In [1]:

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

DATA COLLECTION

In [2]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\uber - uber.csv")
a
```

Out[2]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude
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 8:22:21	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 1:09:00	7.5	2014-03-14 01:09:00 UTC	-73.984722	40.736837
199997	27804658	2009-06-29 0:42:00	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 4:08:00	14.1	2010-05-15 04:08:00 UTC	-73.984395	40.720077

200000 rows × 9 columns

In [3]:

```
b=a.head(100)
b
```

Out[3]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dro
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 8:22:21	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	
...	...	...	...	...	...	...	
95	25431833	2015-04-11 8:47:47	9.5	2015-04-11 08:47:47 UTC	-73.978432	40.752399	
96	44792012	2011-10-03 20:29:00	4.5	2011-10-03 20:29:00 UTC	-73.990055	40.756413	
97	18571020	2010-04-26 3:12:44	3.3	2010-04-26 03:12:44 UTC	-73.982326	40.731314	
98	37942404	2011-11-18 9:51:00	30.9	2011-11-18 09:51:00 UTC	-73.995888	40.759078	
99	29024472	2009-08-30 14:03:55	26.9	2009-08-30 14:03:55 UTC	-73.990137	40.756007	

100 rows × 9 columns

DATA CLEANING AND PRE-PROCESSING

In [4]:

```
b.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         100 non-null    int64  
 1   key                100 non-null    object 
 2   fare_amount        100 non-null    float64
 3   pickup_datetime    100 non-null    object 
 4   pickup_longitude   100 non-null    float64
 5   pickup_latitude    100 non-null    float64
 6   dropoff_longitude  100 non-null    float64
 7   dropoff_latitude   100 non-null    float64
 8   passenger_count    100 non-null    int64  
dtypes: float64(5), int64(2), object(2)
memory usage: 7.2+ KB
```

In [5]:

```
b.describe()
```

Out[5]:

	Unnamed: 0	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropc
count	1.000000e+02	100.000000	100.000000	100.000000	100.000000	
mean	2.810554e+07	11.065700	-71.019759	39.123621	-71.015479	
std	1.635033e+07	9.029756	14.569902	8.026358	14.569028	
min	2.268700e+05	2.500000	-74.013173	0.000000	-74.016152	
25%	1.422691e+07	5.475000	-73.992601	40.733982	-73.989142	
50%	2.710896e+07	8.100000	-73.982002	40.752764	-73.979396	
75%	4.480811e+07	12.600000	-73.968615	40.765572	-73.960980	
max	5.508597e+07	56.800000	0.000000	40.850558	0.000000	

In [6]:

```
b.columns
```

Out[6]:

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

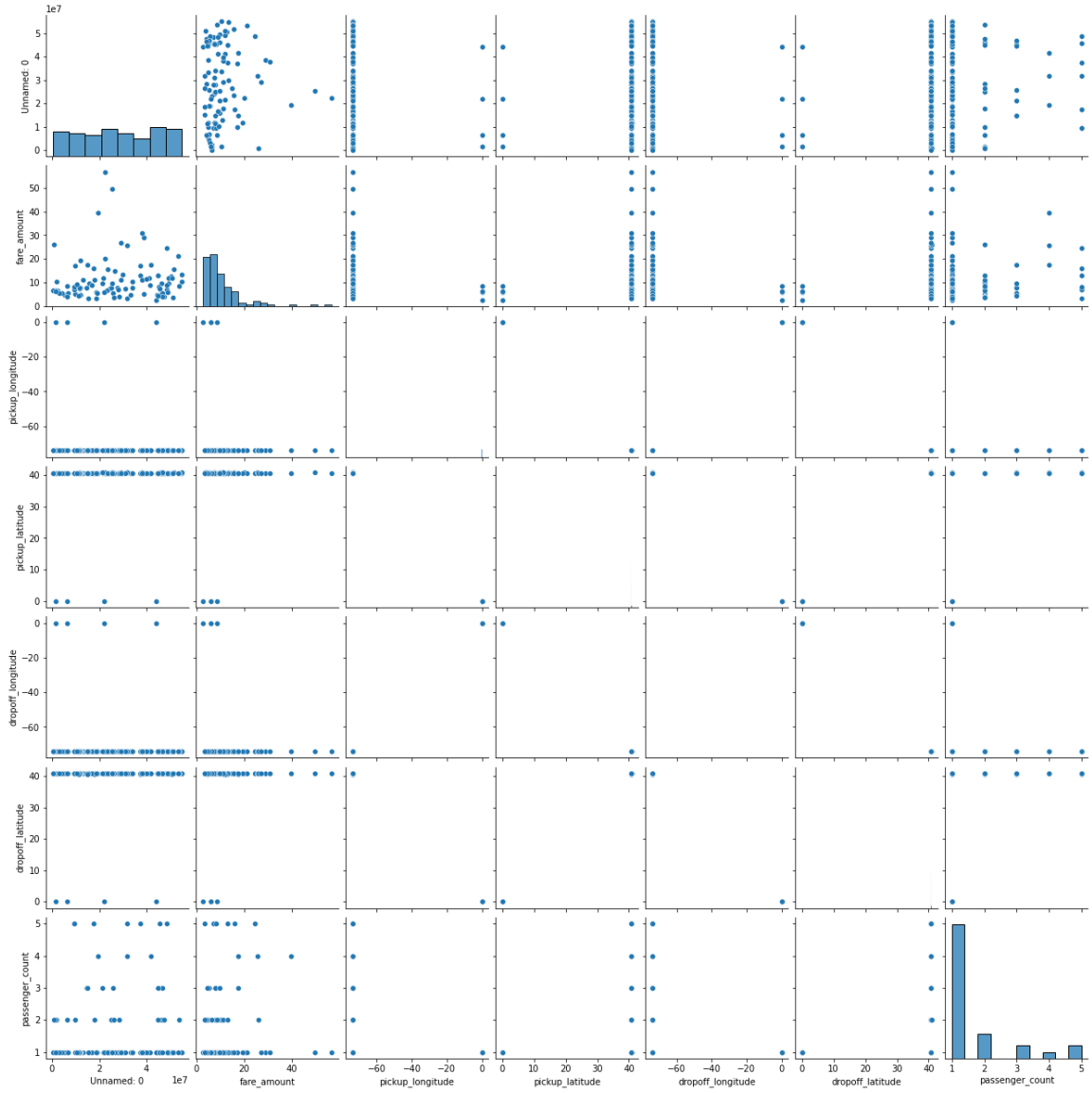
EDA AND VISUALIZATION

In [7]:

```
sns.pairplot(b)
```

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x1d1255655b0>
```



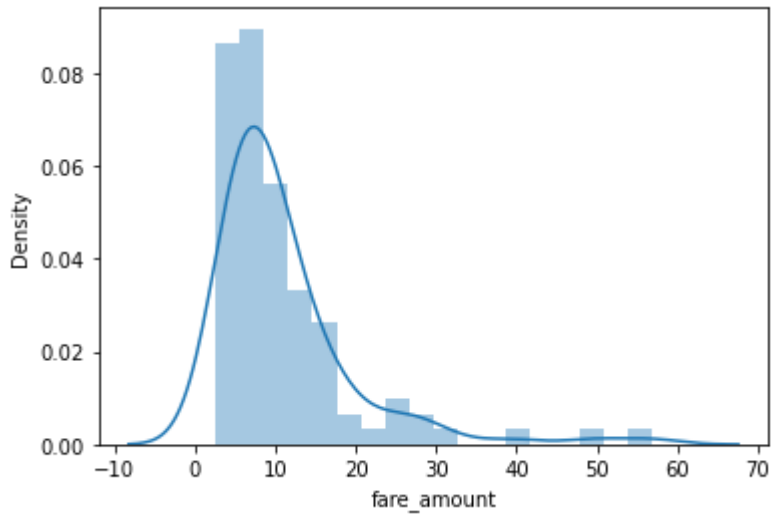
In [8]:

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

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

Out[8]:

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



In [9]:

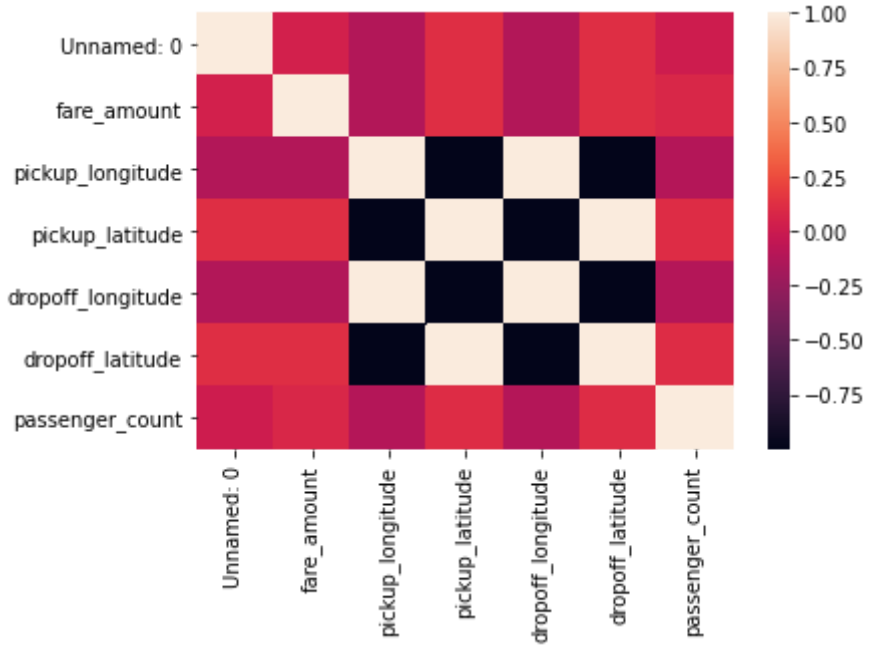
```
f=b[['Unnamed: 0', 'key', 'fare_amount',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count']]
```

In [10]:

```
sns.heatmap(f.corr())
```

Out[10]:

```
<AxesSubplot:>
```



In [11]:

```
x=f[['Unnamed: 0',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count']]
y=f['fare_amount']
```

In [12]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.5)
```

In [13]:

```
from sklearn.linear_model import LinearRegression

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

Out[13]:

```
LinearRegression()
```

In [14]:

```
print(lr.intercept_)
```

```
8.437358801228955
```

In [15]:

```
r=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
r
```

Out[15]:

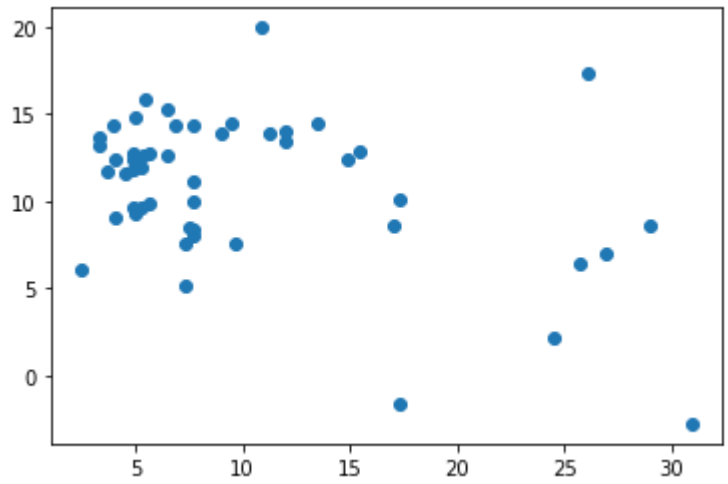
	Co-efficient
Unnamed: 0	-2.238157e-08
pickup_longitude	1.505395e+02
pickup_latitude	-6.396591e+01
dropoff_longitude	-1.254377e+02
dropoff_latitude	1.096878e+02
passenger_count	-1.344439e+00

In [16]:

```
u=lr.predict(x_test)
plt.scatter(y_test,u)
```

Out[16]:

```
<matplotlib.collections.PathCollection at 0x1d1385df310>
```



In [17]:

```
print(lr.score(x_test,y_test))
```

```
-0.8059419740583886
```

In [18]:

```
lr.score(x_train,y_train)
```

Out[18]:

```
0.13040306175737582
```

RIDGE 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]:

```
-0.13593947856344957
```

LASSO REGRESSION

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]:

```
-0.08315022367105751
```

In [24]:

```
from sklearn.linear_model import ElasticNet
p=ElasticNet()
p.fit(x_train,y_train)
```

Out[24]:

```
ElasticNet()
```

In [25]:

```
print(p.coef_)
```

```
[-3.79391261e-08 -0.00000000e+00  0.00000000e+00 -8.29128322e-02
  0.00000000e+00 -8.59669000e-02]
```

In [26]:

```
print(p.intercept_)
```

```
7.462688322376646
```

In [27]:

```
print(p.predict(x_test))
```

```
[12.04149499 11.82251729 13.3124312  12.87751838 11.57158612 11.66941705
 11.66854488 11.94704898 12.06157654 12.03837717 13.016298   13.08614663
 12.77364495 12.61874536 11.64455307 11.56715823 11.73880082 11.99518505
 12.80684451 13.17653629 12.50765496 12.3401362  11.30910708 13.38877154
 12.35503803 12.34904933 13.06193067 12.79988426 13.06016124 11.78218962
 12.47221931 11.428496   11.65889525 13.37546326 12.54744491 13.35477039
 12.09778035 12.93643265 13.46555087  5.69998346 12.03629919 11.67378508
 12.78349387 12.22402166 11.78382843 12.40525102 13.3950293  13.07083663
 11.43003677 13.05179608]
```

In [28]:

```
prediction=p.predict(x_test)
print(p.score(x_test,y_test))
```

```
-0.10454253764013721
```

In [29]:

```
from sklearn import metrics
```

In [30]:

```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolytre Error: 6.662339984153889
```

In [31]:

```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 59.91668313406558
```

In [32]:

```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root Mean Squared Error: 7.740586743526977
```

In [ ]:

```

```