In [1]:

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

In [2]:

```
df1=pd.read_csv(r'C:\Users\user\Downloads\7_uber.csv')
df1
```

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]:

```
df=df1.head(100)
df
```

Out[3]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_la
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.7
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.7
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.7
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.7
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.7
...	...	...	...	...	...	
95	25431833	2015-04-11 08:47:47.0000001	9.5	2015-04-11 08:47:47 UTC	-73.978432	40.7
96	44792012	2011-10-03 20:29:00.000000179	4.5	2011-10-03 20:29:00 UTC	-73.990055	40.7
97	18571020	2010-04-26 03:12:44.0000001	3.3	2010-04-26 03:12:44 UTC	-73.982326	40.7
98	37942404	2011-11-18 09:51:00.000000166	30.9	2011-11-18 09:51:00 UTC	-73.995888	40.7
99	29024472	2009-08-30 14:03:55.0000002	26.9	2009-08-30 14:03:55 UTC	-73.990137	40.7

100 rows × 9 columns

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	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]:

```
df.columns
```

Out[6]:

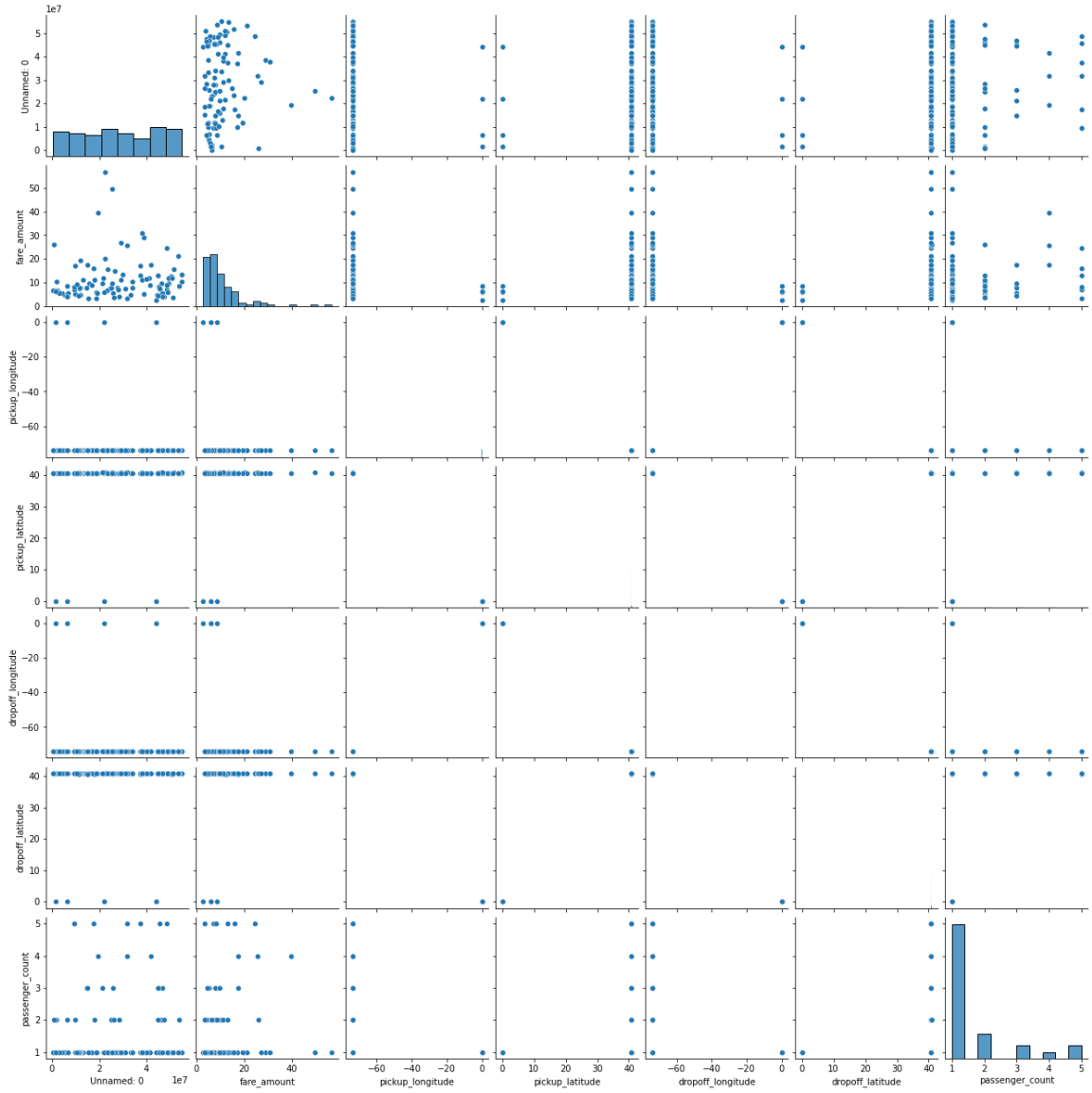
```
Index(['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count'],
      dtype='object')
```

In [7]:

```
sns.pairplot(df)
```

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x1fa309f8c10>
```



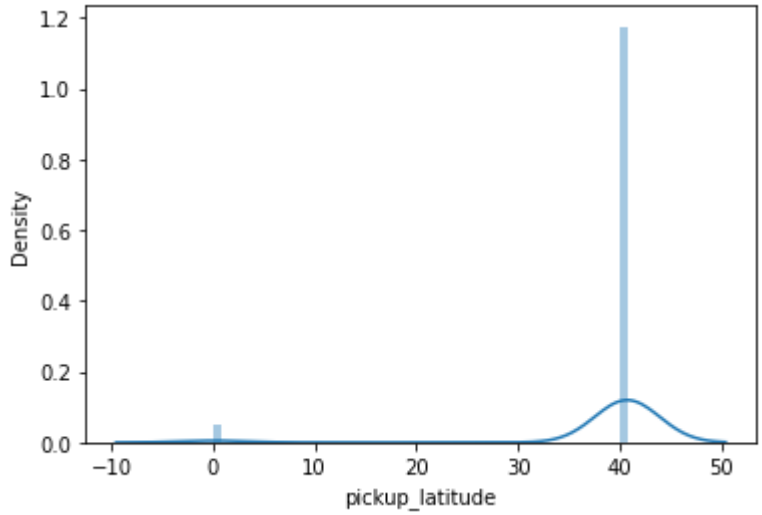
In [8]:

```
sns.distplot(df['pickup_latitude'])
```

```
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='pickup_latitude', ylabel='Density'>
```

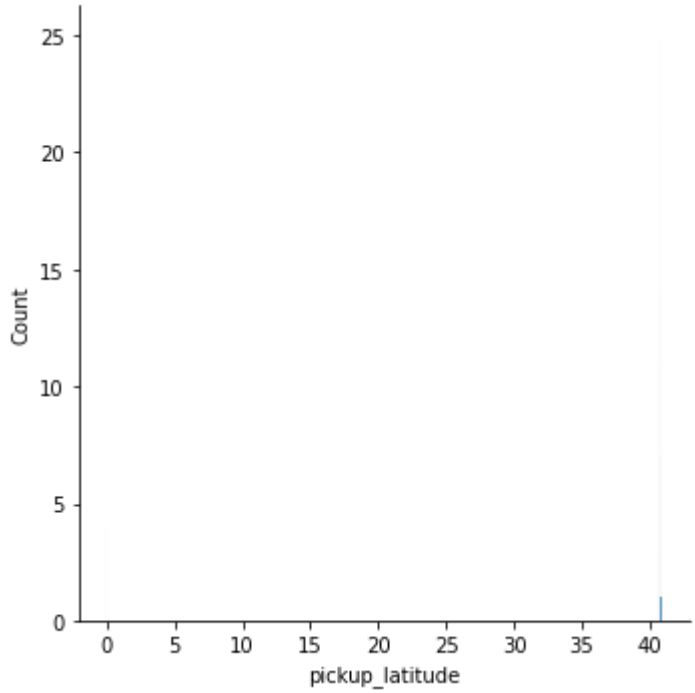


In [9]:

```
sns.displot(df["pickup_latitude"])
```

Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x1fa41f09d90>
```



In [10]:

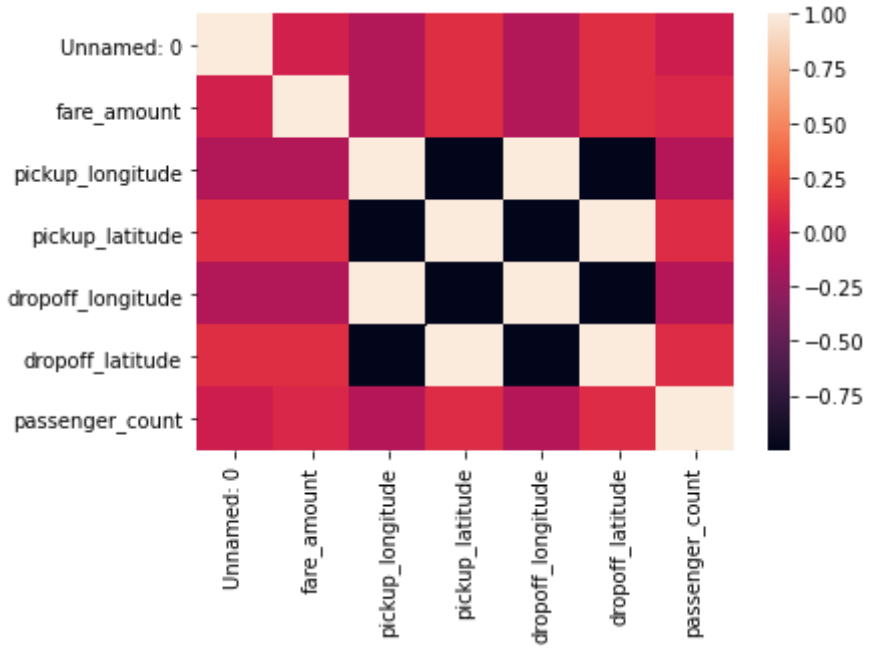
```
df1=df[['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count']]
```

In [11]:

```
sns.heatmap(df1.corr())
```

Out[11]:

<AxesSubplot:>



In [12]:

```
df2=df.dropna(axis=1)
df2
```

Out[12]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_la
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.7
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.7
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.7
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.7
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.7
...	...	...	...	...	...	
95	25431833	2015-04-11 08:47:47.0000001	9.5	2015-04-11 08:47:47 UTC	-73.978432	40.7
96	44792012	2011-10-03 20:29:00.000000179	4.5	2011-10-03 20:29:00 UTC	-73.990055	40.7
97	18571020	2010-04-26 03:12:44.0000001	3.3	2010-04-26 03:12:44 UTC	-73.982326	40.7
98	37942404	2011-11-18 09:51:00.000000166	30.9	2011-11-18 09:51:00 UTC	-73.995888	40.7
99	29024472	2009-08-30 14:03:55.0000002	26.9	2009-08-30 14:03:55 UTC	-73.990137	40.7

100 rows × 9 columns

In [13]:

```
x=df1[['Unnamed: 0', 'fare_amount',
       'pickup_longitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count']]
y=df1[['pickup_latitude']]
```

In [14]:

```
from sklearn.model_selection import train_test_split
```

In [15]:

```
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)#ValueError: Input contains NaN, infinity or a value too large for
```

Out[16]:

```
LinearRegression()
```

In [17]:

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

```
[-0.00775665]
```

In [18]:

```
coef= pd.DataFrame(lr.coef_)
coef
```

Out[18]:

	0	1	2	3	4	5
0	1.312174e-10	0.000541	0.619222	-0.750594	0.76145	0.002467

In [19]:

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

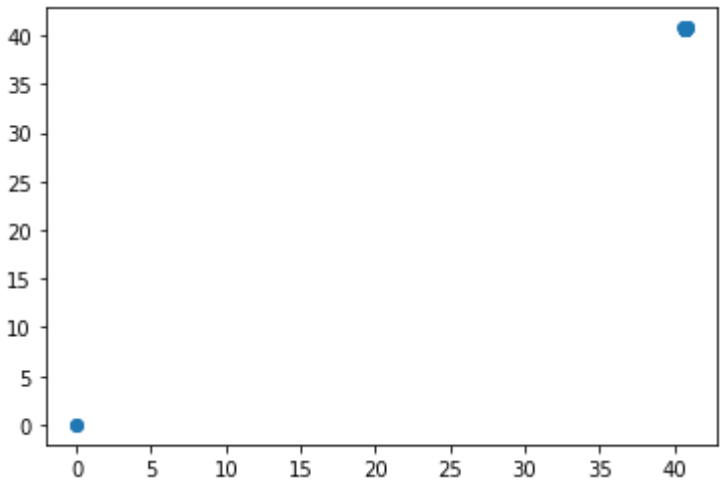
```
0.9999882567934155
```

In [20]:

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

Out[20]:

```
<matplotlib.collections.PathCollection at 0x1fa496193d0>
```



In [21]:

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

Out[21]:

```
0.9999882567934155
```

In [22]:

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

Out[22]:

```
0.9999904201257689
```

In [23]:

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

In [24]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[24]:

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

In [25]:

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

Out[25]:

```
0.9999954815590651
```

In [26]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[26]:

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

In [27]:

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

Out[27]:

```
0.9855277857881688
```