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

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

Out[2]:

	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	
...	...	...	...	...	...	...	...	...	...	
7653	6.2017	9.0	Sweden	29650.0	Gothenburg	12.0	Checkout	6322.323	0.0	3
7654	6.2017	9.0	Sweden	29650.0	Gothenburg	16.0	Customer Services	4270.479	0.0	
7655	6.2017	9.0	Sweden	29650.0	Gothenburg	11.0	Delivery	0	0.0	
7656	6.2017	9.0	Sweden	29650.0	Gothenburg	17.0	others	2224.929	0.0	
7657	6.2017	9.0	Sweden	29650.0	Gothenburg	18.0	all	39652.2	0.0	3

7658 rows × 14 columns

In [3]:
```
d.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 [4]:
```
d.head()
```

Out[4]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursLease	Sales units
0	10.2016	1.0	United Kingdom	88253.0	London (I)	1.0	Dry	3184.764	0.0	398560.0
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen	1582.941	0.0	82725.0
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other	47.205	0.0	438400.0
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	0.0	309425.0
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	0.0	165515.0

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

Out[5]:

	Time index	StoreID	Dept_ID	HoursLease	Sales units	Turnover	Customer
count	7650.000000	7650.000000	7650.000000	7650.000000	7.650000e+03	7.650000e+03	0.0
mean	5.000000	61995.220000	9.470588	22.036078	1.076471e+06	3.721393e+06	NaN
std	2.582158	29924.581631	5.337429	133.299513	1.728113e+06	6.003380e+06	NaN
min	1.000000	12227.000000	1.000000	0.000000	0.000000e+00	0.000000e+00	NaN
25%	3.000000	29650.000000	5.000000	0.000000	5.457125e+04	2.726798e+05	NaN
50%	5.000000	75400.500000	9.000000	0.000000	2.932300e+05	9.319575e+05	NaN
75%	7.000000	87703.000000	14.000000	0.000000	9.175075e+05	3.264432e+06	NaN
max	9.000000	98422.000000	18.000000	3984.000000	1.124296e+07	4.271739e+07	NaN

In [6]:
```
d.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]:
```
d.index
```

Out[7]: RangeIndex(start=0, stop=7658, step=1)

In [8]:
```
d=d.head(100)
d
```

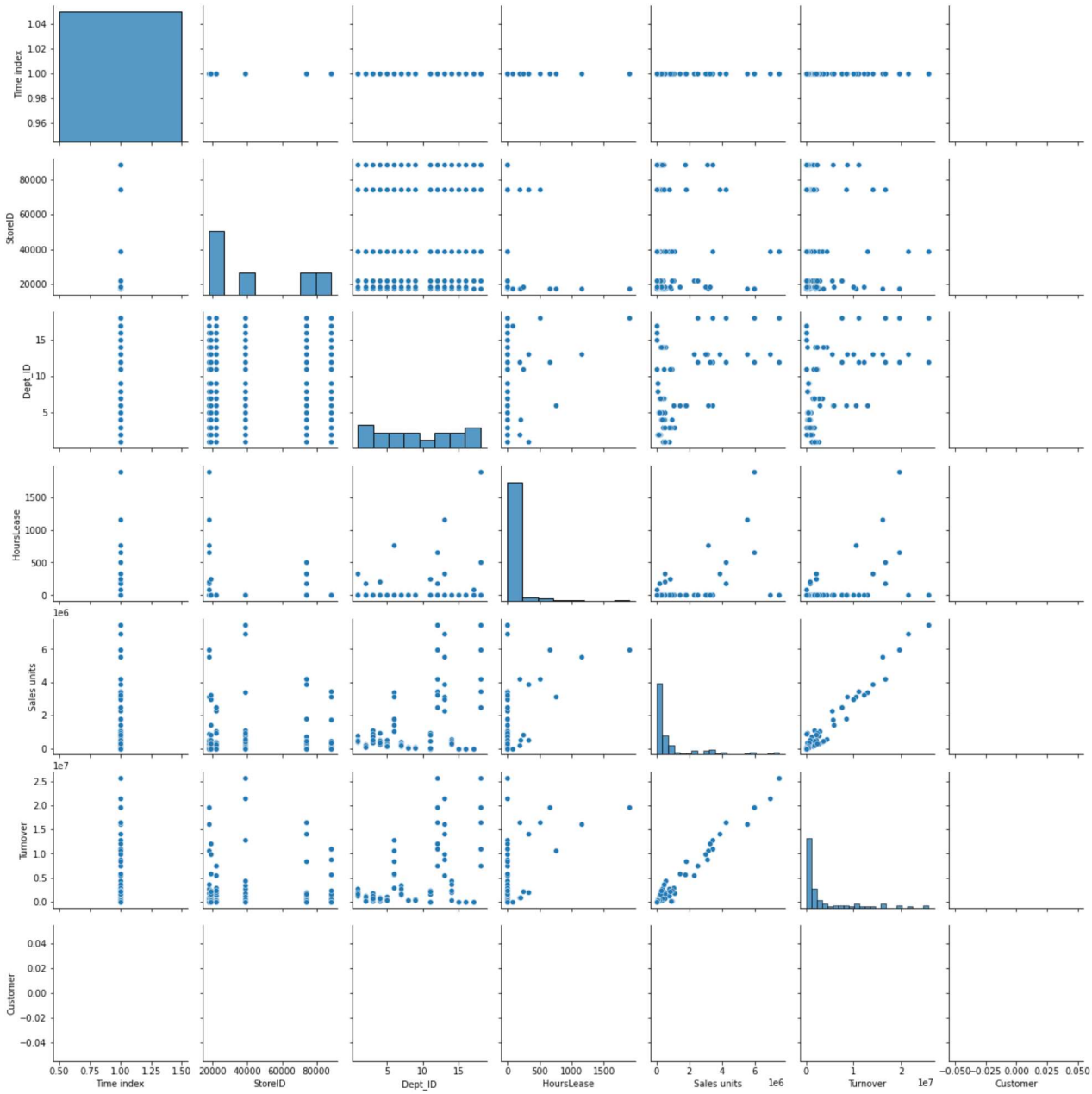
Out[8]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursLease	Sal uni
0	10.2016	1.0	United Kingdom	88253.0	London (I)	1.0	Dry	3184.764	0.0	39856(
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen	1582.941	0.0	82725
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other	47.205	0.0	43840(
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	0.0	30942[
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	0.0	16551[
...	...	...	...	...	...	...	...	...	...	
95	10.2016	1.0	United Kingdom	18808.0	London (II)	14.0	Non Food	7817.148	0.0	30150(
96	10.2016	1.0	United Kingdom	18808.0	London (II)	15.0	Admin	5110.728	0.0	2[
97	10.2016	1.0	United Kingdom	18808.0	London (II)	12.0	Checkout	6209.031	0.0	326224(
98	10.2016	1.0	United Kingdom	18808.0	London (II)	16.0	Customer Services	3115.53	0.0	2[
99	10.2016	1.0	United Kingdom	18808.0	London (II)	11.0	Delivery	7209.777	246.0	84361[

100 rows × 14 columns

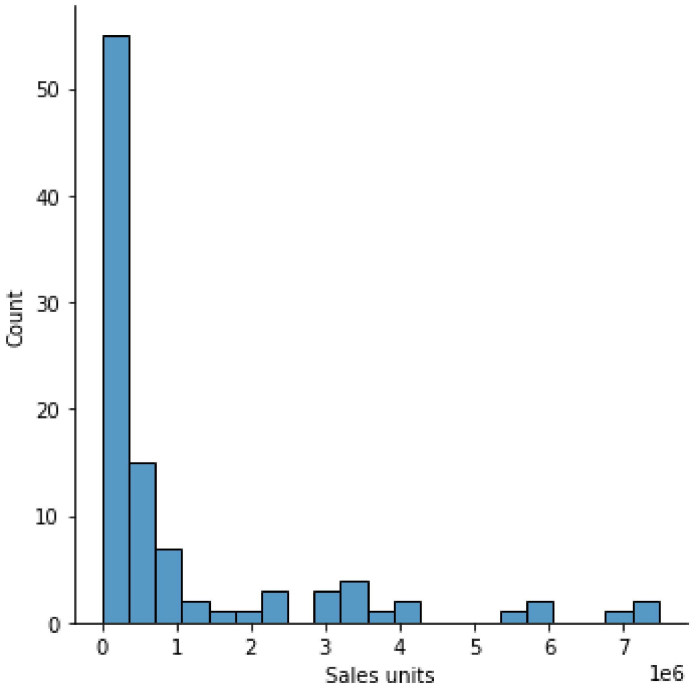
In [9]:
```
sns.pairplot(d)
```

Out[9]: <seaborn.axisgrid.PairGrid at 0x22c1991e670>



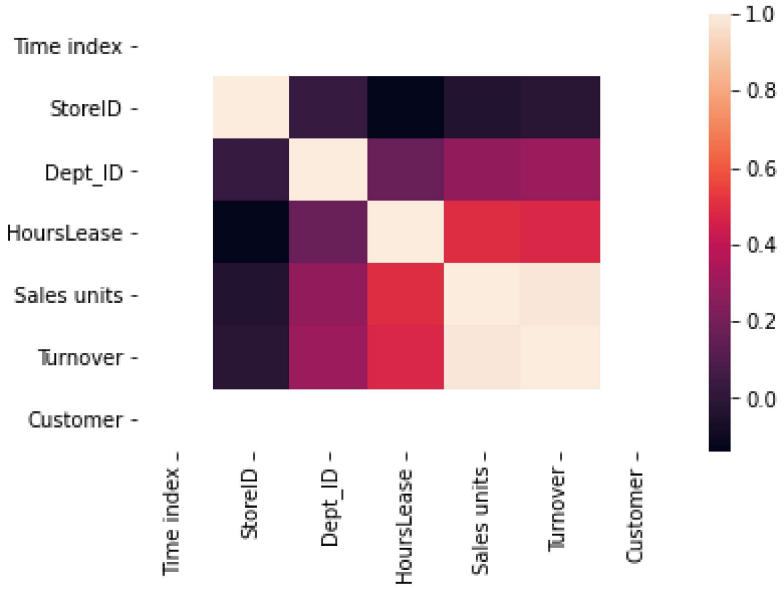
In [10]:
```
sns.displot(d['Sales units'])
```

Out[10]:
```
<seaborn.axisgrid.FacetGrid at 0x22c1c099820>
```



In [11]:
```
d1=d[['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
       'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
       'Customer', 'Area (m2)', 'Opening hours']]
sns.heatmap(d1.corr())
```

Out[11]: <AxesSubplot:>



In [12]:
```
x=d1[['MonthYear', 'Time index','StoreID','Dept_ID','HoursOwn', 'HoursLease', 'Turnover
y=d1['Sales units']
```

In [13]:
```
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 [14]:
```
from sklearn.linear_model import LinearRegression
```

In [15]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[15]:
```
LinearRegression()
```

In [16]:
```
print(lr.intercept_)
```

```
106150.93808447546
```

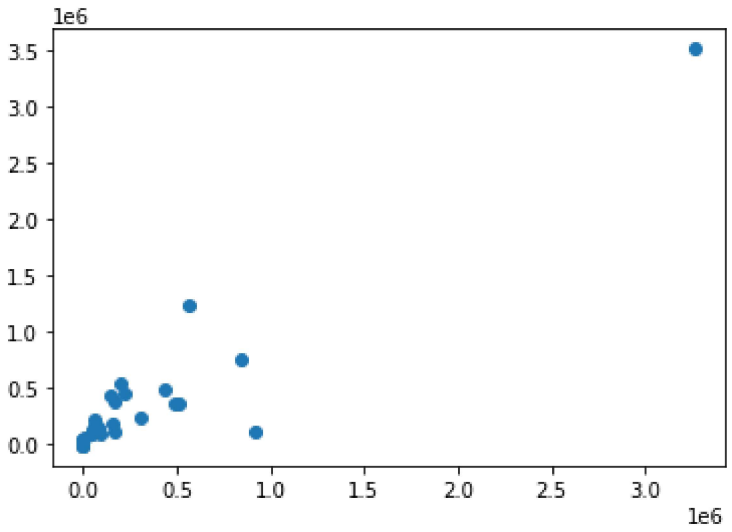
In [17]:
```
coeff =pd.DataFrame(lr.coef_,x.columns,columns=["Co-efficient"])
coeff
```

Out[17]:

	Co-efficient
MonthYear	0.000000e+00
Time index	-3.034548e-07
StoreID	-9.741773e-01
Dept_ID	-3.101410e+03
HoursOwn	-3.912454e-02
HoursLease	1.591050e+02
Turnover	2.984325e-01
Area (m2)	-5.137880e+00

In [18]:
```
prediction =lr.predict(x_test)
py.scatter(y_test,prediction)
```

Out[18]:
```
<matplotlib.collections.PathCollection at 0x22c1e6b18b0>
```



In [19]:
```
print(lr.score(x_test,y_test))
```

```
0.8560899084782179
```

In [20]:
```
print(lr.score(x_train,y_train))
```

```
0.9764615700111092
```

In [21]:
```
from sklearn.linear_model import Ridge,Lasso
```

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

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

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

Out[23]:
```
0.8560817382584147
```

In [24]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[24]:
```
Lasso(alpha=10)
```

In [25]:
```
la.score(x_test,y_test)
```

Out[25]:
```
0.8560897190216925
```

In [26]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[26]:
```
ElasticNet()
```

In [27]:
```
print(en.coef_)
```

```
[-0.00000000e+00  0.00000000e+00 -9.73228911e-01 -3.02266523e+03
 -5.49420387e-02  1.59182524e+02  2.98461871e-01 -5.15636356e+00]
```

In [28]:
```
print(en.intercept_)
```

```
105535.9279539762
```

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

```
0.8560615377349597
```

In [30]:
```
from sklearn import metrics
```

In [33]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 135639.1438373063
```

In [34]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 52132479159.11114
```

In [35]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

```
Root Mean Squared Error: 228325.38001525617
```

In [ ]:
```

```