In [1]:

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

In [2]:

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

Out[2]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	picku
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	
...	...	...	...	...	...	
199995	42598914	2012-10-28 10:49:00.00000053	3.0	2012-10-28 10:49:00 UTC	-73.987042	
199996	16382965	2014-03-14 01:09:00.0000008	7.5	2014-03-14 01:09:00 UTC	-73.984722	
199997	27804658	2009-06-29 00:42:00.00000078	30.9	2009-06-29 00:42:00 UTC	-73.986017	
199998	20259894	2015-05-20 14:56:25.0000004	14.5	2015-05-20 14:56:25 UTC	-73.997124	
199999	11951496	2010-05-15 04:08:00.00000076	14.1	2010-05-15 04:08:00 UTC	-73.984395	

200000 rows × 9 columns

In [3]:

```
a=a.head(10)
a
```

Out[3]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_lati
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.73
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.72
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.74
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.79
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.74
5	44470845	2011-02-12 02:27:09.0000006	4.9	2011-02-12 02:27:09 UTC	-73.969019	40.75
6	48725865	2014-10-12 07:04:00.0000002	24.5	2014-10-12 07:04:00 UTC	-73.961447	40.69
7	44195482	2012-12-11 13:52:00.00000029	2.5	2012-12-11 13:52:00 UTC	0.000000	0.00
8	15822268	2012-02-17 09:32:00.00000043	9.7	2012-02-17 09:32:00 UTC	-73.975187	40.74
9	50611056	2012-03-29 19:06:00.000000273	12.5	2012-03-29 19:06:00 UTC	-74.001065	40.74

In [4]:

```
# to find
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         10 non-null     int64  
 1   key                10 non-null     object 
 2   fare_amount        10 non-null     float64
 3   pickup_datetime    10 non-null     object 
 4   pickup_longitude   10 non-null     float64
 5   pickup_latitude    10 non-null     float64
 6   dropoff_longitude  10 non-null     float64
 7   dropoff_latitude   10 non-null     float64
 8   passenger_count    10 non-null     int64  
dtypes: float64(5), int64(2), object(2)
memory usage: 848.0+ bytes
```

In [5]:

```
# to display summary of statastic
a.describe()
```

Out[5]:

	Unnamed: 0	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropo
count	1.000000e+01	10.000000	10.000000	10.000000	10.000000	
mean	3.443881e+07	10.350000	-66.580708	36.667971	-66.570116	
std	1.342943e+07	6.460693	23.394088	12.883834	23.390384	
min	1.582227e+07	2.500000	-74.005043	0.000000	-74.002720	
25%	2.465233e+07	5.850000	-73.998451	40.730757	-73.989303	
50%	3.601534e+07	8.700000	-73.975656	40.741278	-73.967167	
75%	4.485598e+07	12.800000	-73.963340	40.745346	-73.962684	
max	5.061106e+07	24.500000	0.000000	40.790844	0.000000	

In [6]:

```
# to display colum heading
a.columns
```

Out[6]:

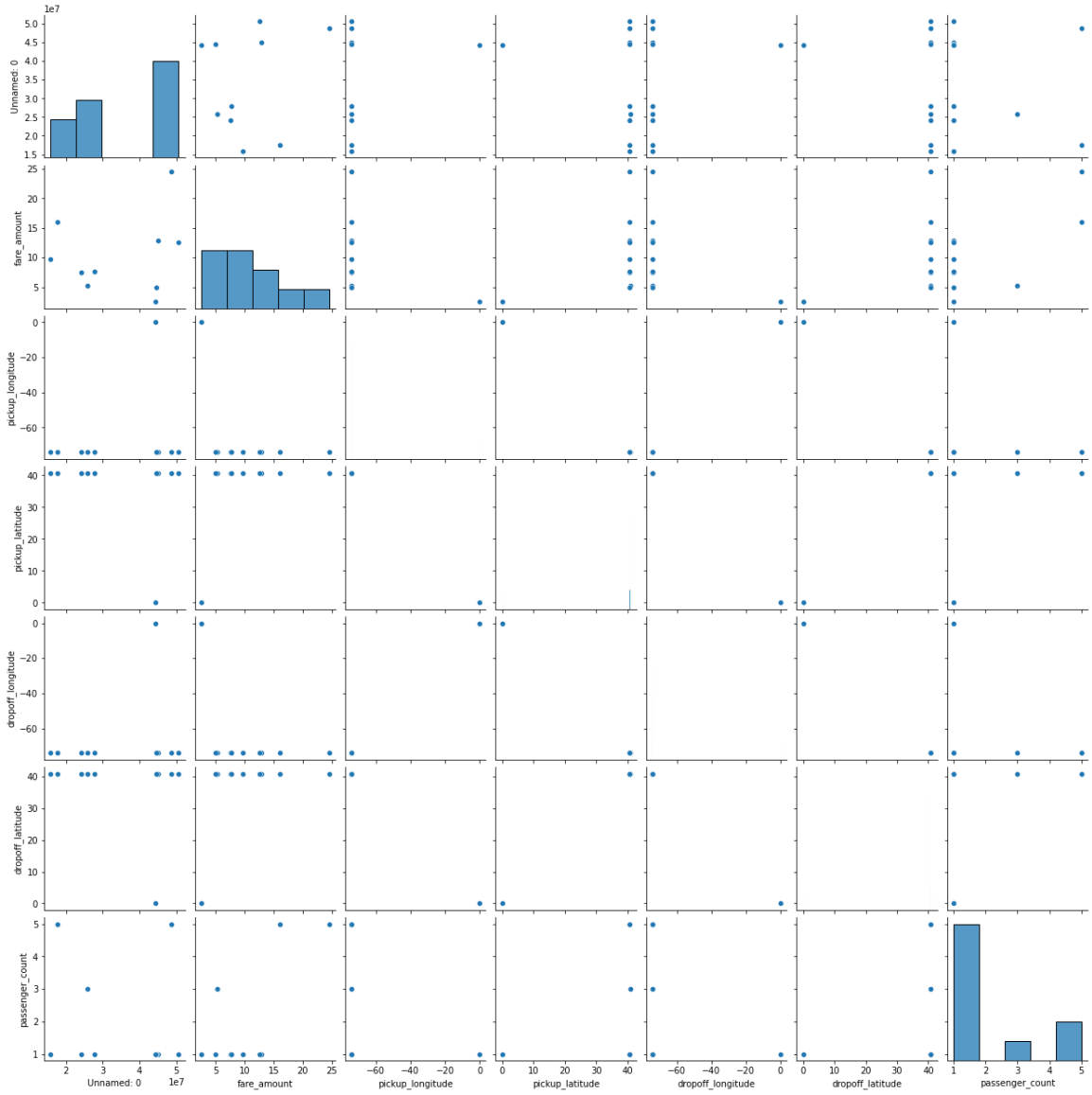
```
Index(['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count'],
      dtype='object')
```

