A real estate agent want help to predict the house price for regions in Usa.he gave us the dataset to work on to use linear Regression model.Create a model that helps him to estimate

Data Collection

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

In [2]:
```
#import the dataset
data=pd.read_csv(r"C:\Users\user\Desktop\Vicky\7_uber.csv")[0:500]
```

In [3]:
```
#to display top 10 rows
data.head()
```

Out[3]:

	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dro
0	2015-05-07 06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.999512	
1	2009-07-17 56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.994710	
2	2009-08-24).00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73.962565	
3	2009-06-26 21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.965316	
4	2014-08-28 000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.973082	

In [4]:
```
#to display null values
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         500 non-null    int64  
 1   key                500 non-null    object 
 2   fare_amount        500 non-null    float64
 3   pickup_datetime    500 non-null    object 
 4   pickup_longitude   500 non-null    float64
 5   pickup_latitude    500 non-null    float64
 6   dropoff_longitude  500 non-null    float64
 7   dropoff_latitude   500 non-null    float64
 8   passenger_count    500 non-null    int64  
dtypes: float64(5), int64(2), object(2)
memory usage: 35.3+ KB
```

In [5]:
```
data.shape
```

Out[5]:
```
(500, 9)
```

In [6]:
```
#to display summary of statistics
data.describe()
```

Out[6]:

	Unnamed: 0	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_l
count	5.000000e+02	500.000000	500.000000	500.000000	500.000000	500.
mean	2.737940e+07	10.708720	-72.053865	39.692497	-72.201155	39.
std	1.607155e+07	8.334145	11.784239	6.491541	11.333432	6.
min	1.862090e+05	2.500000	-74.030417	0.000000	-74.027813	0.
25%	1.250293e+07	6.000000	-73.992804	40.735994	-73.991571	40.
50%	2.749836e+07	8.100000	-73.982352	40.752445	-73.980784	40.
75%	4.157492e+07	12.500000	-73.968724	40.765865	-73.965878	40.
max	5.519870e+07	57.330000	0.001782	40.850558	0.000875	40.

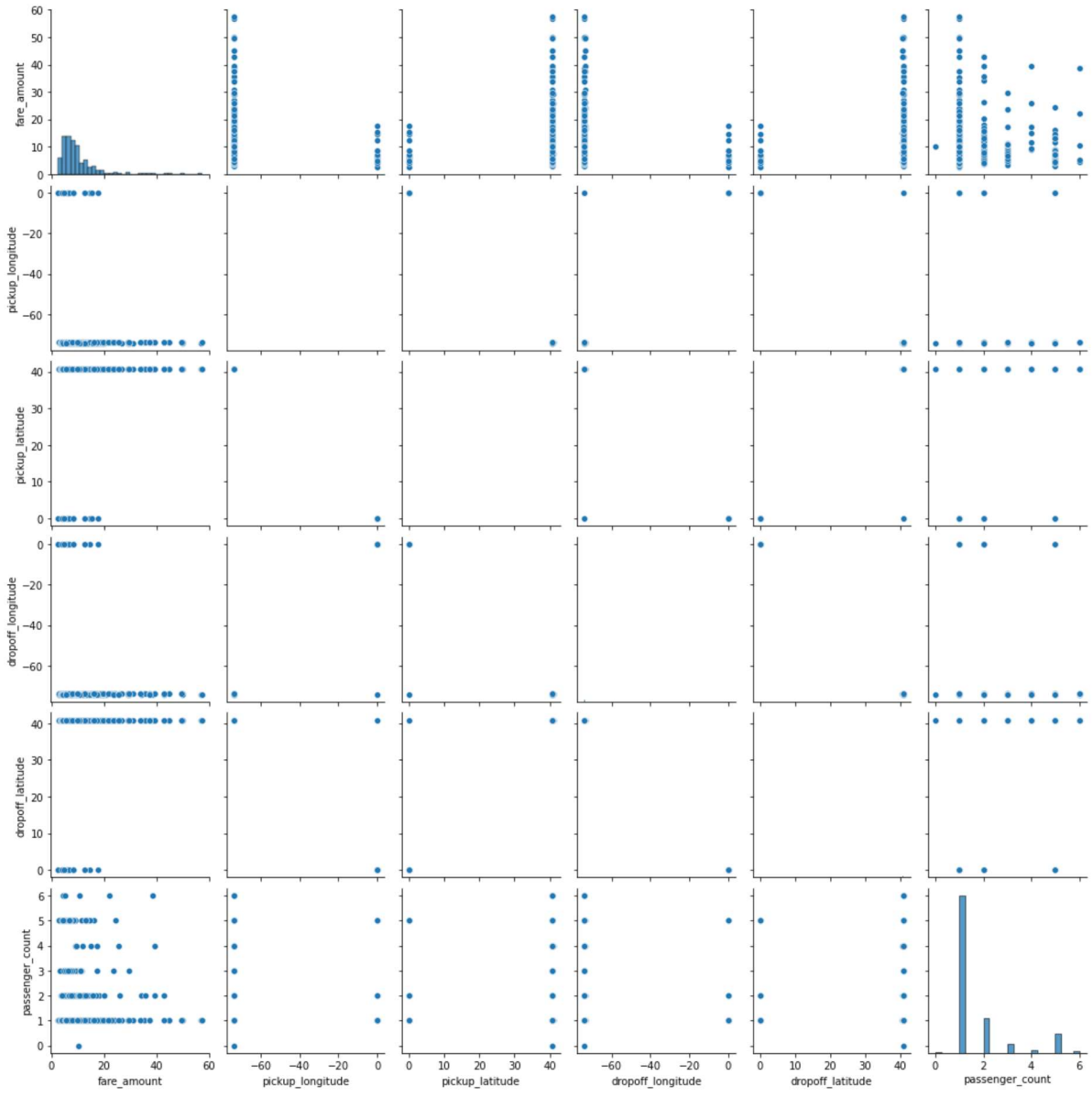
In [7]:
```
#to display columns name
data.columns
```

Out[7]:
```
Index(['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count'],
      dtype='object')
```

In [8]:
```
data1=data[['fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count']]
```

In [9]:
```
sns.pairplot(data1)
```

Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x238e814af10>
```

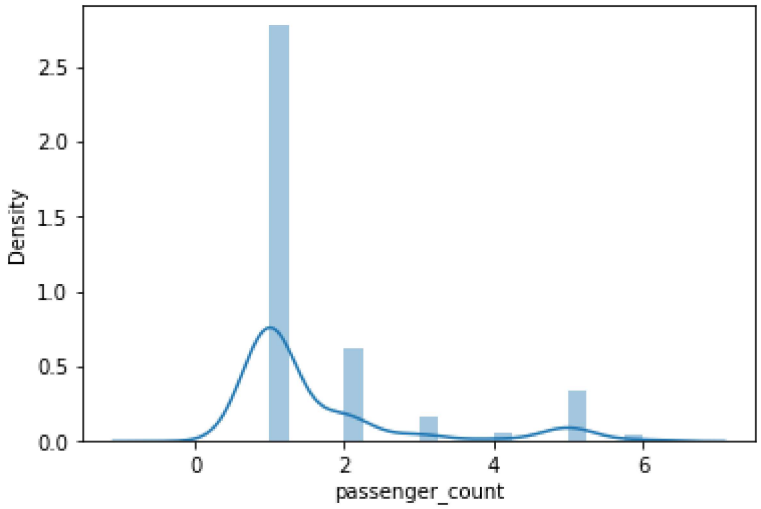


EDA and Visualization

In [10]:
```
sns.distplot(data['passenger_count'])
```

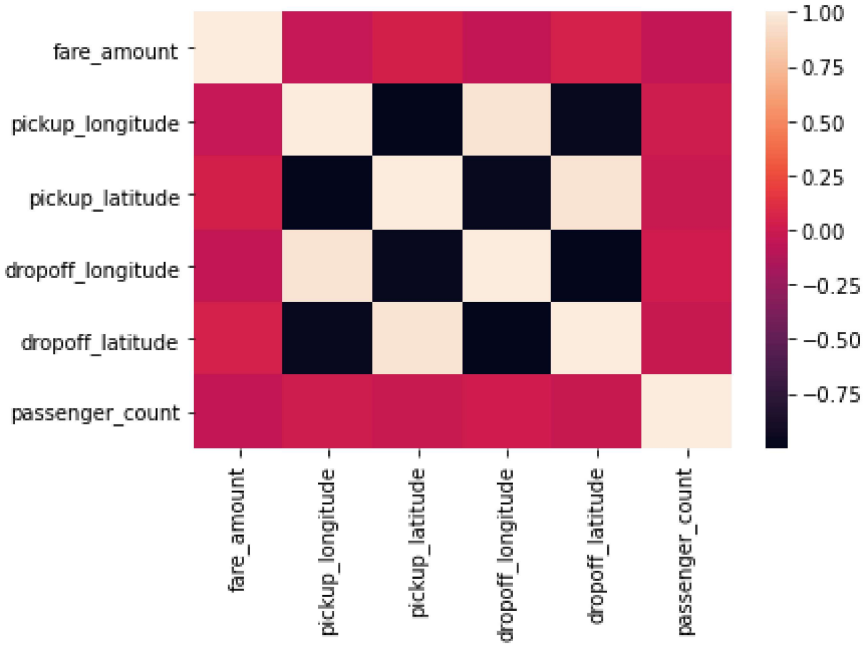
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

Out[10]: <AxesSubplot:xlabel='passenger_count', ylabel='Density'>



In [11]:
```
sns.heatmap(data1.corr())
```

Out[11]: <AxesSubplot:>



To train the model

we are going to train the linear regression model ;We need to split the two variable x and y where x in independent variable (input) and y is dependent of x(output) so we could ignore

In [13]:
```
x=data1[[ "passenger_count","fare_amount"]]
y=data1["passenger_count"]
```

In [14]:
```
#To split test and train data
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.6)
```

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

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

In [16]:
```
lr.intercept_
```

Out[16]:
```
4.440892098500626e-16
```

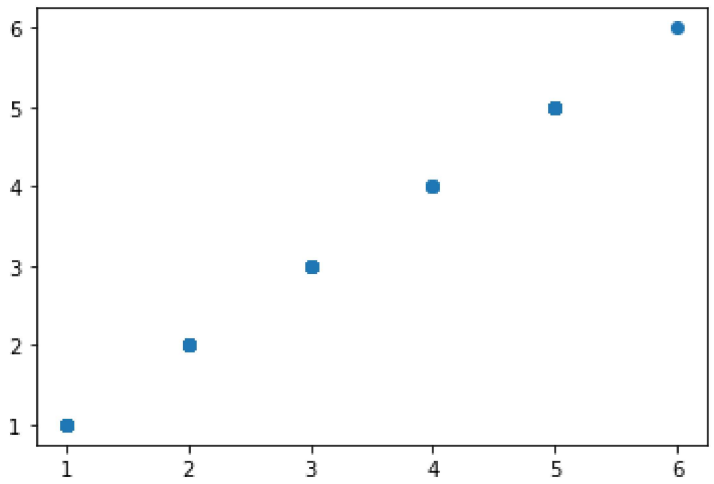
In [17]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=["Co-efficient"])
coeff
```

Out[17]:

	Co-efficient
passenger_count	1.000000e+00
fare_amount	-4.032178e-18

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

Out[18]:
```
<matplotlib.collections.PathCollection at 0x23885be19a0>
```



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

Out[23]:
```
1.0
```

In [24]:
```
lr.score(x_train,y_train)
```

Out[24]:
```
1.0
```

In [25]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [26]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[26]:
```
0.9989057314266017
```

In [27]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
la.score(x_test,y_test)
```

Out[27]:
```
-0.00787561784147539
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: