Problem Statement

Linear Regression

Import Libraries

In [1]:

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

In [2]:

```
a=pd.read_csv("uber.csv")
a
```

Out[2]:

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

200000 rows × 9 columns

To display top 10 rows

In [3]:

```
c=a.head(15)
c
```

Out[3]:

	id	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_lo
0	24238194	2015- 05-07 19:52:06	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73
1	27835199	2009- 07-17 20:04:56	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73
2	44984355	2009- 08-24 21:45:00	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73
3	25894730	2009- 06-26 08:22:21	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73
4	17610152	2014- 08-28 17:47:00	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73
5	44470845	2011- 02-12 02:27:09	4.9	2011-02-12 02:27:09 UTC	-73.969019	40.755910	-73
6	48725865	2014- 10-12 07:04:00	24.5	2014-10-12 07:04:00 UTC	-73.961447	40.693965	-73
7	44195482	2012- 12-11 13:52:00	2.5	2012-12-11 13:52:00 UTC	0.000000	0.000000	0
8	15822268	2012- 02-17 09:32:00	9.7	2012-02-17 09:32:00 UTC	-73.975187	40.745767	-74
9	50611056	2012- 03-29 19:06:00	12.5	2012-03-29 19:06:00 UTC	-74.001065	40.741787	-73
10	2205147	2015- 05-22 17:32:27	6.5	2015-05-22 17:32:27 UTC	-73.974388	40.746952	-73
11	6379048	2011- 05-23 22:15:00	8.5	2011-05-23 22:15:00 UTC	0.000000	0.000000	0
12	31892535	2011- 05-17 14:03:00	3.3	2011-05-17 14:03:00 UTC	-73.966378	40.804440	-73
13	13012786	2011- 06-25	10.9	2011-06-25 11:19:00 UTC	-73.953352	40.767382	-73

	id	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_lor
		11:19:00					
14	48411337	2010-04-06 22:20:27	6.9	2010-04-06 22:20:27 UTC	-73.973370	40.755193	-73

To find Missing values

In [4]:

```
c.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15 entries, 0 to 14
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 15 non-null     int64  
 1   key                15 non-null     object 
 2   fare_amount        15 non-null     float64
 3   pickup_datetime    15 non-null     object 
 4   pickup_longitude   15 non-null     float64
 5   pickup_latitude    15 non-null     float64
 6   dropoff_longitude  15 non-null     float64
 7   dropoff_latitude   15 non-null     float64
 8   passenger_count    15 non-null     int64  
dtypes: float64(5), int64(2), object(2)
memory usage: 1.2+ KB
```

To display summary of statistics

In [5]:

```
a.describe()
```

Out[5]:

	id	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_la
count	2.000000e+05	200000.000000	200000.000000	200000.000000	199999.000000	199999.0
mean	2.771250e+07	11.359955	-72.527638	39.935885	-72.525292	39.9
std	1.601382e+07	9.901776	11.437787	7.720539	13.117408	6.7
min	1.000000e+00	-52.000000	-1340.648410	-74.015515	-3356.666300	-881.9
25%	1.382535e+07	6.000000	-73.992065	40.734796	-73.991407	40.7
50%	2.774550e+07	8.500000	-73.981823	40.752592	-73.980093	40.7
75%	4.155530e+07	12.500000	-73.967153	40.767158	-73.963659	40.7
max	5.542357e+07	499.000000	57.418457	1644.421482	1153.572603	872.6

To display column heading

In [6]:

```
a.columns
```

Out[6]:

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

Pairplot

In [7]:

```
s=a.dropna(axis=1)
s
```

Out[7]:

