Linear Regression-Salesworkload

In [4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [5]:
```
df = pd.read_csv("C:\\Users\\santh\Downloads\\6_Salesworkload1 - 6_Salesworklo
df = df[['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
       'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover', 'Are
df
```

Out[5]:

	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	6.2017	9.0	Sweden	29650.0	Gothenburg	12.0	Checkout	6322.323	
7654	6.2017	9.0	Sweden	29650.0	Gothenburg	16.0	Customer Services	4270.479	
7655	6.2017	9.0	Sweden	29650.0	Gothenburg	11.0	Delivery	0	
7656	6.2017	9.0	Sweden	29650.0	Gothenburg	17.0	others	2224.929	
7657	6.2017	9.0	Sweden	29650.0	Gothenburg	18.0	all	39652.2	

7650 rows × 13 columns

In [6]:
```
df.head()
```

Out[6]:

	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

Data cleaning and pre processing

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7650 entries, 0 to 7657
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   MonthYear      7650 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  Area (m2)      7650 non-null   object 
 12  Opening hours  7650 non-null   object 
dtypes: float64(6), object(7)
memory usage: 836.7+ KB
```

In [8]:
```
df.describe()
```

Out[8]:

	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

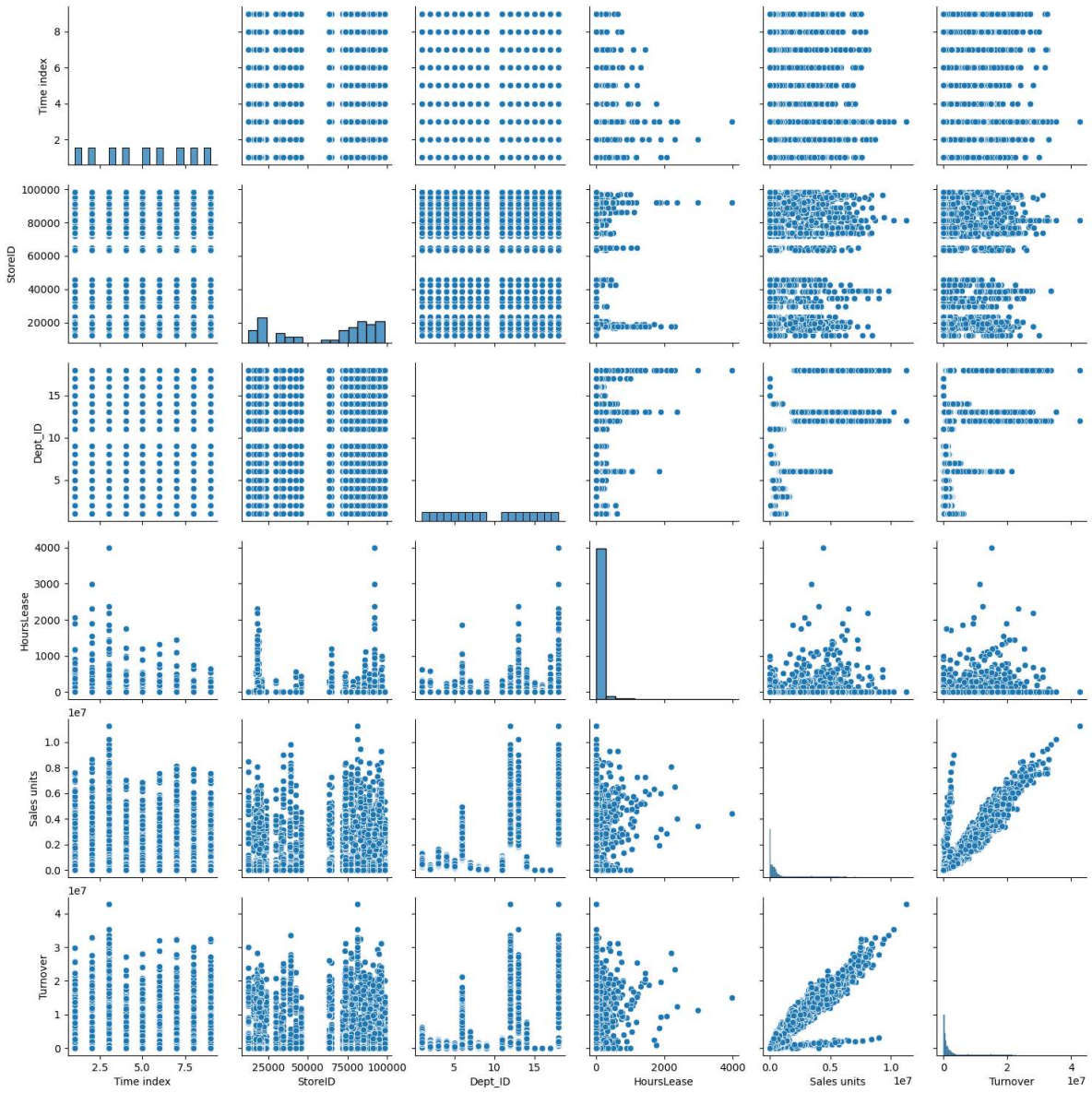
In [9]:
```
df.columns
```

Out[9]:
```
Index(['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
       'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
       'Area (m2)', 'Opening hours'],
      dtype='object')
```

EDA and VISUALIZATION

In [10]:
```
sns.pairplot(df)
```

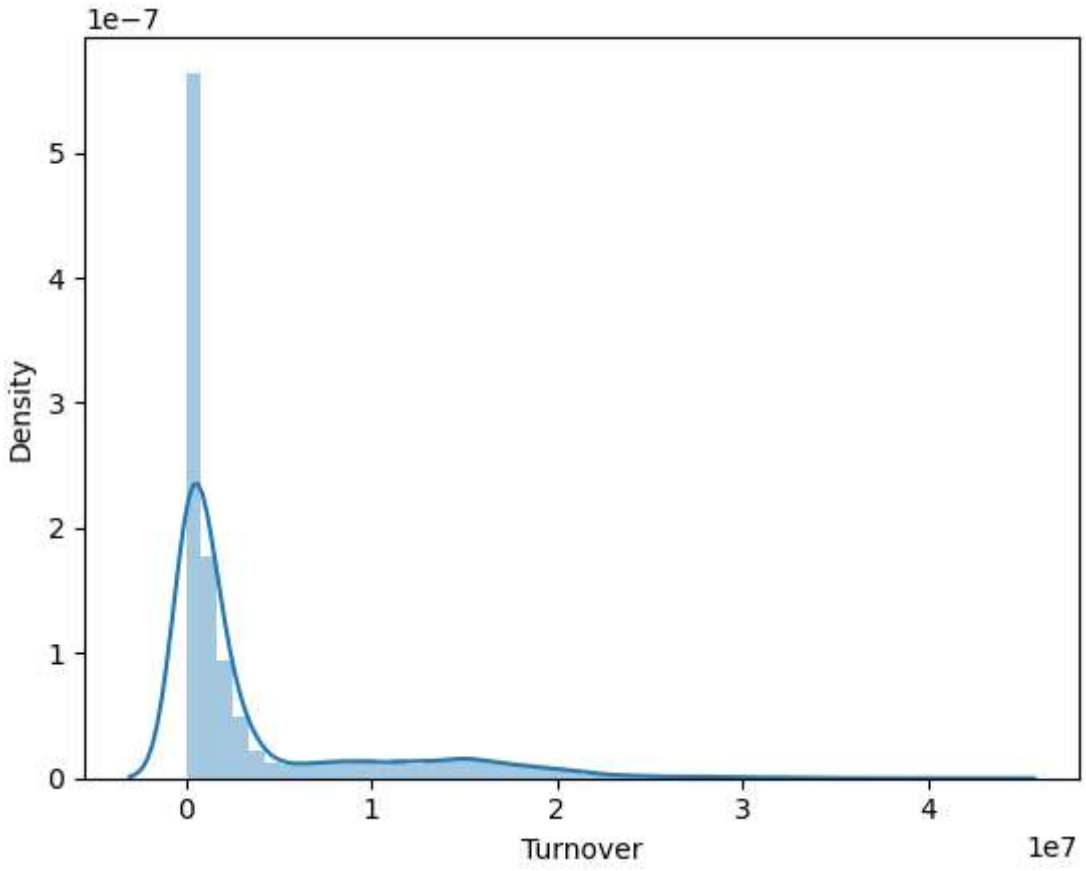
Out[10]:
```
<seaborn.axisgrid.PairGrid at 0x1adec2cbe80>
```



In [11]:
```
sns.distplot(df["Turnover"])
```

```
C:\Users\santh\AppData\Local\Temp\ipykernel_14036\255617326.py:1: UserWarnin
g:

`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://gis
t.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(df["Turnover"])
```

Out[11]: <Axes: xlabel='Turnover', ylabel='Density'>

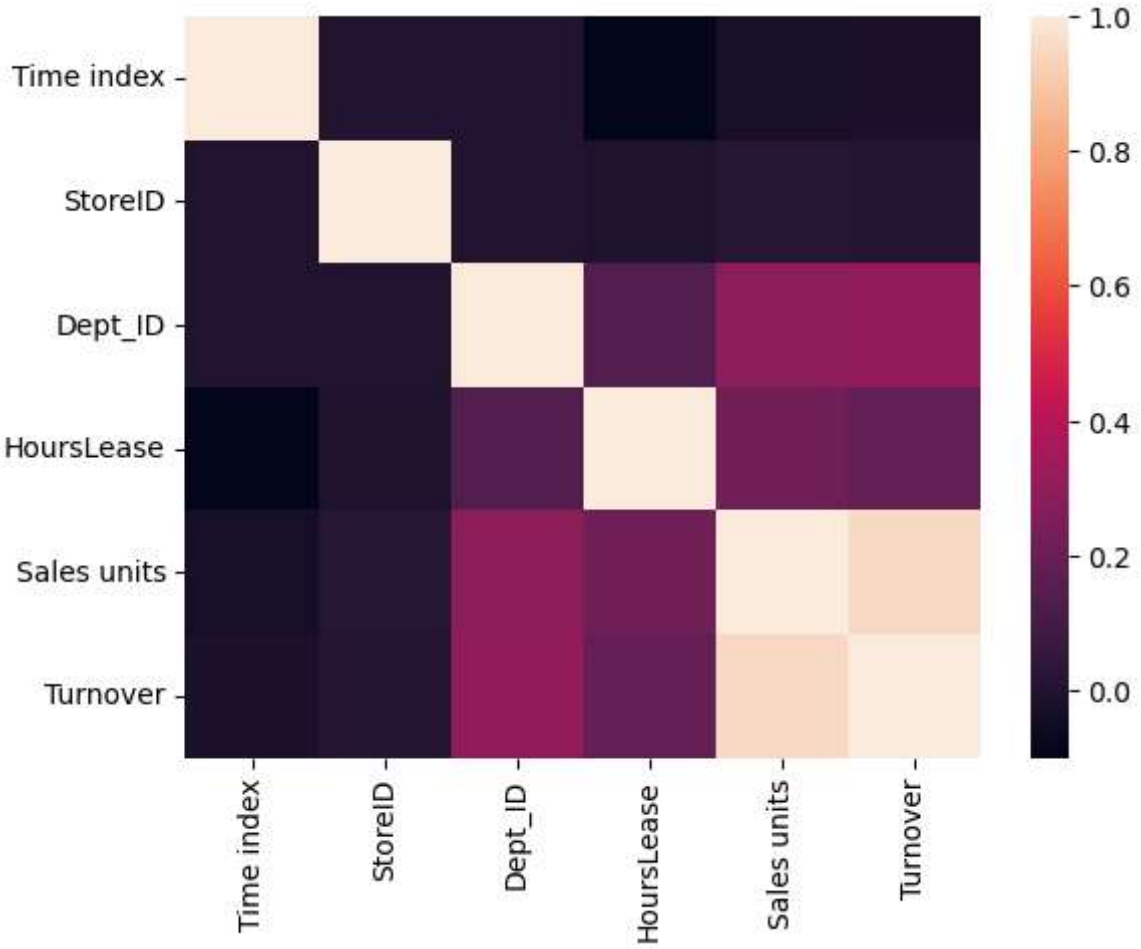


In [12]:
```
df1 = df[['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
       'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
       'Area (m2)', 'Opening hours']]
```

In [13]:
```
sns.heatmap(df1.corr())
```

```
C:\Users\santh\AppData\Local\Temp\ipykernel_14036\781785195.py:1: FutureWarni
ng: The default value of numeric_only in DataFrame.corr is deprecated. In a f
uture 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[13]: <Axes: >



In [14]:
```
df1.fillna(1)
```

Out[14]:

	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	6.2017	9.0	Sweden	29650.0	Gothenburg	12.0	Checkout	6322.323	
7654	6.2017	9.0	Sweden	29650.0	Gothenburg	16.0	Customer Services	4270.479	
7655	6.2017	9.0	Sweden	29650.0	Gothenburg	11.0	Delivery	0	
7656	6.2017	9.0	Sweden	29650.0	Gothenburg	17.0	others	2224.929	
7657	6.2017	9.0	Sweden	29650.0	Gothenburg	18.0	all	39652.2	

7650 rows × 13 columns

In [15]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7650 entries, 0 to 7657
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   MonthYear      7650 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  Area (m2)      7650 non-null   object 
 12  Opening hours  7650 non-null   object 
dtypes: float64(6), object(7)
memory usage: 836.7+ KB
```

In [16]:
```
x = df1[[ 'Time index', 'StoreID', 'Dept_ID',
        'HoursLease', 'Sales units']]
y = df1['Turnover']
```

split the data into training and test data

In [17]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3)
```

In [18]:
```
lr = LinearRegression()
lr.fit(x_train, y_train)
```

Out[18]:
```
LinearRegression
LinearRegression()
```

In [19]:
```
lr.intercept_
```

Out[19]:
```
-266638.52476934064
```

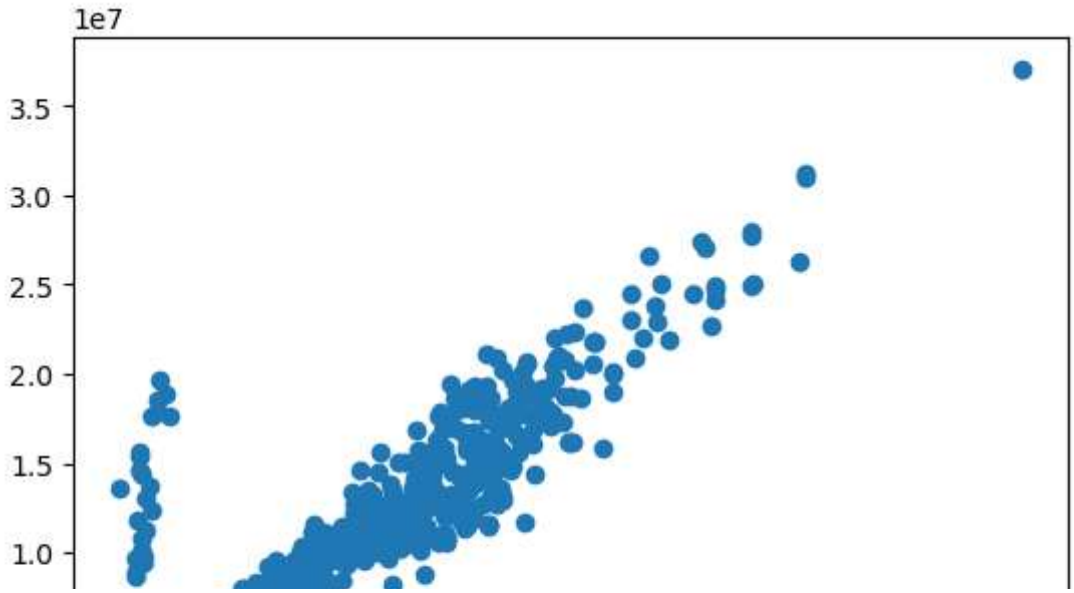
In [20]:
```
coeff = pd.DataFrame(lr.coef_, x.columns, columns =['Co-efficient'])
coeff
```

Out[20]:

	Co-efficient
Time index	28850.342432
StoreID	0.056697
Dept_ID	37261.732046
HoursLease	-1129.855491
Sales units	3.250276

In [21]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test, prediction)
```

Out[21]:
```
<matplotlib.collections.PathCollection at 0x1ade7b09cf0>
```



In [23]:
```
lr.score(x_test,y_test)
```

Out[23]:
```
0.9082622480070864
```

In [ ]:

In [ ]: