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

In [2]:
```python
df=pd.read_csv(r"C:\Users\user\Downloads\6_Salesworkload1.csv")
df.fillna(0,inplace=True)
df
```

Out[2]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursLease	Sa ui
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	8272
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
...	...	...	...	...	...	...	...	...	...	
7653	06.2017	9.0	Sweden	29650.0	Gothenburg	12.0	Checkout	6322.323	0.0	388653
7654	06.2017	9.0	Sweden	29650.0	Gothenburg	16.0	Customer Services	4270.479	0.0	24
7655	06.2017	9.0	Sweden	29650.0	Gothenburg	11.0	Delivery	0	0.0	
7656	06.2017	9.0	Sweden	29650.0	Gothenburg	17.0	others	2224.929	0.0	24
7657	06.2017	9.0	Sweden	29650.0	Gothenburg	18.0	all	39652.2	0.0	388653

7658 rows × 14 columns

In [3]:
```python
df.head()
```

Out[3]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursLease	Sales units	Tu
0	10.2016	1.0	United Kingdom	88253.0	London (I)	1.0	Dry	3184.764	0.0	398560.0	122
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen	1582.941	0.0	82725.0	38
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other	47.205	0.0	438400.0	65
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	0.0	309425.0	49
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	0.0	165515.0	32

In [4]:
```
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     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
```

In [5]:
```
import seaborn as sns
```

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

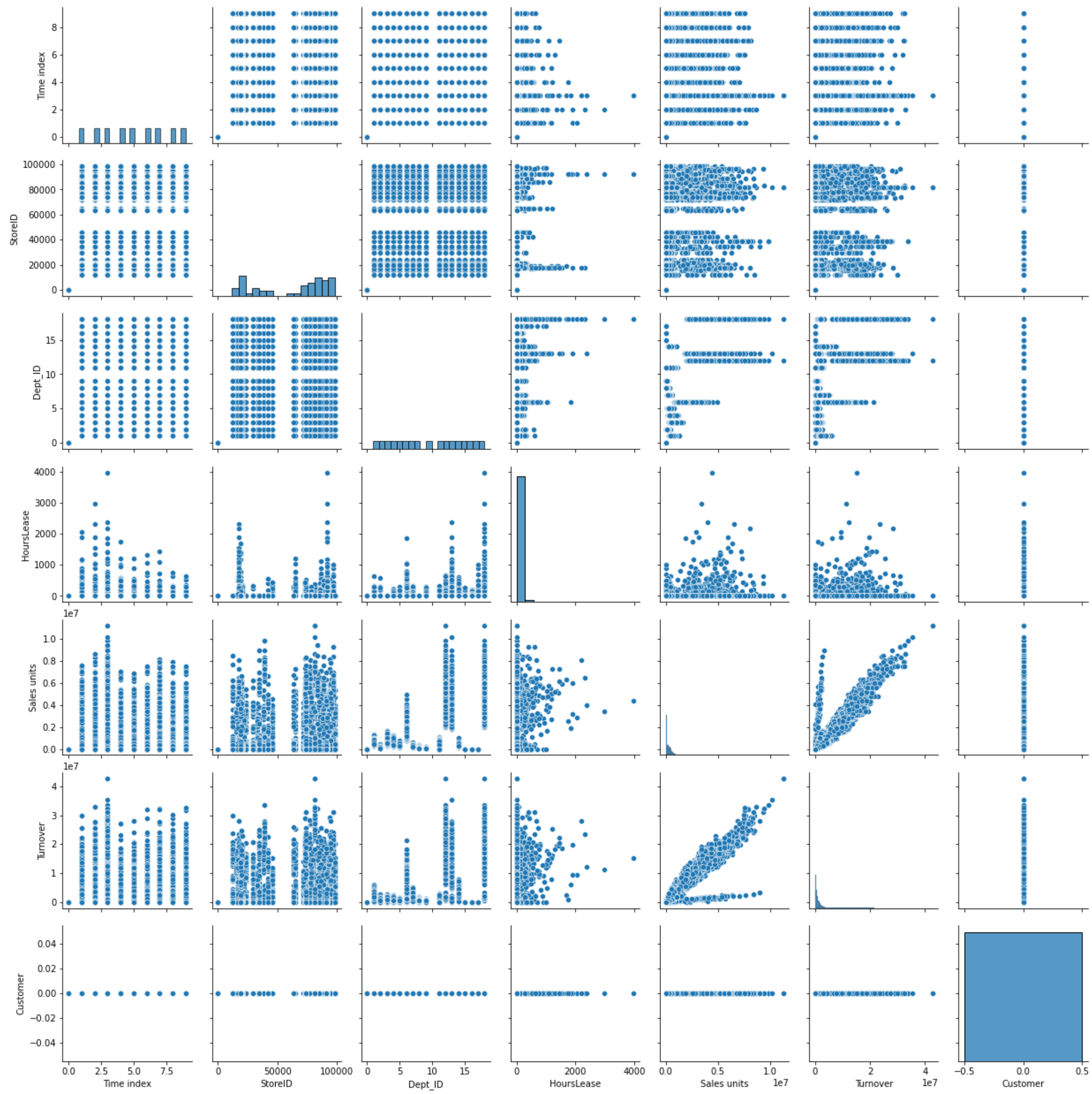
Out[6]:

	Time index	StoreID	Dept_ID	HoursLease	Sales units	Turnover	Customer
count	7658.000000	7658.000000	7658.000000	7658.000000	7.658000e+03	7.658000e+03	7658.0
mean	4.994777	61930.456124	9.460695	22.013058	1.075346e+06	3.717505e+06	0.0
std	2.585859	29975.929873	5.343407	133.231761	1.727560e+06	6.001448e+06	0.0
min	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000e+00	0.0
25%	3.000000	29650.000000	5.000000	0.000000	5.441375e+04	2.720558e+05	0.0
50%	5.000000	73949.000000	9.000000	0.000000	2.927625e+05	9.300810e+05	0.0
75%	7.000000	87703.000000	14.000000	0.000000	9.154812e+05	3.251488e+06	0.0
max	9.000000	98422.000000	18.000000	3984.000000	1.124296e+07	4.271739e+07	0.0

In [ ]:

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

Out[7]: <seaborn.axisgrid.PairGrid at 0x1921cc61be0>

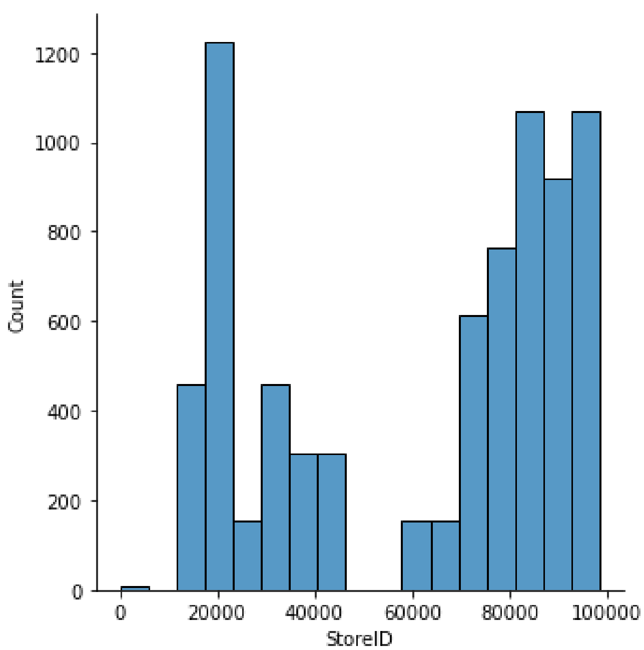


In [8]:
```
df1=df.drop(['Country'],axis=1)
df1
df1=df1.drop(df1.index[1537:])
df1.isna().sum()
```

Out[8]:
```
MonthYear        0
Time index       0
StoreID          0
City             0
Dept_ID          0
Dept. Name       0
HoursOwn         0
HoursLease       0
Sales units      0
Turnover         0
Customer         0
Area (m2)        0
Opening hours    0
dtype: int64
```

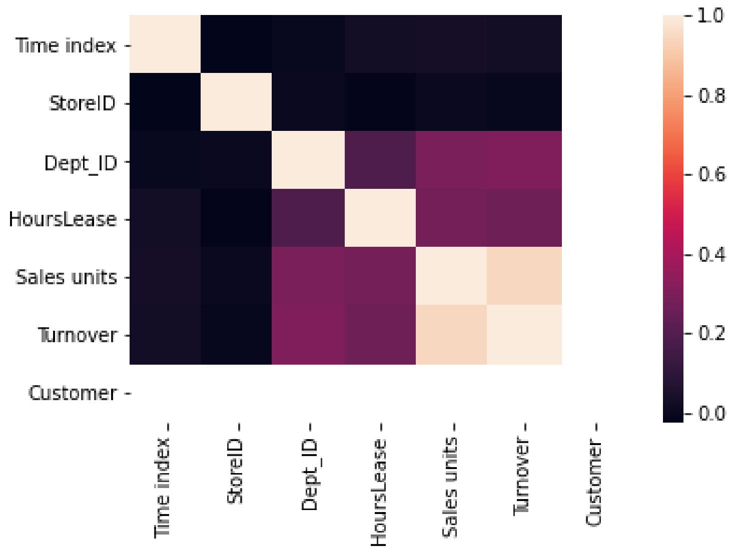
In [9]:
```
sns.displot(df['StoreID'])
```

Out[9]: <seaborn.axisgrid.FacetGrid at 0x19219e54610>



In [10]:
```
sns.heatmap(df1.corr())
```

Out[10]: <AxesSubplot:>



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

In [12]:
```
df1.isna().sum()
```

Out[12]:
```
MonthYear        0
Time index       0
StoreID          0
City             0
Dept_ID          0
Dept. Name       0
HoursOwn         0
HoursLease       0
Sales units      0
Turnover         0
Customer         0
Area (m2)        0
Opening hours    0
dtype: int64
```

In [13]:
```
y=df1['Turnover']
x=df1.drop(['Turnover','MonthYear','City','Opening hours','Dept. Name','Customer','Turn
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
print(x_train)
```

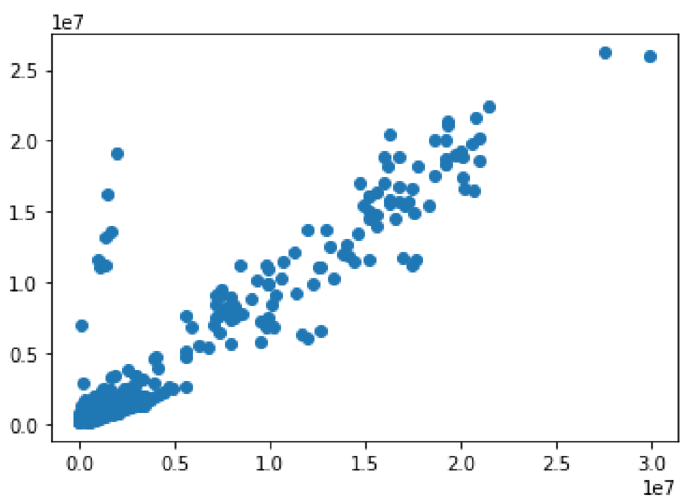
```
       Time index  StoreID  Dept_ID   HoursOwn  HoursLease  Sales units
1147          2.0  85696.0      7.0   5680.335        84.0     240170.0
572           1.0  12227.0     15.0   3974.661         0.0        165.0
1515          2.0  79785.0      2.0   2278.428         0.0      92455.0
1460          2.0  34378.0     11.0          0         0.0         10.0
1217          2.0  19000.0      9.0   1595.529         0.0      41060.0
...           ...      ...      ...        ...         ...          ...
662           1.0  98422.0     18.0  46276.635         0.0    3508675.0
783           1.0  73762.0      2.0   2426.337         0.0     182250.0
1246          2.0  20166.0      5.0   1992.051         0.0     243510.0
1242          2.0  20166.0      1.0   3449.112         0.0     557495.0
542           1.0  78325.0     17.0   2193.459         0.0        155.0

[1075 rows x 6 columns]
```

In [14]:
```
model=LinearRegression()
model.fit(x_train,y_train)
model.intercept_
```

Out[14]: 71487.15112321172

In [15]:
```
prediction=model.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[15]: <matplotlib.collections.PathCollection at 0x1922262a670>



In [16]:
```
model.score(x_test,y_test)
```

Out[16]: 0.8844082866440194

In [17]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [18]:
```python
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```
Out[18]:
```
Ridge(alpha=10)
```

In [19]:
```python
rr.score(x_test,y_test)
```
Out[19]:
```
0.8844074382998153
```

In [20]:
```python
la =Lasso(alpha=10)
la.fit(x_train,y_train)
```
Out[20]:
```
Lasso(alpha=10)
```

In [21]:
```python
la.score(x_test,y_test)
```
Out[21]:
```
0.884408249300056
```

In [22]:
```python
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```
Out[22]:
```
ElasticNet()
```

In [23]:
```python
print(en.coef_)
```
```
[-8.49670897e+03 -2.83657677e+00  2.80818729e+04  2.60727171e+01
  3.91533921e+01  3.07359204e+00]
```

In [24]:
```python
print(en.intercept_)
```
```
51211.900752046145
```

In [25]:
```python
print(en.predict(x_test))
```
```
 12081313.10615316    318344.55719734    161808.63620064 20386487.79747214
  1104261.62469585   1985757.29088112   1515331.45219829    154903.21553394
  1250542.62909192    546798.39424928    866228.38947379   1267149.35330232
  1049485.95428433  17454798.87082483  18277896.92037161    526152.89263141
   322212.53734578    298812.19160697   1101221.19750973    245088.25842468
   341348.9338572    1142780.75439107    335986.78491951    487380.15303624
  2018371.9873523     532543.62839248   1670797.07024807    314843.80322381
  1946713.52384728    526580.01513346   4691939.55389973    407220.35528643
   527200.7612785    7427458.75733612   1590263.56950927    736289.56119032
  1815862.82154166  18885560.24321736    383570.58251211    529899.46255776
   415955.72426126    314857.84317711    360335.27297      1483195.17038797
  2833594.396895     1772604.38529096    269436.8816465   15378673.112774
  8980402.98260365    422353.62165203    120944.20925279    142384.21800239
   171731.73735571   7451817.89593718    466892.72012039    336644.33983013
  6034595.82323731  11392712.30266854  16354156.15313049   1533643.13924037
   870754.86557243   5316510.32895393   8452937.38126709    277696.84394422
  1007142.9373982     292356.86817356    395279.80168817   1120197.39862718
   937535.45049221   1330982.36569508    246103.72548528   7537691.05049852
  3879704.8851028     400243.89950139]
```

In [26]:
```
print(en.score(x_test,y_test))
```

```
0.8844007933875692
```

In [27]:
```
from sklearn import metrics
```

In [28]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 1003905.7077692078
```

In [29]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 4032719243820.1797
```

In [30]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

```
Root Mean Squared Error: 2008163.1516936515
```