kaviyadevi 20106064

In [1]:

```
#to import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
#to import dataset
data1=pd.read_csv(r"C:\Users\user\Downloads\7_uber - 7_uber.csv")
data1
```

Out[2]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	drop
0	24238194	2015-05-07 19:52:06	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	
1	27835199	2009-07-17 20:04:56	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	
2	44984355	2009-08-24 21:45:00	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	
3	25894730	2009-06-26 08:22:21	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	
4	17610152	2014-08-28 17:47:00	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	
...	...	...	...	...	...	...	
199995	42598914	2012-10-28 10:49:00	3.0	2012-10-28 10:49:00 UTC	-73.987042	40.739367	
199996	16382965	2014-03-14 01:09:00	7.5	2014-03-14 01:09:00 UTC	-73.984722	40.736837	
199997	27804658	2009-06-29 00:42:00	30.9	2009-06-29 00:42:00 UTC	-73.986017	40.756487	
199998	20259894	2015-05-20 14:56:25	14.5	2015-05-20 14:56:25 UTC	-73.997124	40.725452	
199999	11951496	2010-05-15 04:08:00	14.1	2010-05-15 04:08:00 UTC	-73.984395	40.720077	

200000 rows × 9 columns

In [3]:
```
#to display top 5 rows
data=data1.head(200)
data
```

Out[3]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_
0	24238194	2015-05-07 19:52:06	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-
1	27835199	2009-07-17 20:04:56	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-
2	44984355	2009-08-24 21:45:00	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-
3	25894730	2009-06-26 08:22:21	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-
4	17610152	2014-08-28 17:47:00	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-
...	...	...	...	...	...	...	
195	49202586	2014-05-28 01:00:00	14.5	2014-05-28 01:00:00 UTC	-74.005477	40.738575	-
196	51452192	2009-05-12 10:32:00	24.0	2009-05-12 10:32:00 UTC	-73.981558	40.783752	-
197	45317989	2012-08-07 20:53:18	10.5	2012-08-07 20:53:18 UTC	-73.965930	40.805358	-
198	41858701	2009-09-24 16:21:42	8.9	2009-09-24 16:21:42 UTC	-73.952080	40.790119	-
199	13472186	2011-04-03 00:01:40	14.1	2011-04-03 00:01:40 UTC	-74.000190	40.718336	-

200 rows × 9 columns

DATA CLEANING AND PREPROCESSING

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         200 non-null    int64  
 1   key                200 non-null    object 
 2   fare_amount        200 non-null    float64
 3   pickup_datetime    200 non-null    object 
 4   pickup_longitude   200 non-null    float64
 5   pickup_latitude    200 non-null    float64
 6   dropoff_longitude  200 non-null    float64
 7   dropoff_latitude   200 non-null    float64
 8   passenger_count    200 non-null    int64  
dtypes: float64(5), int64(2), object(2)
memory usage: 14.2+ KB
```

In [5]:
```
#to display summary of statistics
data.describe()
```

Out[5]:

	Unnamed: 0	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_lati
count	2.000000e+02	200.000000	200.000000	200.000000	200.000000	200.00
mean	2.779091e+07	10.620050	-71.388553	39.327046	-71.387016	39.32
std	1.578378e+07	8.023976	13.629815	7.508297	13.629487	7.50
min	2.268700e+05	2.500000	-74.015122	0.000000	-74.016152	0.00
25%	1.418957e+07	6.000000	-73.992744	40.736897	-73.989371	40.73
50%	2.799295e+07	8.100000	-73.982225	40.753583	-73.979274	40.75
75%	4.126453e+07	12.125000	-73.968338	40.766672	-73.962785	40.77
max	5.519870e+07	56.800000	0.001782	40.850558	0.000875	40.89

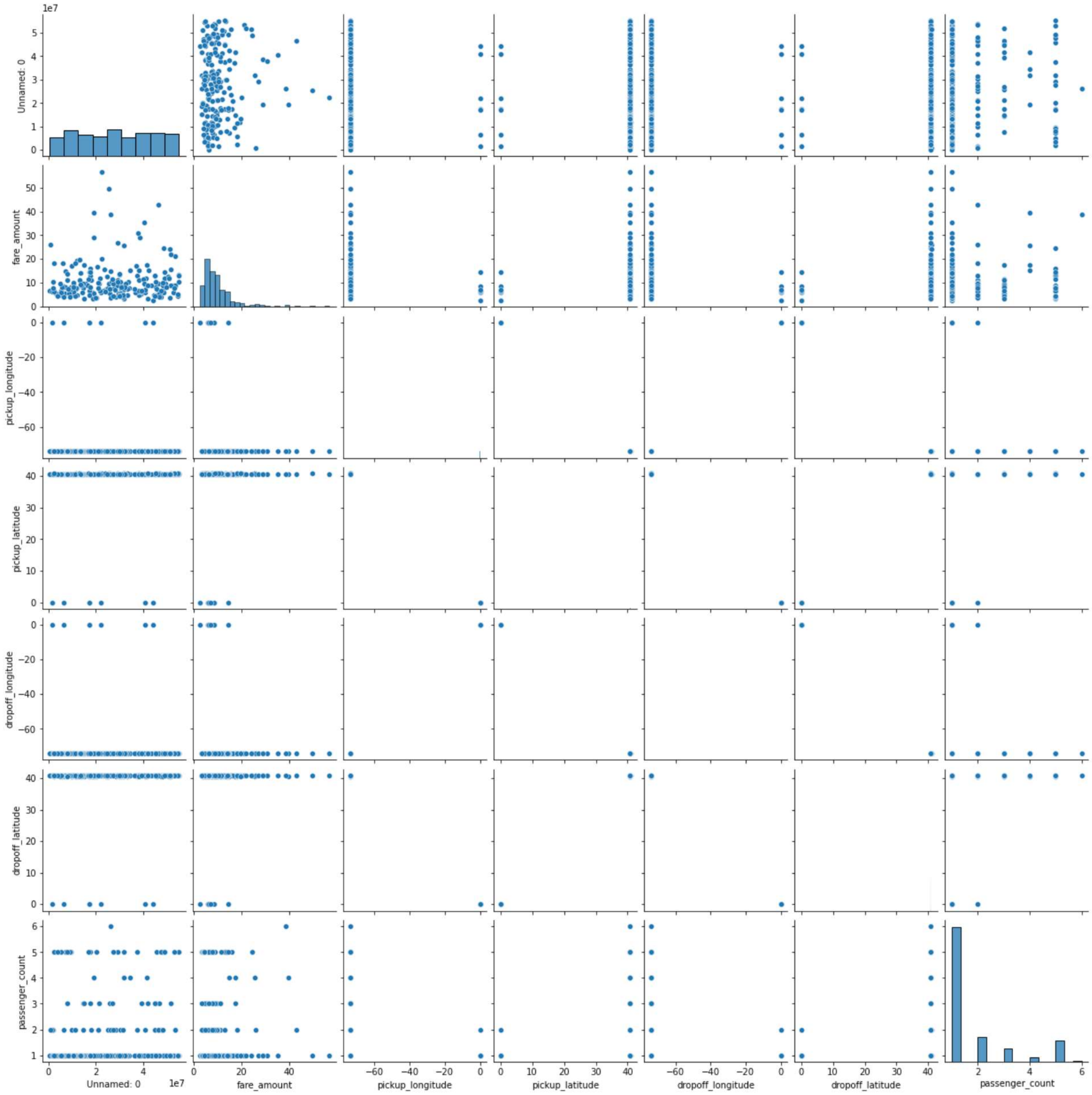
In [6]:
```
#to display the column heading
data.columns
```

Out[6]:
```
Index(['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count'],
      dtype='object')
```

EDA and DATA VISUALIZATION

In [7]:
```
sns.pairplot(data)
```

Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x22fb57ed820>
```



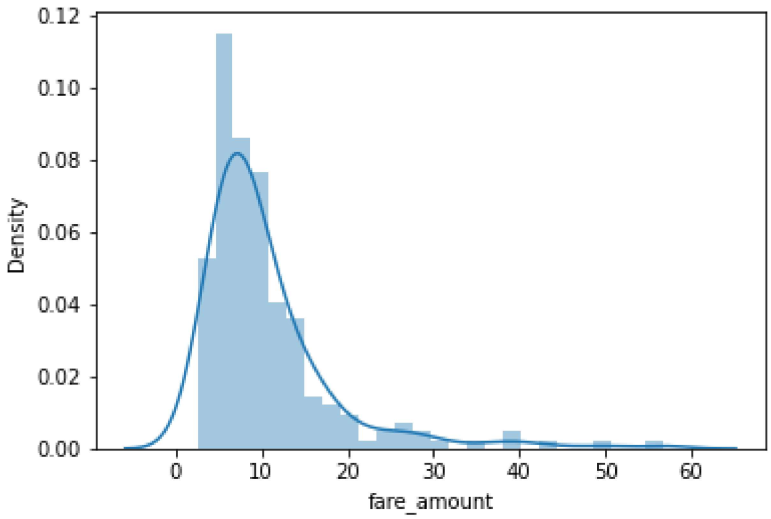
In [8]:

```
sns.distplot(data['fare_amount'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

Out[8]:

```
<AxesSubplot:xlabel='fare_amount', ylabel='Density'>
```

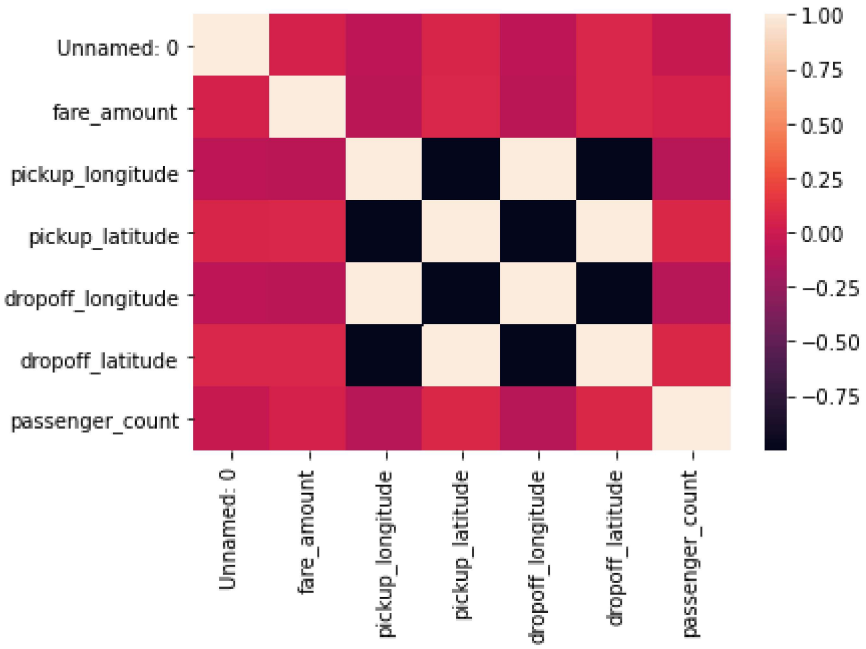


In [11]:

```
df=data[['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count']]
```

In [12]:
```
sns.heatmap(df.corr())
```

Out[12]:
```
<AxesSubplot:>
```



TRAINING MODEL

In [13]:
```
x=df[['pickup_longitude', 'pickup_latitude', 'dropoff_longitude','dropoff_latitud
y=df[['fare_amount']]
```

In [14]:
```
#to split my dataset into trainning and test
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [15]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

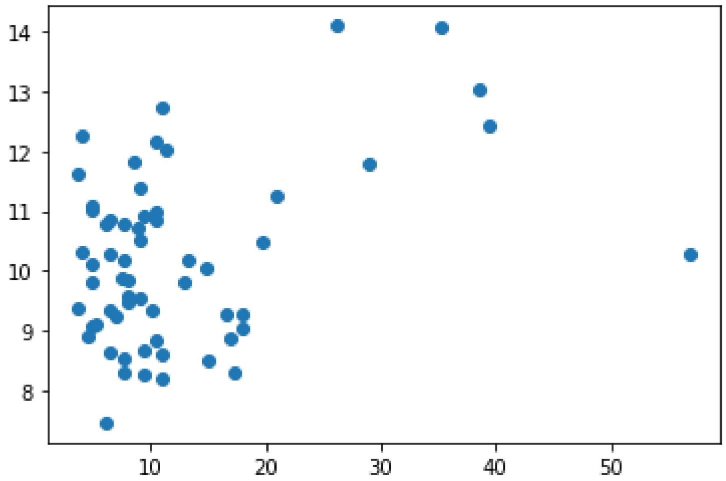
Out[15]:
```
LinearRegression()
```

In [16]:
```
#to find intercept
print(lr.intercept_)
```

```
[7.60835568]
```

In [17]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[17]:
```
<matplotlib.collections.PathCollection at 0x22fcc5a21c0>
```



In [18]:
```
print(lr.score(x_test,y_test))
```

```
0.04538298035867061
```

RIDGE AND LASSO REGRESSION

In [19]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [20]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[20]:
```
Ridge(alpha=10)
```

In [21]:
```
rr.score(x_test,y_test)
```

Out[21]:
```
-0.05278160832023482
```

In [22]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[22]:
```
Lasso(alpha=10)
```

In [23]:
```
la.score(x_test,y_test)
```

Out[23]:
```
-0.056041047709260994
```

In [24]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[24]:
```
ElasticNet()
```

In [25]:
```
print(en.coef_)
```

```
[-0.00852009  0.         -0.02340496  0.         -0.        ]
```

In [26]:
```
print(en.predict(x_test))
```

```
[10.01981069 10.01961678 10.0195846  10.01781808 10.01914812 10.01961083
 10.01893757 10.01785182 10.0202087  10.02020181 10.01971636 10.01947523
 10.01918567 10.01956863 10.01878375 10.01958358 10.01956893 10.01952649
 10.02039825 10.01906104 10.01631723 10.01958536 10.02036157 10.01966399
 10.0199183   7.6577922  10.01906798 10.01995257 10.01962596 10.01971264
 10.0190412  10.02007035 10.01941289 10.01991139 10.01955201 10.0193181
 10.01857914 10.01998333 10.01976073 10.01962461 10.01935984 10.02003851
 10.01528182 10.01883033 10.01804167 10.02017321 10.01986961 10.01958406
 10.01977546 10.02039835 10.01907911 10.01961409 10.01897744 10.01949854
 10.01960819 10.01852073 10.01903095 10.01918528 10.01885797 10.01927063]
```

In [27]:
```
print(en.score(x_test,y_test))
```

```
-0.04918612859435023
```

In [28]:
```
from sklearn import metrics
```

In [29]:
```
print("Mean Absolute error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute error 5.868756045570609
```

In [30]:
```
print("Mean Squared error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared error 93.1911195021879
```

In [31]:
```
print("Root Mean Absolute error",np.sqrt(metrics.mean_squared_error(y_test,predic
```

```
Root Mean Absolute error 9.653554759889639
```