EDA with Data Collection, Data Cleaning and Pre-processing with

Uber dataset

Data cleaning and Pre-procrossing

To import library

In [1]:

```
import numpy as np
import pandas as pd
```

To import dataset

In [2]:

```
d=pd.read_csv(r"c:\Users\user\Downloads\7_uber.csv")
d
```

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

To get top 10 record

In [3]:

```
d.head(10)
```

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

To get last 10

In [4]:

```
d.tail(10)
```

Out[4]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	picku
199990	9577367	2015-05-24 22:05:56.0000002	12.0	2015-05-24 22:05:56 UTC	-73.987106	
199991	13512837	2015-06-08 10:49:14.0000001	17.5	2015-06-08 10:49:14 UTC	-73.981453	
199992	20566507	2010-01-30 16:24:00.000000199	8.9	2010-01-30 16:24:00 UTC	-74.003548	
199993	28359558	2012-09-29 19:51:27.0000006	9.5	2012-09-29 19:51:27 UTC	-73.987798	
199994	3189201	2014-01-31 14:42:00.000000181	12.0	2014-01-31 14:42:00 UTC	-73.983070	
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	

To describe statistics Analysis

In [5]:

```
d.describe()
```

Out[5]:

	Unnamed: 0	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dro
count	2.000000e+05	200000.000000	200000.000000	200000.000000	199999.000000	19
mean	2.771250e+07	11.359955	-72.527638	39.935885	-72.525292	
std	1.601382e+07	9.901776	11.437787	7.720539	13.117408	
min	1.000000e+00	-52.000000	-1340.648410	-74.015515	-3356.666300	
25%	1.382535e+07	6.000000	-73.992065	40.734796	-73.991407	
50%	2.774550e+07	8.500000	-73.981823	40.752592	-73.980093	
75%	4.155530e+07	12.500000	-73.967154	40.767158	-73.963658	
max	5.542357e+07	499.000000	57.418457	1644.421482	1153.572603	

To get rows and columns

In [6]:

```
np.shape(d)
```

Out[6]:

```
(200000, 9)
```

To get number of elements

In [7]:

```
np.size(d)
```

Out[7]:

```
1800000
```

To get the missing value

In [8]:

```
d.isna()
```

Out[8]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	d
0	False	False	False	False	False	False	
1	False	False	False	False	False	False	
2	False	False	False	False	False	False	
3	False	False	False	False	False	False	
4	False	False	False	False	False	False	
...	...	...	...	...	...	...	
199995	False	False	False	False	False	False	
199996	False	False	False	False	False	False	
199997	False	False	False	False	False	False	
199998	False	False	False	False	False	False	
199999	False	False	False	False	False	False	

200000 rows × 9 columns

To drop the missing elements

In [9]:

```
d.dropna(axis=1,how='any')
```

Out[9]:

	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 × 7 columns

In [10]:

```
d["fare_amount"]
```

Out[10]:

```
0          7.5
1          7.7
2         12.9
3          5.3
4         16.0
          ...
199995     3.0
199996     7.5
199997    30.9
199998    14.5
199999    14.1
Name: fare_amount, Length: 200000, dtype: float64
```

Visualization

In [11]:

```
data=pd.DataFrame(d[['fare_amount','passenger_count']][0:500])
data
```

Out[11]:

	fare_amount	passenger_count
0	7.5	1
1	7.7	1
2	12.9	1
3	5.3	3
4	16.0	5
...	...	...
495	25.7	1
496	8.0	1
497	10.5	2
498	5.5	1
499	10.0	1

500 rows × 2 columns

In [12]:

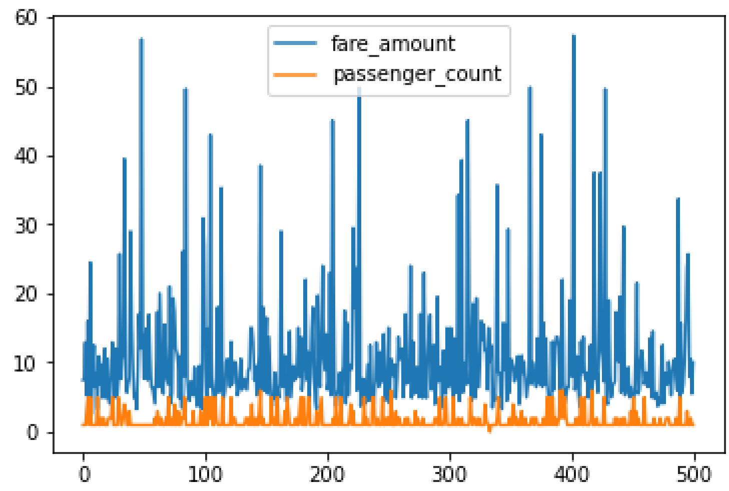
```
import matplotlib.pyplot as pp
```

In [13]:

```
data.plot.line()
```

Out[13]:

```
<AxesSubplot:>
```

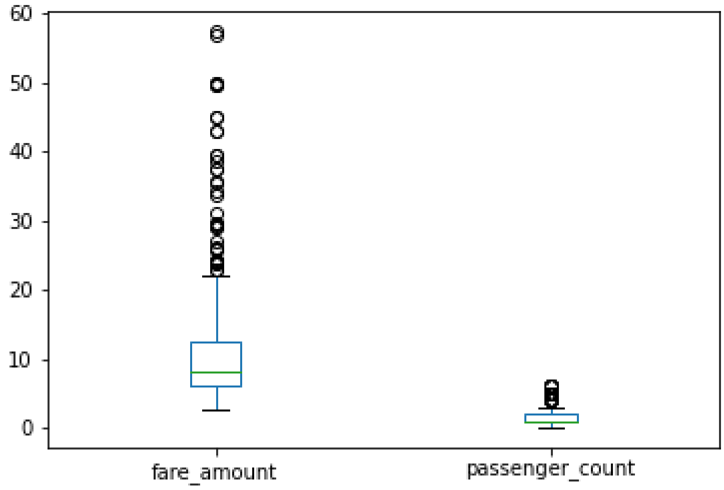


In [14]:

```
data.plot.box()
```

Out[14]:

```
<AxesSubplot:>
```

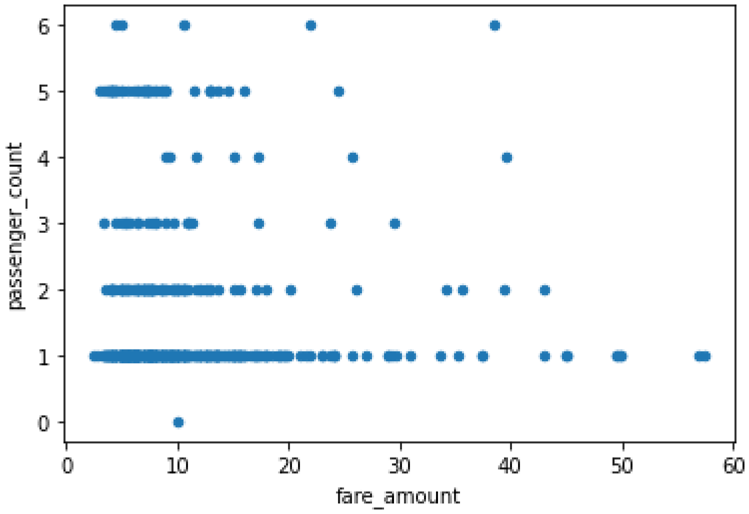


In [15]:

```
data.plot.scatter(x="fare_amount",y="passenger_count")
```

Out[15]:

```
<AxesSubplot:xlabel='fare_amount', ylabel='passenger_count'>
```

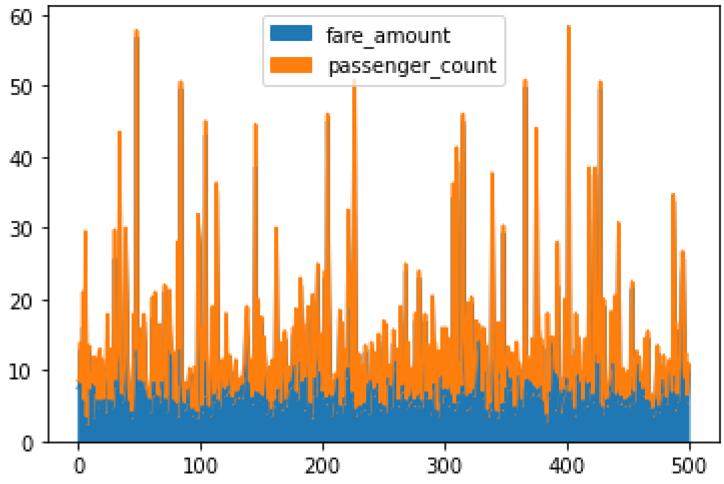


In [16]:

```
data.plot.area()
```

Out[16]:

```
<AxesSubplot:>
```

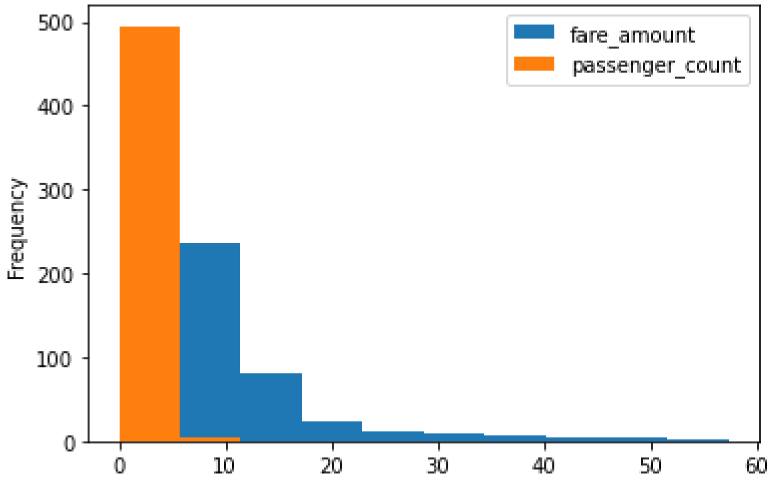


In [17]:

```
data.plot.hist()
```

Out[17]:

```
<AxesSubplot:ylabel='Frequency'>
```

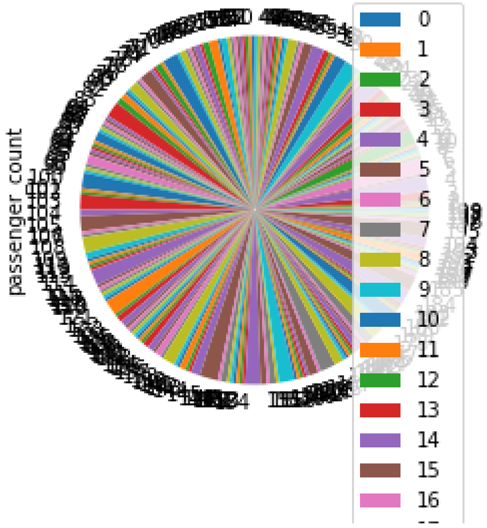


In [18]:

```
data=pd.DataFrame(d[['fare_amount','passenger_count']][0:200])
data.plot.pie(y="passenger_count")
```

Out[18]:

```
<AxesSubplot:ylabel='passenger_count'>
```



Statistics

Mean,median,mode,describe

In [19]:

```
data=pd.DataFrame(d[['fare_amount','passenger_count']][0:500])
data
```

Out[19]:

	fare_amount	passenger_count
0	7.5	1
1	7.7	1
2	12.9	1
3	5.3	3
4	16.0	5
...	...	...
495	25.7	1
496	8.0	1
497	10.5	2
498	5.5	1
499	10.0	1

500 rows × 2 columns

In [20]:

```
print(data.mean())
```

```
fare_amount       10.70872
passenger_count    1.66400
dtype: float64
```

In [21]:

```
print(data.median())
```

```
fare_amount       8.1
passenger_count   1.0
dtype: float64
```

In [22]:

```
print(data.mode())
```

```
   fare_amount  passenger_count
0          6.5                1
```

In [23]:

```
data.fillna(value=1)
```

Out[23]:

	fare_amount	passenger_count
0	7.5	1
1	7.7	1
2	12.9	1
3	5.3	3
4	16.0	5
...	...	...
495	25.7	1
496	8.0	1
497	10.5	2
498	5.5	1
499	10.0	1

500 rows × 2 columns

In [24]:

```
print(data.describe())
```

```
       fare_amount  passenger_count
count   500.000000       500.000000
mean     10.708720         1.664000
std       8.334145         1.267405
min       2.500000         0.000000
25%       6.000000         1.000000
50%       8.100000         1.000000
75%      12.500000         2.000000
max      57.330000         6.000000
```

Sum,cumsum,count,min,max

In [25]:

```
print(data.sum())
```

```
fare_amount        5354.36
passenger_count     832.00
dtype: float64
```

In [26]:

```
print(data.cumsum())
```

```
     fare_amount  passenger_count
0           7.50                1
1          15.20                2
2          28.10                3
3          33.40                6
4          49.40               11
..           ...              ...
495      5320.36              827
496      5328.36              828
497      5338.86              830
498      5344.36              831
499      5354.36              832

[500 rows x 2 columns]
```

In [27]:

```
print(data.count())
```

```
fare_amount        500
passenger_count    500
dtype: int64
```

In [28]:

```
print(data.min())
```

```
fare_amount        2.5
passenger_count    0.0
dtype: float64
```

In [29]:

```
print(data.max())
```

```
fare_amount        57.33
passenger_count     6.00
dtype: float64
```

covariance and correlation (spearman and pearsons)

In [30]:

```
data1=data['fare_amount'][0:10]
data1
```

Out[30]:

```
0     7.5
1     7.7
2    12.9
3     5.3
4    16.0
5     4.9
6    24.5
7     2.5
8     9.7
9    12.5
Name: fare_amount, dtype: float64
```

In [31]:

```
data2=data['passenger_count'][0:10]
data2
```

Out[31]:

```
0    1
1    1
2    1
3    3
4    5
5    1
6    5
7    1
8    1
9    1
Name: passenger_count, dtype: int64
```

In [32]:

```
from numpy import cov
print(cov(data1,data2))
```

```
[[41.74055556  7.67777778]
 [ 7.67777778  2.88888889]]
```

In [33]:

```
from scipy.stats import pearsonr
print(pearsonr(data1,data2))
```

```
(0.6991832347843764, 0.024444145792245162)
```

In [34]:

```
from scipy.stats import spearmanr
print(spearmanr(data1,data2))
```

```
SpearmanrResult(correlation=0.509395451638894, pvalue=0.1326052475011008)
```

In [ ]:

```
```