DATA COLLECTION

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

In [2]:
```
# To Import Dataset
sd=pd.read_csv(r"c:\Users\user\Downloads\\Salesworkload.csv")
sd
```

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	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]:

```
# to display top 10 rows
sd.head(10)
```

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	3
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	4
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	0.0	3
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	0.0	1
5	10.2016	1.0	United Kingdom	88253.0	London (I)	6.0	Meat	8270.316	0.0	17
6	10.2016	1.0	United Kingdom	88253.0	London (I)	13.0	Food	16468.251	0.0	31
7	10.2016	1.0	United Kingdom	88253.0	London (I)	7.0	Clothing	4698.471	0.0	2
8	10.2016	1.0	United Kingdom	88253.0	London (I)	8.0	Household	1183.272	0.0	
9	10.2016	1.0	United Kingdom	88253.0	London (I)	9.0	Hardware	2029.815	0.0	

DATA CLEANING AND PRE_PROCESSING

In [4]:
```
sd.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]:
```
# to display summary of statistics
sd.describe()
```

Out[5]:

	Time index	StoreID	Dept_ID	HoursLease	Sales units	Turnover	Custon
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	N
std	2.582158	29924.581631	5.337429	133.299513	1.728113e+06	6.003380e+06	N
min	1.000000	12227.000000	1.000000	0.000000	0.000000e+00	0.000000e+00	N
25%	3.000000	29650.000000	5.000000	0.000000	5.457125e+04	2.726798e+05	N
50%	5.000000	75400.500000	9.000000	0.000000	2.932300e+05	9.319575e+05	N
75%	7.000000	87703.000000	14.000000	0.000000	9.175075e+05	3.264432e+06	N
max	9.000000	98422.000000	18.000000	3984.000000	1.124296e+07	4.271739e+07	N

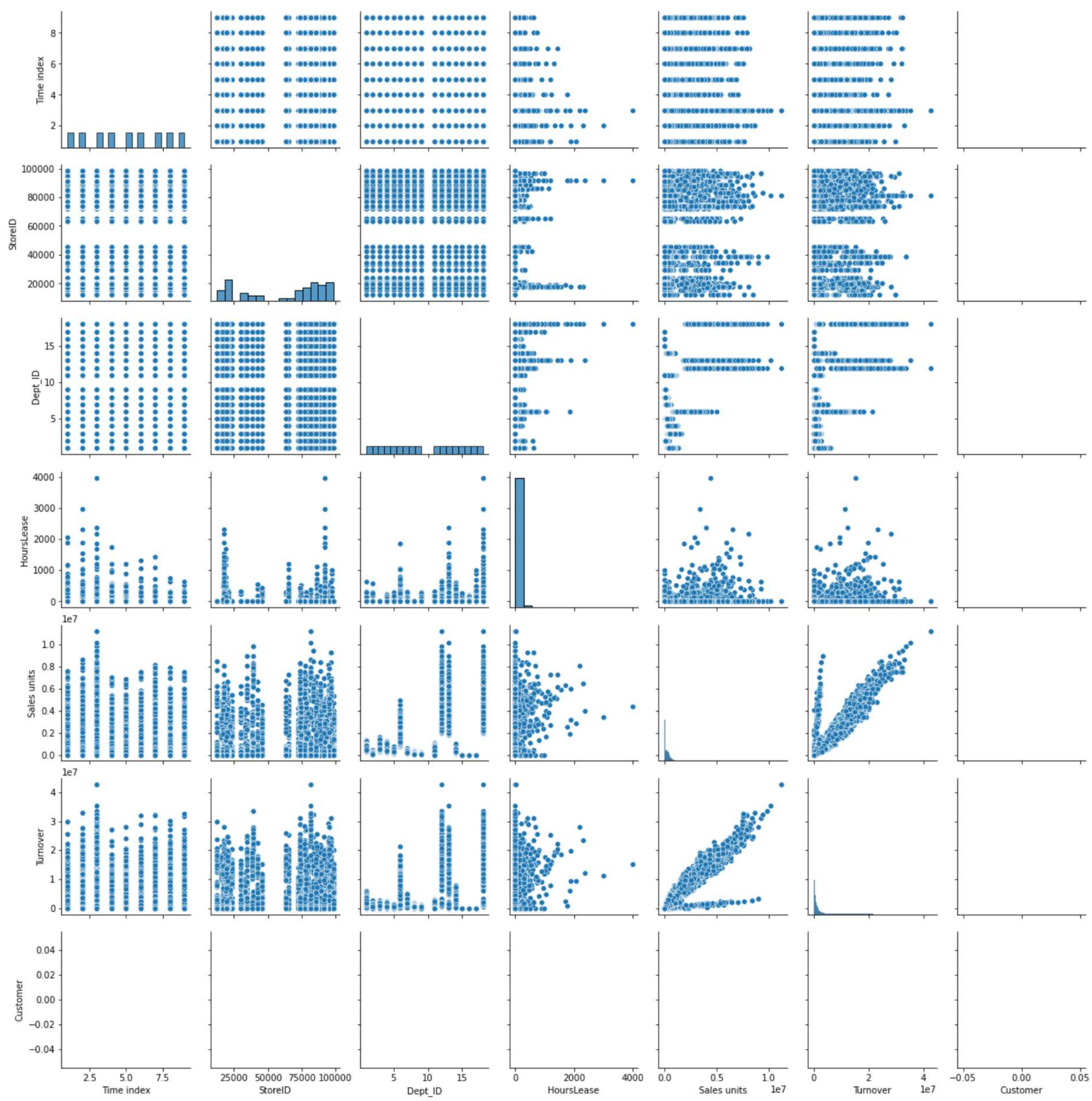
In [6]:
```
#to display colums heading
sd.columns
```

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')
```

EDA and visualization

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

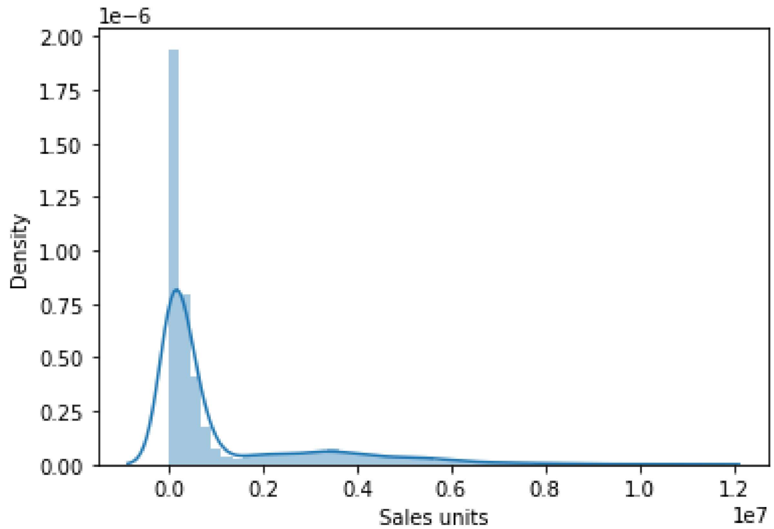
Out[7]: <seaborn.axisgrid.PairGrid at 0x2ee1600ea90>



In [8]:
```python
sns.distplot(sd['Sales units'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `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 flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

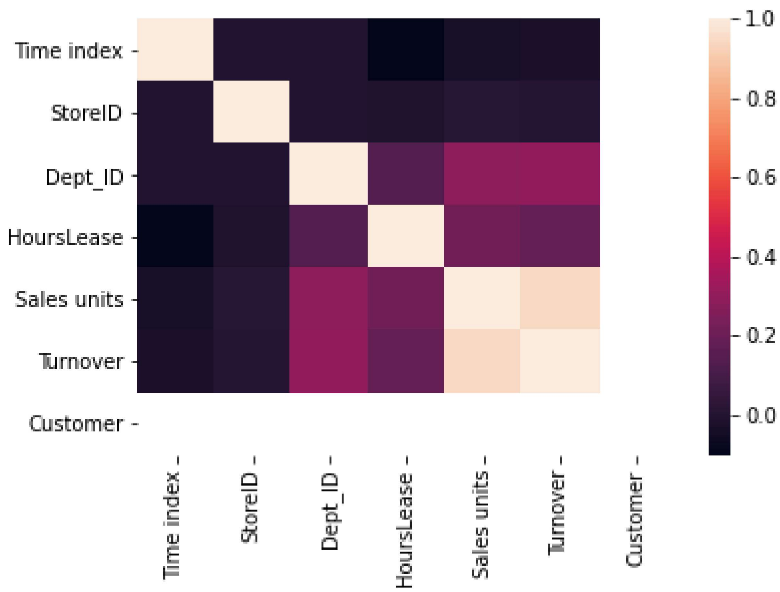
Out[8]: <AxesSubplot:xlabel='Sales units', ylabel='Density'>



In [9]:
```python
sd1=sd[['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
        'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
        'Customer', 'Area (m2)', 'Opening hours']]
```

In [10]:
```python
sns.heatmap(sd1.corr())
```

Out[10]: <AxesSubplot:>



TO TRAIN THE MODEL _MODEL BUILDING

we are goint train Liner Regression model; we need to split out the data into two varibles x and y where x is independent on x (output) and y is dependent on x(output) adress coloumn as it is not required our model

In [11]:
```
dss=sd.head(200)
dss
```

Out[11]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursLeas
0	10.2016	1.0	United Kingdom	88253.0	London (I)	1.0	Dry	3184.764	0.(
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen	1582.941	0.(
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other	47.205	0.(
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	0.(
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	0.(
...	...	...	...	...	...	...	...	...	..
195	10.2016	1.0	The Netherlands	95434.0	Den Haag	8.0	Household	2127.372	0.(
196	10.2016	1.0	The Netherlands	95434.0	Den Haag	9.0	Hardware	2158.842	0.(
197	10.2016	1.0	The Netherlands	95434.0	Den Haag	14.0	Non Food	9887.874	0.(
198	10.2016	1.0	The Netherlands	95434.0	Den Haag	15.0	Admin	5589.072	0.(
199	10.2016	1.0	The Netherlands	95434.0	Den Haag	12.0	Checkout	6781.785	0.(

200 rows × 14 columns

In [12]:
```
x= dss[['Time index', 'Dept_ID', 'HoursOwn','Turnover']]
y=dss[ 'Sales units']
```

In [13]:
```
# To split my dataset  into training data and test data
from sklearn .model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.4)
```

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

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

Out[14]: LinearRegression()

In [15]:
```python
print(lr.intercept_)
```

-18444.916812692536

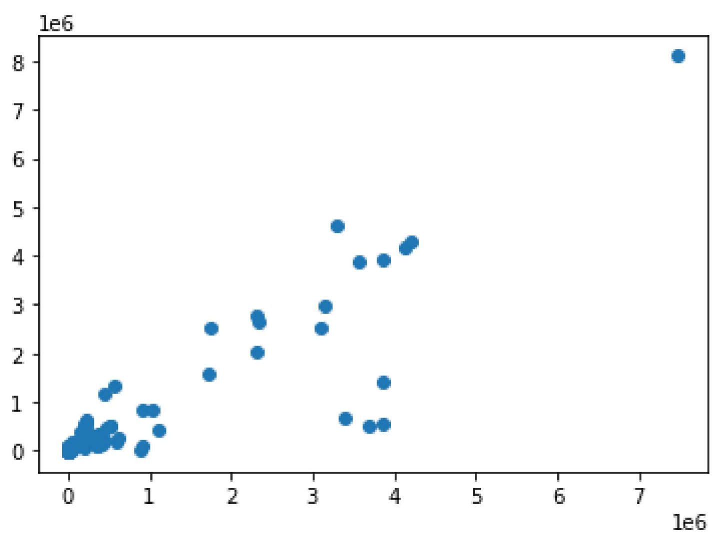
In [16]:
```python
coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
Time index	0.000000
Dept_ID	-540.602054
HoursOwn	23.867138
Turnover	0.247450

In [17]:
```python
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[17]: <matplotlib.collections.PathCollection at 0x2ee1b13b700>



In [18]:
```python
print(lr.score(x_test,y_test))
```

0.7448928717869541

In [19]:
```python
lr.score(x_train,y_train)
```

Out[19]: 0.9260720292667706

In [20]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [21]:
```
dr=Ridge(alpha=10)
dr.fit(x_train,y_train)
```

Out[21]:
```
Ridge(alpha=10)
```

In [22]:
```
dr.score(x_test,y_test)
```

Out[22]:
```
0.7448925287377836
```

In [23]:
```
dr.score(x_train,y_train)
```

Out[23]:
```
0.9260720292277196
```

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]:
```
0.7448927954923499
```

In [26]:
```
la.score(x_train,y_train)
```

Out[26]:
```
0.926072029264835
```

ElasticNet

In [27]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[27]:
```
ElasticNet()
```

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

```
[ 0.00000000e+00 -5.28038499e+02  2.38638871e+01  2.47451792e-01]
```

In [29]:
```
print(en.intercept_)
```

```
-18548.002691796748
```

In [30]:
```
prediction=en.predict(x_test)
```

In [31]:
```
print(en.score(x_test,y_test))
```

```
0.7448908480351606
```

Evaluation metrics

In [32]:
```
from sklearn import metrics
```

In [34]:
```
print("mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
mean Absolute Error: 335439.95295565453
```

In [35]:
```
print("mean squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
mean squared Error: 531595124526.4042
```

In [36]:
```
print("Root mean Absolytre Error:",np.sqrt(metrics.mean_squared_error(y_test,pr
```

```
Root mean Absolytre Error: 729105.7018885562
```

In [ ]:
```

```