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

In [2]:
```
data=pd.read_csv(r"C:\Users\user\Downloads\6_Salesworkload1 - 6_Salesworkload1
data
```

Out[2]:

	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	

7658 rows × 14 columns

In [3]:

```
df=data.head(100)
df
```

Out[3]:

	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
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
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	0.0
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	0.0
...	...	...	...	...	...	...	...	...	...
95	10.2016	1.0	United Kingdom	18808.0	London (II)	14.0	Non Food	7817.148	0.0
96	10.2016	1.0	United Kingdom	18808.0	London (II)	15.0	Admin	5110.728	0.0
97	10.2016	1.0	United Kingdom	18808.0	London (II)	12.0	Checkout	6209.031	0.0
98	10.2016	1.0	United Kingdom	18808.0	London (II)	16.0	Customer Services	3115.53	0.0
99	10.2016	1.0	United Kingdom	18808.0	London (II)	11.0	Delivery	7209.777	246.0

100 rows × 14 columns

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   MonthYear      100 non-null    object 
 1   Time index     100 non-null    float64
 2   Country        100 non-null    object 
 3   StoreID        100 non-null    float64
 4   City           100 non-null    object 
 5   Dept_ID        100 non-null    float64
 6   Dept. Name     100 non-null    object 
 7   HoursOwn       100 non-null    object 
 8   HoursLease     100 non-null    float64
 9   Sales units    100 non-null    float64
 10  Turnover       100 non-null    float64
 11  Customer       0 non-null      float64
 12  Area (m2)      100 non-null    object 
 13  Opening hours  100 non-null    object 
dtypes: float64(7), object(7)
memory usage: 11.1+ KB
```

In [5]:
```
df.describe()
```

Out[5]:

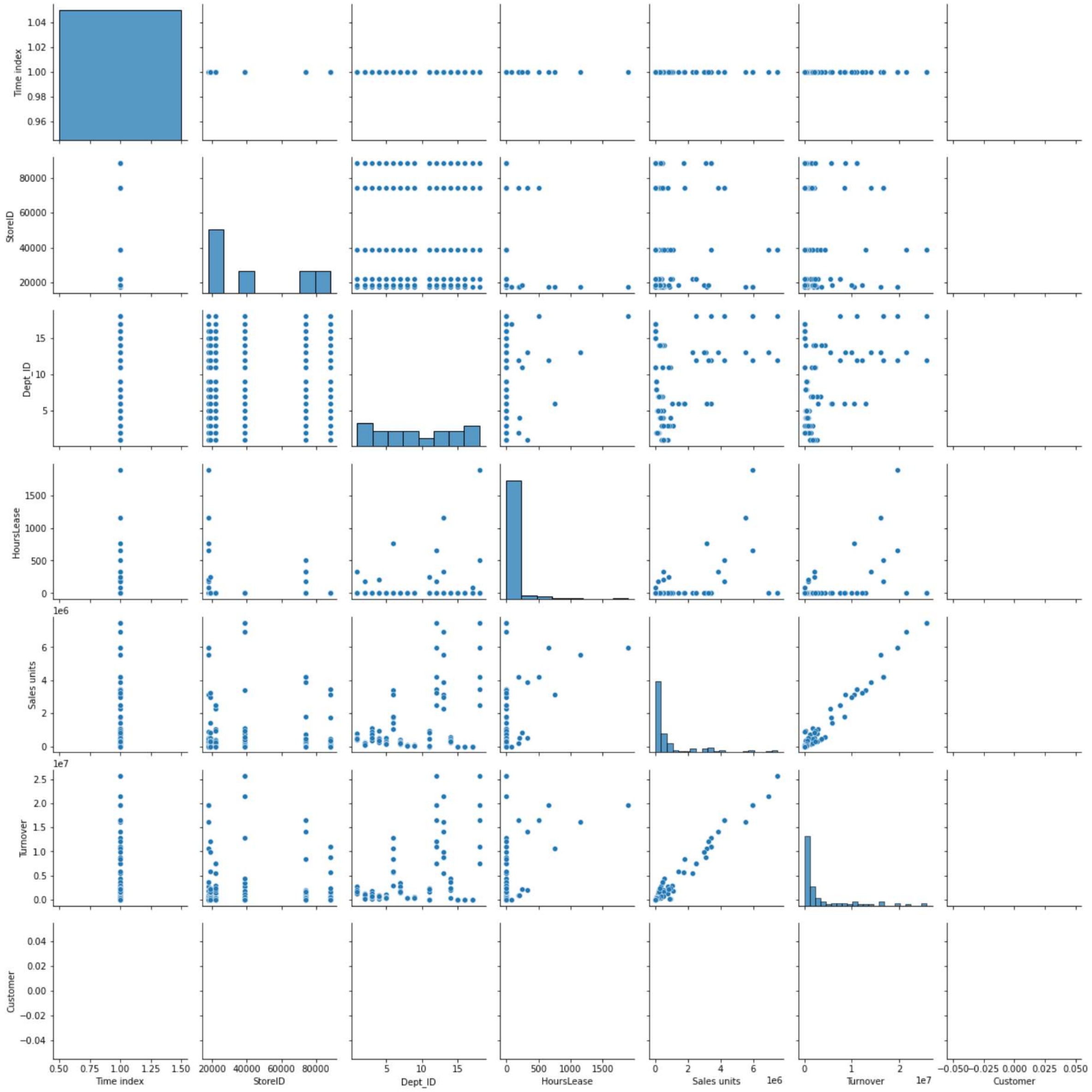
	Time index	StoreID	Dept_ID	HoursLease	Sales units	Turnover	Customer
count	100.0	100.000000	100.000000	100.000000	1.000000e+02	1.000000e+02	0.0
mean	1.0	43781.340000	9.310000	65.340000	1.063110e+06	3.590811e+06	NaN
std	0.0	28146.505061	5.292829	249.349222	1.769242e+06	5.968009e+06	NaN
min	1.0	17647.000000	1.000000	0.000000	0.000000e+00	0.000000e+00	NaN
25%	1.0	18808.000000	5.000000	0.000000	5.300125e+04	2.702460e+05	NaN
50%	1.0	38976.000000	9.000000	0.000000	3.072850e+05	8.339250e+05	NaN
75%	1.0	73949.000000	14.000000	0.000000	9.195138e+05	2.966010e+06	NaN
max	1.0	88253.000000	18.000000	1896.000000	7.476680e+06	2.571973e+07	NaN

In [6]:
```
df.columns
```

Out[6]:
```
Index(['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
       'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
       'Customer', 'Area (m2)', 'Opening hours'],
      dtype='object')
```

In [7]:
```
sns.pairplot(df)
```

Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x1967c60fd00>
```



In [8]:
```
da=df[['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
       'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
       'Customer', 'Area (m2)', 'Opening hours']]
da
```

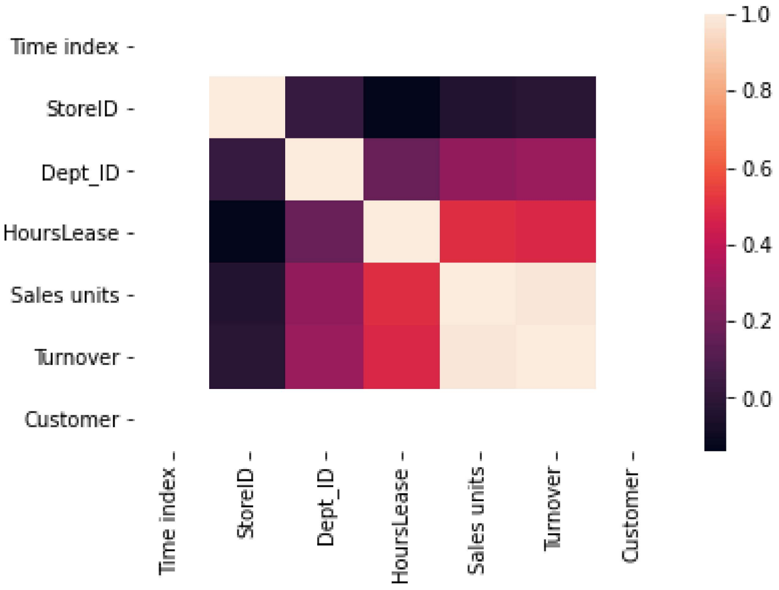
Out[8]:

	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
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
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	0.0
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	0.0
...	...	...	...	...	...	...	...	...	...
95	10.2016	1.0	United Kingdom	18808.0	London (II)	14.0	Non Food	7817.148	0.0
96	10.2016	1.0	United Kingdom	18808.0	London (II)	15.0	Admin	5110.728	0.0
97	10.2016	1.0	United Kingdom	18808.0	London (II)	12.0	Checkout	6209.031	0.0
98	10.2016	1.0	United Kingdom	18808.0	London (II)	16.0	Customer Services	3115.53	0.0
99	10.2016	1.0	United Kingdom	18808.0	London (II)	11.0	Delivery	7209.777	246.0

100 rows × 14 columns

```
In [9]: sns.heatmap(da.corr())

Out[9]: <AxesSubplot:>
```



```
In [10]: x=da[['MonthYear', 'Time index', 'Dept_ID', 'HoursOwn', 'HoursLease', 'Sales u
         y=da['Turnover']
```

```
In [11]: 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 [12]: from sklearn.linear_model import LinearRegression

         lr=LinearRegression()
         lr.fit(x_train,y_train)

Out[12]: LinearRegression()
```

```
In [13]: print(lr.intercept_)

         -259254.98998099798
```

In [14]:

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

Out[14]:

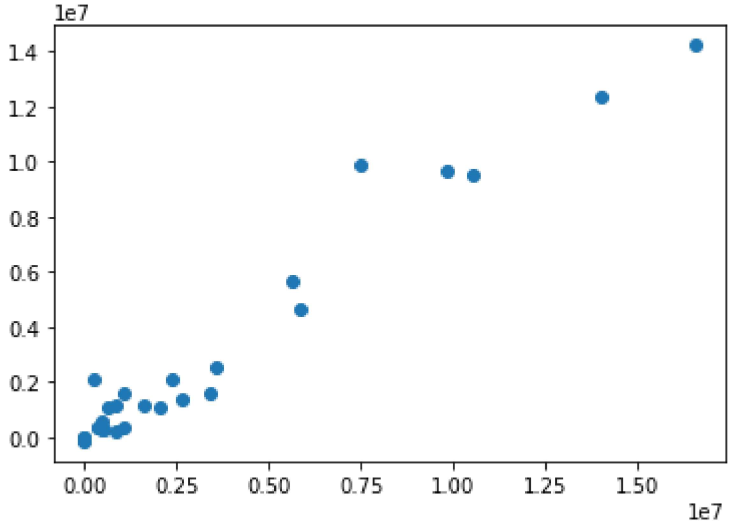
	Co-efficient
MonthYear	0.000000e+00
Time index	1.528216e-07
Dept_ID	5.265083e+03
HoursOwn	2.913973e+01
HoursLease	-7.318791e+02
Sales units	2.910219e+00
Area (m2)	5.768828e+01

In [15]:

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

Out[15]:

```
<matplotlib.collections.PathCollection at 0x19601ca3430>
```



In [16]:

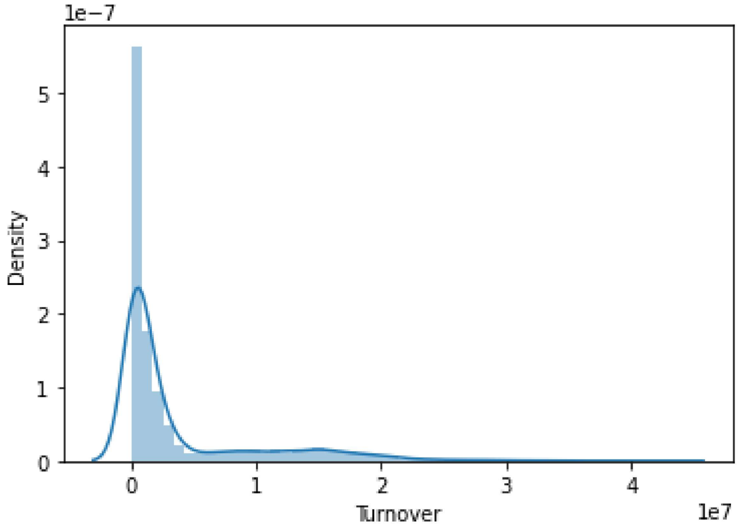
```
print(lr.score(x_test,y_test))
```

```
0.9493351758128525
```

In [17]:
```
sns.distplot(data['Turnover'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

Out[17]:
```
<AxesSubplot:xlabel='Turnover', ylabel='Density'>
```



In [18]:
```
print(lr.score(x_train,y_train))
```

```
0.9818934125426865
```

In [19]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [20]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[20]:
```
Ridge(alpha=10)
```

In [21]:
```
rr.score(x_test,y_test)
```

Out[21]:
```
0.9493374086792508
```

In [22]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[22]:
```
Lasso(alpha=10)
```

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

Out[23]:
```
0.9493352067627612
```

In [ ]: