In [1]:
```
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 [2]:
```
df = pd.read_csv("6_Salesworkload1.csv")
df = df[['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
       'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover', 'Area (m2)',
df
```

Out[2]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursLease	Sa ur
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

7650 rows × 13 columns

In [3]:
```
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

Data cleaning and pre processing

In [4]:
```
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 [5]:
```
df.describe()
```

Out[5]:

	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

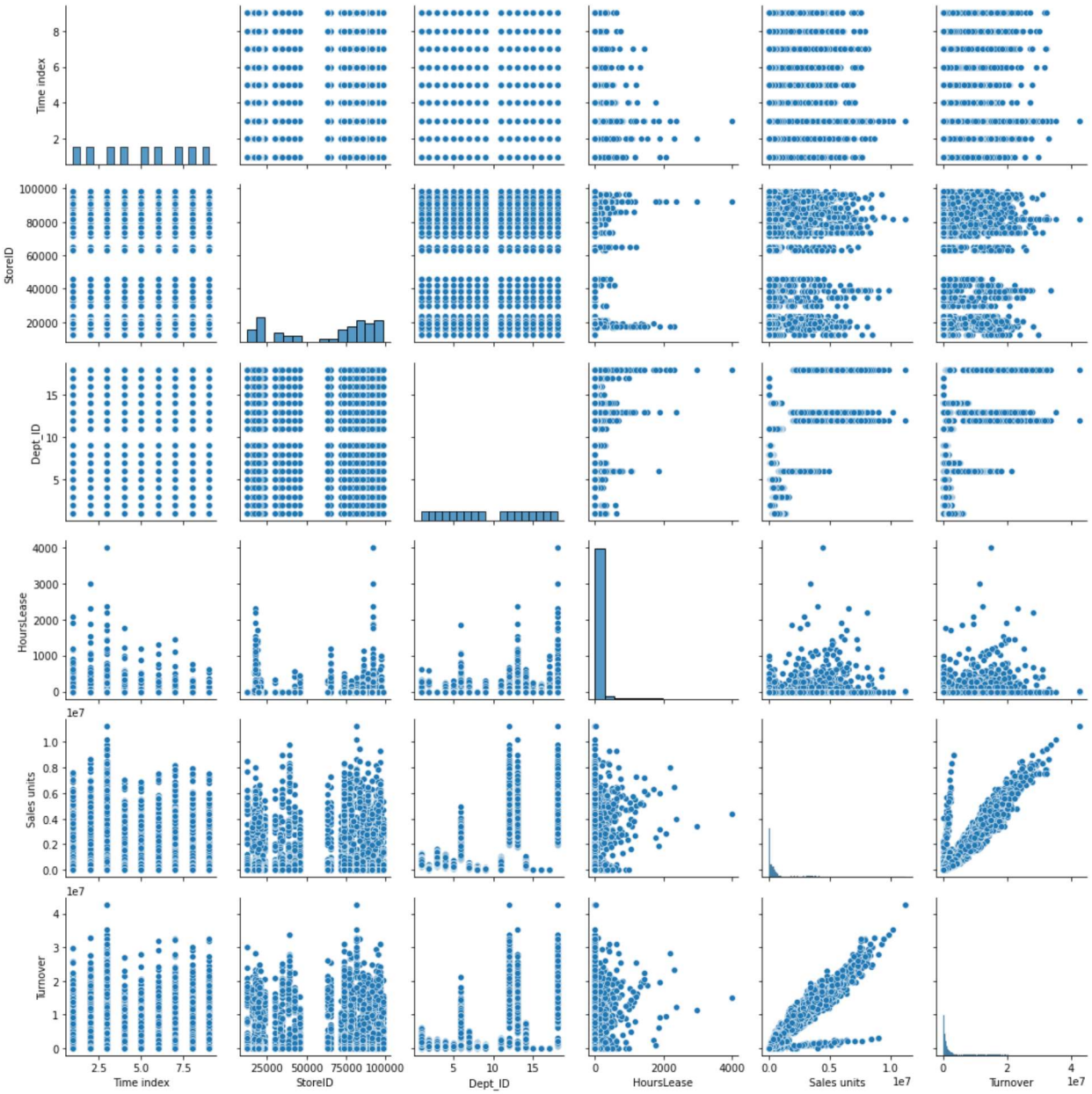
In [6]:
```
df.columns
```

Out[6]:
```
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 [7]:
```
sns.pairplot(df)
```

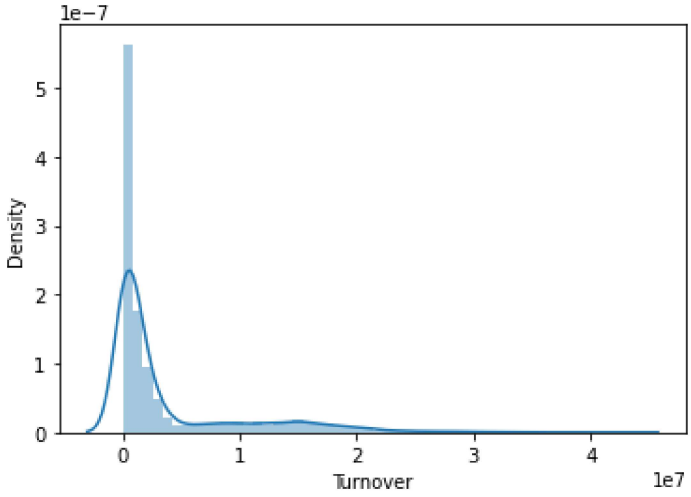
Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x2204d8d7c70>
```



In [8]:
```
sns.distplot(df["Turnover"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarnin
g: `distplot` is a deprecated function and will be removed in a future version. Please
adapt your code to use either `displot` (a figure-level function with similar flexibil
ity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

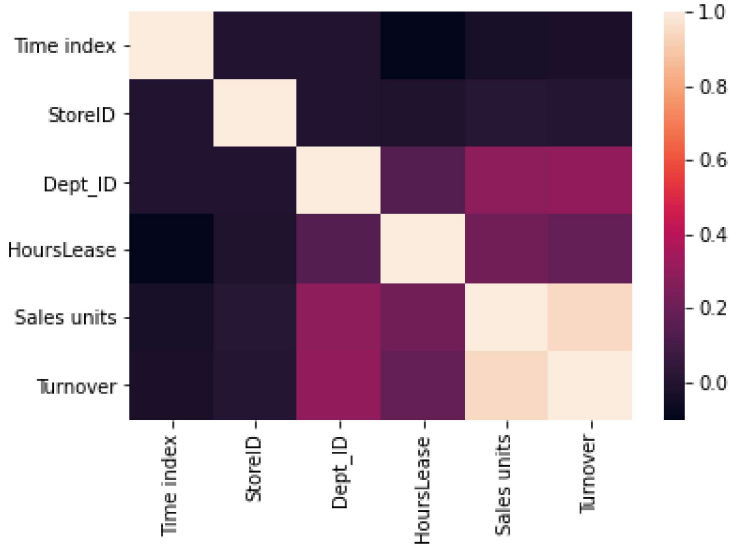
Out[8]: <AxesSubplot:xlabel='Turnover', ylabel='Density'>



In [9]:
```
df1 = df[['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
       'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
       'Area (m2)', 'Opening hours']]
```

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

Out[10]: <AxesSubplot:>



In [11]:
```
df1.fillna(1)
```

Out[11]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursLease	Sa u
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

7650 rows × 13 columns

In [12]:
```
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 [13]:
```
x = df1[[ 'Time index', 'StoreID', 'Dept_ID',
        'HoursLease', 'Sales units']]
y = df1['Turnover']
```

split the data into training and test data

In [14]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3)
```

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

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

In [16]:
```
lr.intercept_
```

Out[16]:
```
-149257.00780653395
```

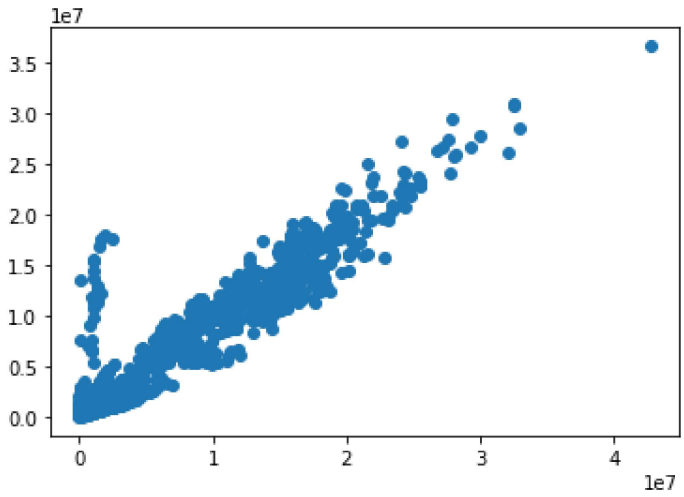
In [17]:
```
coeff = pd.DataFrame(lr.coef_, x.columns, columns =['Co-efficient'])
coeff
```

Out[17]:

	Co-efficient
Time index	21307.295834
StoreID	-0.867540
Dept_ID	34198.649362
HoursLease	-647.190945
Sales units	3.238854

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

Out[18]:
```
<matplotlib.collections.PathCollection at 0x22050a57eb0>
```



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

Out[19]:
```
0.9137256307047066
```

In [20]:
```
from sklearn.linear_model import ElasticNet
en = ElasticNet()
en.fit(x_train,y_train)
```

Out[20]:
```
ElasticNet()
```

In [21]:
```
print(en.coef_)
```

```
[ 1.98261822e+04 -8.70188601e-01  3.35624690e+04 -6.47821609e+02
  3.23933160e+00]
```

In [22]:
```
print(en.intercept_)
```

```
-136129.6512203263
```

In [23]:
```
print(en.predict(x_test))
```

```
[  430186.71509285   414874.51593134  1236368.90771275 ...
 19045097.03077549   711298.95331123  9569529.39784102]
```

In [24]:
```
print(en.score(x_test,y_test))
```

```
0.913710254809619
```

In [25]:
```
from sklearn import metrics
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

```
Mean Absolytre Error: 979214.8781805587
Mean Squared Error: 3125196318733.213
Root Mean Squared Error: 1767822.4794173234
```

In [ ]: