UBER EDA

-PROJECT 1-

Group 5

Rebecca Kwon

Daniel Garza

Turgut Ozsirkinti

Neil Tipton

Data

KAGGLE.COM

Raw data found while exploring Kaggle.com

REQUIREMENTS

Number of columns, rows
Type of Data: longitude, latitude, time stamps, pickup and drop off distinguished

DATA CONTENT

Is the data interesting?
Is it worth exploring?

UBER EDA PROJECT 1

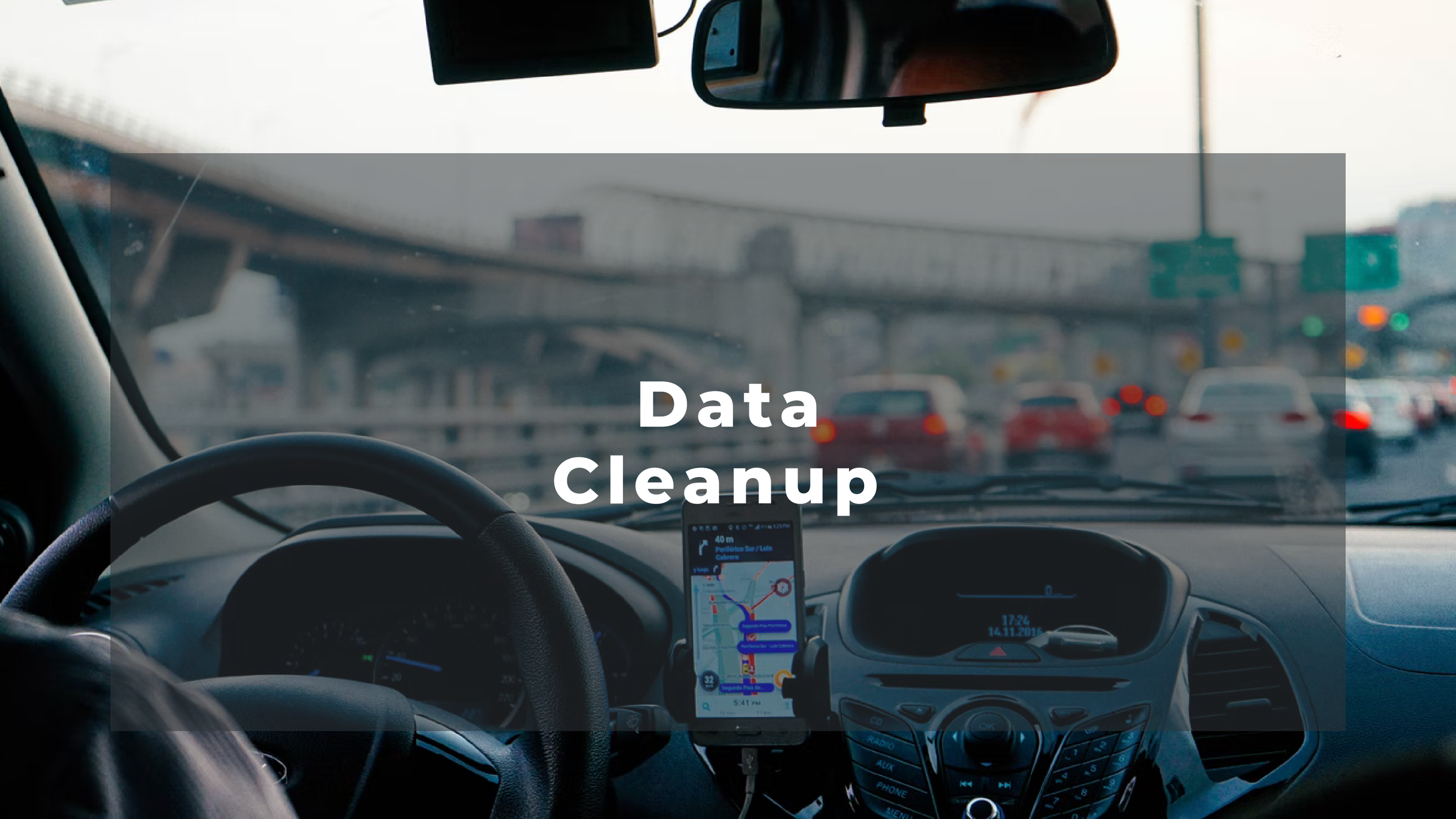
Motivation & Questions

RELEVANT DATA TO EXPLORE

Explore relevant data to current transportation method
Good source of raw data

QUESTIONS

What are the most popular pick up times?
What are the most popular pick up locations?
Do you save more money when you ride with others?
Is there a correlation between income and the number of Uber rides?
Which city spends the most on Uber rides?



Data Cleanup

```
#Dependencies
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from scipy import stats
from datetime import datetime
import warnings
warnings.filterwarnings("ignore")

# Incorporated citipy to determine city based on latitude and longitude
from citipy import citipy
```

DEPENDENCIES

```
#path to uber csv

file="uber.csv"

#read into data
uber_df= pd.read_csv(file)


uber_df=uber_df.sample(1000)
```

PATH TO CSV

Large data file, pulling random 1000 rows

```
# List for holding lat_lngs and cities
lat_lngs = []
cities = []

pickup_lats=uber_df["pickup_latitude"]
pickup_lngs=uber_df["pickup_longitude"]
#lat_lngs=zip(pickup_lats,pickup_lngs)

uber_df["City"]="x"

for index, row in uber_df.iterrows():
    city=citipy.nearest_city(row["pickup_latitude"],row["pickup_longitude"]).city_name
    uber_df.loc[index,"City"]=city

# Print the city count to confirm sufficient count
len(cities)
uber_df
```

PICK UP CITY NAMES USING CITIPY

use iterrows for loop to populate a column named "City" to capture the cities for each longitude and latitude pair

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
191863	11110663	2010-06-12 23:10:56.0000004	8.1	2010-06-12 23:10:56 UTC	-73.982489	40.768430	-73.974112	40.748273	2
116814	45749738	2010-11-24 16:55:00.00000081	9.3	2010-11-24 16:55:00 UTC	-73.992055	40.749317	-73.982737	40.739670	1
46007	18035915	2009-02-13 20:19:00.00000050	5.7	2009-02-13 20:19:00 UTC	-74.002198	40.734558	-73.989863	40.734395	6
101427	51933839	2012-04-05 14:10:00.00000080	8.1	2012-04-05 14:10:00 UTC	-73.984708	40.742742	-73.973418	40.757738	3
86545	18501346	2010-05-29 19:22:00.000000135	3.7	2010-05-29 19:22:00 UTC	-73.986800	40.755780	-73.992427	40.749445	5
...	...	...	...	...	...	...	...	...	...
181536	35279223	2011-02-06 15:00:00.00000067	28.9	2011-02-06 15:00:00 UTC	-73.969205	40.785475	-73.861455	40.768230	2
32524	7173143	2010-12-08 17:52:00.000000224	5.7	2010-12-08 17:52:00 UTC	-73.993432	40.727197	-74.003297	40.723602	1
39084	47267175	2012-07-18 11:42:00.000000205	7.3	2012-07-18 11:42:00 UTC	-73.979025	40.740390	-73.997830	40.733592	1
71962	44974467	2009-12-23 18:48:00.00000080	8.9	2009-12-23 18:48:00 UTC	-73.976372	40.759642	-73.989490	40.771400	1
148911	40023153	2009-04-01 17:55:00.000000151	7.3	2009-04-01 17:55:00 UTC	-74.000150	40.742835	-73.980290	40.739325	1

RAW DATA

Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	City
24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.999512	40.723217	1	new york
27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.994710	40.750325	1	new york
44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73.962565	40.772647	1	new york
25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.965316	40.803349	3	guttenberg
17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.973082	40.761247	5	new york
...	...	...	...	...	...	...	...	...	...
13439193	2011-05-04 06:39:00.00000044	5.7	2011-05-04 06:39:00 UTC	-73.969720	40.757577	-73.953782	40.766960	1	guttenberg
32405310	2011-11-23 20:43:20.0000002	8.1	2011-11-23 20:43:20 UTC	-73.993784	40.757054	-73.980018	40.775632	3	weehawken
51612001	2010-01-11 20:58:00.00000035	8.5	2010-01-11 20:58:00 UTC	-73.972338	40.765078	-73.954527	40.783833	5	guttenberg
937243	2013-06-12 17:01:24.0000001	5.5	2013-06-12 17:01:24 UTC	-73.979054	40.784730	-73.982970	40.775048	1	guttenberg
47946613	2011-12-11 21:48:22.0000001	9.7	2011-12-11 21:48:22 UTC	-73.983675	40.729944	-73.992416	40.758208	1	new york

UBER_DF DATAFRAME

```python
# vectorized haversine function
def haversine(lat1, lon1, lat2, lon2, to_radians=True, earth_radius=6371):
    """
    slightly modified version: of http://stackoverflow.com/a/29546836/2901002

    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees or in radians)

    All (lat, lon) coordinates must have numeric dtypes and be of equal length.

    """
    if to_radians:
        lat1, lon1, lat2, lon2 = np.radians([lat1, lon1, lat2, lon2])

    a = np.sin((lat2-lat1)/2.0)**2 + \
        np.cos(lat1) * np.cos(lat2) * np.sin((lon2-lon1)/2.0)**2

    return earth_radius * 2 * np.arcsin(np.sqrt(a))
```

HAVERSINE FUNCTION

Define new function from stackoverflow to convert latitude longitude coordinate differences as Distance (mi)

```python
#distance

dropoff_lats=uber_df["pickup_longitude"]
dropoff_lngs=uber_df["pickup_longitude"]

uber_df["Distance"]=0

for index, row in uber_df.iterrows():
    distance=haversine(row["pickup_latitude"],row["pickup_longitude"],\
                       row["dropoff_latitude"], row["dropoff_longitude"],to_radians=True, earth_radius=6371)
    uber_df.loc[index,"Distance"]=distance
```

DISTANCE

Use Haversine function to calculate distances and store in a new column "Distance" for the dataframe

```python
#binning

#set time of day bins
timeofday=[0, 5, 12, 17, 24]

uber_df.pickup_datetime= pd.to_datetime(uber_df.pickup_datetime)

#Labels for bins
times=["Night","Morning","Afternoon","Evening"]

uber_df['Time of Day'] = pd.cut(uber_df.pickup_datetime.dt.hour, timeofday, labels=times, right=False)
```

BINNING

Binning based on time of day to four categories: Night, Morning, Afternoon, Evening

```python
#remove distance 0

uber_df_cleaned=uber_df.loc[uber_df["Distance"]!=0]
```

REMOVE 0 DISTANCES

key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	City	Distance	Time of Day
2013-05-23 17:21:00.000000131	12.00	2013-05-23 17:21:00+00:00	-73.967680	40.762665	-73.992487	40.742867	1	guttenberg	3.035245	Evening
2015-03-20 15:39:31.0000005	17.50	2015-03-20 15:39:31+00:00	-73.942474	40.751007	-73.938972	40.801731	1	new york	5.647957	Afternoon
2015-01-17 15:05:35.0000008	5.50	2015-01-17 15:05:35+00:00	-73.990585	40.734936	-73.998161	40.729328	5	new york	0.892353	Afternoon
2012-02-19 21:27:00.00000077	4.90	2012-02-19 21:27:00+00:00	-73.951598	40.714208	-73.957473	40.722232	1	new york	1.020405	Evening
2012-01-04 09:12:29.0000002	11.30	2012-01-04 09:12:29+00:00	-73.946756	40.772440	-73.984445	40.733046	1	guttenberg	5.409869	Morning
2013-11-07 17:41:00.00000038	8.00	2013-11-07 17:41:00+00:00	-73.978865	40.782282	-73.966972	40.807457	3	guttenberg	2.972975	Evening

UBER_DF_CLEANED

Cleaned dataframe used as base-dataframe for all consequent plots/maps

Data Analysis

Do Income & Number of rides correlate?

```
from geopy.geocoders import Nominatim
geolocator = Nominatim(user_agent="geoapiExercises")

uber_df_cleaned["Zipcode"]=0



for index, row in uber_df_cleaned.iterrows():
    try:
        location = geolocator.reverse(str(row["pickup_latitude"])+","+str(row["pickup_longitude"]))
        zipcode=list(location.raw.values())[7]["postcode"]
        uber_df_cleaned.loc[index,"Zipcode"]=zipcode

    except:
        print(str(row["pickup_latitude"])+","+str(row["pickup_longitude"])+" zipcode not found, passing... ")
        pass
```

```
40.758325,-73.9725 zipcode not found, passing...
40.779248,-73.962154 zipcode not found, passing...
0.00039,-0.003813 zipcode not found, passing...
0.0,0.0 zipcode not found, passing...
40.779069,-73.962325 zipcode not found, passing...
40.740127,-74.005832 zipcode not found, passing...
40.739565,-74.00641 zipcode not found, passing...
40.739685,-74.006242 zipcode not found, passing...
40.777242,-73.982357 zipcode not found, passing...
```

GEOPY

Using geopy and reverse locator, picked up the zip codes for each latitude and longitude pair.

Passed an exception for any pair not found and skipped as to not cause any errors.

Do Income & Number of rides correlate?

```
#where zipcodes were not found, leave behind
zipcodez=uber_df_cleaned.loc[uber_df_cleaned["Zipcode"]!=0]
```

```
#average year of date times

year=round(uber_df_cleaned.pickup_datetime.dt.year.mean(),0)
year
```

```
2012.0
```

```
from census import Census

# Census API Key
from api_key import c_key
c = Census(c_key, year=2012)

census_data = c.acs5.get(("NAME", "B19013_001E"), {'for': 'zip code tabulation area:*'})

# Convert to DataFrame
census_pd = pd.DataFrame(census_data)

# Column Reordering
census_pd = census_pd.rename(columns={"B19013_001E": "Household Income",
                                      "NAME": "Name", "zip code tabulation area": "Zipcode"})
census_pd
```

	Name	Household Income	state	Zipcode
0	ZCTA5 02655	73323.0	25	02655
1	ZCTA5 02657	46031.0	25	02657
2	ZCTA5 02659	51466.0	25	02659
3	ZCTA5 02660	48617.0	25	02660
4	ZCTA5 02663	21667.0	25	02663

Removed any rows with longitude/latitude pairs that did not return a zip code

Return average year of data set to obtain census data from

CENSUS

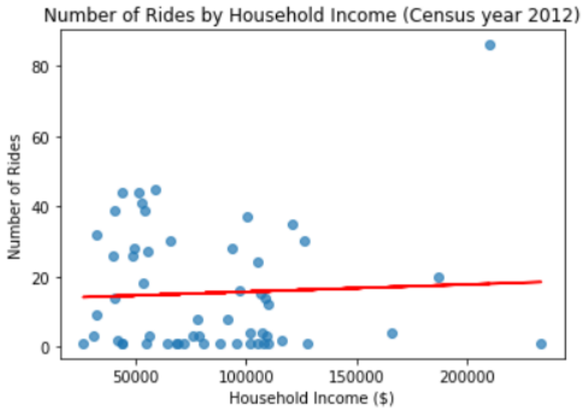
Use Census API to obtain Zip Code and House hold income

Do Income & Number of rides correlate?

```
zipcoded=pd.merge(zipcodez,census_pd,how="left",on="Zipcode").dropna()
zipcoded=zipcoded.loc[(zipcoded["Household Income"].astype(float))>0]
```

```
#count number of rides per zipcode
byincome=zipcoded.groupby("Household Income")["fare_amount"].count()
```

```
# Create a Scatter Plot for income and number of rides
x_values = zipcoded["Household Income"].unique().astype(float)
y_values = byincome



slope,inte,r,p,std_err=stats.linregress(x_values,y_values)
fitline=slope*x_values + inte

plt.scatter(x_values,y_values,alpha=.7)
plt.plot(x_values,fitline,"r-")
plt.xlabel('Household Income ($)')
plt.ylabel(f'Number of Rides')

plt.title(f'Number of Rides by Household Income (Census year 2012)')

plt.show()
```



Left merged on Zipcode
Drop any NAs and ensure numbers are being passed for income

Groupby Household income

NUMBER OF RIDES BY HOUSEHOLD INCOME

Line regression shows no correlation
However, clustering shows more common income number of rides under 50 rides versus the *outlier* $200,000 higher number of rides.

Most Popular Cities

```
#sorting the datarame with City, Pickup Lats & Lngs, Dropoff Lats & Lngs

all_locations = uber_df_cleaned[['City', 'pickup_latitude', 'pickup_longitude', 'dropoff_latitude', 'dropoff_longitude']]\
.groupby(uber_df_cleaned['City'])
all_locations.head()
```

	City	pickup_latitude	pickup_longitude	dropoff_latitude	dropoff_longitude
106090	new york	40.728868	-74.005293	40.728378	-73.999908
110072	new york	40.726930	-73.992586	40.723715	-74.003482
139813	guttenberg	40.756078	-73.977623	40.782177	-73.946245
169796	guttenberg	40.775897	-73.986980	40.752103	-73.979778
73022	guttenberg	40.772012	-73.956052	40.793197	-73.965940
64926	edgewater	40.791664	-73.953013	40.764618	-73.962201
130385	edgewater	40.790523	-73.942588	40.750188	-73.994981

SORTING DATAFRAME

Used groupby on the cleaned dataframe with specific columns

```
#finding out the top cities
city_group = pd.DataFrame(uber_df_cleaned, columns=['City', 'pickup_latitude', 'pickup_longitude', \
                                                    'dropoff_latitude', 'dropoff_longitude'])
top_cities = city_group[['City']].apply(pd.Series.value_counts).head(5).reset_index()["index"].tolist()
top_cities
```

```
['new york', 'guttenberg', 'weehawken', 'edgewater', 'woodmere']
```

Found top cities by counting rides for each city and stored those cities into a list using reset_index and tolist.

Most Popular Cities

```
#creating the dataframe to get the lats and lngs for pickup citites
city_group = uber_df_cleaned.groupby("City")[["pickup_latitude", "pickup_longitude"]].mean()

top_pickups = city_group.loc[top_cities]
top_pickups
```

City	pickup_latitude	pickup_longitude
new york	40.731622	-73.990774
guttenberg	40.768933	-73.970303
weehawken	40.754736	-73.989986
edgewater	40.787213	-73.941153
woodmere	40.645841	-73.786274

```
#creating the dataframe to get the lats and lngs for dropoff citites

city_group = uber_df_cleaned.groupby("City")[["dropoff_latitude", "dropoff_longitude"]].mean()

top_dropoffs = city_group.loc[top_cities]
top_dropoffs
```

City	dropoff_latitude	dropoff_longitude
new york	40.629111	-73.777258
guttenberg	40.757837	-73.971581
weehawken	40.747643	-73.979164
edgewater	40.769573	-73.958176
woodmere	40.748866	-73.959923

SETTING UP MAPS

Used groupby and loc to retrieve the pick up and drop off location longitude and latitude pair using the top cities list

Most Popular Cities

```
# Configure gmaps
from api_key import g_key
gmaps.configure(api_key=g_key)
```

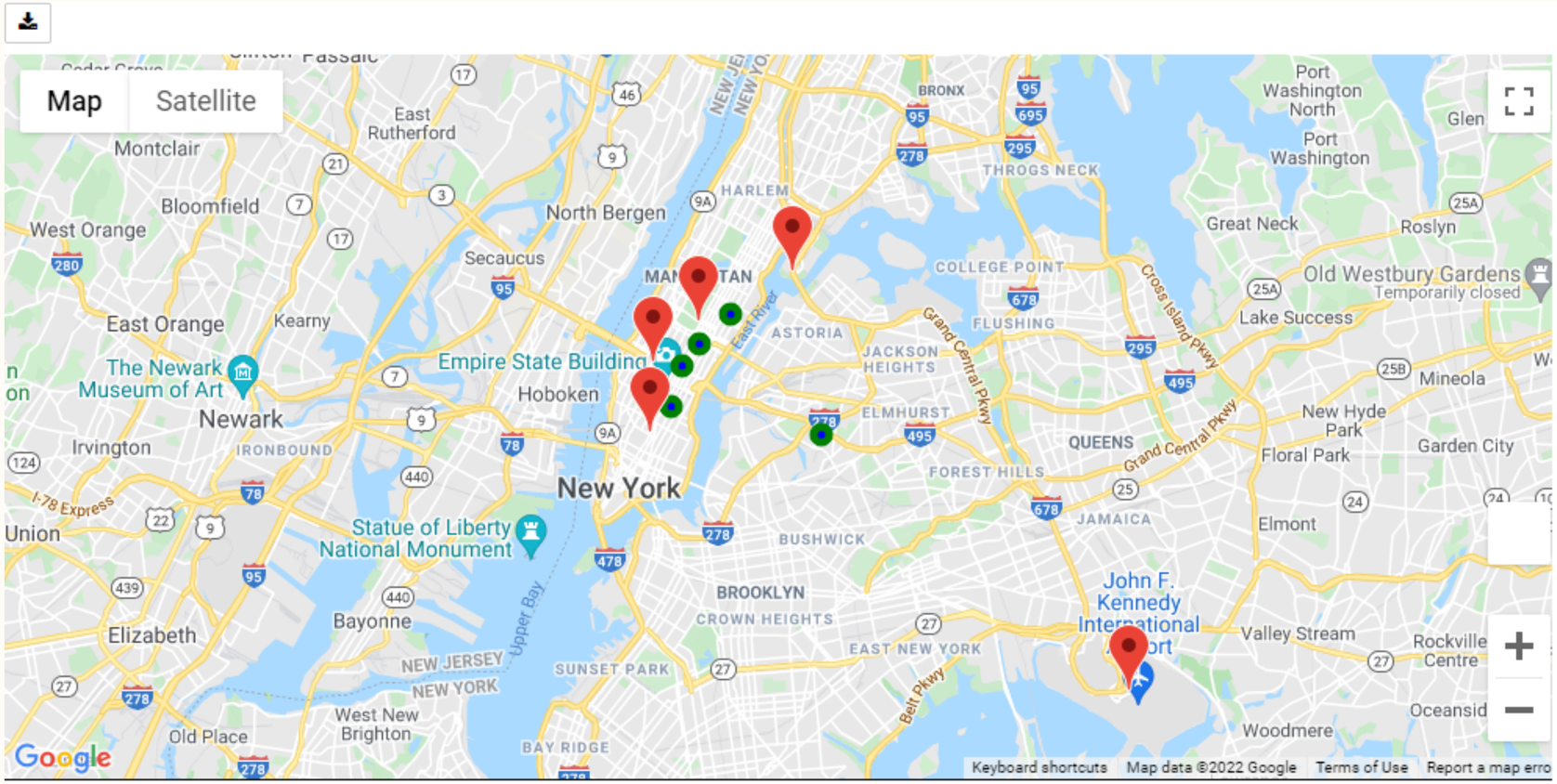
configure gmaps with the API Key located in a separate file

```
#setting pickup locations
pickup_locations = top_pickups[["pickup_latitude", "pickup_longitude"]]
pickup_locations

#creating figure layout
figure_layout = {'width': '1000px',
                 'height': '500px',
                 'border': '1px solid black',
                 'padding': '0.01px',
                 'margin': '0 auto 0 auto'}
fig = gmaps.figure(layout=figure_layout)

#adding markers
markers = gmaps.marker_layer(pickup_locations[["pickup_latitude", "pickup_longitude"]])
fig.add_layer(markers)

#displaying the figure
fig
```

Set up the locations and figure specifications

Place markers for each location

Most Popular Cities



Pick Up Locations

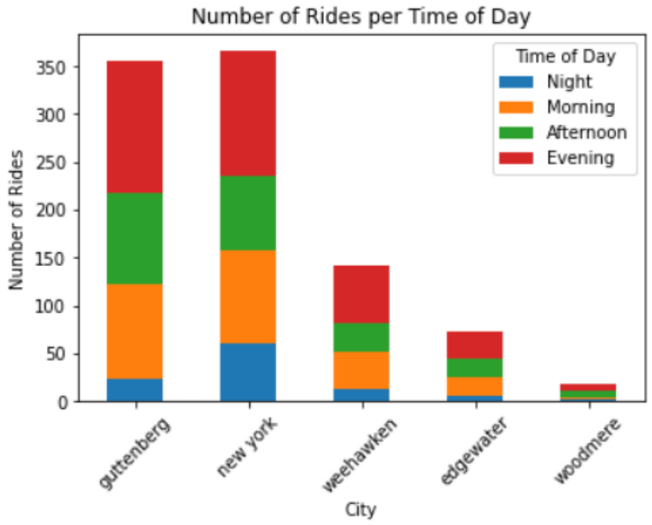
Drop off Locations

City Evaluation

```
topcities=uber_df_cleaned.groupby(["City"]).size().sort_values(ascending=False).head(5).reset_index()["City"].tolist()
topcities
```

```
['new york', 'guttenberg', 'weehawken', 'edgewater', 'woodmere']
```

```
bycities=uber_df_cleaned.groupby(["City","Time of Day"]).size()[topcities].sort_values(ascending=False).head(20).unstack()\
.plot(kind='bar', stacked=True)

# Set a title for the chart
plt.title("Number of Rides per Time of Day")
plt.ylabel("Number of Rides")
plt.xticks(rotation=45)
plt.show()
plt.tight_layout()
```



BREAKOUT OUT BY TOP RIDING CITIES AND TIME OF DAY

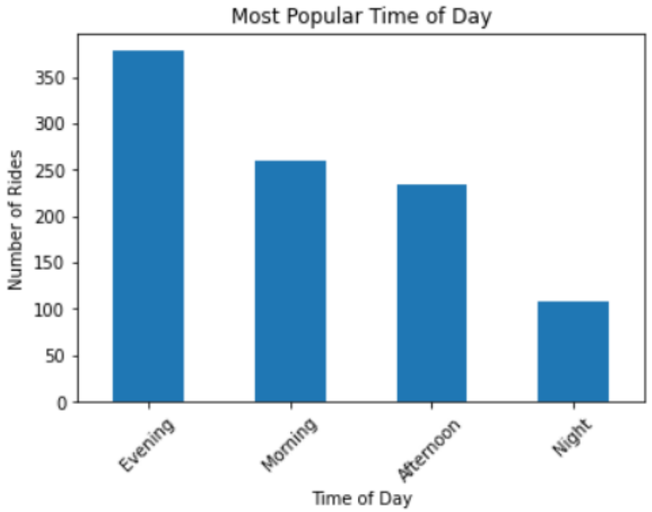
Groupby City and Time of Day to view a stacked bar graph

Of the dataset evaluated, Guttenberg and New York had the highest number of rides, most of them occurring in the evening.

This evaluation could find that other cities might travel more during the day than the night or vise versa.

City Evaluation

```
bycities=uber_df_cleaned.groupby(["Time of Day"]).size().sort_values(ascending=False).plot(kind='bar')

# Set a title for the chart
plt.title("Most Popular Time of Day")
plt.ylabel("Number of Rides")
plt.xticks(rotation=45)
plt.show()
plt.tight_layout()
```



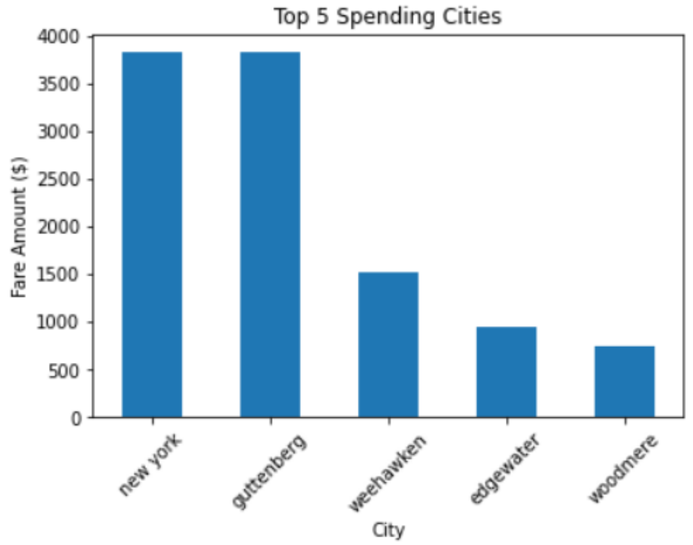
MOST POPULAR TIME OF DAY TO TRAVEL

Groupby Time of Day to view the ride counts for each time of day category

Most rides happen in the evening between the hours of 12AM and 5AM

City Evaluation

```
bycities=uber_df_cleaned.groupby("City")["fare_amount"].sum().sort_values(ascending=False).head(5).plot(kind='bar')

# Set a title for the chart
plt.title("Top 5 Spending Cities")
plt.ylabel("Fare Amount ($)")
plt.xticks(rotation=45)
plt.show()
plt.tight_layout()
```



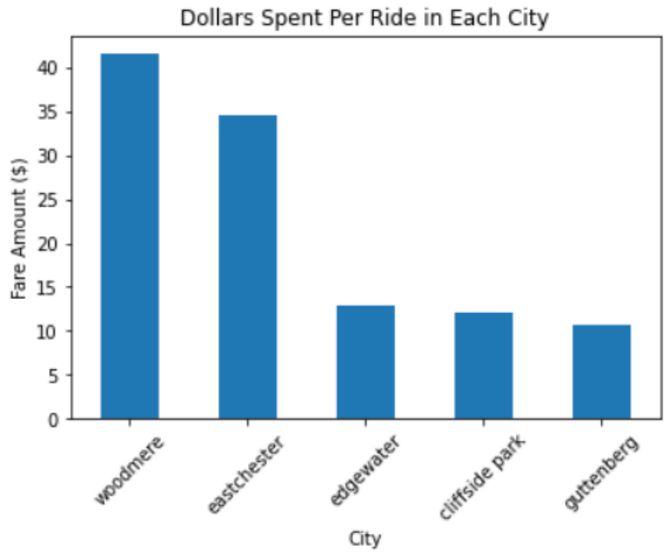
TOP 5 TOTAL SPENDING CITIES

Groupby City & Fare Amount Sums to view the top spending cities.

Unsurprisingly, New York and Guttenberg (in NJ right outside of Manhattan) were the top spenders. This doesn't give a full picture due to the sheer population in these two cities.

City Evaluation

```
bycities=uber_df_cleaned.groupby("City")["fare_amount"].sum()
ridespercity=uber_df_cleaned.groupby("City")["Distance"].count()

percityspent=bycities/ridespercity
percityspent.sort_values(ascending=False).head(5).plot(kind="bar")

# Set a title for the chart
plt.title("Dollars Spent Per Ride in Each City")
plt.ylabel("Fare Amount ($)")
plt.xticks(rotation=45)
plt.show()
plt.tight_layout()
```



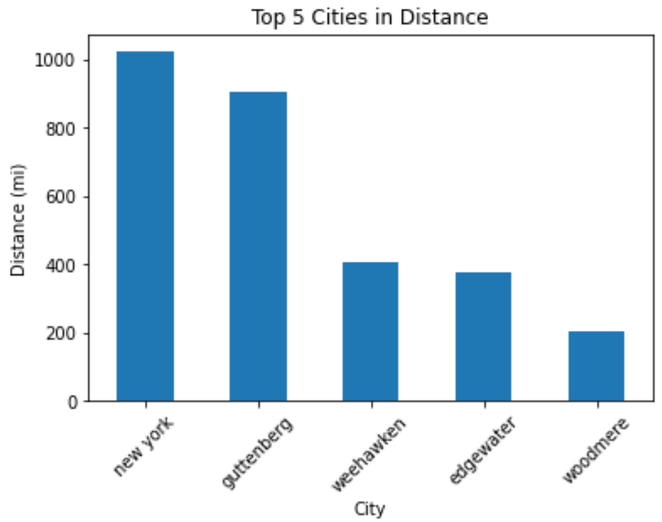
DOLLARS SPENT PER RIDE

Groupby City & Fare Amount Sums and divide by the count of rides for each city. Sort to find top 5

Surprisingly Guttenberg and New York are not the highest on this list for dollars spent per ride in each city.

City Evaluation

```
bycities=new_uber_df_cleaned.groupby("City")["Distance"].sum().sort_values(ascending=False).head(5).plot(kind='bar')

# Set a title for the chart
plt.title("Top 5 Cities in Distance")
plt.ylabel("Distance (mi)")
plt.xticks(rotation=45)
plt.show()
plt.tight_layout()
```



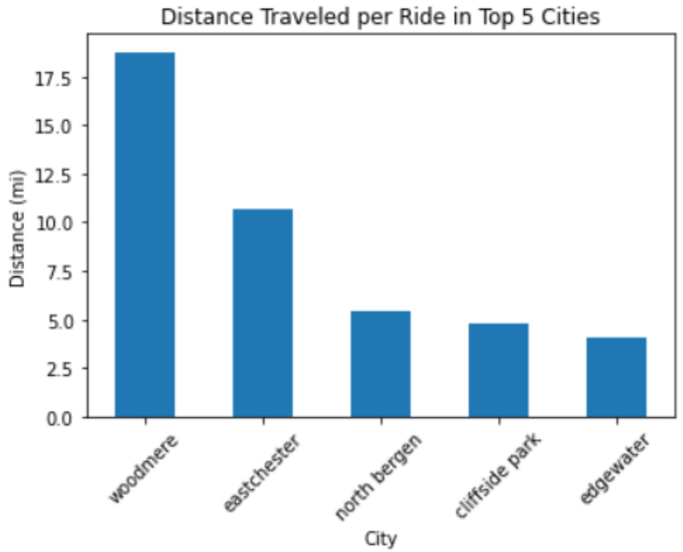
TOP 5 CITIES BY DISTANCE

Groupby City & Distance Sums.
Sort to find top 5

Similar to the spending analysis, this does not provide a full picture due to the sheer number of rides/ people.

City Evaluation

```
bycities=new_uber_df_cleaned.groupby("City")["Distance"].sum()
percityspent=bycities/ridespercity
percityspent.sort_values(ascending=False).head(5).plot(kind="bar")
# Set a title for the chart
plt.title("Distance Traveled per Ride in Top 5 Cities")
plt.ylabel("Distance (mi)")
plt.xticks(rotation=45)
plt.show()
plt.tight_layout()
```



DISTANCE TRAVELED PER RIDE IN TOP 5 CITIES

Groupby City & Distance Sums and divide by the count of rides for each city. Sort to find top 5

Woodmere and Eastchester 20-30 mi away from central Manhattan. Commutes into the city may factor into this analysis.

Distance & Fare

```
# Distance Basic Stats - Daniel
uber_stats_df = pd.DataFrame({
    "Average Distance": [uber_df_cleaned["Distance"].mean()],
    "Total of Distances": [uber_df_cleaned["Distance"].sum()],
    "Number of Trips": [uber_df_cleaned["Distance"].count()],
    "Minimum Distance": [uber_df_cleaned["Distance"].min()],
    "Max Distance": [uber_df_cleaned["Distance"].max()]})

uber_stats_df.style.format({"Number of Unique Distances":"{:,.2f}",
                            "Average Distance":"{:,.2f}",
                            "Total of Distances":"{:,.2f}"})
```

	Average Distance	Total of Distances	Number of Trips	Minimum Distance	Max Distance
0	21.26	20,881.83	982	0.000140	8667.287756

```
# Fare Amount Basic Stats - Daniel
uber_fare_dist_df = pd.DataFrame({
    "Average Fare Amount": [uber_df_cleaned["fare_amount"].mean()],
    "Total Fare Amounts": [uber_df_cleaned["fare_amount"].sum()],
    "Number of Trips": [uber_df_cleaned["fare_amount"].count()],
    "Minimum Fare Amount": [uber_df_cleaned["fare_amount"].min()],
    "Max Fare Amount": [uber_df_cleaned["fare_amount"].max()]})



uber_fare_dist_df.style.format({"Average Fare Amount":"${:,.2f}",
                            "Total Fare Amounts":"${:,.2f}",
                            "Minimum Fare Amount":"${:,.2f}"})
```

	Average Fare Amount	Total Fare Amounts	Number of Trips	Minimum Fare Amount	Max Fare Amount
0	$11.49	$11,278.65	982	$2.50	83.150000

BASIC STATS FOR DISTANCE & FARE AMOUNTS

Dataframes created to house basic stats for Distance and Fare analysis

Max Distance shows irregularity in data

Distance & Fare

```
# Identify the trip that has a distance over 8000 - Daniel
uber_df_cleaned[uber_df_cleaned['Distance'] > 8000]
```

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	
114468	51581870	2012-06-29 09:21:00.00000040	49.8	2012-06-29 09:21:00+00:00	-73.99085	40.747615	0.000000	0.000000	1	
16028	29195565	2011-02-08 07:53:30.0000003	6.1	2011-02-08 07:53:30+00:00	0.00000	0.000000	-74.001273	40.742693	0	tak

```
# Filter dataframe with a Distance less than 100 - Daniel
new_uber_df_cleaned = uber_df_cleaned[uber_df_cleaned['Distance'] < 100]
new_uber_df_cleaned
```

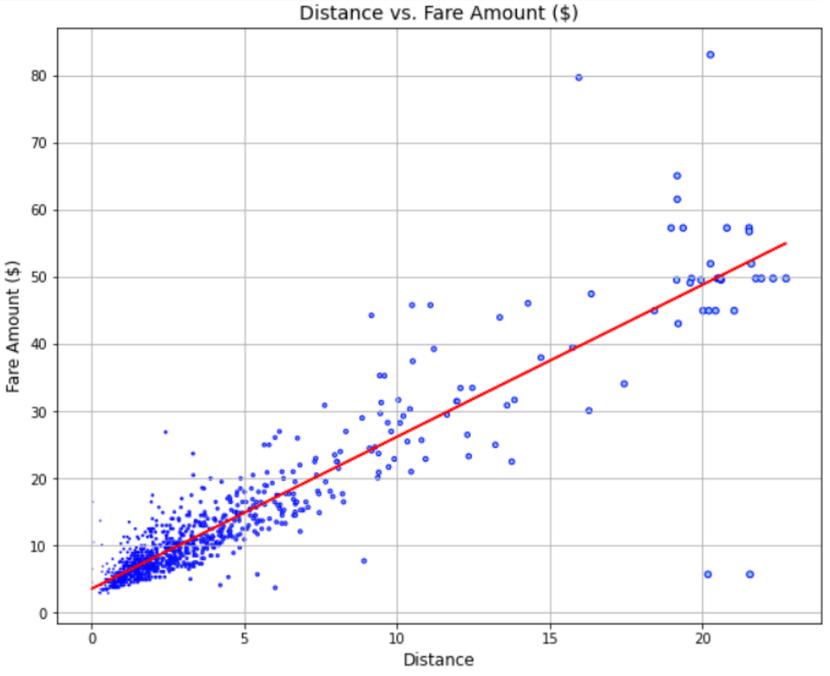
MORE CLEAN UP

While trying to analyze data, found outliers in Distance

Filtered out the trips with more than 100 miles

Distance & Fare

```
# Distance vs Price plot with Linear Regression - Daniel
from scipy.stats import linregress

(slope, intercept, rvalue, pvalue, stderr) = linregress(uber_dist, uber_fare)
regress_values = uber_dist * slope + intercept

# Create scatter plot
plt.figure(figsize=(10, 8))
plt.scatter(uber_dist, uber_fare, uber_dist, marker="o", facecolors="lightskyblue", edgecolors="blue")
plt.plot(uber_dist,regress_values,"r-")

line_eq=f"y={round(slope,2)}x+{round(intercept,2)}"

# Assign to the plot, title, x label and y label
plt.title("Distance vs. Fare Amount ($)", fontsize=14)
plt.xlabel("Distance", fontsize=12)
plt.ylabel("Fare Amount ($)", fontsize=12)
plt.grid()
plt.annotate(line_eq,(25,25),fontsize=15,color="red")

# Print plot
plt.show()
```




```
r value: 0.9097839290653043
```

DISTANCE VS FARE AMOUNT

Scatter plot created for Distance vs Fare Amount

Linear Regression performed on the same plot

R value shows high positive correlation between Distance and Fare Amount

Pick Up times

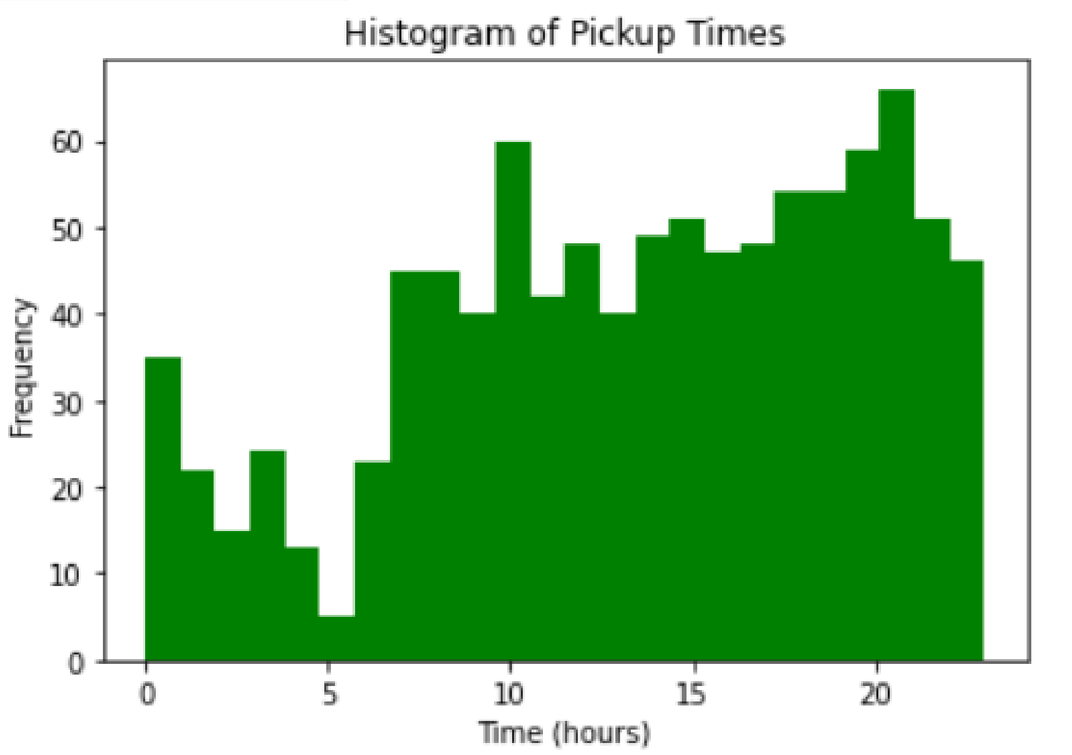
```
# Most Popular Pick up times & how it correlates to price (plot, basic stats dataframe) -Neil
#Convert pickup_datetime to datetime object
uber_df_cleaned["pickup_datetime"] = pd.to_datetime(uber_df_cleaned["pickup_datetime"])

#Extract time from datetime object
uber_df_cleaned["pickup_hour"] = uber_df_cleaned.loc[:,"pickup_datetime"].dt.hour



#Generate summary stats, histogram
t=uber_df_cleaned["pickup_hour"]
t_mean = uber_df_cleaned["pickup_hour"].mean()
t_med = uber_df_cleaned["pickup_hour"].median()
t_mode = uber_df_cleaned["pickup_hour"].mode()
t_corr = t.corr(uber_df_cleaned["fare_amount"], method='pearson')
t_df = pd.DataFrame({"Mean Pickup Time (hours)":t_mean,
                     "Median Pickup Time (hours)":t_med,
                     "Modal Pickup Time (hours)":t_mode,
                    "Correlation between time and fare":t_corr})


plt.hist(t, bins=24, range =(t.min(), t.max()), color='green')
plt.ylabel("Frequency")
plt.xlabel("Time (hours)")
plt.title("Histogram of Pickup Times")
t_df
```

	Mean Pickup Time (hours)	Median Pickup Time (hours)	Modal Pickup Time (hours)	Correlation between time and fare
0	13.538697	14.0	21	-0.005735

PICK UP TIMES STATS

Basic Stats for pick up times stored in dataframe

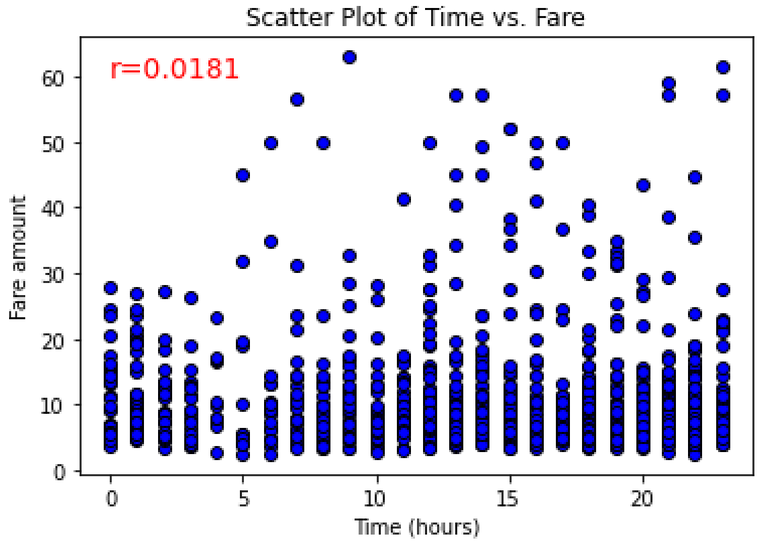
Use of to_datetime to obtain the hour



Pick Up times & Fare amount

```
plt.scatter(uber_df_cleaned["pickup_hour"],uber_df_cleaned["fare_amount"], facecolors="blue", edgecolors="black")
plt.ylabel("Fare amount")
plt.xlabel("Time (hours)")
plt.title("Scatter Plot of Time vs. Fare")
plt.annotate(f"r={round(t_corr,4)}",xy=(0,60),fontsize=14,color="red")
```

```
Text(0, 60, 'r=0.0181')
```



PICK UP TIMES & FARE AMOUNT

Low r value shows and scatter plot shows no significant correlation between pick up times and fare amount

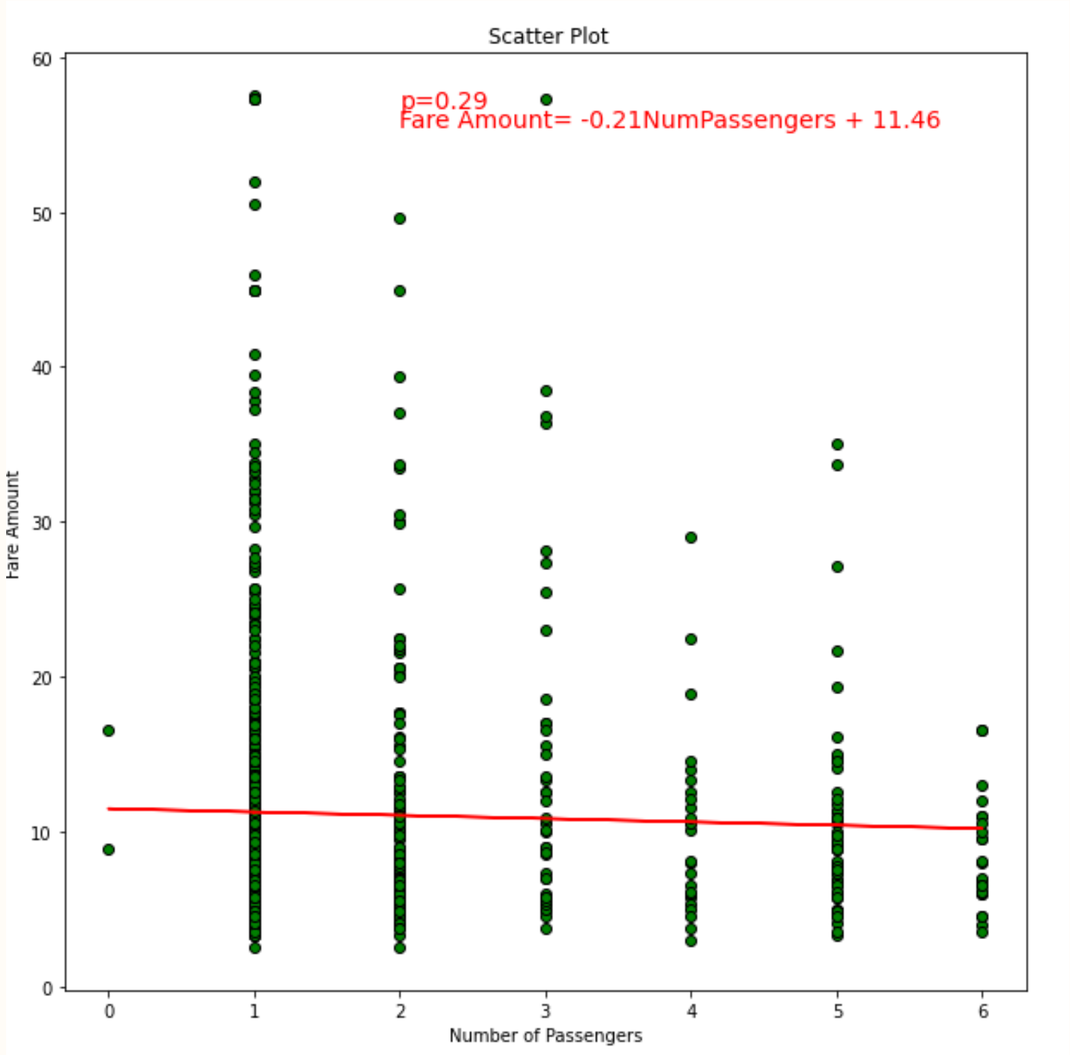
Number of Passengers

```
# Price regression on number of passengers (plot, stats)- Neil
y1 = new_uber_df_cleaned["fare_amount"]
x1 = new_uber_df_cleaned["passenger_count"]

#Generate Regression
(slope, intercept, rvalue, pvalue, stderr) = stats.linregress(x1, y1)
regress_values = slope*x1 + intercept
r_squared = rvalue**2
line_eq = "Fare Amount= " + str(round(slope,2)) + "NumPassengers + " + str(round(intercept,2))


#Generate Plot
plt.scatter(x1,y1, facecolors="green", edgecolors="black")
plt.gcf().set_size_inches(10, 10)
plt.annotate(line_eq,xy=(2,max(y1)-2),fontsize=14,color="red")
plt.annotate(f"p={round(pvalue,2)}",xy=(2,max(y1)-.8),xycoords='data',fontsize=14,color="red")
plt.plot(x1,regress_values,"r-")
plt.xlabel("Number of Passengers")
plt.ylabel("Fare Amount")
plt.title(" Scatter Plot")
```

NUMBER OF PASSENGERS

Obtained Fare amount and passenger count to analyze the correlation between the two variables.

Number of Passengers



NUMBER OF PASSENGERS

Scatter plot was created and linear regression was attempted, but showed no useful correlation between Number of Passengers and Fare amount

Pick Up times

```
# Price regression if passengers > 2 (plot, stats)- Neil
uber_df_cleaned['Over 2 passengers'] = uber_df_cleaned['passenger_count']>2


y2 = uber_df_cleaned["fare_amount"]
x2 = uber_df_cleaned['Over 2 passengers']

#Generate Regression
(slope2, intercept2, rvalue2, pvalue2, stderr2) = stats.linregress(x2, y2)
regress_values2 = slope2*x2 + intercept
r_squared2 = rvalue2**2
line_eq2 = "Fare Amount= " + str(round(slope2,2)) + "DummyVar+ " + str(round(intercept2,2))

print(f"the p value is {pvalue} and the r-squared is {r_squared}")
print(line_eq2)
```

```
the p value is 0.29099781991223933 and the r-squared is 0.0011638120025431143
Fare Amount= -0.35DummyVar+ 11.1
```

DUMMY VARIABLE

Evaluated dataset for rides with more than 2 passengers for additional analysis

Conclusions

ARE INCOME & NUMBER OF RIDES CORRELATED?

WHAT ARE THE LOCATIONS FOR THE MOST POPULAR DROP OFF AND PICK UP PLACES?

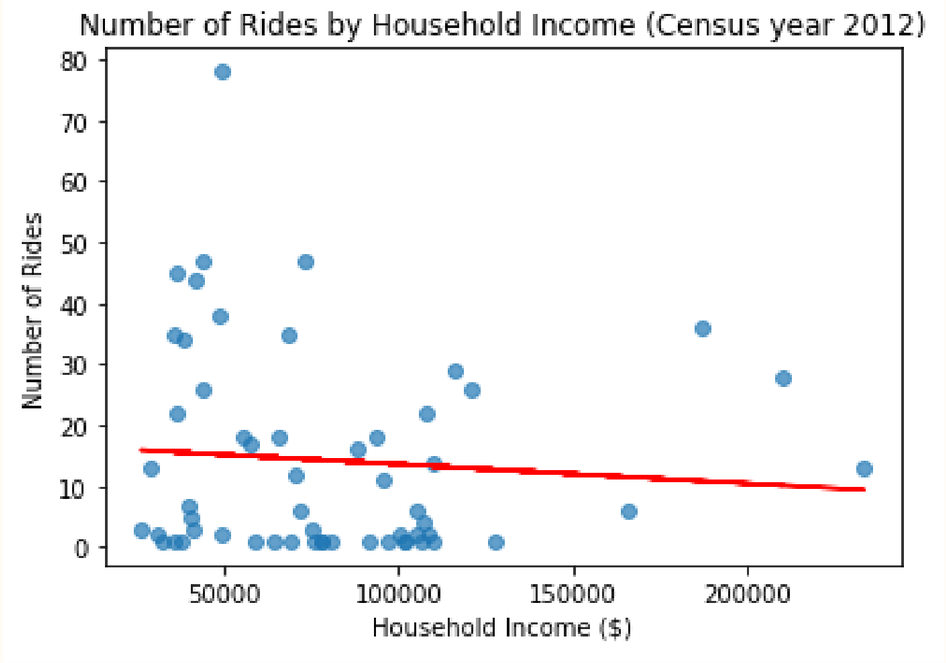
ARE DISTANCE AND FARE AMOUNT CORRELATED?

WHAT ARE THE MOST POPULAR TIMES OF TRAVEL?

ARE PICK UP TIMES AND FARE CORRELATED?

DO NUMBER OF PASSENGERS AFFECT FARE AMOUNT?

```
The R value is -0.08
```



CONCLUSIONS

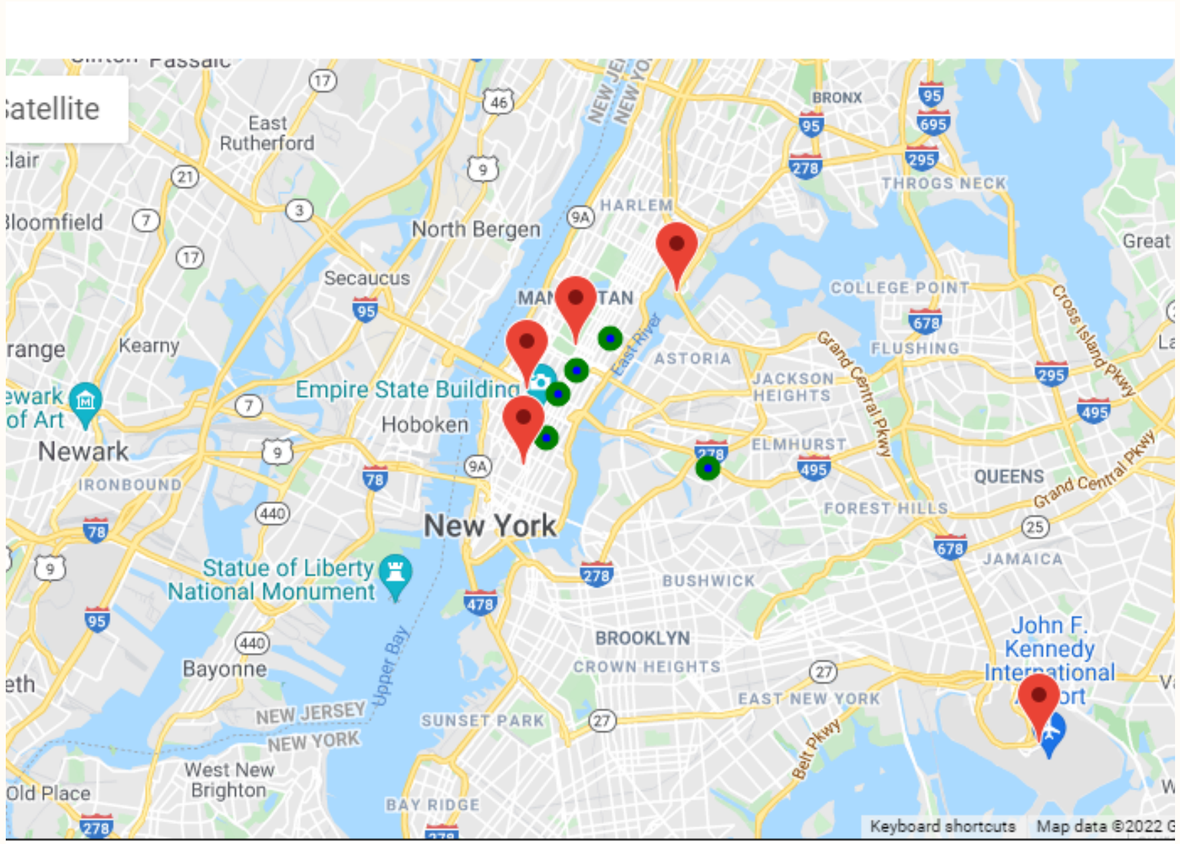
Are income & number of rides correlated?

LOW R VALUE

Shows little to none correlation between income and number of rides

INITIAL ASSUMPTION

People who make more money take more uber rides



Pick Up Locations

Drop off Locations

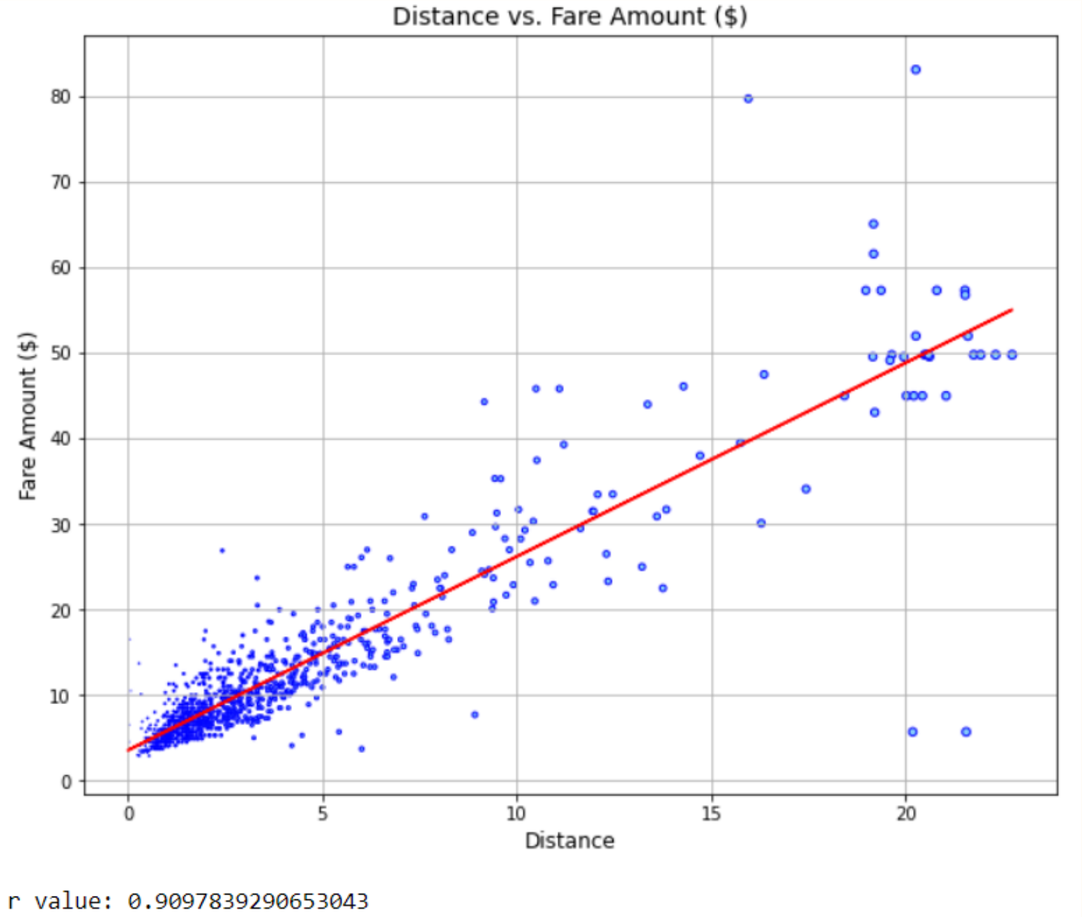
CONCLUSIONS

What are the locations for the most popular drop off and pick up places?

MEAN OF ALL LOCATIONS TO FIND MOST POPULAR LAT & LONG PAIR

JFK AIRPORT IS A POPULAR PICK UP LOCATION

Data set is New York central




```
r value: 0.9097839290653043
```

CONCLUSIONS

Are distance and fare amount correlated?

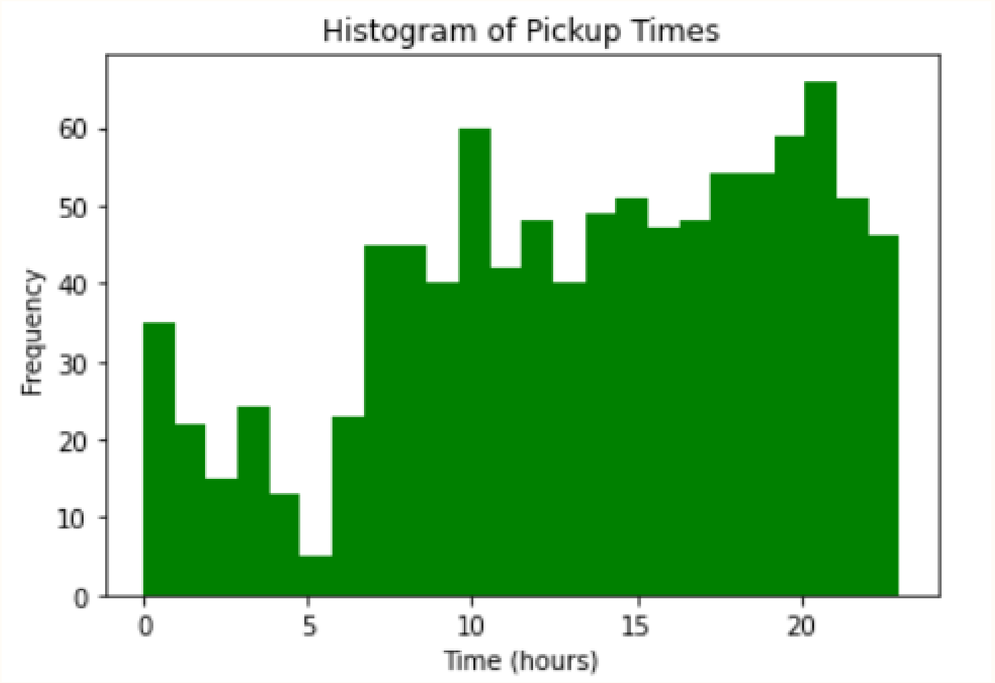
HIGH R VALUE

R Value is 0.9, indicates high positive correlation

The longer the distance, the more you will pay

CLUSTERING

Most trips are happening below 5 miles



CONCLUSIONS

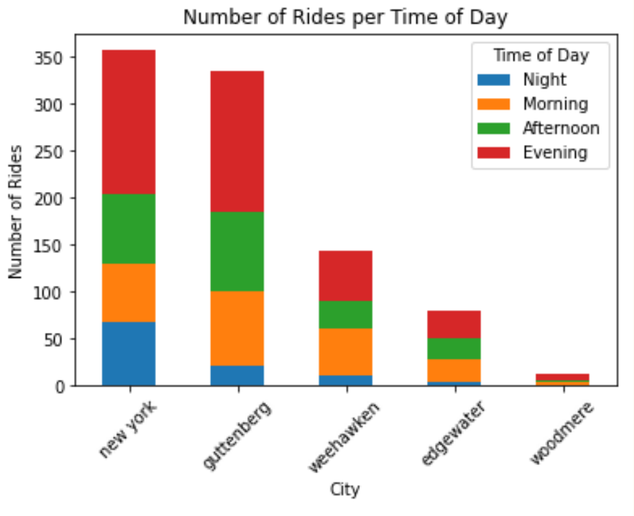
What are the most popular times of travel?

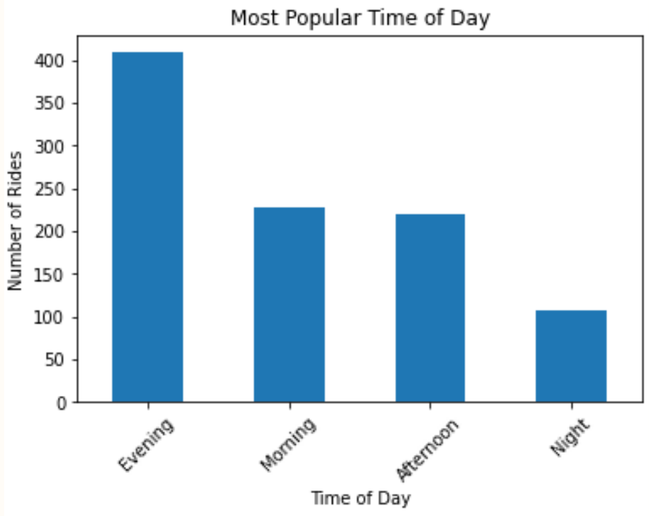
MODAL PICK UP TIME

Modal pick up time is 21 - 9 PM

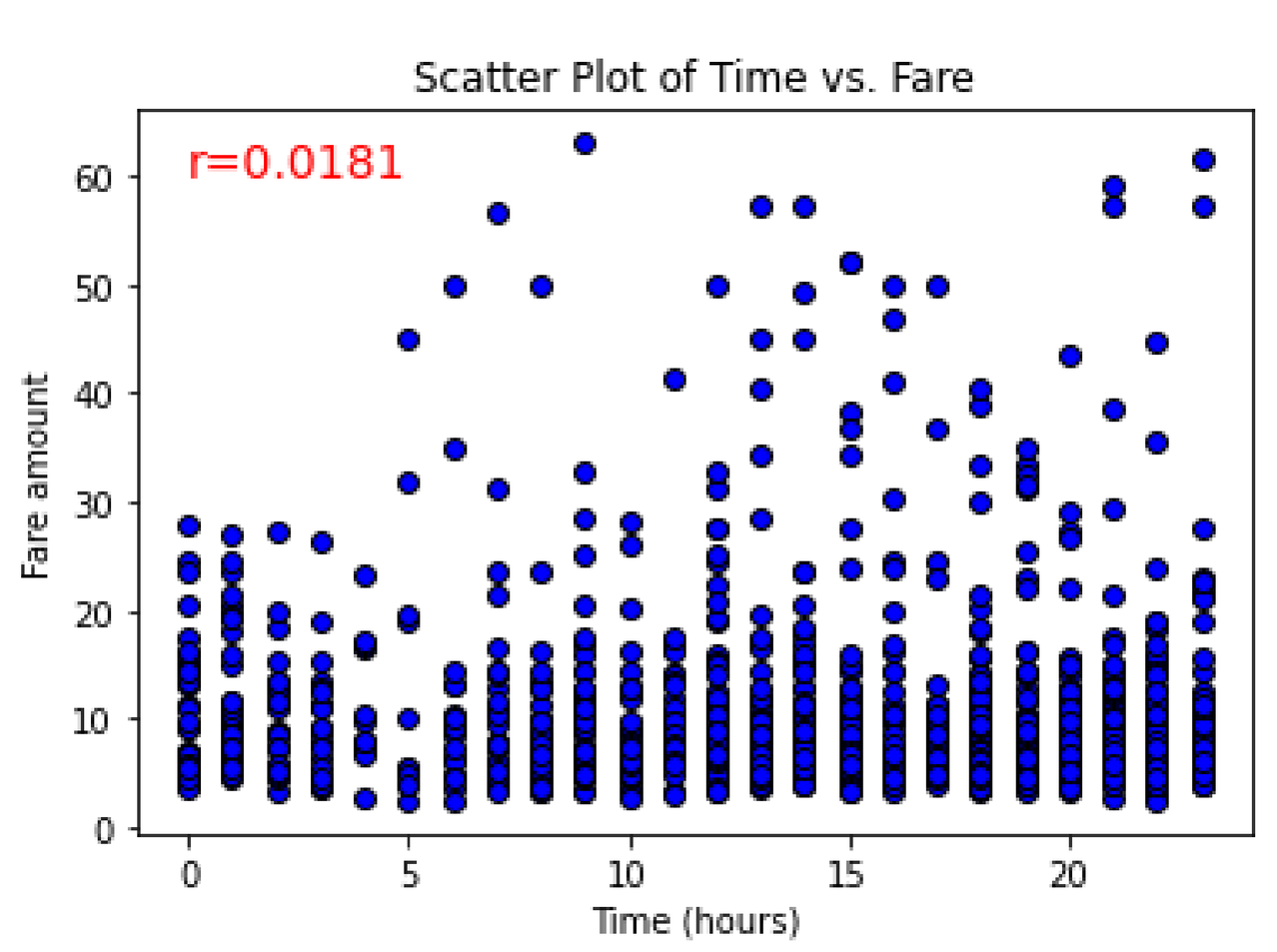
TIME OF DAY

For the top 5 cities and overall data set, Evening times (17-24) from 5 PM to Midnight were the most popular





	Mean Pickup Time (hours)	Median Pickup Time (hours)	Modal Pickup Time (hours)	Correlation between time and fare
0	13.497942	14.0	21	0.018146



CONCLUSIONS

Are Pick up times and Fare correlated?

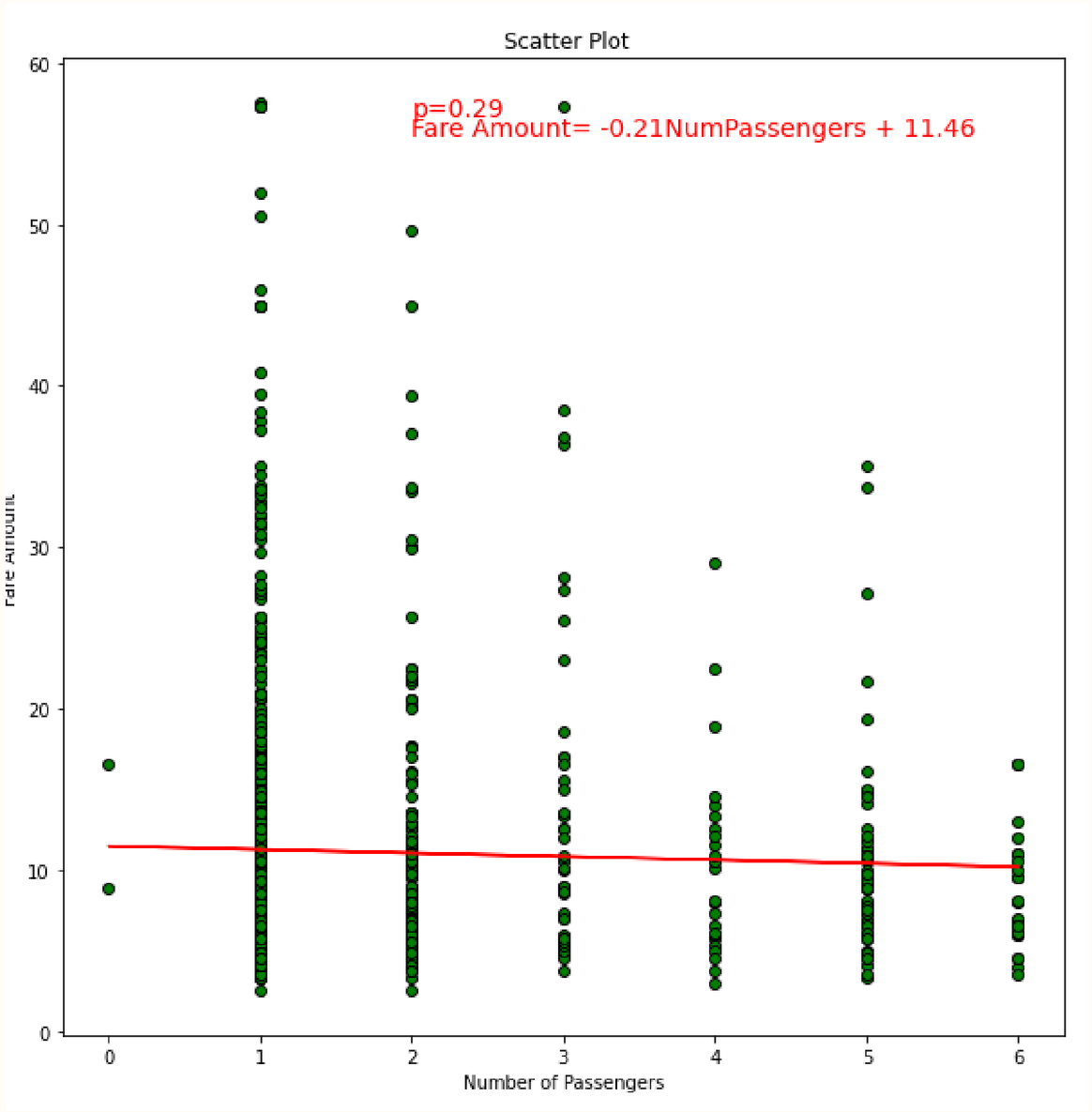
INITIAL ASSUMPTION

Fares would spike at popular times of the day

LOW R VALUE

R value shows little to no correlation between pick up times and fare

	Mean Pickup Time (hours)	Median Pickup Time (hours)	Modal Pickup Time (hours)	Correlation between time and fare
0	13.497942	14.0	21	0.018146



CONCLUSIONS

Does the number of passengers affect the fare amount?

LOW P VALUE

P value shows little to no correlation between number of passengers and fare

Questions