

D4

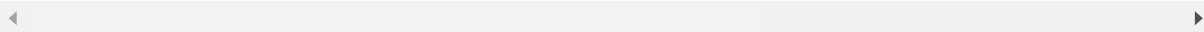
```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [4]: df=pd.read_csv(r"C:\Users\user\Downloads\6_Salesworkload1.csv")
df
```

Out[4]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursLe
0	10.2016	1.0	United Kingdom	88253.0	London (I)	1.0	Dry	3184.764	
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen	1582.941	
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other	47.205	
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	
...
7653	06.2017	9.0	Sweden	29650.0	Gothenburg	12.0	Checkout	6322.323	
7654	06.2017	9.0	Sweden	29650.0	Gothenburg	16.0	Customer Services	4270.479	
7655	06.2017	9.0	Sweden	29650.0	Gothenburg	11.0	Delivery	0	
7656	06.2017	9.0	Sweden	29650.0	Gothenburg	17.0	others	2224.929	
7657	06.2017	9.0	Sweden	29650.0	Gothenburg	18.0	all	39652.2	

7658 rows × 14 columns



In [5]: `df.head(10)`

Out[5]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursLease	
0	10.2016	1.0	United Kingdom	88253.0	London (I)	1.0	Dry	3184.764	0.0	3
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen	1582.941	0.0	
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other	47.205	0.0	4
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	0.0	3
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	0.0	1
5	10.2016	1.0	United Kingdom	88253.0	London (I)	6.0	Meat	8270.316	0.0	17
6	10.2016	1.0	United Kingdom	88253.0	London (I)	13.0	Food	16468.251	0.0	31
7	10.2016	1.0	United Kingdom	88253.0	London (I)	7.0	Clothing	4698.471	0.0	2
8	10.2016	1.0	United Kingdom	88253.0	London (I)	8.0	Household	1183.272	0.0	
9	10.2016	1.0	United Kingdom	88253.0	London (I)	9.0	Hardware	2029.815	0.0	

In [6]: `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
```

```
In [10]: dff=df.drop("Customer",axis=1)
```

```
In [12]: dft=dff.dropna()
```

```
In [13]: dft.isnull().sum()
```

```
Out[13]: MonthYear      0
Time index      0
Country         0
StoreID         0
City           0
Dept_ID        0
Dept. Name      0
HoursOwn        0
HoursLease      0
Sales units     0
Turnover        0
Area (m2)       0
Opening hours   0
dtype: int64
```

```
In [14]: dft.describe()
```

```
Out[14]:
```

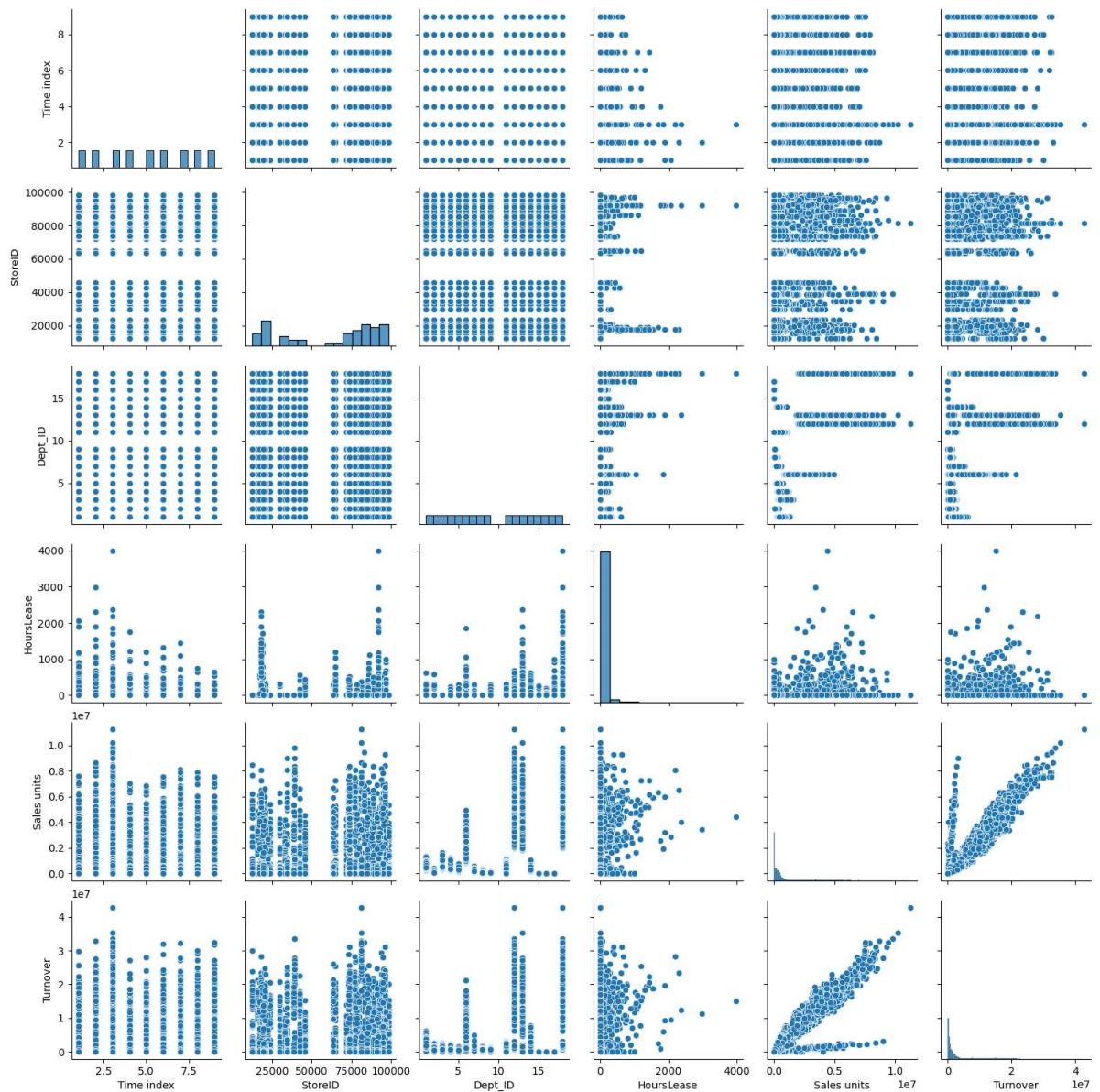
	Time index	StoreID	Dept_ID	HoursLease	Sales units	Turnover
count	7650.000000	7650.000000	7650.000000	7650.000000	7.650000e+03	7.650000e+03
mean	5.000000	61995.220000	9.470588	22.036078	1.076471e+06	3.721393e+06
std	2.582158	29924.581631	5.337429	133.299513	1.728113e+06	6.003380e+06
min	1.000000	12227.000000	1.000000	0.000000	0.000000e+00	0.000000e+00
25%	3.000000	29650.000000	5.000000	0.000000	5.457125e+04	2.726798e+05
50%	5.000000	75400.500000	9.000000	0.000000	2.932300e+05	9.319575e+05
75%	7.000000	87703.000000	14.000000	0.000000	9.175075e+05	3.264432e+06
max	9.000000	98422.000000	18.000000	3984.000000	1.124296e+07	4.271739e+07

```
In [15]: dft.columns
```

```
Out[15]: Index(['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
               'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
               'Area (m2)', 'Opening hours'],
              dtype='object')
```

```
In [16]: sns.pairplot(dft)
```

```
Out[16]: <seaborn.axisgrid.PairGrid at 0x25afb9fd690>
```



```
In [19]: sns.distplot(dft["Dept_ID"])
```

C:\Users\user\AppData\Local\Temp\ipykernel_1572\3941466768.py:1: UserWarning:

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

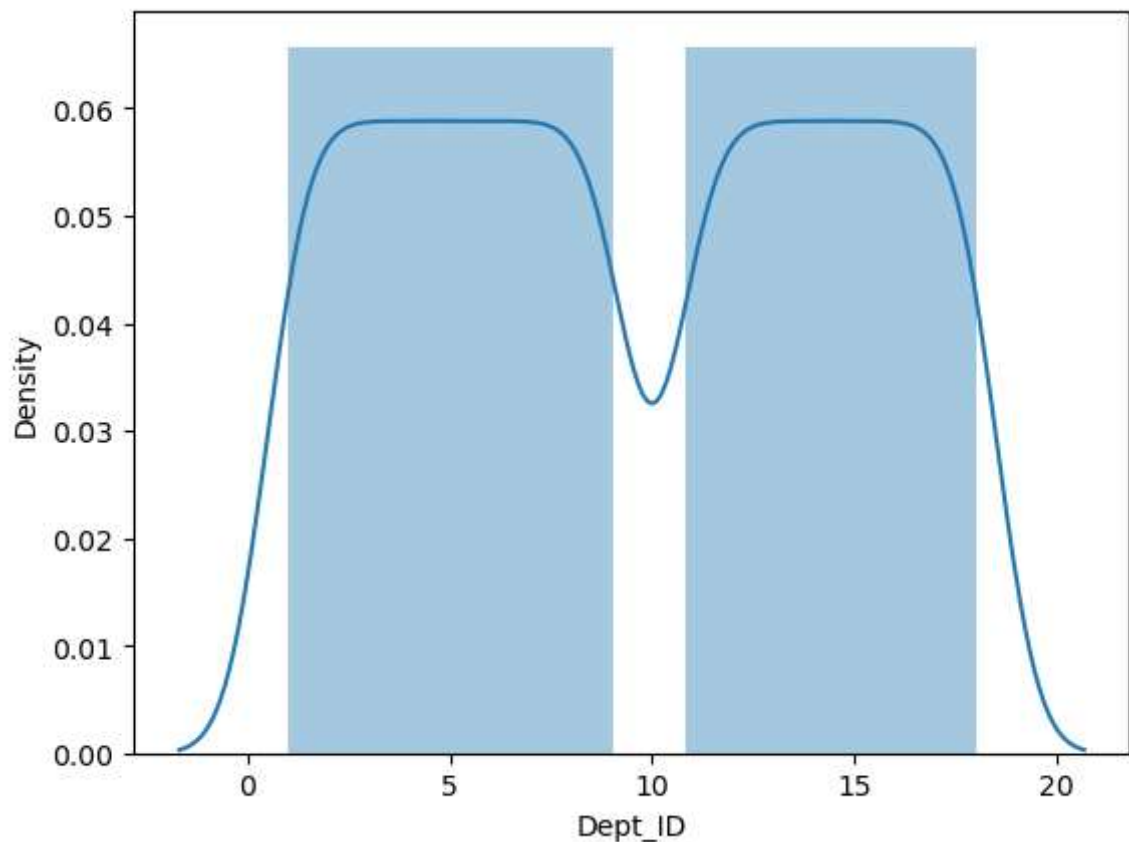
Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see

<https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751> (<https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>)

```
sns.distplot(dft["Dept_ID"])
```

Out[19]: <Axes: xlabel='Dept_ID', ylabel='Density'>

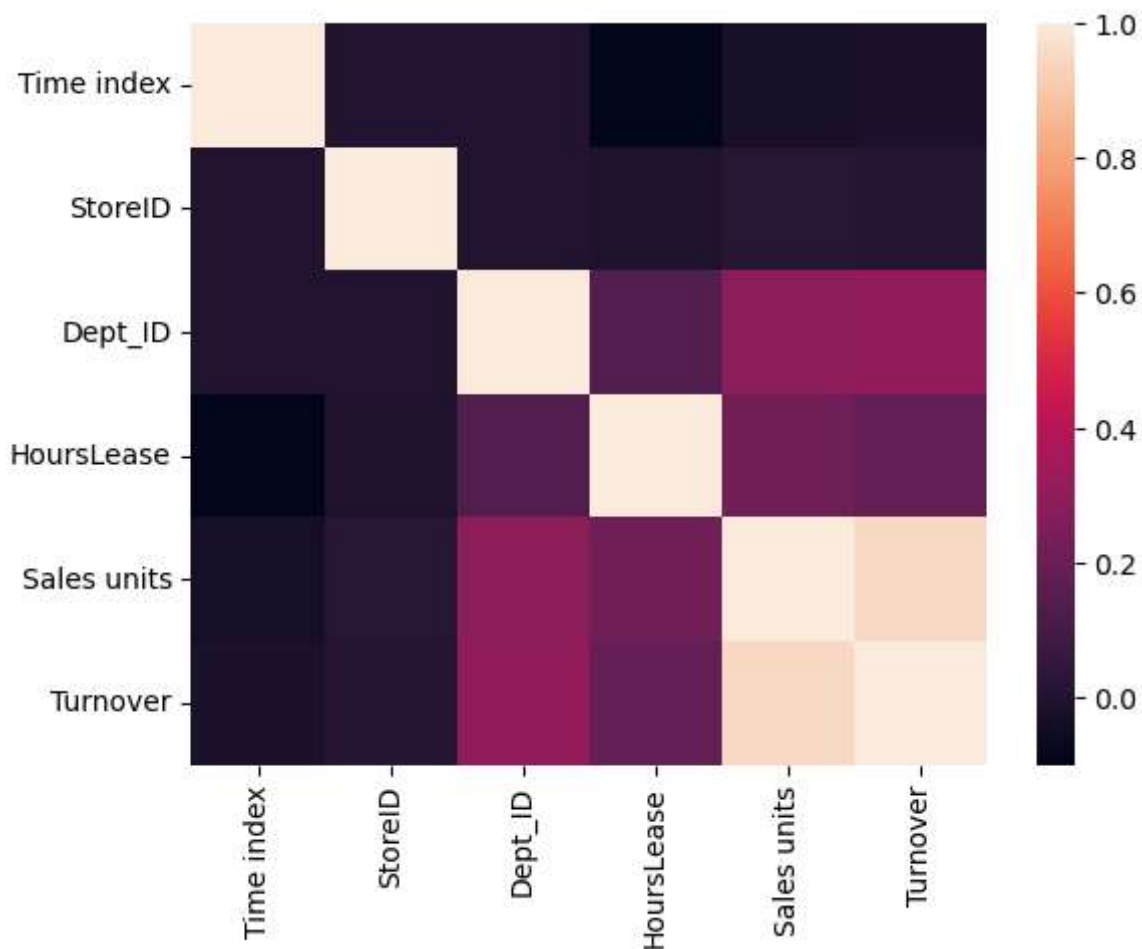


