120A3051

Shreya Idate

Batch: E3

EXPERIMENT NO. 2

Aim: Data preparation using NumPy and Pandas

a. Obtain a listing of all records that are outliers according to the any field. Print out a listing of the 10 largest values for that field.

b. Do the following for the any field.

i. Standardize the variable.

ii. Identify how many outliers there are and identify the most extreme outlier

Theory:

What is Pandas?

Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

Why Use Pandas?

Pandas allows us to analyze big data and make conclusions based on statistical theories. Pandas can clean messy data sets, and make them readable and relevant. Relevant data is very important in data science.

What are Outliers?

An outlier of a dataset is defined as a value that is more than 3 standard deviations from the mean. Removing outliers from a pandas. DataFrame removes any rows in the DataFrame which contain an outlier. Outlier calculations are performed separately for each column.

What is nlargest?

DataFrame.nlargest(n, columns, keep='first') [source]

Return the first n rows ordered by columns in descending order. Return the first n rows with the largest values in columns, in descending order. The columns that are not specified are returned as well, but not used for ordering.

What is group by?

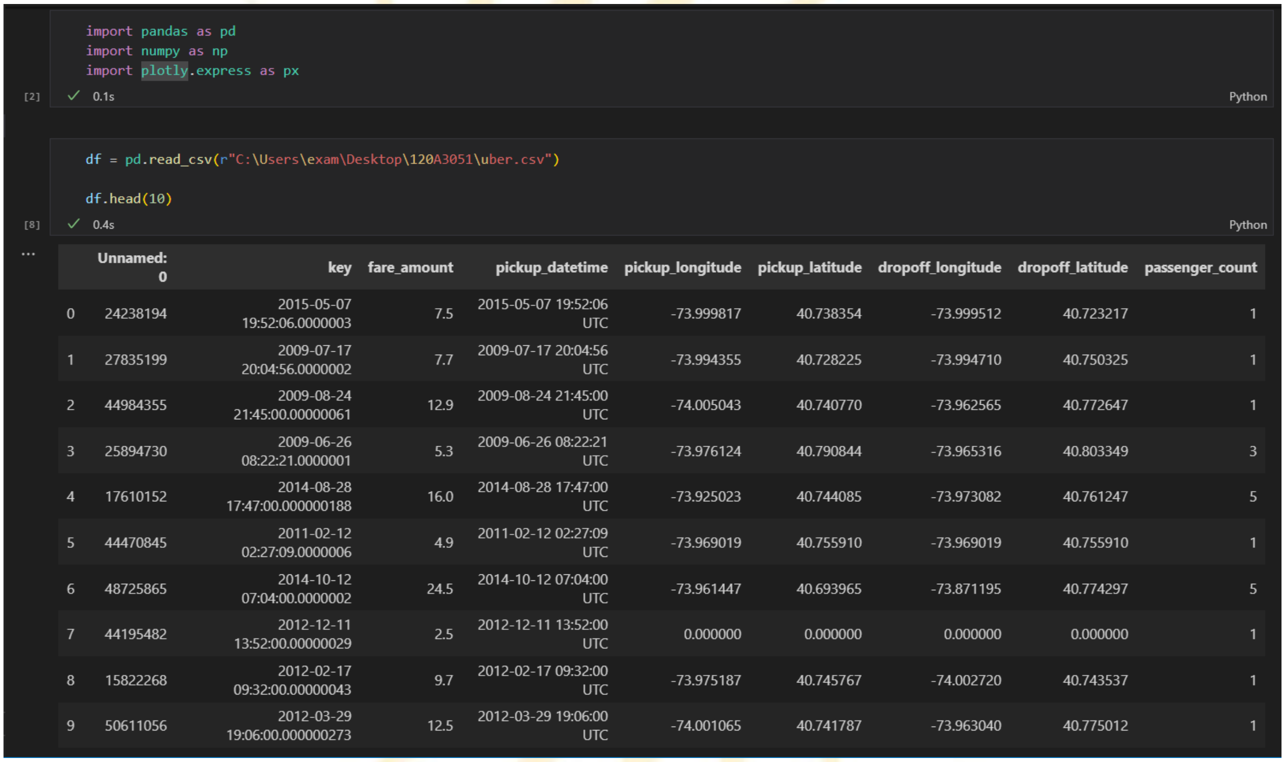
DataFrame.groupby(by=None, axis=0, level=None, as_index=True, sort=True, group_keys=True, squeeze=NoDefault.no_default, observed=False, dropna=True) [source]
Group DataFrame using a mapper or by a Series of columns. A groupby operation involves some

combination of splitting the object, applying a function, and combining the results. This can be used to group large amounts of data and compute operations on these groups.

What is mean?
DataFrame.mean(axis=NoDefault.no_default, skipna=True, level=None, numeric_only=None, **kwargs) [source]

Return the mean of the values over the requested axis.

Program:



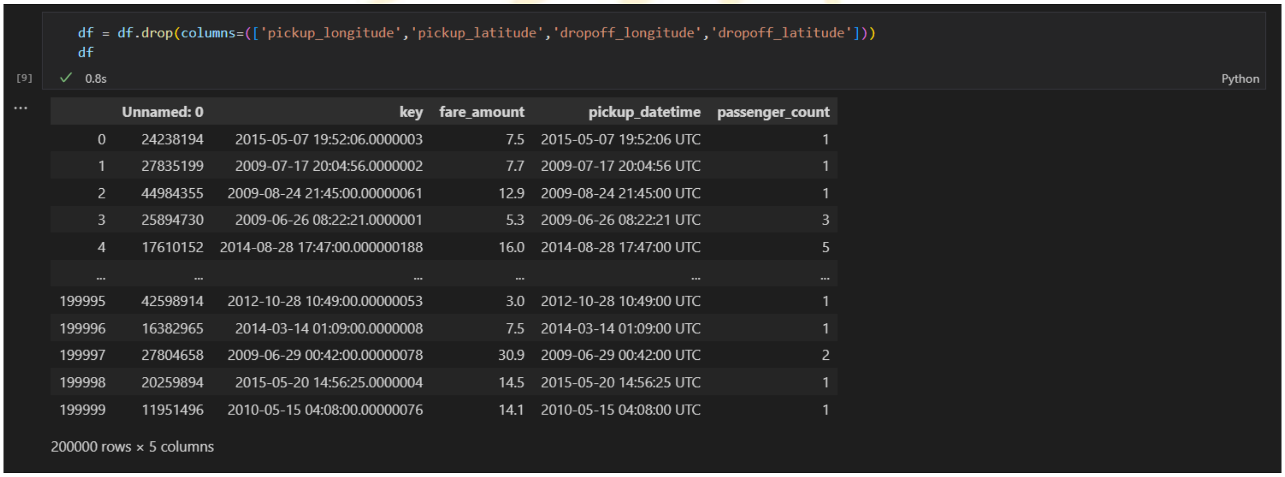
```
import pandas as pd
import numpy as np
import plotly.express as px
```

[2] ✓ 0.1s Python

```
df = pd.read_csv(r"C:\Users\exam\Desktop\120A3051\uber.csv")

df.head(10)
```

[8] ✓ 0.4s Python

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.999512	40.723217	1
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.994710	40.750325	1
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73.962565	40.772647	1
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.965316	40.803349	3
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.973082	40.761247	5
5	44470845	2011-02-12 02:27:09.0000006	4.9	2011-02-12 02:27:09 UTC	-73.969019	40.755910	-73.969019	40.755910	1
6	48725865	2014-10-12 07:04:00.0000002	24.5	2014-10-12 07:04:00 UTC	-73.961447	40.693965	-73.871195	40.774297	5
7	44195482	2012-12-11 13:52:00.00000029	2.5	2012-12-11 13:52:00 UTC	0.000000	0.000000	0.000000	0.000000	1
8	15822268	2012-02-17 09:32:00.00000043	9.7	2012-02-17 09:32:00 UTC	-73.975187	40.745767	-74.002720	40.743537	1
9	50611056	2012-03-29 19:06:00.000000273	12.5	2012-03-29 19:06:00 UTC	-74.001065	40.741787	-73.963040	40.775012	1

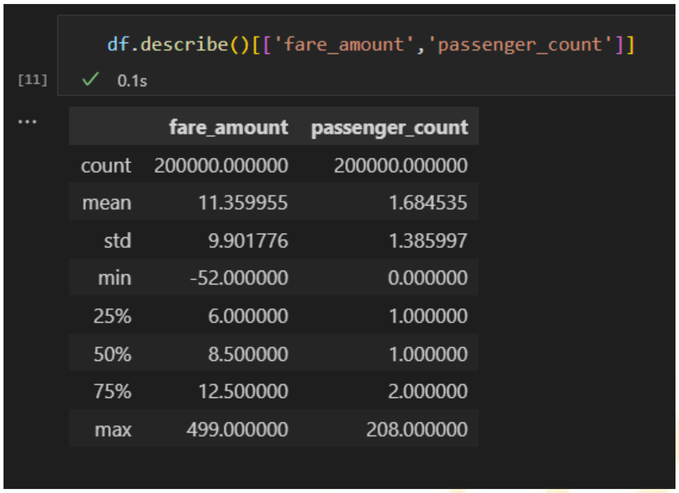


```
df = df.drop(columns=(['pickup_longitude','pickup_latitude','dropoff_longitude','dropoff_latitude']))
df
```

[9] ✓ 0.8s Python

	Unnamed: 0	key	fare_amount	pickup_datetime	passenger_count
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	1
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	1
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	1
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	3
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	5
...	...	...	...	...	...
199995	42598914	2012-10-28 10:49:00.00000053	3.0	2012-10-28 10:49:00 UTC	1
199996	16382965	2014-03-14 01:09:00.0000008	7.5	2014-03-14 01:09:00 UTC	1
199997	27804658	2009-06-29 00:42:00.00000078	30.9	2009-06-29 00:42:00 UTC	2
199998	20259894	2015-05-20 14:56:25.0000004	14.5	2015-05-20 14:56:25 UTC	1
199999	11951496	2010-05-15 04:08:00.00000076	14.1	2010-05-15 04:08:00 UTC	1

200000 rows × 5 columns

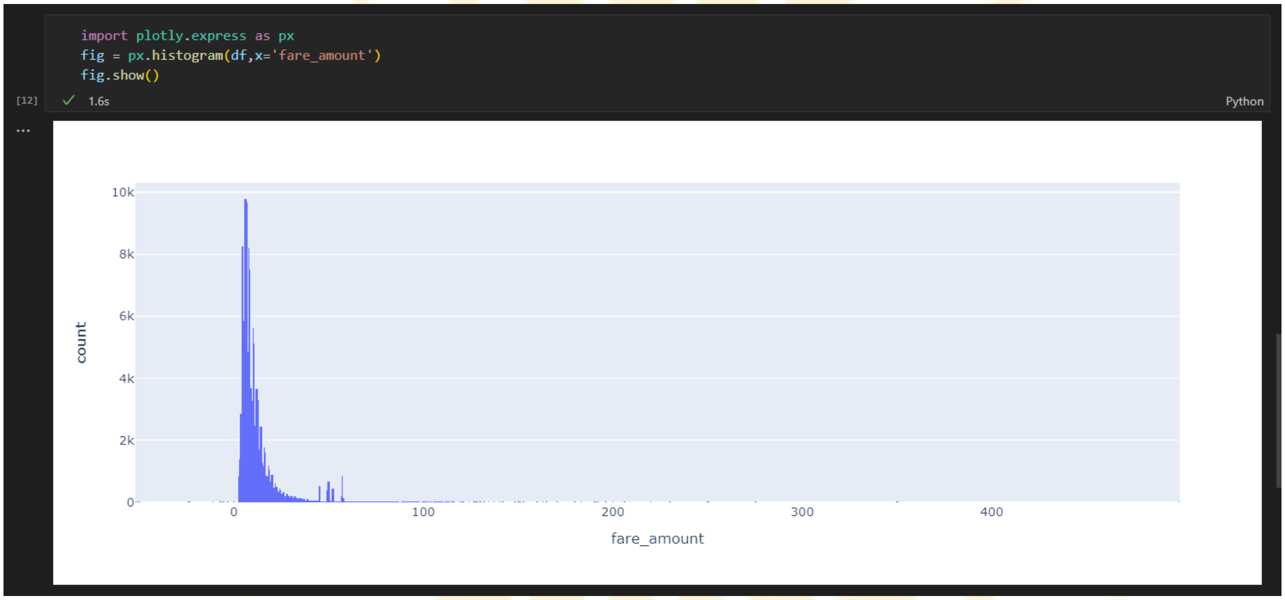


```
df.describe()[['fare_amount','passenger_count']]
```

[11] ✓ 0.1s

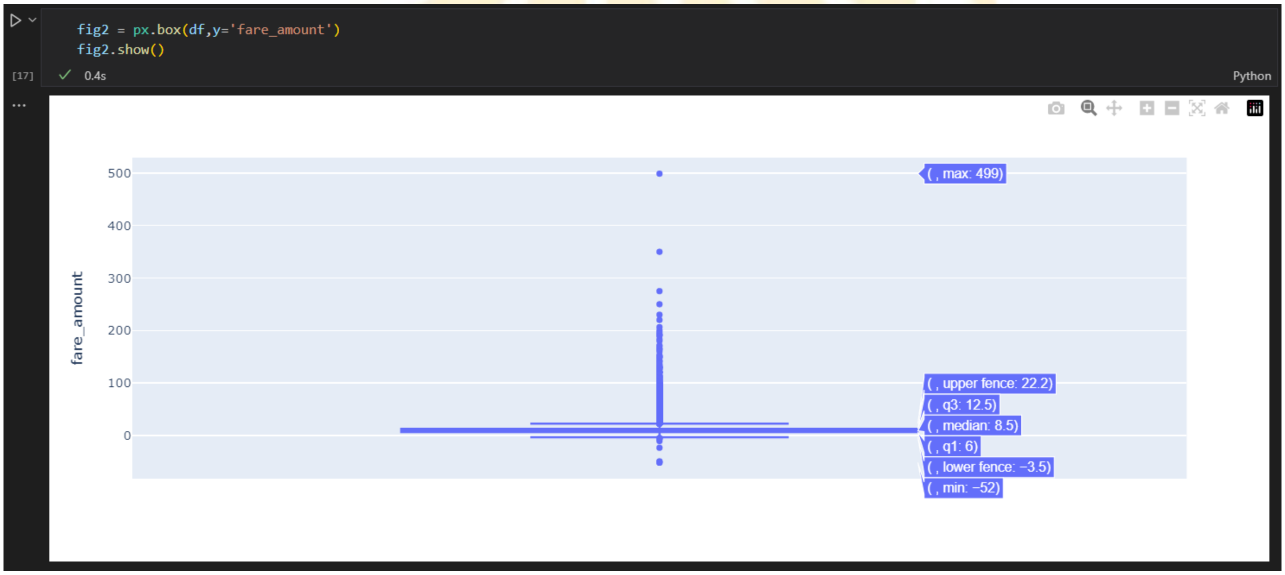
	fare_amount	passenger_count
count	200000.000000	200000.000000
mean	11.359955	1.684535
std	9.901776	1.385997
min	-52.000000	0.000000
25%	6.000000	1.000000
50%	8.500000	1.000000
75%	12.500000	2.000000
max	499.000000	208.000000

With x=fare_amount



```
import plotly.express as px
fig = px.histogram(df,x='fare_amount')
fig.show()
```

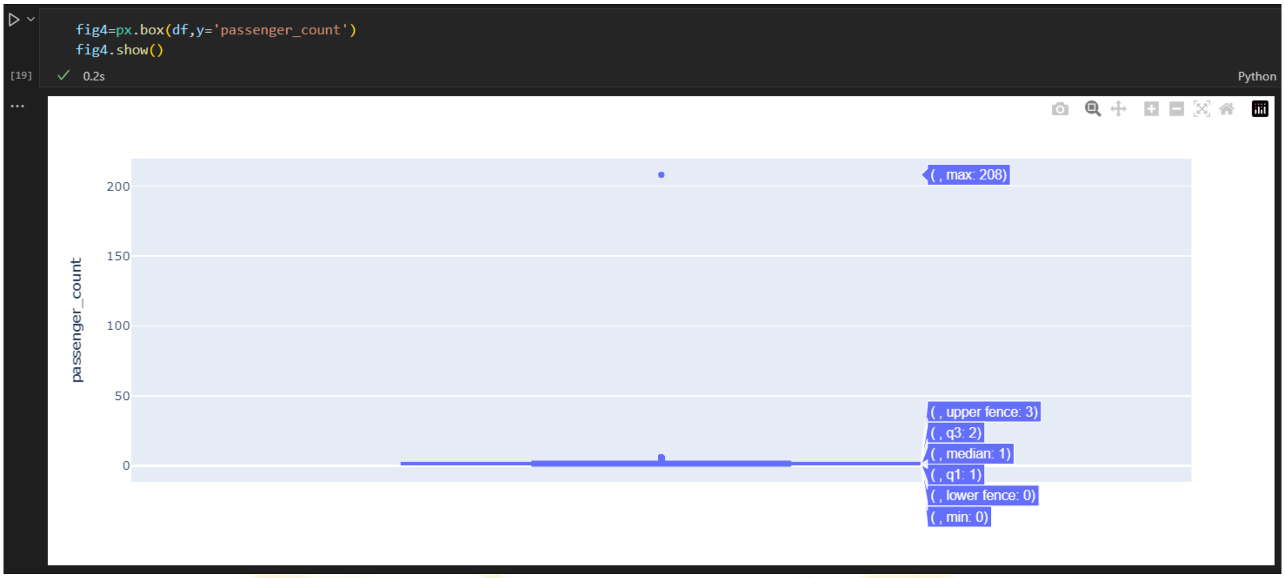
[12] ✓ 1.6s Python



```
fig2 = px.box(df,y='fare_amount')
fig2.show()
```

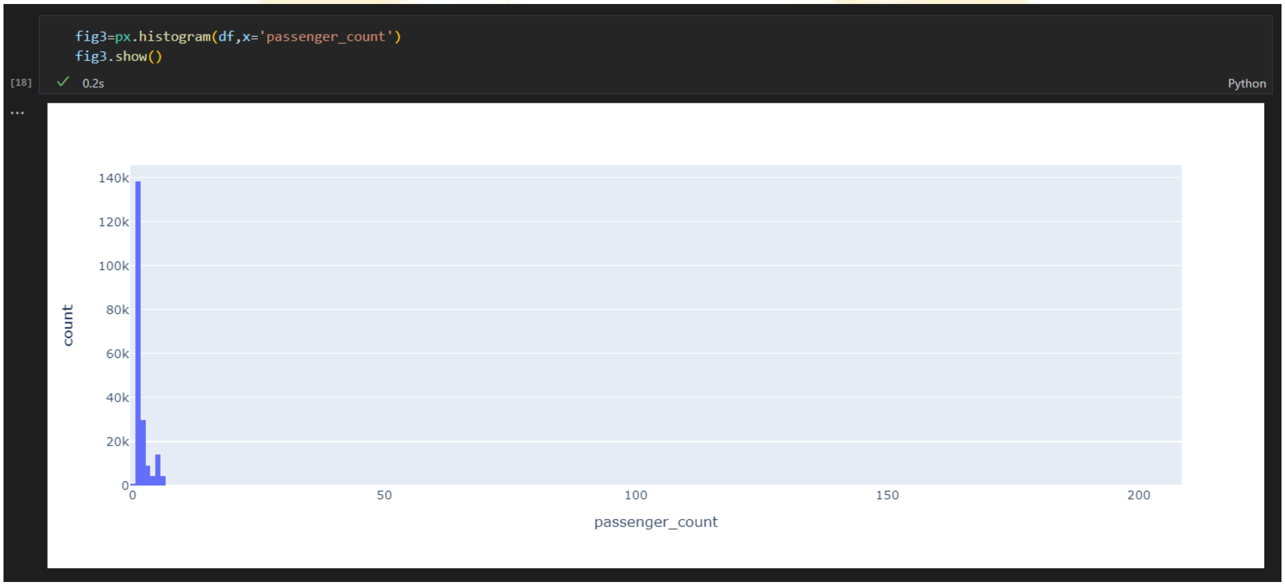
[17] ✓ 0.4s Python

With x=passenger_count



```
fig4=px.box(df,y='passenger_count')
fig4.show()
```

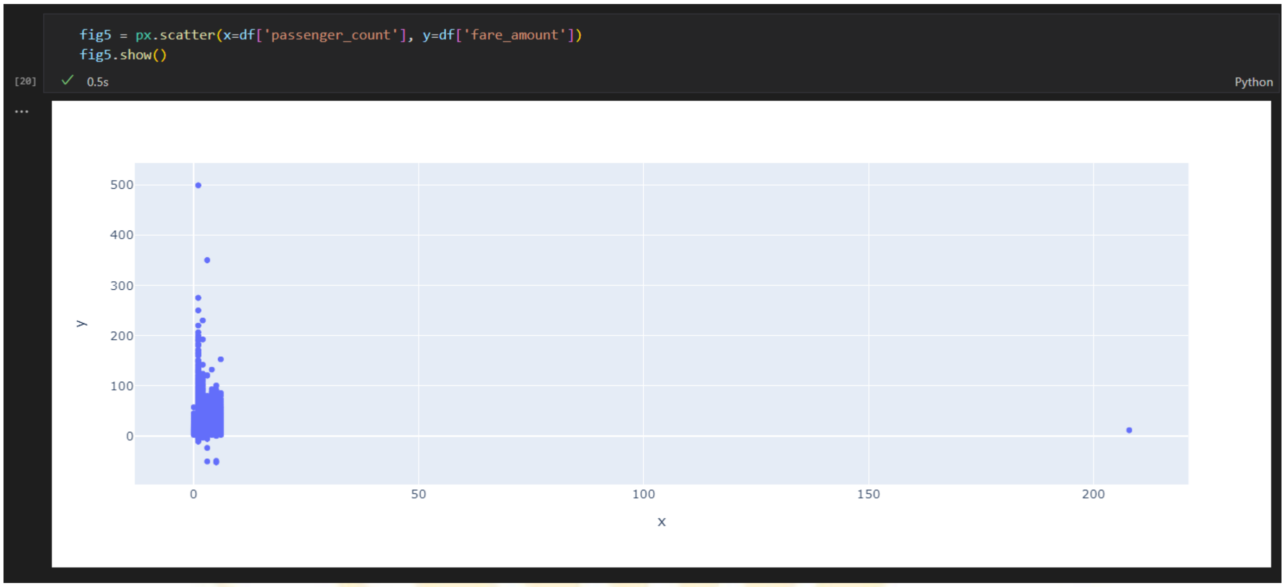
[19] ✓ 0.2s Python



```
fig3=px.histogram(df,x='passenger_count')
fig3.show()
```

[18] ✓ 0.2s Python

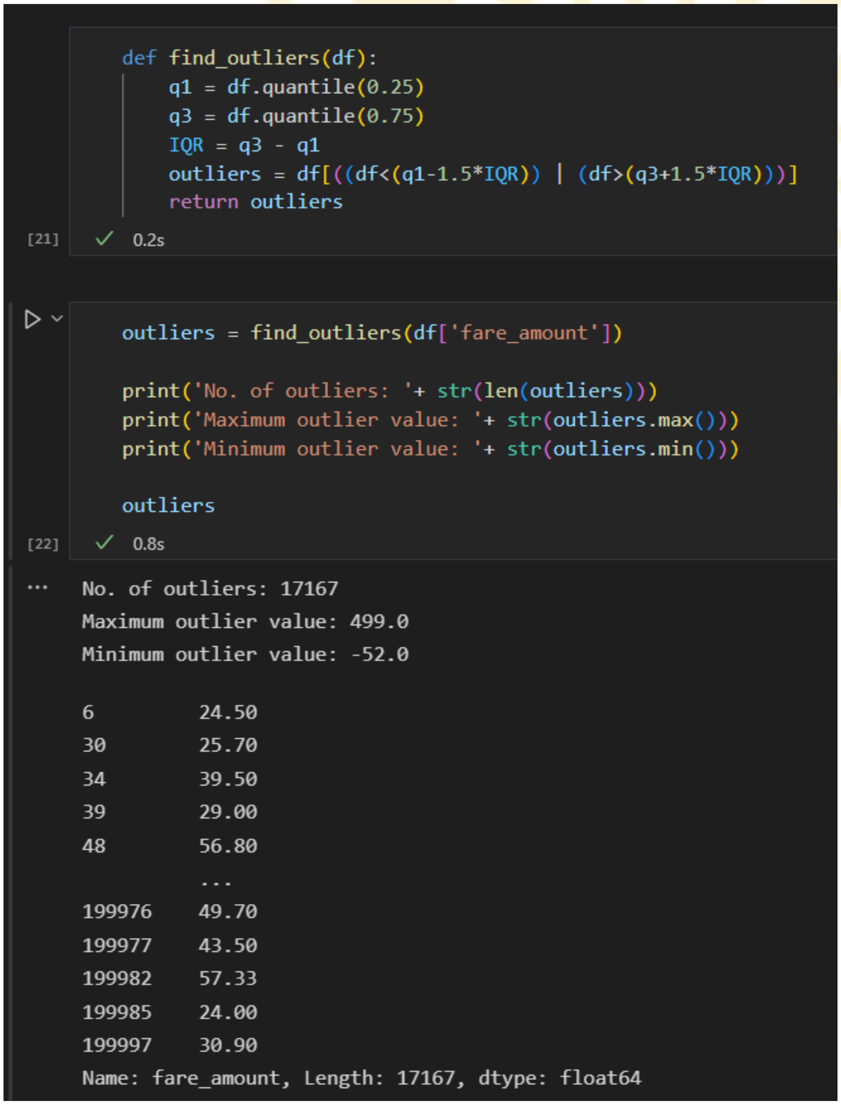
Scatter with x n y



```
fig5 = px.scatter(x=df['passenger_count'], y=df['fare_amount'])
fig5.show()
```

[20] ✓ 0.5s Python

Finding outliers using statistical methods



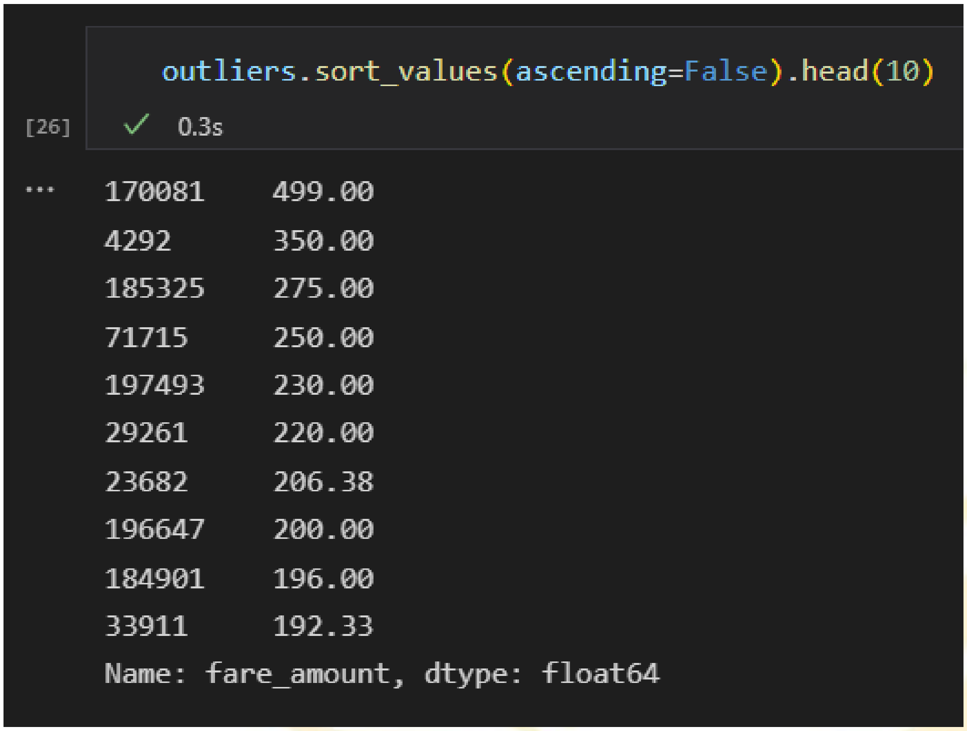
```
def find_outliers(df):
    q1 = df.quantile(0.25)
    q3 = df.quantile(0.75)
    IQR = q3 - q1
    outliers = df[((df<(q1-1.5*IQR)) | (df>(q3+1.5*IQR)))]
    return outliers
```

[21] ✓ 0.2s

```
outliers = find_outliers(df['fare_amount'])

print('No. of outliers: '+ str(len(outliers)))
print('Maximum outlier value: '+ str(outliers.max()))
print('Minimum outlier value: '+ str(outliers.min()))

outliers
```

[22] ✓ 0.8s

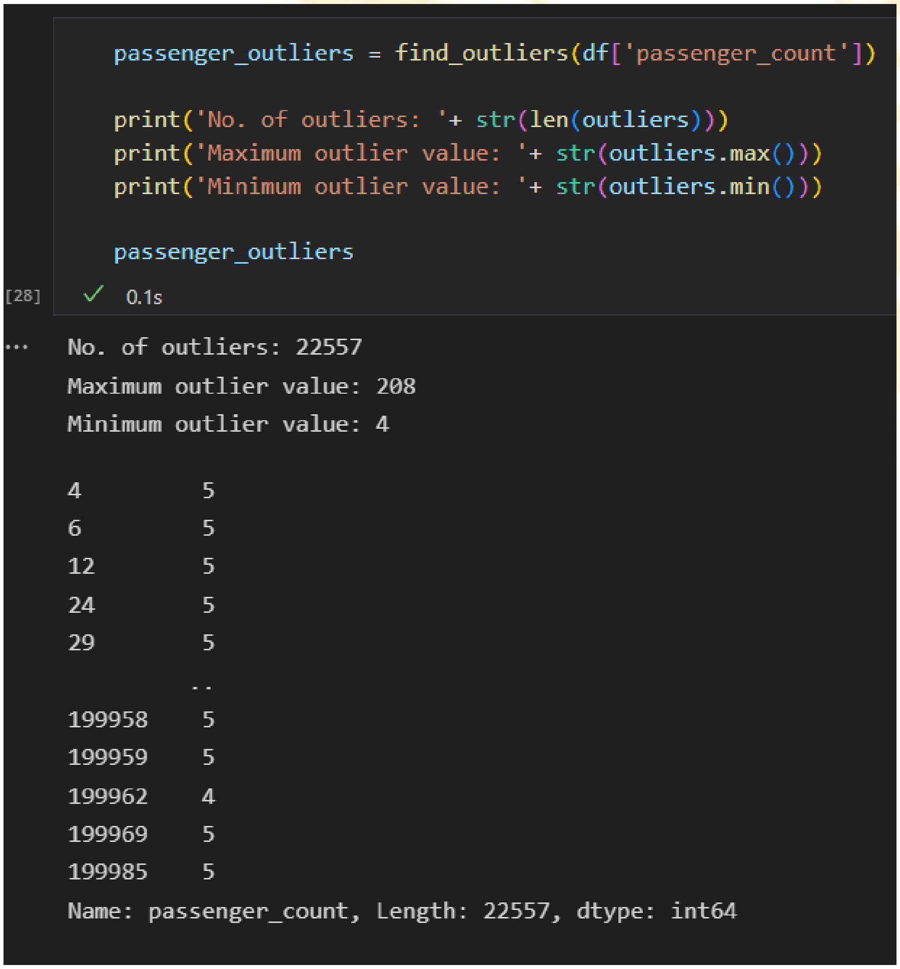
```
No. of outliers: 17167
Maximum outlier value: 499.0
Minimum outlier value: -52.0

6         24.50
30        25.70
34        39.50
39        29.00
48        56.80
          ...
199976    49.70
199977    43.50
199982    57.33
199985    24.00
199997    30.90
Name: fare_amount, Length: 17167, dtype: float64
```



```
outliers.sort_values(ascending=False).head(10)
```

[26] ✓ 0.3s

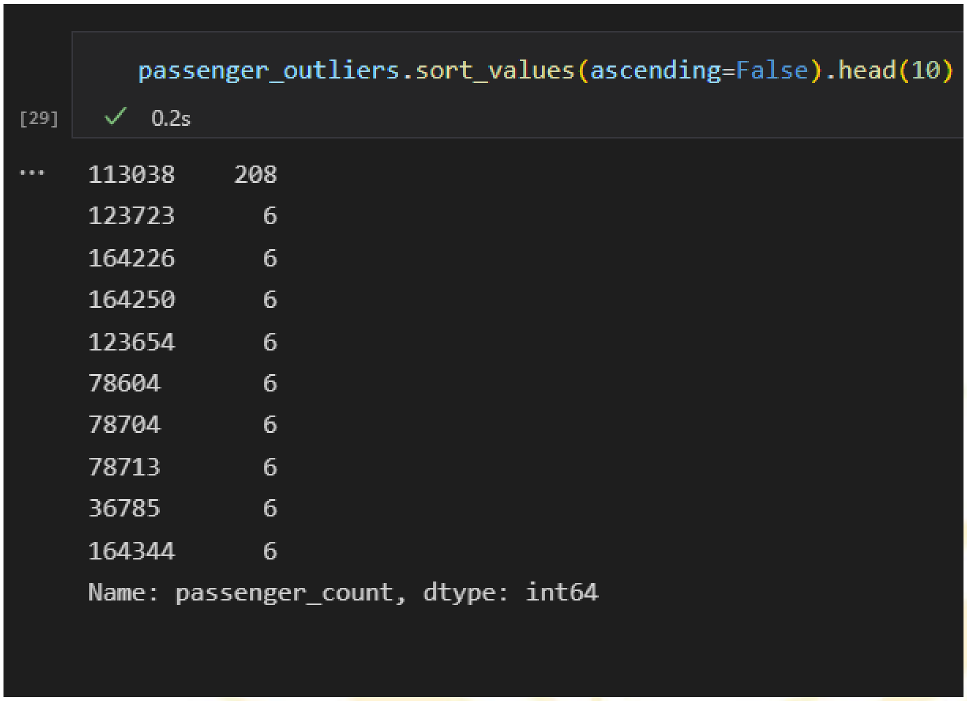
```
170081    499.00
4292      350.00
185325    275.00
71715     250.00
197493    230.00
29261     220.00
23682     206.38
196647    200.00
184901    196.00
33911     192.33
Name: fare_amount, dtype: float64
```



```
passenger_outliers = find_outliers(df['passenger_count'])

print('No. of outliers: '+ str(len(outliers)))
print('Maximum outlier value: '+ str(outliers.max()))
print('Minimum outlier value: '+ str(outliers.min()))

passenger_outliers
```

[28] ✓ 0.1s

```
No. of outliers: 22557
Maximum outlier value: 208
Minimum outlier value: 4

4         5
6         5
12        5
24        5
29        5
         ..
199958    5
199959    5
199962    4
199969    5
199985    5
Name: passenger_count, Length: 22557, dtype: int64
```



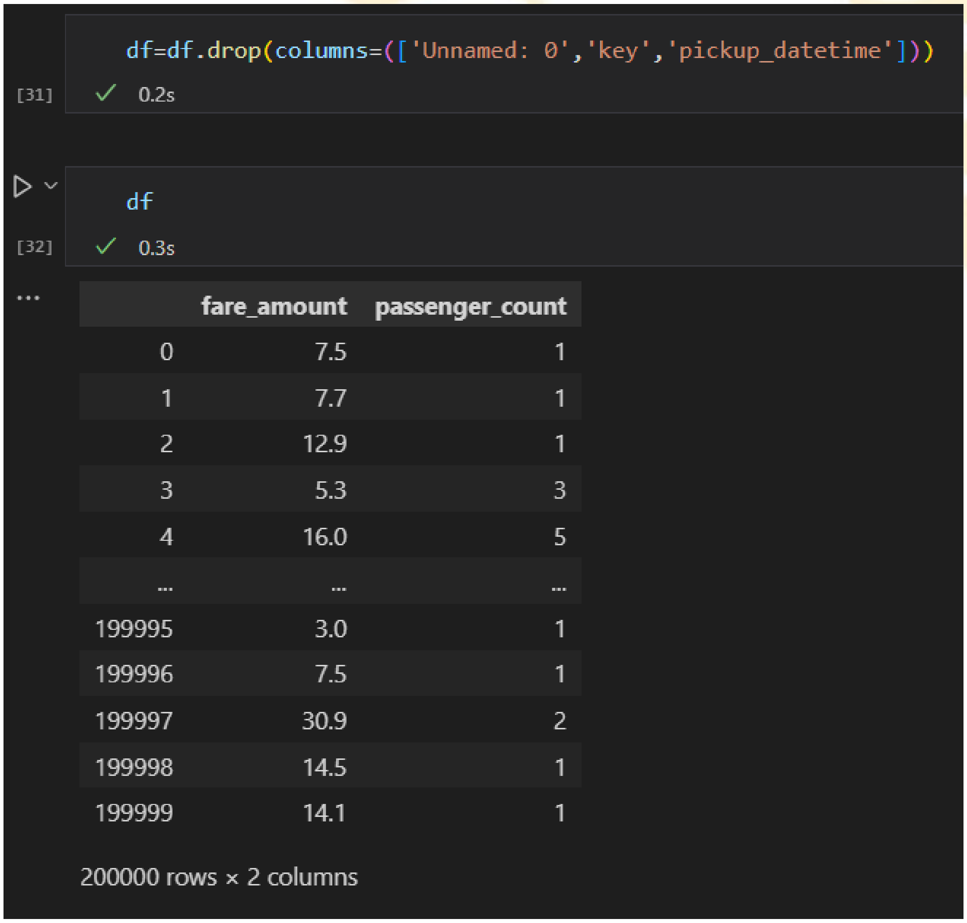
```
passenger_outliers.sort_values(ascending=False).head(10)
```

[29] ✓ 0.2s

```
113038    208
123723      6
164226      6
164250      6
123654      6
78604       6
78704       6
78713       6
36785       6
164344      6
Name: passenger_count, dtype: int64
```

Working with outliers using statistical methods

Drop unnecessary columns



```
df=df.drop(columns=(['Unnamed: 0','key','pickup_datetime']))
```

[31] ✓ 0.2s

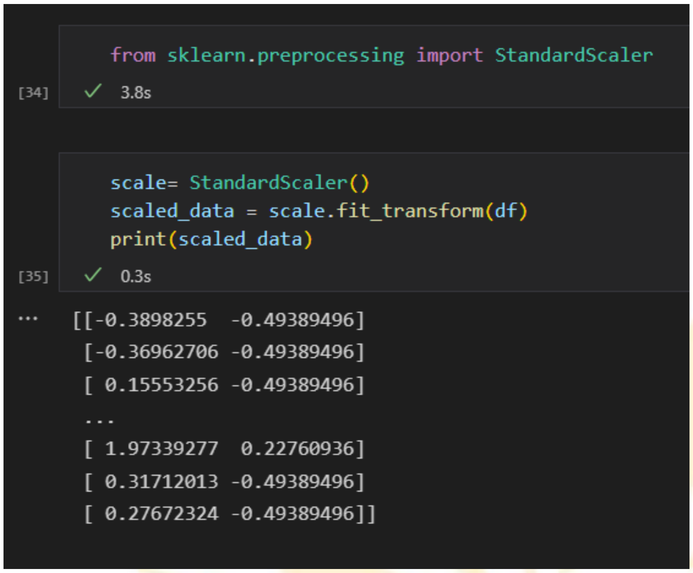
```
df
```

[32] ✓ 0.3s

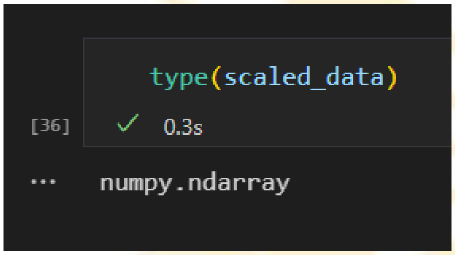
```
        fare_amount  passenger_count
     0          7.5                1
     1          7.7                1
     2         12.9                1
     3          5.3                3
     4         16.0                5
   ...          ...              ...
199995          3.0                1
199996          7.5                1
199997         30.9                2
199998         14.5                1
199999         14.1                1

200000 rows × 2 columns
```

STANDARDIZE THE DATASET

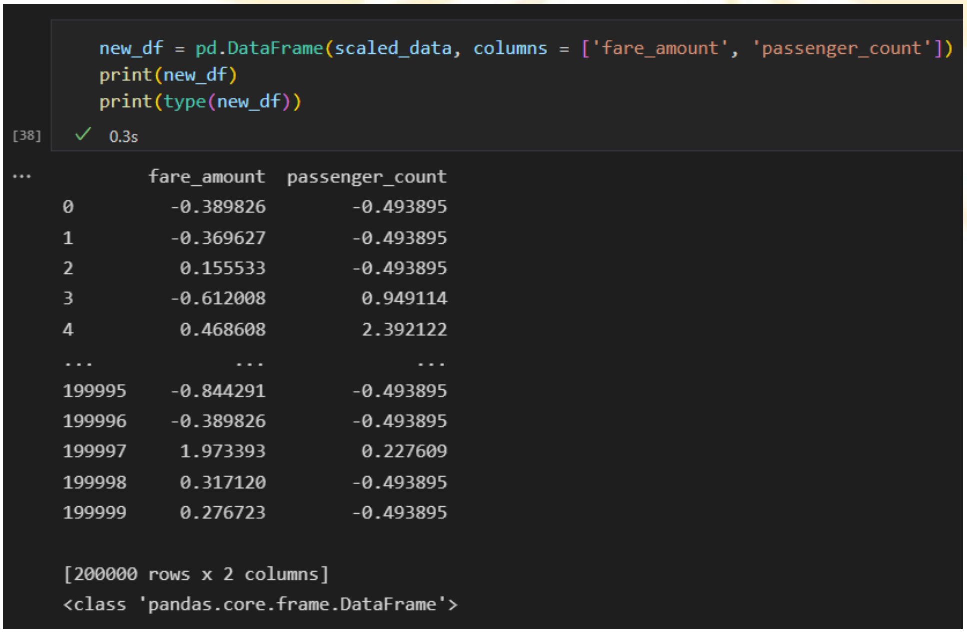
pip install -U scikit-learn scipy matplotlib



```
[34]  from sklearn.preprocessing import StandardScaler
      ✓ 3.8s

      scale= StandardScaler()
      scaled_data = scale.fit_transform(df)
      print(scaled_data)
[35]  ✓ 0.3s

...   [[-0.3898255  -0.49389496]
       [-0.36962706 -0.49389496]
       [ 0.15553256 -0.49389496]
       ...
       [ 1.97339277  0.22760936]
       [ 0.31712013 -0.49389496]
       [ 0.27672324 -0.49389496]]
```



```
      type(scaled_data)
[36]  ✓ 0.3s

...   numpy.ndarray
```



```
      new_df = pd.DataFrame(scaled_data, columns = ['fare_amount', 'passenger_count'])
      print(new_df)
      print(type(new_df))
[38]  ✓ 0.3s

...          fare_amount  passenger_count
      0         -0.389826        -0.493895
      1         -0.369627        -0.493895
      2          0.155533        -0.493895
      3         -0.612008         0.949114
      4          0.468608         2.392122
      ...             ...              ...
      199995    -0.844291        -0.493895
      199996    -0.389826        -0.493895
      199997     1.973393         0.227609
      199998     0.317120        -0.493895
      199999     0.276723        -0.493895

      [200000 rows x 2 columns]
      <class 'pandas.core.frame.DataFrame'>
```

Conclusion: Successful Data preparation using NumPy and Pandas.