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

200000 rows × 7 columns

In [8]:

```
s.columns
```

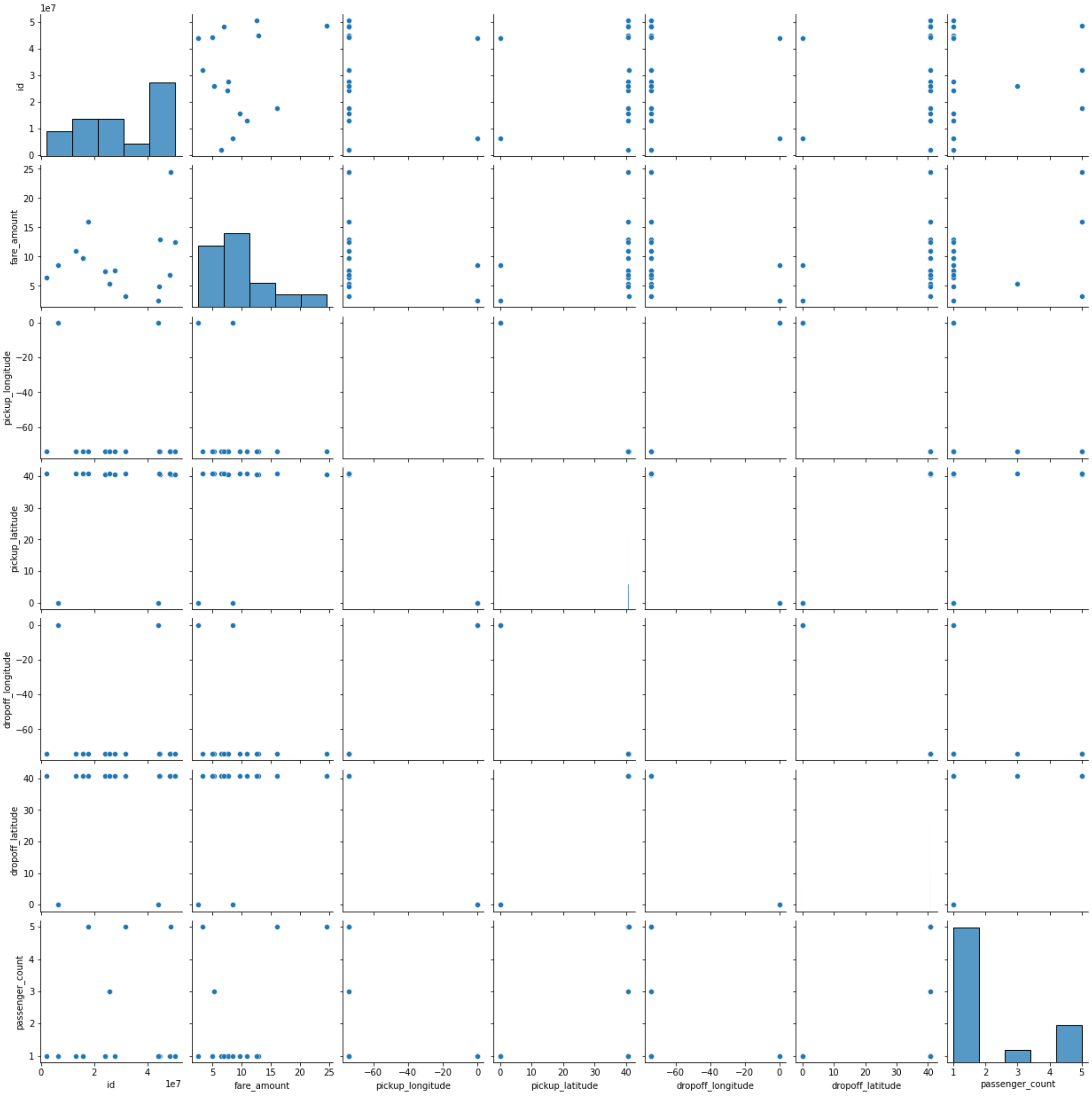
Out[8]:

```
Index(['id', 'key', 'fare_amount', 'pickup_datetime', 'pickup_longitude',
       'pickup_latitude', 'passenger_count'],
      dtype='object')
```

In [9]:

```
sns.pairplot(c)
```

Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x16ef8f5bb20>
```

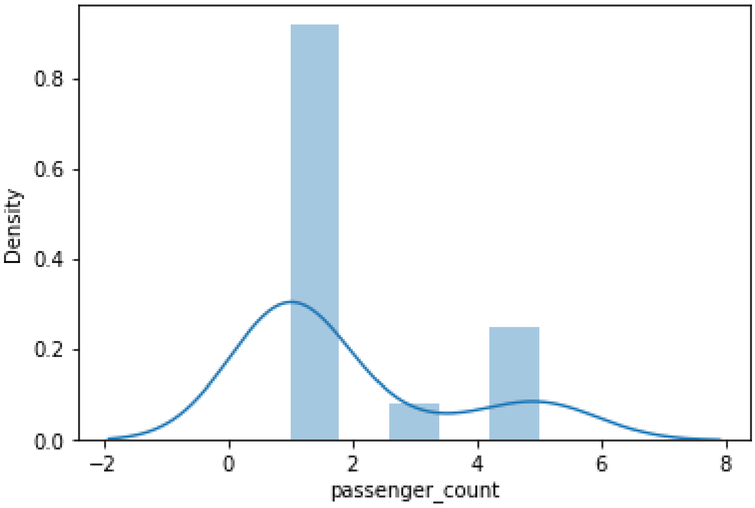


Distribution Plot

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

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

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



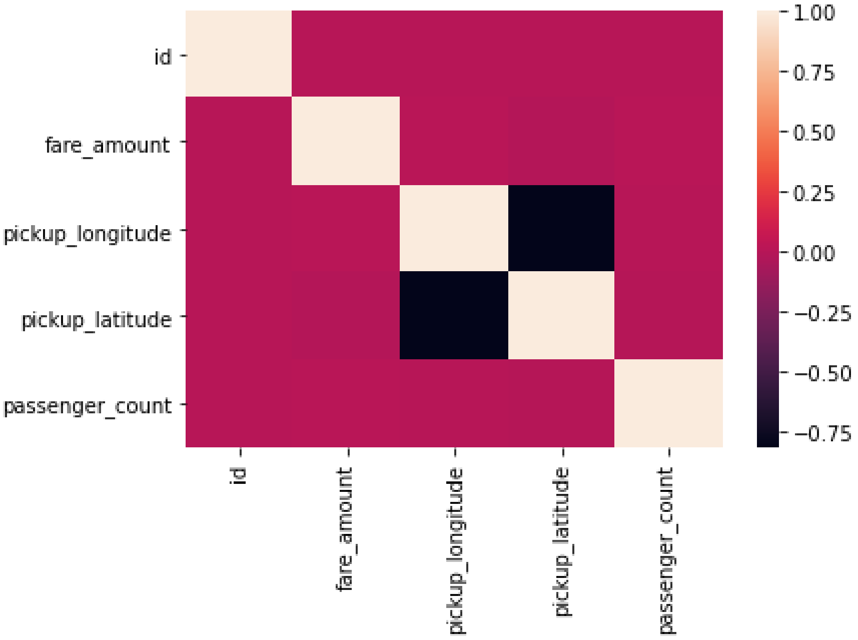
Correlation

In [11]:

```
b=a[['id', 'key', 'fare_amount', 'pickup_datetime', 'pickup_longitude',
       'pickup_latitude', 'passenger_count']]
sns.heatmap(b.corr())
```

Out[11]:

```
<AxesSubplot:>
```



Train the model - Model Building

In [12]:

```
g=c[['id']]
h=c['passenger_count']
```

To split dataset into training end test

In [13]:

```
from sklearn.model_selection import train_test_split
g_train,g_test,h_train,h_test=train_test_split(g,h,test_size=0.6)
```

To run the model

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

In [15]:
```
lr=LinearRegression()
lr.fit(g_train,h_train)
```

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

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

```
-0.2548871183257817
```

Coeffecient

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

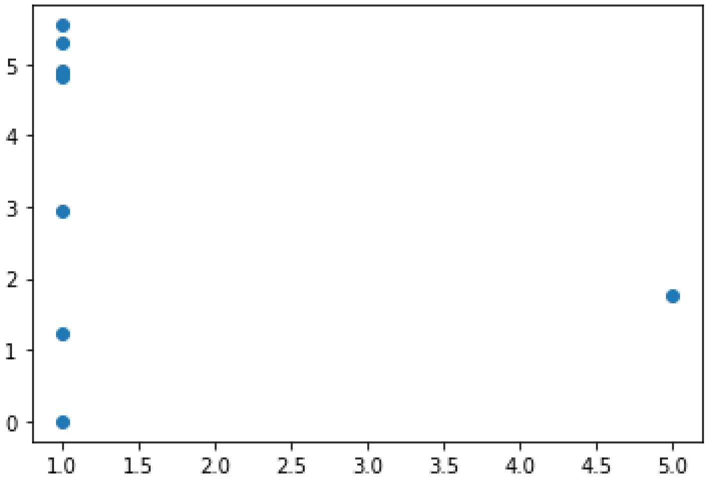
Out[17]:

	Co-effecient
id	1.146062e-07

Best Fit line

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

Out[18]:
```
<matplotlib.collections.PathCollection at 0x16e8bde6f40>
```



To find score

In [19]:
```
print(lr.score(g_test,h_test))
```

```
-5.951547640410228
```

Import Lasso and ridge

```
In [20]: from sklearn.linear_model import Ridge,Lasso
```

Ridge

```
In [21]: ri=Ridge(alpha=5)
         ri.fit(g_train,h_train)

Out[21]: Ridge(alpha=5)
```

```
In [22]: ri.score(g_test,h_test)

Out[22]: -5.951547640410194
```

```
In [23]: ri.score(g_train,h_train)

Out[23]: 0.707448300378085
```

Lasso

```
In [24]: l=Lasso(alpha=6)
         l.fit(g_train,h_train)

Out[24]: Lasso(alpha=6)
```

```
In [25]: l.score(g_test,h_test)

Out[25]: -5.951545291204958
```

```
In [26]: ri.score(g_train,h_train)

Out[26]: 0.707448300378085
```

```
In [ ]:
```