In [7]:

```
sns.pairplot(a)
```

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x243d5c2b4c0>
```

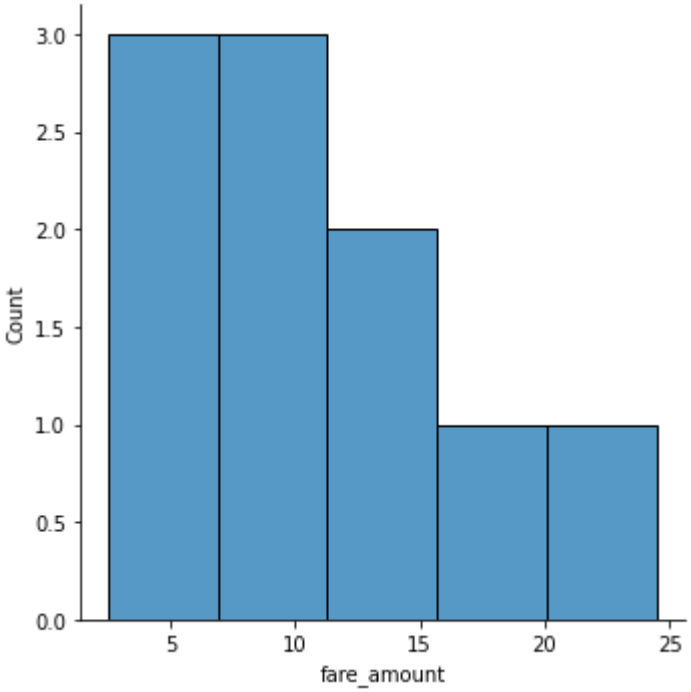


In [8]:

```
sns.displot(a["fare_amount"])
```

Out[8]:

```
<seaborn.axisgrid.FacetGrid at 0x243e4469490>
```



In [9]:

```
b=a[['fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude']]
b
```

Out[9]:

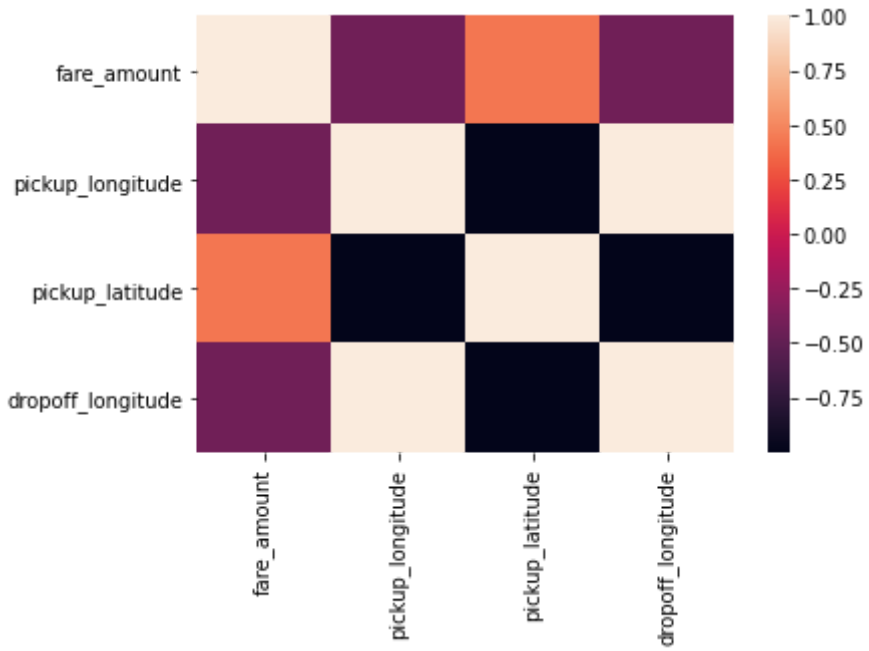
	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude
0	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.999512
1	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.994710
2	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73.962565
3	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.965316
4	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.973082
5	4.9	2011-02-12 02:27:09 UTC	-73.969019	40.755910	-73.969019
6	24.5	2014-10-12 07:04:00 UTC	-73.961447	40.693965	-73.871195
7	2.5	2012-12-11 13:52:00 UTC	0.000000	0.000000	0.000000
8	9.7	2012-02-17 09:32:00 UTC	-73.975187	40.745767	-74.002720
9	12.5	2012-03-29 19:06:00 UTC	-74.001065	40.741787	-73.963040

In [10]:

```
sns.heatmap(b.corr())
```

Out[10]:

<AxesSubplot:>



In [16]:

```
x=a[['fare_amount',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude']]
y=a['pickup_latitude']
```

In [17]:

```
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 [18]:

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[18]:

```
LinearRegression()
```

In [19]:

```
lr.intercept_
```

Out[19]:

```
-2.1316282072803006e-14
```

In [20]:

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

Out[20]:

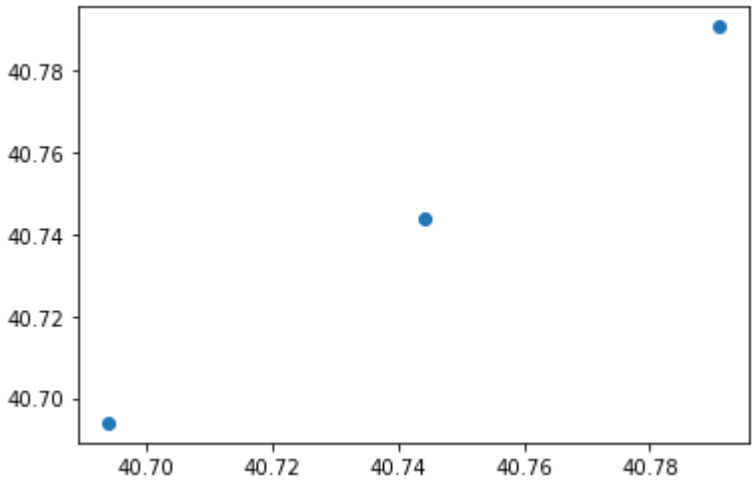
	Co-efficient
fare_amount	3.331211e-16
pickup_longitude	1.439144e-16
pickup_latitude	1.000000e+00
dropoff_longitude	-1.889804e-16

In [21]:

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

Out[21]:

```
<matplotlib.collections.PathCollection at 0x243e5579a90>
```



In [22]:

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

Out[22]:

```
1.0
```

In [23]:

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

Out[23]:

```
1.0
```

In [24]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [25]:

```
rr=Ridge(alpha=10)
rr.fit(x_test,y_test)
```

Out[25]:

```
Ridge(alpha=10)
```

In [26]:

```
rr.score(x_test,y_test)
```

Out[26]:

```
0.9900090107281445
```

In [27]:

```
la=Lasso(alpha=10)
la.fit(x_test,y_test)
```

Out[27]:

```
Lasso(alpha=10)
```

In [28]:

```
la.score(x_test,y_test)
```

Out[28]:

```
0.0
```

In [29]:

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

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_coordinat
e_descent.py:530: ConvergenceWarning: Objective did not converge. You migh
t want to increase the number of iterations. Duality gap: 0.14583617914194
28, tolerance: 0.14227670749997787
  model = cd_fast.enet_coordinate_descent(
```

Out[29]:

```
ElasticNet()
```

In [30]:

```
en.coef_
```

Out[30]:

```
array([ 0.        , -0.33678428,  0.        , -0.2129699 ])
```

In [31]:

```
en.intercept_
```

Out[31]:

```
0.05739223479763922
```

In [32]:

```
prediction=en.predict(x_test)
prediction
```

Out[32]:

```
array([40.72377396, 40.70821787, 40.69878604])
```

In [33]:

```
en.score(x_test,y_test)
```

Out[33]:

```
-0.23716998533755218
```

EVALUATION METRICS

In [34]:

```
from sklearn import metrics
```

In [35]:

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

```
Mean Absolute Error: 0.03591940241993304
```

In [36]:

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

```
Mean Squared Error 0.0019360279444952542
```

In [37]:

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

```
Root Mean Squared Error 0.04400031754993655
```

In [ ]:

```

```