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

In [2]:
```
df = pd.read_csv(r"C:\Users\user\Downloads\6_Salesworkload1 (1).csv")
df
```

Out[2]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursLea
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 [3]:
```
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 [5]:
```
df1 = df[0:1000]
df1
```

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
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
...	...	...	...	...	...	...	...	...	...
995	11.2016	2.0	Poland	23623.0	Poznan	8.0	Household	1671.057	0.0
996	11.2016	2.0	Poland	23623.0	Poznan	9.0	Hardware	1516.854	0.0
997	11.2016	2.0	Poland	23623.0	Poznan	14.0	Non Food	5834.538	0.0
998	11.2016	2.0	Poland	23623.0	Poznan	15.0	Admin	3707.166	0.0
999	11.2016	2.0	Poland	23623.0	Poznan	12.0	Checkout	6312.882	0.0

1000 rows × 14 columns

In [7]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   MonthYear      1000 non-null   object 
 1   Time index     999 non-null    float64
 2   Country        999 non-null    object 
 3   StoreID        999 non-null    float64
 4   City           999 non-null    object 
 5   Dept_ID        999 non-null    float64
 6   Dept. Name     999 non-null    object 
 7   HoursOwn       999 non-null    object 
 8   HoursLease     999 non-null    float64
 9   Sales units    999 non-null    float64
 10  Turnover       999 non-null    float64
 11  Customer       0 non-null      float64
 12  Area (m2)      999 non-null    object 
 13  Opening hours  999 non-null    object 
dtypes: float64(7), object(7)
memory usage: 109.5+ KB
```

In [8]:
```
df1.describe()
```

Out[8]:

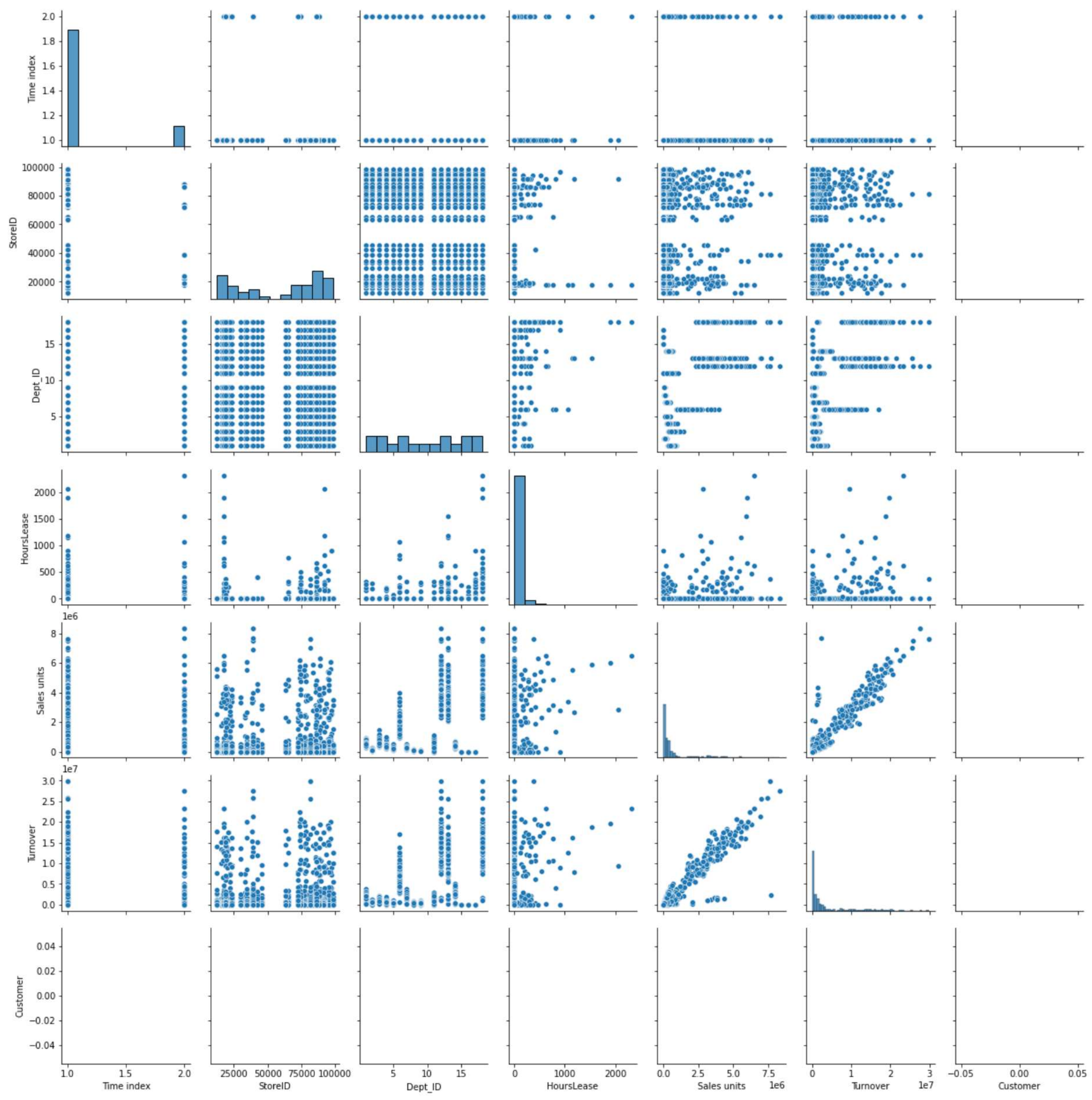
	Time index	StoreID	Dept_ID	HoursLease	Sales units	Turnover	Customer
count	999.000000	999.000000	999.000000	999.000000	9.990000e+02	9.990000e+02	0.0
mean	1.149149	60168.337337	9.446446	36.406406	1.049182e+06	3.549660e+06	NaN
std	0.356414	30094.482461	5.334022	170.339961	1.664266e+06	5.714799e+06	NaN
min	1.000000	12227.000000	1.000000	0.000000	0.000000e+00	0.000000e+00	NaN
25%	1.000000	23623.000000	5.000000	0.000000	5.602250e+04	2.414370e+05	NaN
50%	1.000000	73762.000000	9.000000	0.000000	3.022100e+05	8.376510e+05	NaN
75%	1.000000	86208.000000	14.000000	0.000000	9.207575e+05	3.180158e+06	NaN
max	2.000000	98422.000000	18.000000	2314.000000	8.351080e+06	2.988887e+07	NaN

In [9]:
```
df1.columns
```

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

In [10]:
```
sns.pairplot(df1)
```

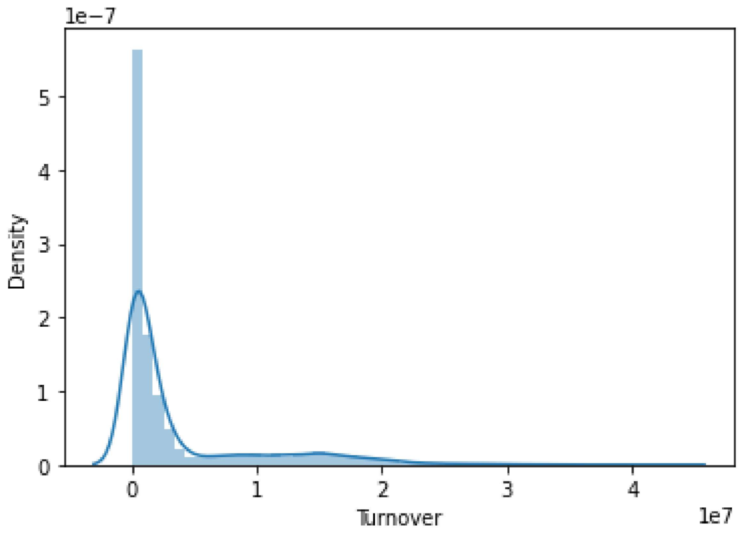
Out[10]:
```
<seaborn.axisgrid.PairGrid at 0x25489cbafd0>
```



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

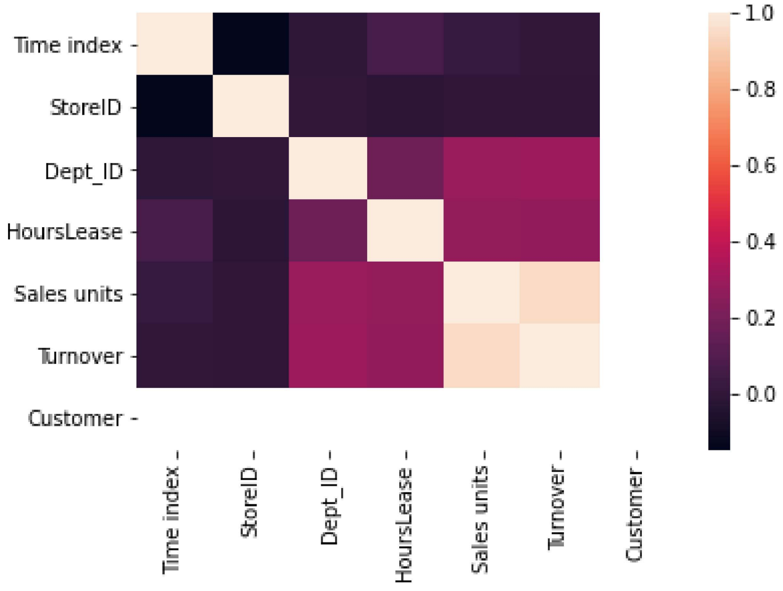
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

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



In [12]:
```
df2 = df1[['HoursOwn', 'HoursLease', 'Sales units', 'Turnover']]
sns.heatmap(df1.corr())
```

Out[12]:
```
<AxesSubplot:>
```



In [13]:
```
x = df2[['HoursOwn', 'HoursLease', 'Sales units']]
y = df2['Turnover']
```

In [16]:
```
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 [15]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(x_train,y_train)
```

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

In [17]:
```
print(lr.intercept_)
```

```
82365.4029589505
```

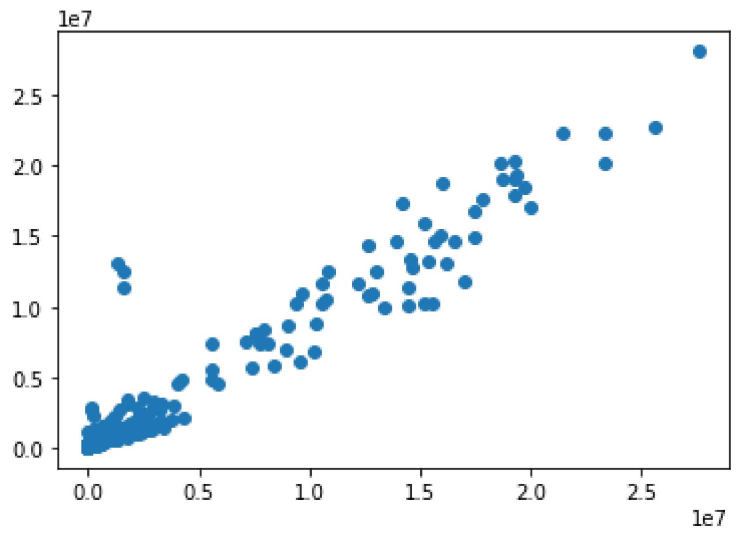
In [18]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[18]:

	Co-efficient
HoursOwn	37.774189
HoursLease	68.289672
Sales units	3.009753

In [19]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[19]:
```
<matplotlib.collections.PathCollection at 0x2548ec4b520>
```



In [20]:
```
print(lr.score(x_test,y_test))
```

```
0.9274311730976958
```

In [ ]: