018629, llcrnrlat=40.704889,
    urcrnrlon=-73.905500, urcrnrlat=40.831201)

# Pickup map
    ax1 = fig.add_subplot(121)
    ax1.set_title('Pickups between %04d to %04d hours' %(hour_start*100,hour_end*100))

    map.drawcoastlines()
    map.drawcountries()
    map.drawrivers()
    map.fillcontinents(color='coral',lake_color='aqua')
    map.drawmapboundary(fill_color='aqua')

    pickup_df = pd.DataFrame(data=tripData[tripData['pickup_longitude'] != 0][['picku
                columns=['pickup_datetime','pickup_longitude','pickup_latitude'])
    count = 1

    for index, row in pickup_df.iterrows():
        if count > 5000:
            break
```

```
                pickup_time = time.strptime(row['pickup_datetime'], "%Y-%m-%d %H:%M:%S").tm_ho
                if ((pickup_time >= hour_start) and (pickup_time<hour_end)):
                    x1,y1 = map(row['pickup_longitude'], row['pickup_latitude'])
                    map.plot(x1, y1, 'bo', markersize=2)
                    count += 1

            plt.show()

        draw_pickup_dropoff_maps(8, 10)
```

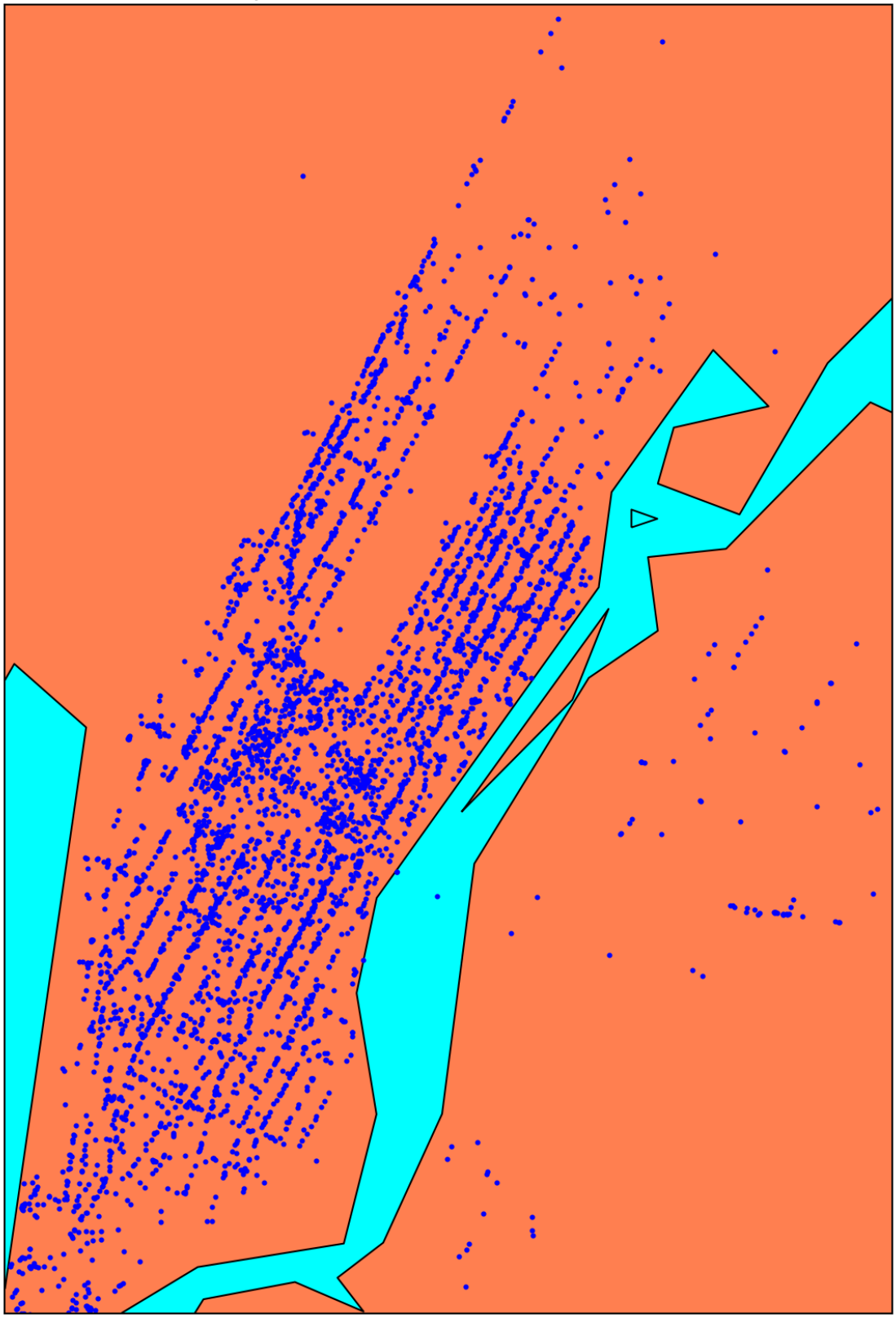
```
C:\Users\Dell\Anaconda3\lib\site-packages\mpl_toolkits\basemap\__init__.py:1708: MatplotlibDepr
  limb = ax.axesPatch
C:\Users\Dell\Anaconda3\lib\site-packages\mpl_toolkits\basemap\__init__.py:1711: MatplotlibDepr
  if limb is not ax.axesPatch:
```


Pickups between 0800 to 1000 hours

```
In [126]: draw_pickup_dropoff_maps(10, 12)
```

```
C:\Users\Dell\Anaconda3\lib\site-packages\mpl_toolkits\basemap\__init__.py:1708: MatplotlibDepr
  limb = ax.axesPatch
C:\Users\Dell\Anaconda3\lib\site-packages\mpl_toolkits\basemap\__init__.py:1711: MatplotlibDepr
  if limb is not ax.axesPatch:
```

Pickups between 1000 to 1200 hours



In [138]:
```
print(np.min(tripData['pickup_longitude']), np.min(tripData['pickup_latitude']))
print(np.max(tripData['pickup_longitude']), np.max(tripData['pickup_latitude']))

#print(np.min(tripData['dropoff_longitude']), np.min(tripData['dropoff_latitude']))
#print(np.max(tripData['dropoff_longitude']), np.max(tripData['dropoff_latitude']))
```

```
-2113.6499 -3114.3157
80.842125 1609.2006
```