```
In [21]: df1=df1[['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',  
                'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',  
                'Area (m2)', 'Opening hours']]
```

In [22]: `sns.heatmap(df1.corr())`

C:\Users\user\AppData\Local\Temp\ipykernel_1572\781785195.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
`sns.heatmap(df1.corr())`

Out[22]: <Axes: >



In [23]: `x=df1[['Time index', 'StoreID', 'HoursLease', 'Sales units', 'Turnover']]`
`y=df1['Dept_ID']`

In [24]: `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 [25]: `from sklearn.linear_model import LinearRegression`
`lr=LinearRegression()`
`lr.fit(x_train,y_train)`

Out[25]: `LinearRegression`
`LinearRegression()`

```
In [26]: print(lr.intercept_)
```

8.336804095058563

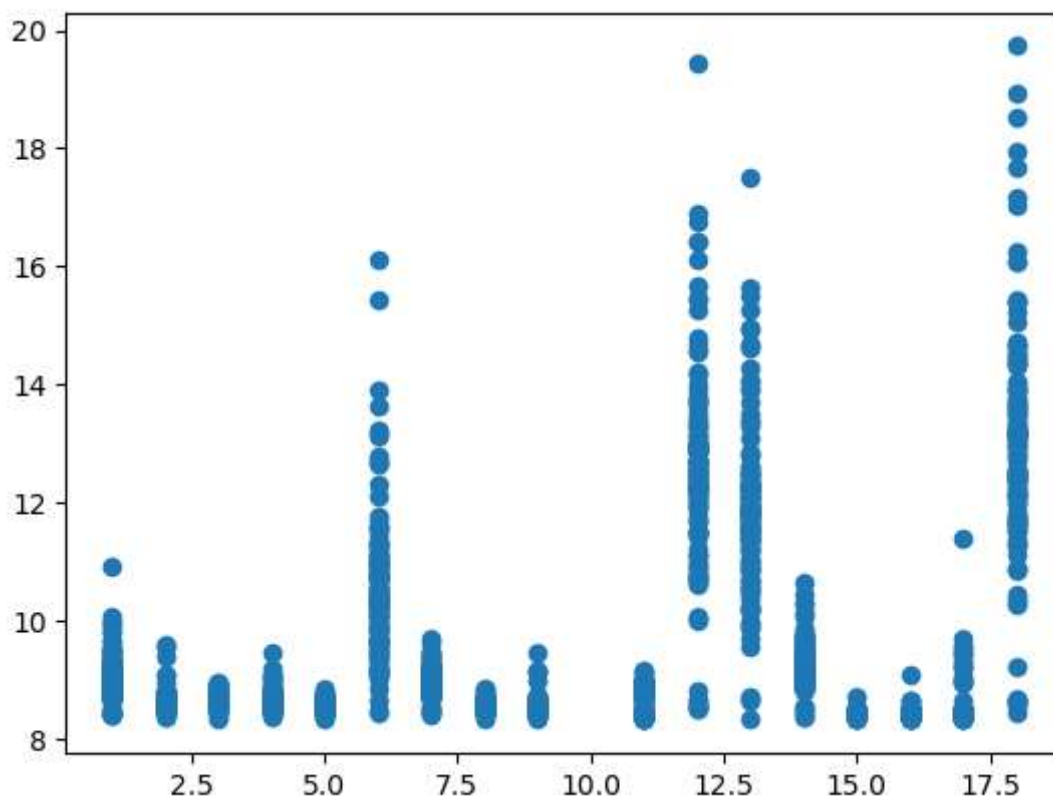
```
In [27]: coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])  
coeff
```

Out[27]:

	Co-efficient
Time index	1.708390e-02
StoreID	3.798626e-07
HoursLease	3.338348e-03
Sales units	-4.758073e-08
Turnover	2.703195e-07

```
In [28]: prediction=lr.predict(x_test)  
plt.scatter(y_test,prediction)
```

Out[28]: <matplotlib.collections.PathCollection at 0x25a824d1fd0>



```
In [29]: print(lr.score(x_test,y_test))
```

0.08660675097239212

In []: