Problem Statement

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 of what the house would sell for

Import libraries

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
# To import dataset
df=pd.read_csv('uber csv')
df
```

Out[3]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_l
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.
...	...	...	...	...	...	...
199995	42598914	2012-10-28 10:49:00.00000053	3.0	2012-10-28 10:49:00 UTC	-73.987042	40.
199996	16382965	2014-03-14 01:09:00.0000008	7.5	2014-03-14 01:09:00 UTC	-73.984722	40.
199997	27804658	2009-06-29 00:42:00.00000078	30.9	2009-06-29 00:42:00 UTC	-73.986017	40.
199998	20259894	2015-05-20 14:56:25.0000004	14.5	2015-05-20 14:56:25 UTC	-73.997124	40.
199999	11951496	2010-05-15 04:08:00.00000076	14.1	2010-05-15 04:08:00 UTC	-73.984395	40.

200000 rows × 9 columns

In [4]:
```
# To display top 10 rows
df.head(10)
```

Out[4]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitud
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.73835
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.72822
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.74077
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.79084
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.74408
5	44470845	2011-02-12 02:27:09.0000006	4.9	2011-02-12 02:27:09 UTC	-73.969019	40.75591
6	48725865	2014-10-12 07:04:00.0000002	24.5	2014-10-12 07:04:00 UTC	-73.961447	40.69396
7	44195482	2012-12-11 13:52:00.00000029	2.5	2012-12-11 13:52:00 UTC	0.000000	0.00000
8	15822268	2012-02-17 09:32:00.00000043	9.7	2012-02-17 09:32:00 UTC	-73.975187	40.74576
9	50611056	2012-03-29 19:06:00.000000273	12.5	2012-03-29 19:06:00 UTC	-74.001065	40.74178

Data Cleaning and Pre-Processing

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 9 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   Unnamed: 0         200000 non-null  int64  
 1   key                200000 non-null  object 
 2   fare_amount        200000 non-null  float64
 3   pickup_datetime    200000 non-null  object 
 4   pickup_longitude   200000 non-null  float64
 5   pickup_latitude    200000 non-null  float64
 6   dropoff_longitude  199999 non-null  float64
 7   dropoff_latitude   199999 non-null  float64
 8   passenger_count    200000 non-null  int64  
dtypes: float64(5), int64(2), object(2)
memory usage: 13.7+ MB
```

In [6]:
```
df.describe()
```

Out[6]:

	Unnamed: 0	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff
count	2.000000e+05	200000.000000	200000.000000	200000.000000	199999.000000	19999
mean	2.771250e+07	11.359955	-72.527638	39.935885	-72.525292	3
std	1.601382e+07	9.901776	11.437787	7.720539	13.117408	
min	1.000000e+00	-52.000000	-1340.648410	-74.015515	-3356.666300	-88
25%	1.382535e+07	6.000000	-73.992065	40.734796	-73.991407	4
50%	2.774550e+07	8.500000	-73.981823	40.752592	-73.980093	4
75%	4.155530e+07	12.500000	-73.967154	40.767158	-73.963658	4
max	5.542357e+07	499.000000	57.418457	1644.421482	1153.572603	87

In [7]:
```
df.columns
```

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

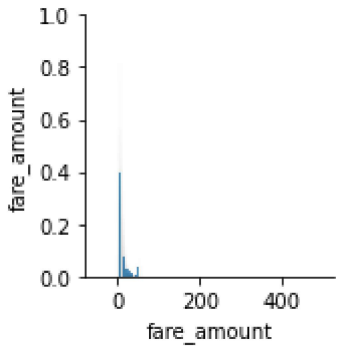
In [8]:
```
a = df.dropna(axis='columns')
a.columns
```

Out[8]:
```
Index(['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'passenger_count'],
      dtype='object')
```

EDA and Visualization

In [9]:
```
sns.pairplot(a[['key', 'fare_amount', 'pickup_datetime']])
```

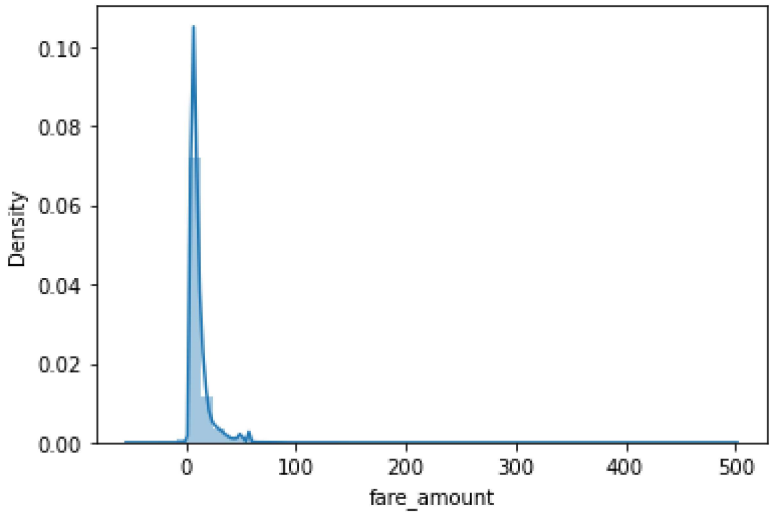
Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x14025214ac0>
```



In [10]:
```
sns.distplot(a['fare_amount'])
```

```
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)
```

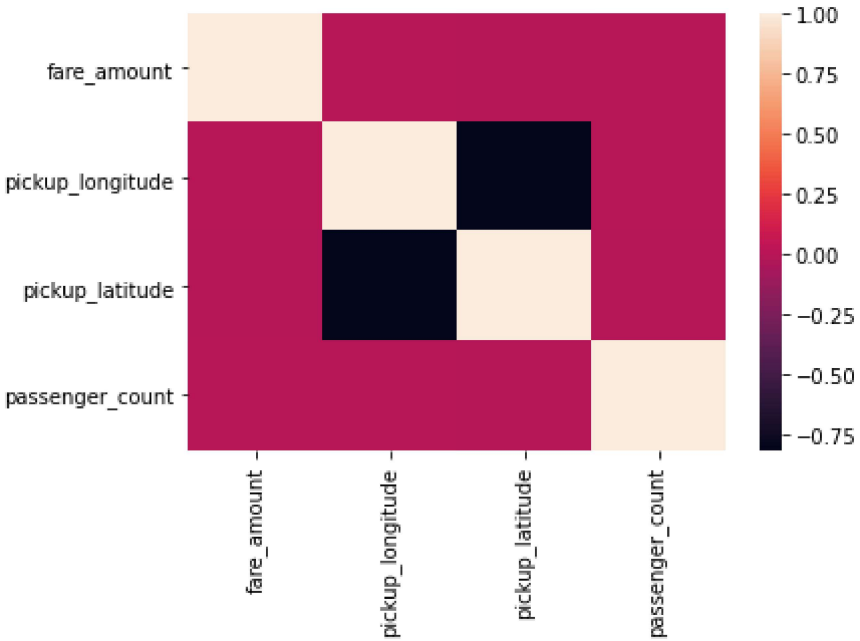
Out[10]: <AxesSubplot:xlabel='fare_amount', ylabel='Density'>



In [11]:
```
a1=a[[ 'key', 'fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'passenger_count']]
```

In [12]:
```
sns.heatmap(a1.corr())
```

Out[12]: <AxesSubplot:>



To Train the Model - Model Building

We are going to train Linear Regression model;We need to split out data into two variables x and y where x is independent variable (input) and y is dependent on x(output). We could ignore address column as it is not required for our model.

In [13]:
```
x=a1[['pickup_longitude', 'pickup_latitude', 'passenger_count']]
y=a1['fare_amount']
```

To split my dataset into training and test data

In [14]:
```
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 [15]:
```
from sklearn.linear_model import LinearRegression

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

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

In [16]:
```
print(lr.intercept_)
```

```
12.042520132095321
```

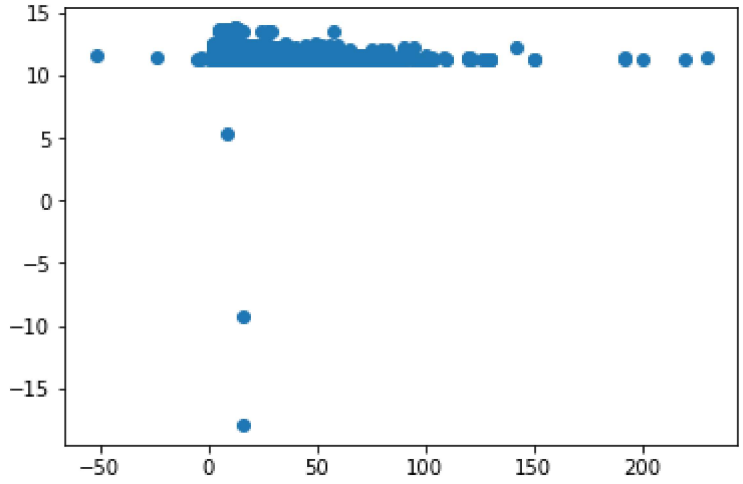
In [17]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[17]:

	Co-efficient
pickup_longitude	0.001967
pickup_latitude	-0.016641
passenger_count	0.071399

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

Out[18]: <matplotlib.collections.PathCollection at 0x1402cc4a0d0>



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

```
-0.00012519981048186857
```

In [20]:
```
from sklearn.linear_model import ElasticNet
en = ElasticNet()
en.fit(x_train,y_train)
```

Out[20]: ElasticNet()

In [21]:
```
print(en.coef_)
```

```
[ 0.00627848 -0.          0.        ]
```

In [22]:
```
print(en.intercept_)
```

```
11.811550041405129
```

In [23]:
```

print(en.predict(x_test))
```

```
[11.34717696 11.34740924 11.34699547 ... 11.34831669 11.34709035
 11.34699909]
```

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

```
5.66200607901024e-05
```

In [25]:
```
from sklearn import metrics
```

In [26]:
```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolytre Error: 6.037938945527556
```

In [27]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 96.77118612067
```

In [28]:
```
print("Root Mean Squared Error:",metrics.mean_squared_error(y_test,prediction)
```

```
Root Mean Squared Error: 96.77118612067
```