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

In [2]:
```
d=pd.read_csv(r"C:\Users\Admin\Downloads\uber - uber.csv")
d
```

Out[2]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longit
0	24238194	2015-05-07 19:52:06	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.999
1	27835199	2009-07-17 20:04:56	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.994
2	44984355	2009-08-24 21:45:00	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73.962
3	25894730	2009-06-26 8:22:21	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.965
4	17610152	2014-08-28 17:47:00	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.973
...	...	...	...	...	...	...	...
199995	42598914	2012-10-28 10:49:00	3.0	2012-10-28 10:49:00 UTC	-73.987042	40.739367	-73.986
199996	16382965	2014-03-14 1:09:00	7.5	2014-03-14 01:09:00 UTC	-73.984722	40.736837	-74.006
199997	27804658	2009-06-29 0:42:00	30.9	2009-06-29 00:42:00 UTC	-73.986017	40.756487	-73.858
199998	20259894	2015-05-20 14:56:25	14.5	2015-05-20 14:56:25 UTC	-73.997124	40.725452	-73.983
199999	11951496	2010-05-15 4:08:00	14.1	2010-05-15 04:08:00 UTC	-73.984395	40.720077	-73.985

200000 rows × 9 columns

In [3]:
```
df=d.head(100)
```

In [4]:
```
df.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]:
```
df.describe()
```

Out[5]:

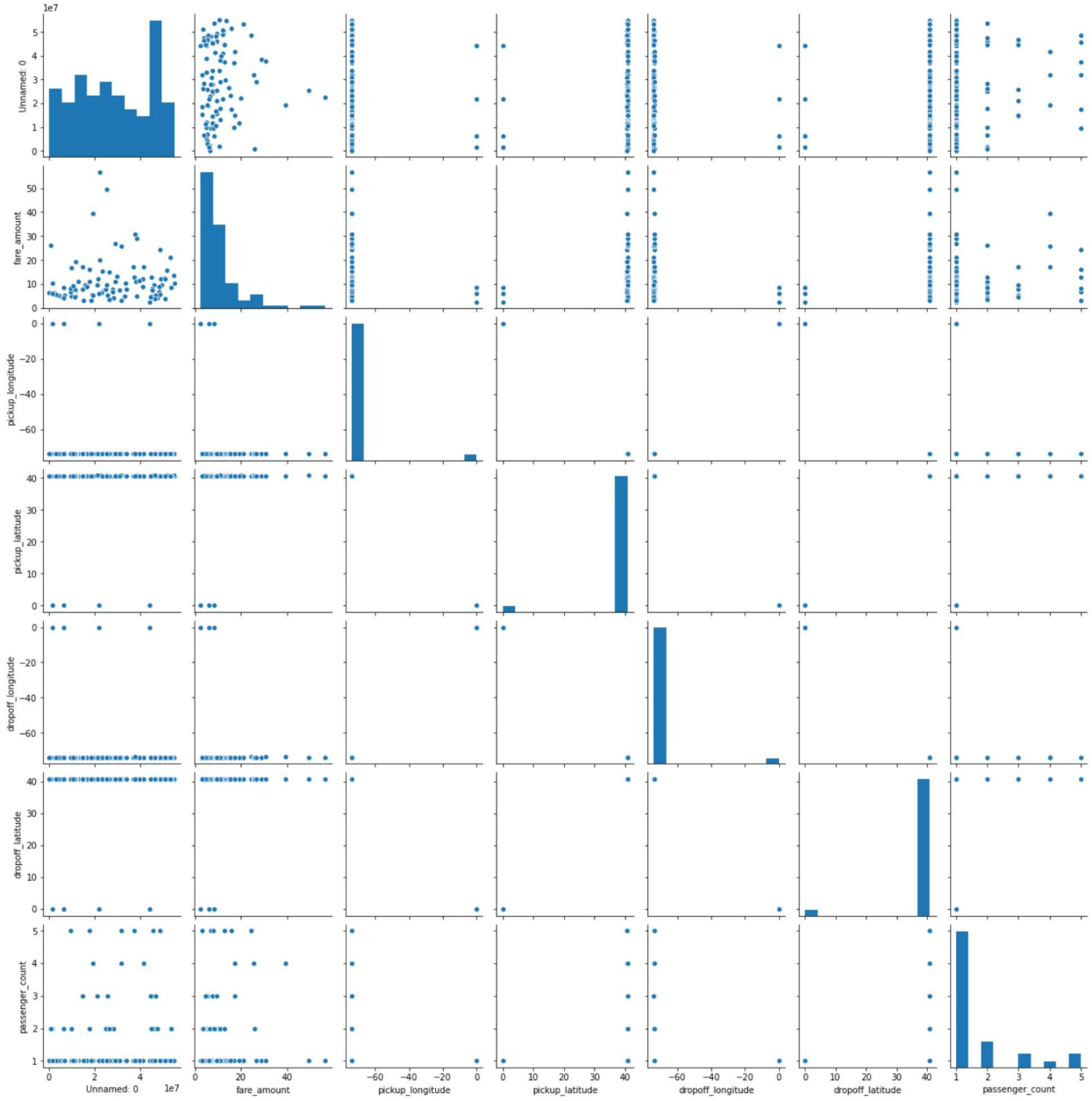
	Unnamed: 0	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	pas
count	1.000000e+02	100.000000	100.000000	100.000000	100.000000	100.000000	
mean	2.810554e+07	11.065700	-71.019759	39.123621	-71.015479	39.126295	
std	1.635033e+07	9.029756	14.569902	8.026358	14.569028	8.026905	
min	2.268700e+05	2.500000	-74.013173	0.000000	-74.016152	0.000000	
25%	1.422691e+07	5.475000	-73.992601	40.733982	-73.989142	40.733759	
50%	2.710896e+07	8.100000	-73.982002	40.752764	-73.979396	40.757083	
75%	4.480811e+07	12.600000	-73.968615	40.765572	-73.960980	40.770287	
max	5.508597e+07	56.800000	0.000000	40.850558	0.000000	40.876687	

In [6]:
```
df.columns
```

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

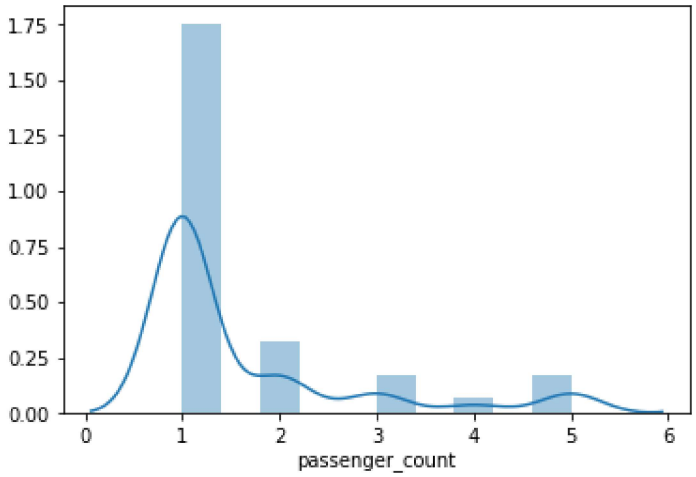
In [7]:
```
sns.pairplot(df)
```

Out[7]: <seaborn.axisgrid.PairGrid at 0x24c7f41db80>



In [8]:
```
sns.distplot(df['passenger_count'])
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x24c0220ca90>
```



In [9]:
```
df1=df[[ 'fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count']]
df1
```

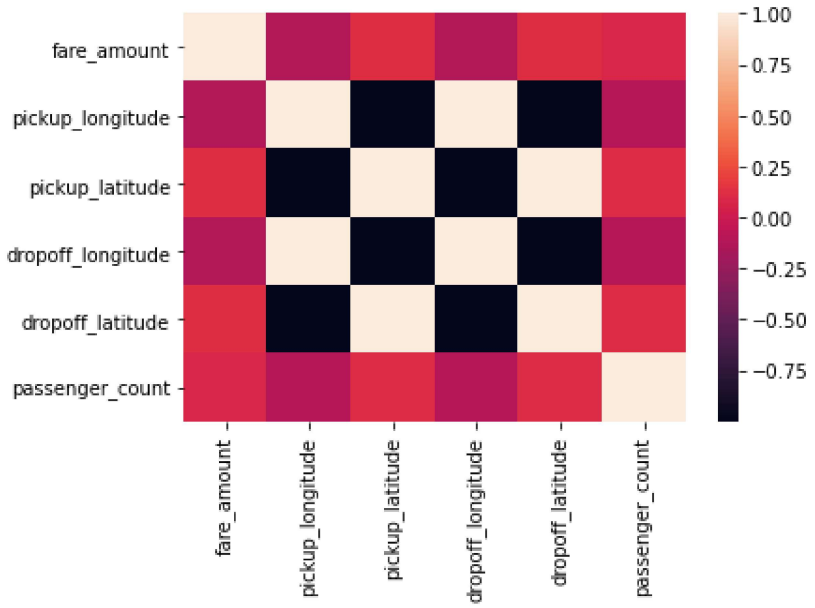
Out[9]:

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	pas
0	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.999512	40.723217	
1	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.994710	40.750325	
2	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73.962565	40.772647	
3	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.965316	40.803349	
4	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.973082	40.761247	
...	...	...	...	...	...	...	...
95	9.5	2015-04-11 08:47:47 UTC	-73.978432	40.752399	-74.000427	40.742119	
96	4.5	2011-10-03 20:29:00 UTC	-73.990055	40.756413	-73.983047	40.756727	
97	3.3	2010-04-26 03:12:44 UTC	-73.982326	40.731314	-73.989649	40.734398	
98	30.9	2011-11-18 09:51:00 UTC	-73.995888	40.759078	-73.865005	40.770452	
99	26.9	2009-08-30 14:03:55 UTC	-73.990137	40.756007	-73.929361	40.774553	

100 rows × 7 columns

In [10]:
```
sns.heatmap(df1.corr())
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x24c02985580>
```



In [14]:
```
x=df1[['fare_amount',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude']]
y=df1['passenger_count']
```

In [15]:
```
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 [16]:
```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

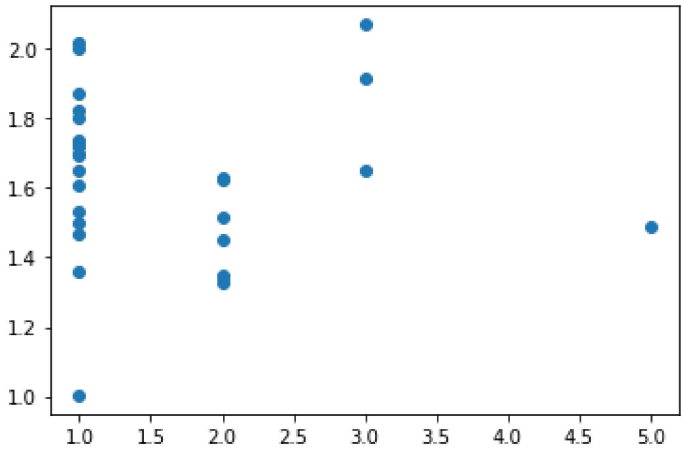
Out[16]:
```
LinearRegression()
```

In [17]:
```
print(lr.intercept_)
```

```
0.9881903663278926
```

In [18]:
```
prediction= lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[18]: <matplotlib.collections.PathCollection at 0x24c02ec8d60>



In [19]:
```
print(lr.score(x_test,y_test))
```

```
-0.09377998485007377
```

In [20]:
```
print(lr.score(x_train,y_train))
```

```
0.06735004195710648
```

In [21]:
```
from sklearn.linear_model import Ridge,Lasso
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[21]: Ridge(alpha=10)

In [22]:
```
rr.score(x_test,y_test)
```

Out[22]: -0.00023882811437436757

In [23]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[23]: Lasso(alpha=10)

In [24]:
```
la.score(x_test,y_test)
```

Out[24]: -0.012979699106975051

In [25]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[25]: ElasticNet()

In [26]:
```
print(en.intercept_)
```

```
1.1345902220816857
```

In [27]:
```
print(en.predict(x_test))
```

```
[1.71704748 1.69902    1.69714058 1.16445041 1.76537334 1.65995223
 1.69269898 1.67165179 1.65841923 1.67149055 1.65973303 1.65416327
 1.67168071 1.67346942 1.65780186 1.67740814 1.659761   1.6976135
 1.65821402 1.65007227 1.70676338 1.67155035 1.68136826 1.66078331
 1.68534113 1.65790287 1.68910469 1.6657416  1.67931758 1.65098498]
```

In [28]:
```
print(en.score(x_test,y_test))
```

```
0.0005892990223963501
```

Evaluation

In [29]:
```
from sklearn import metrics
print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))
print("Mean Squared Error",metrics.mean_squared_error(y_test,prediction))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

```
Mean Absolute Error 0.7771841170079521
Mean Squared Error 0.9248517427454515
Root Mean Squared Error: 0.9616921247184316
```

In [ ]: