

shelfjiwx

July 31, 2023

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

```
[ ]: df=pd.read_csv("/content/6_Salesworkload1.csv")
df
```

```
[ ]:
      MonthYear  Time index      Country  StoreID      City  Dept_ID \
0      10.2016          1.0  United Kingdom  88253.0  London (I)      1.0
1      10.2016          1.0  United Kingdom  88253.0  London (I)      2.0
2      10.2016          1.0  United Kingdom  88253.0  London (I)      3.0
3      10.2016          1.0  United Kingdom  88253.0  London (I)      4.0
4      10.2016          1.0  United Kingdom  88253.0  London (I)      5.0
...      ...      ...      ...      ...      ...
7653    06.2017          9.0      Sweden  29650.0  Gothenburg      12.0
7654    06.2017          9.0      Sweden  29650.0  Gothenburg      16.0
7655    06.2017          9.0      Sweden  29650.0  Gothenburg      11.0
7656    06.2017          9.0      Sweden  29650.0  Gothenburg      17.0
7657    06.2017          9.0      Sweden  29650.0  Gothenburg      18.0

      Dept. Name  HoursOwn  HoursLease  Sales units  Turnover \
0              Dry  3184.764          0.0    398560.0  1226244.0
1             Frozen  1582.941          0.0     82725.0   387810.0
2             other    47.205          0.0   438400.0   654657.0
3              Fish  1623.852          0.0   309425.0   499434.0
4  Fruits & Vegetables  1759.173          0.0   165515.0   329397.0
...      ...      ...      ...      ...      ...
7653          Checkout  6322.323          0.0   3886530.0  14538825.0
7654  Customer Services  4270.479          0.0      245.0        0.0
7655          Delivery         0          0.0        0.0        0.0
7656          others  2224.929          0.0      245.0        0.0
7657             all   39652.2          0.0   3886530.0  15056214.0

      Customer Area (m2) Opening hours
0              NaN    953.04      Type A
1              NaN    720.48      Type A
```

2	NaN	966.72	Type A
3	NaN	1053.36	Type A
4	NaN	1053.36	Type A
...
7653	NaN	#NV	Type A
7654	NaN	#NV	Type A
7655	NaN	#NV	Type A
7656	NaN	#NV	Type A
7657	NaN	#NV	Type A

[7658 rows x 14 columns]

```
[ ]: df.head()
```

```
[ ]:   MonthYear  Time index      Country  StoreID      City  Dept_ID \
0   10.2016         1.0  United Kingdom  88253.0  London (I)    1.0
1   10.2016         1.0  United Kingdom  88253.0  London (I)    2.0
2   10.2016         1.0  United Kingdom  88253.0  London (I)    3.0
3   10.2016         1.0  United Kingdom  88253.0  London (I)    4.0
4   10.2016         1.0  United Kingdom  88253.0  London (I)    5.0
```

	Dept. Name	HoursOwn	HoursLease	Sales units	Turnover \
0	Dry	3184.764	0.0	398560.0	1226244.0
1	Frozen	1582.941	0.0	82725.0	387810.0
2	other	47.205	0.0	438400.0	654657.0
3	Fish	1623.852	0.0	309425.0	499434.0
4	Fruits & Vegetables	1759.173	0.0	165515.0	329397.0

	Customer Area (m2)	Opening hours
0	NaN	953.04
1	NaN	720.48
2	NaN	966.72
3	NaN	1053.36
4	NaN	1053.36

1 DATA CLEANING AND DATA PREPROCESSING

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7658 entries, 0 to 7657
Data columns (total 14 columns):
#   Column      Non-Null Count  Dtype
---  -
0   MonthYear    7658 non-null   object
1   Time index   7650 non-null   float64
2   Country      7650 non-null   object
```

```

3  StoreID          7650 non-null  float64
4  City            7650 non-null  object
5  Dept_ID        7650 non-null  float64
6  Dept. Name     7650 non-null  object
7  HoursOwn       7650 non-null  object
8  HoursLease     7650 non-null  float64
9  Sales units    7650 non-null  float64
10 Turnover       7650 non-null  float64
11 Customer       0 non-null     float64
12 Area (m2)      7650 non-null  object
13 Opening hours  7650 non-null  object
dtypes: float64(7), object(7)
memory usage: 837.7+ KB

```

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

```

[ ]:
count      Time index      StoreID      Dept_ID      HoursLease      Sales units  \
mean         5.000000    61995.220000         9.470588         22.036078    1.076471e+06
std          2.582158    29924.581631         5.337429        133.299513    1.728113e+06
min           1.000000    12227.000000         1.000000         0.000000    0.000000e+00
25%           3.000000    29650.000000         5.000000         0.000000    5.457125e+04
50%           5.000000    75400.500000         9.000000         0.000000    2.932300e+05
75%           7.000000    87703.000000        14.000000         0.000000    9.175075e+05
max           9.000000    98422.000000        18.000000        3984.000000    1.124296e+07

count      Turnover      Customer
mean     3.721393e+06         NaN
std      6.003380e+06         NaN
min      0.000000e+00         NaN
25%      2.726798e+05         NaN
50%      9.319575e+05         NaN
75%      3.264432e+06         NaN
max      4.271739e+07         NaN

```

```
[ ]: df.columns
```

```

[ ]: Index(['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
          'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
          'Customer', 'Area (m2)', 'Opening hours'],
          dtype='object')

```

```

[ ]: df1=df.fillna(1)
df1

```

```
[ ]:      MonthYear  Time index      Country  StoreID      City  Dept_ID  \
0      10.2016      1.0  United Kingdom  88253.0  London (I)      1.0
1      10.2016      1.0  United Kingdom  88253.0  London (I)      2.0
2      10.2016      1.0  United Kingdom  88253.0  London (I)      3.0
3      10.2016      1.0  United Kingdom  88253.0  London (I)      4.0
4      10.2016      1.0  United Kingdom  88253.0  London (I)      5.0
...      ...      ...      ...      ...      ...
7653   06.2017      9.0      Sweden  29650.0  Gothenburg      12.0
7654   06.2017      9.0      Sweden  29650.0  Gothenburg      16.0
7655   06.2017      9.0      Sweden  29650.0  Gothenburg      11.0
7656   06.2017      9.0      Sweden  29650.0  Gothenburg      17.0
7657   06.2017      9.0      Sweden  29650.0  Gothenburg      18.0
```

```
      Dept. Name  HoursOwn  HoursLease  Sales units  Turnover  \
0      Dry  3184.764      0.0    398560.0  1226244.0
1      Frozen  1582.941      0.0     82725.0   387810.0
2      other    47.205      0.0   438400.0   654657.0
3      Fish  1623.852      0.0   309425.0   499434.0
4  Fruits & Vegetables  1759.173      0.0   165515.0   329397.0
...      ...      ...      ...      ...
7653      Checkout  6322.323      0.0   3886530.0  14538825.0
7654  Customer Services  4270.479      0.0     245.0     0.0
7655      Delivery      0      0.0     0.0     0.0
7656      others  2224.929      0.0     245.0     0.0
7657      all  39652.2      0.0   3886530.0  15056214.0
```

```
      Customer Area (m2) Opening hours
0      1.0    953.04      Type A
1      1.0    720.48      Type A
2      1.0    966.72      Type A
3      1.0   1053.36      Type A
4      1.0   1053.36      Type A
...      ...      ...
7653      1.0    #NV      Type A
7654      1.0    #NV      Type A
7655      1.0    #NV      Type A
7656      1.0    #NV      Type A
7657      1.0    #NV      Type A
```

[7658 rows x 14 columns]

```
[ ]: df1=df1.replace('#NV',1)
df1
```

```
[ ]:      MonthYear  Time index      Country  StoreID      City  Dept_ID  \
0      10.2016      1.0  United Kingdom  88253.0  London (I)      1.0
1      10.2016      1.0  United Kingdom  88253.0  London (I)      2.0
```

2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0
...
7653	06.2017	9.0	Sweden	29650.0	Gothenburg	12.0
7654	06.2017	9.0	Sweden	29650.0	Gothenburg	16.0
7655	06.2017	9.0	Sweden	29650.0	Gothenburg	11.0
7656	06.2017	9.0	Sweden	29650.0	Gothenburg	17.0
7657	06.2017	9.0	Sweden	29650.0	Gothenburg	18.0

	Dept. Name	HoursOwn	HoursLease	Sales units	Turnover \
0	Dry	3184.764	0.0	398560.0	1226244.0
1	Frozen	1582.941	0.0	82725.0	387810.0
2	other	47.205	0.0	438400.0	654657.0
3	Fish	1623.852	0.0	309425.0	499434.0
4	Fruits & Vegetables	1759.173	0.0	165515.0	329397.0
...
7653	Checkout	6322.323	0.0	3886530.0	14538825.0
7654	Customer Services	4270.479	0.0	245.0	0.0
7655	Delivery	0	0.0	0.0	0.0
7656	others	2224.929	0.0	245.0	0.0
7657	all	39652.2	0.0	3886530.0	15056214.0

	Customer Area (m2)	Opening hours
0	1.0	953.04 Type A
1	1.0	720.48 Type A
2	1.0	966.72 Type A
3	1.0	1053.36 Type A
4	1.0	1053.36 Type A
...
7653	1.0	1 Type A
7654	1.0	1 Type A
7655	1.0	1 Type A
7656	1.0	1 Type A
7657	1.0	1 Type A

[7658 rows x 14 columns]

```
[ ]: df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7658 entries, 0 to 7657
Data columns (total 14 columns):
#   Column      Non-Null Count  Dtype
---  -
0   MonthYear    7658 non-null   object
1   Time index   7658 non-null   float64
```

```

2   Country          7658 non-null   object
3   StoreID          7658 non-null   float64
4   City             7658 non-null   object
5   Dept_ID          7658 non-null   float64
6   Dept. Name       7658 non-null   object
7   HoursOwn         7658 non-null   object
8   HoursLease       7658 non-null   float64
9   Sales units      7658 non-null   float64
10  Turnover         7658 non-null   float64
11  Customer         7658 non-null   float64
12  Area (m2)        7658 non-null   object
13  Opening hours    7658 non-null   object
dtypes: float64(7), object(7)
memory usage: 837.7+ KB

```

```
[ ]: df1.columns
```

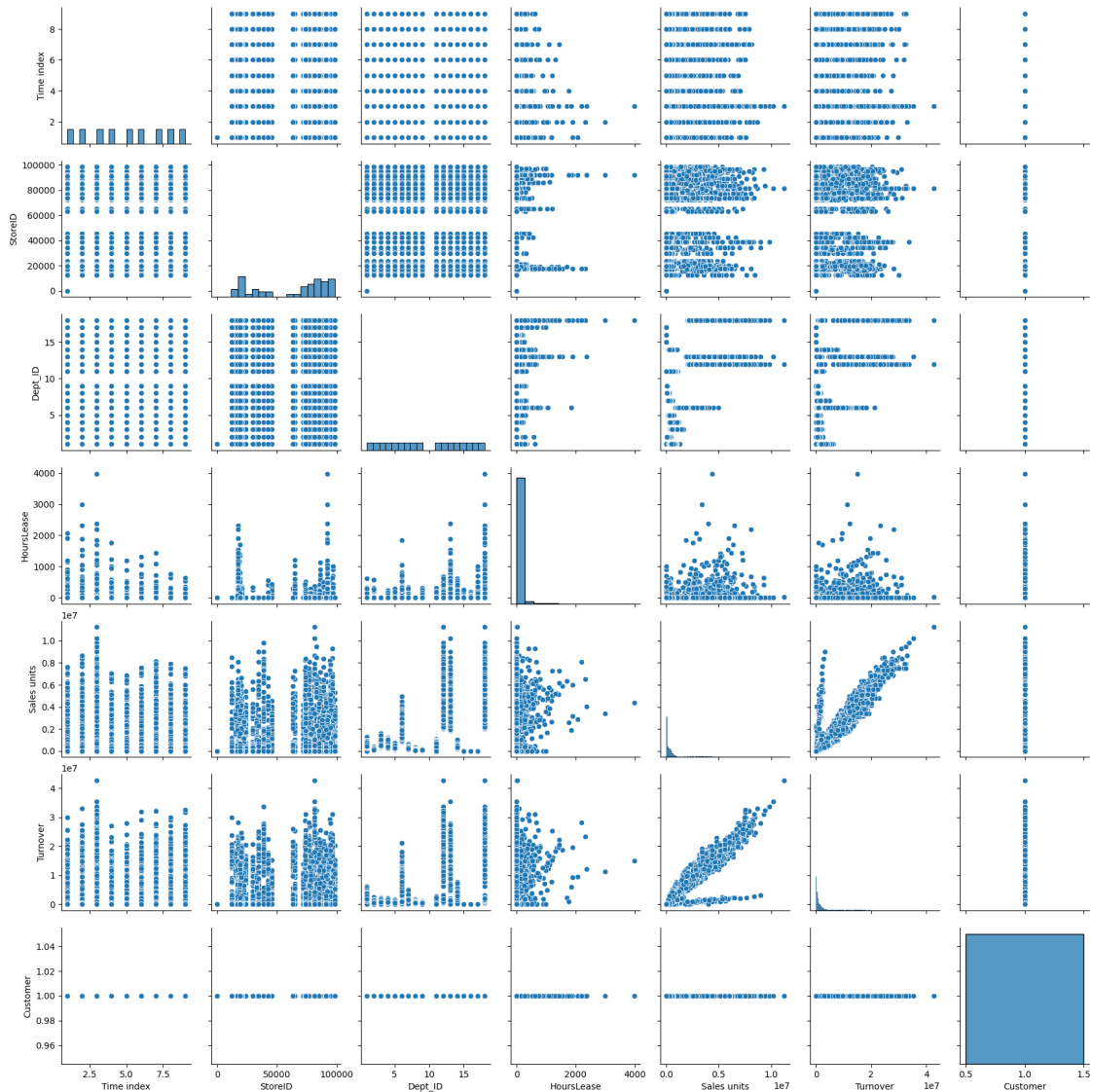
```
[ ]: Index(['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
          'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
          'Customer', 'Area (m2)', 'Opening hours'],
          dtype='object')
```

```
[ ]: df1=df1[[ 'Time index', 'StoreID', 'Dept_ID', 'HoursLease', 'Sales units',
               ↪ 'Turnover', 'Customer']]
```

2 EDA AND VISUALIZATION

```
[ ]: sns.pairplot(df1)
```

```
[ ]: <seaborn.axisgrid.PairGrid at 0x79ccd5d49540>
```



```
[ ]: sns.distplot(df1['Turnover'])
```

<ipython-input-13-098ec87212e1>:1: UserWarning:

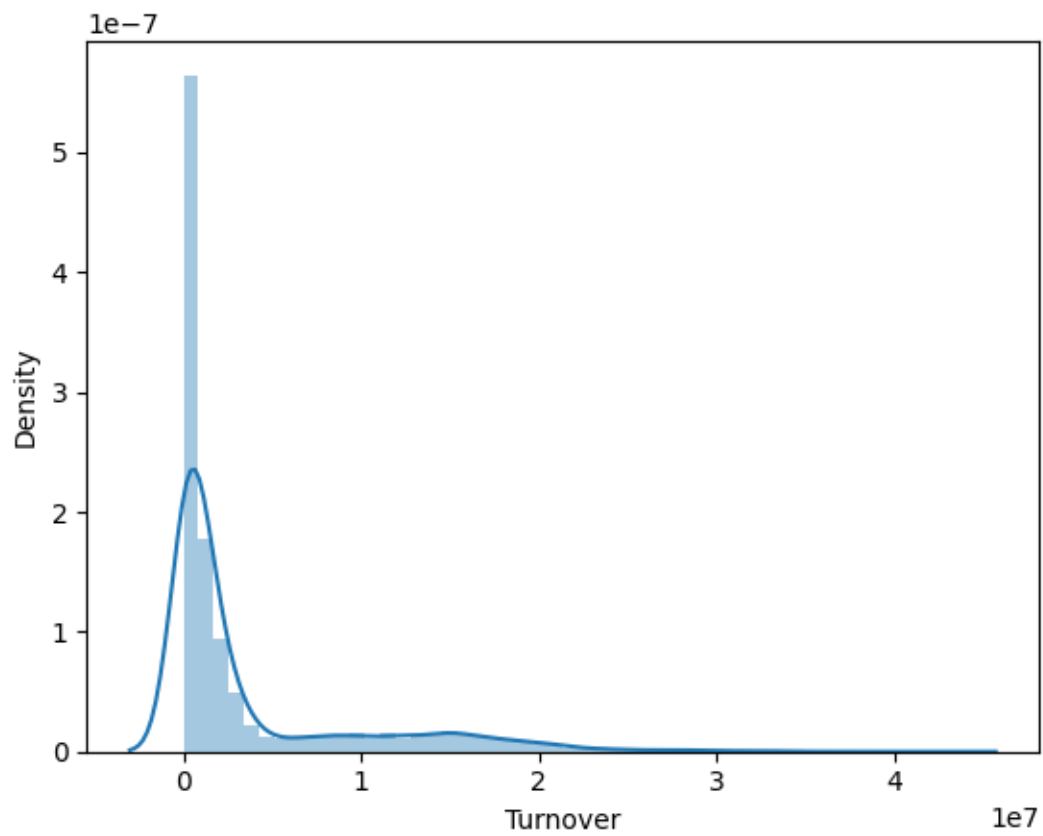
`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see <https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

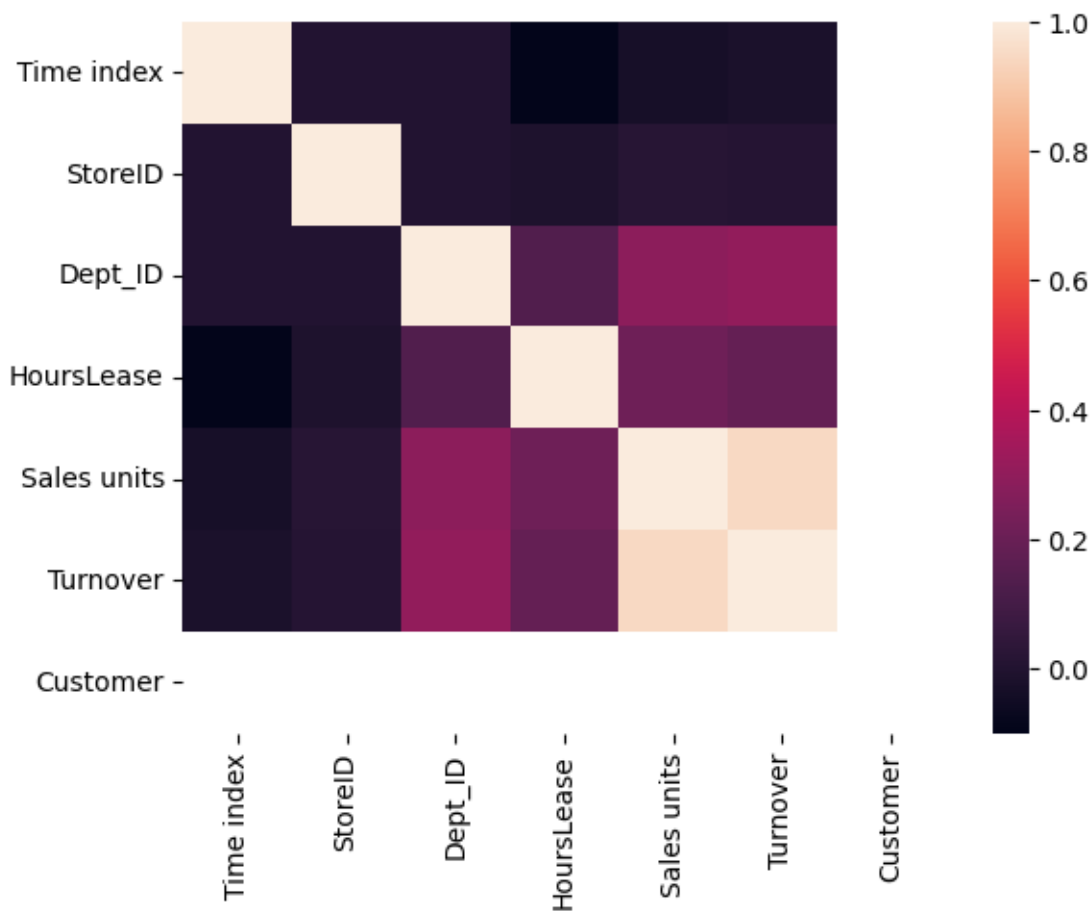
```
sns.distplot(df1['Turnover'])
```

```
[ ]: <Axes: xlabel='Turnover', ylabel='Density'>
```



```
[ ]: sns.heatmap(df1.corr())
```

```
[ ]: <Axes: >
```

3 TO TRAIN THE MODEL AND MODEL BUILDING

```
[ ]: df1
```

```
[ ]:
      Time index  StoreID  Dept_ID  HoursLease  Sales units  Turnover \
0             1.0   88253.0        1.0         0.0   398560.0  1226244.0
1             1.0   88253.0        2.0         0.0    82725.0   387810.0
2             1.0   88253.0        3.0         0.0   438400.0  654657.0
3             1.0   88253.0        4.0         0.0   309425.0  499434.0
4             1.0   88253.0        5.0         0.0  165515.0   329397.0
...           ...      ...      ...      ...      ...      ...
7653           9.0   29650.0       12.0         0.0  3886530.0 14538825.0
7654           9.0   29650.0       16.0         0.0     245.0         0.0
7655           9.0   29650.0       11.0         0.0         0.0         0.0
7656           9.0   29650.0       17.0         0.0     245.0         0.0
7657           9.0   29650.0       18.0         0.0  3886530.0 15056214.0
```

	Customer
0	1.0
1	1.0
2	1.0
3	1.0
4	1.0
...	...
7653	1.0
7654	1.0
7655	1.0
7656	1.0
7657	1.0

[7658 rows x 7 columns]

```
[ ]: x=df1[['Time index', 'StoreID', 'Dept_ID', 'HoursLease', 'Sales_
units','Customer']]
y=df1['Turnover']
```

```
[ ]: 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)
```

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

```
[ ]: LinearRegression()
```

```
[ ]: lr.intercept_
```

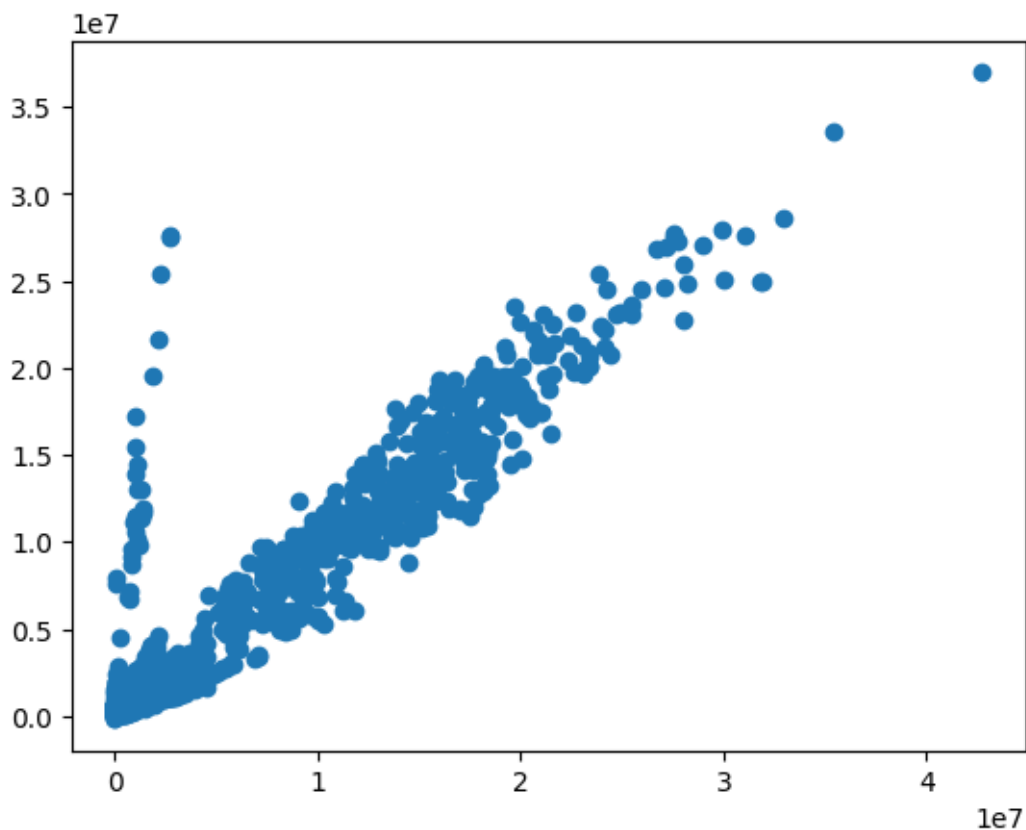
```
[ ]: -224556.94968064642
```

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

```
[ ]:
Time index    21372.090585
StoreID       -0.472052
Dept_ID       39627.736645
HoursLease    -1008.129840
Sales units     3.263712
Customer       0.000000
```

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

```
[ ]: <matplotlib.collections.PathCollection at 0x79cccc4bd090>
```



4 ACCURACY

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

```
[ ]: 0.9037219711584523
```

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

```
[ ]: 0.8967712986141503
```

```
[ ]: from sklearn.linear_model import Ridge,Lasso
      rr=Ridge(alpha=10)
      rr.fit(x_train,y_train)
```

```
[ ]: Ridge(alpha=10)
```

```
[ ]: rr.score(x_train,y_train)
```

```
[ ]: 0.8967712986018467
```

```
[ ]: rr.score(x_test,y_test)
```

```
[ ]: 0.9037219863948696
```

```
[ ]: la=Lasso(alpha=10)  
     la.fit(x_train,y_train)
```

```
[ ]: Lasso(alpha=10)
```

```
[ ]: la.score(x_train,y_train)
```

```
[ ]: 0.8967712986136293
```

```
[ ]: la.score(x_test,y_test)
```

```
[ ]: 0.9037219707380171
```