In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

DATA COLLECTION

In [2]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\6_Salesworkload1 - 6_Salesworkload1.csv")
```

In [3]:

```
b=a.head(10)
b
```

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
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
5	10.2016	1.0	United Kingdom	88253.0	London (I)	6.0	Meat	8270.316	0.0
6	10.2016	1.0	United Kingdom	88253.0	London (I)	13.0	Food	16468.251	0.0
7	10.2016	1.0	United Kingdom	88253.0	London (I)	7.0	Clothing	4698.471	0.0
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]:

```
b.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   MonthYear      10 non-null     object 
 1   Time index     10 non-null     float64
 2   Country        10 non-null     object 
 3   StoreID        10 non-null     float64
 4   City           10 non-null     object 
 5   Dept_ID        10 non-null     float64
 6   Dept. Name     10 non-null     object 
 7   HoursOwn       10 non-null     object 
 8   HoursLease     10 non-null     float64
 9   Sales units    10 non-null     float64
 10  Turnover       10 non-null     float64
 11  Customer       0 non-null      float64
 12  Area (m2)      10 non-null     object 
 13  Opening hours  10 non-null     object 
dtypes: float64(7), object(7)
memory usage: 1.2+ KB
```

In [5]:

```
b.describe()
```

Out[5]:

	Time index	StoreID	Dept_ID	HoursLease	Sales units	Turnover	Customer
count	10.0	10.0	10.000000	10.0	1.000000e+01	1.000000e+01	0.0
mean	1.0	88253.0	5.800000	0.0	6.543725e+05	1.978511e+06	NaN
std	0.0	0.0	3.614784	0.0	9.914003e+05	2.861420e+06	NaN
min	1.0	88253.0	1.000000	0.0	5.491500e+04	2.904000e+05	NaN
25%	1.0	88253.0	3.250000	0.0	1.034225e+05	4.033612e+05	NaN
50%	1.0	88253.0	5.500000	0.0	2.615525e+05	5.770455e+05	NaN
75%	1.0	88253.0	7.750000	0.0	4.284400e+05	1.518067e+06	NaN
max	1.0	88253.0	13.000000	0.0	3.107935e+06	8.714679e+06	NaN

In [6]:

```
c=b.dropna(axis=1)
c
```

Out[6]:

	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
5	10.2016	1.0	United Kingdom	88253.0	London (I)	6.0	Meat	8270.316	0.0
6	10.2016	1.0	United Kingdom	88253.0	London (I)	13.0	Food	16468.251	0.0
7	10.2016	1.0	United Kingdom	88253.0	London (I)	7.0	Clothing	4698.471	0.0
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

In [7]:

```
c.columns
```

Out[7]:

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

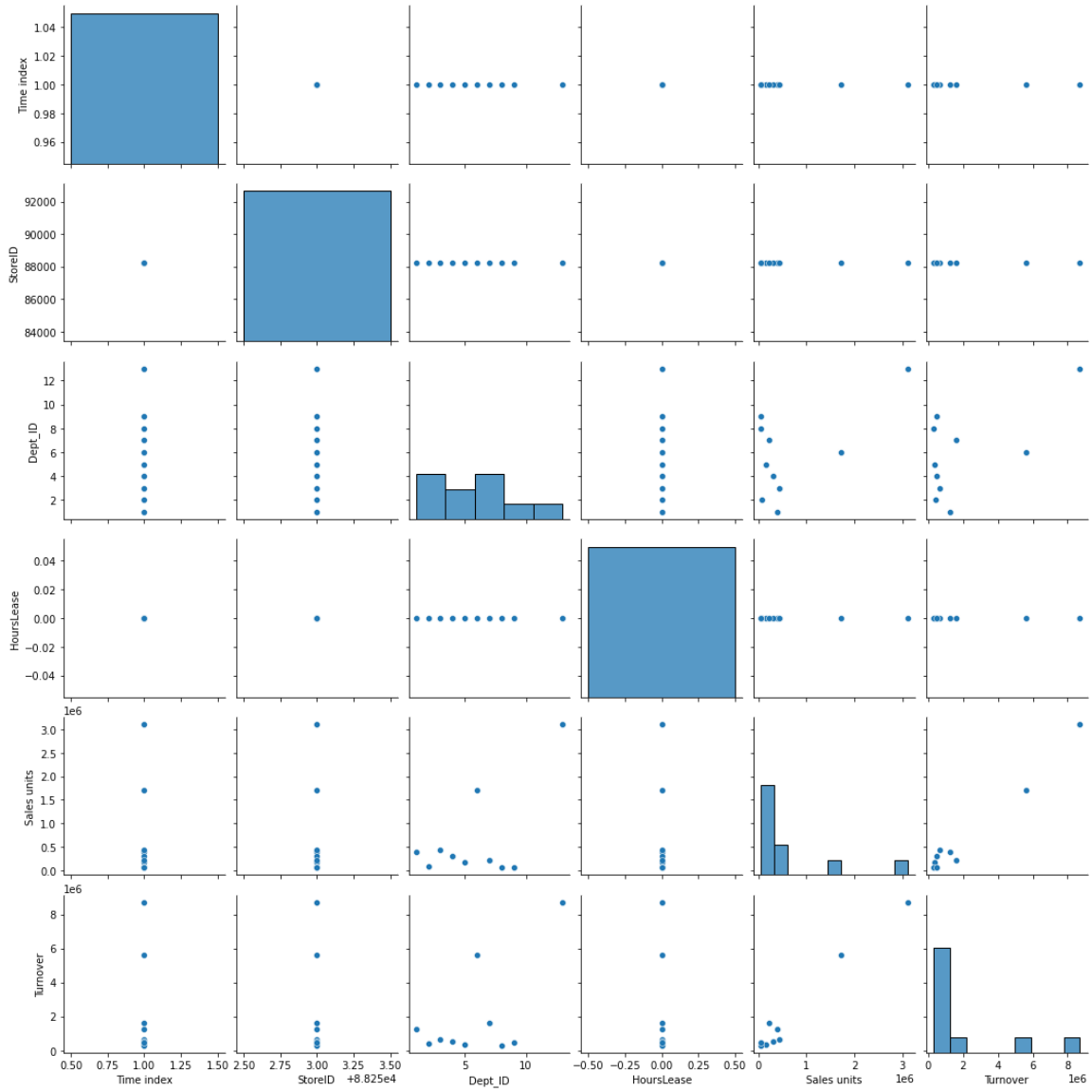
EDA AND VISUALIZATION

In [8]:

```
sns.pairplot(c)
```

Out[8]:

```
<seaborn.axisgrid.PairGrid at 0x290e586b9a0>
```



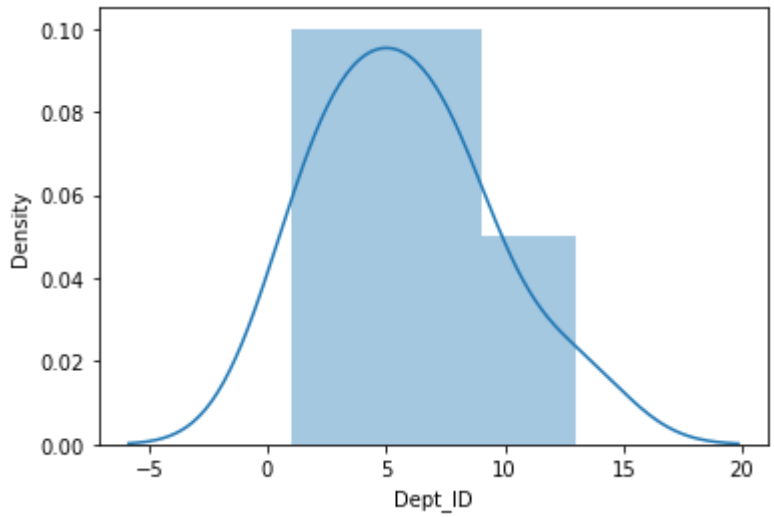
In [9]:

```
sns.distplot(c['Dept_ID'])
```

```
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 fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[9]:

```
<AxesSubplot:xlabel='Dept_ID', ylabel='Density'>
```



In [10]:

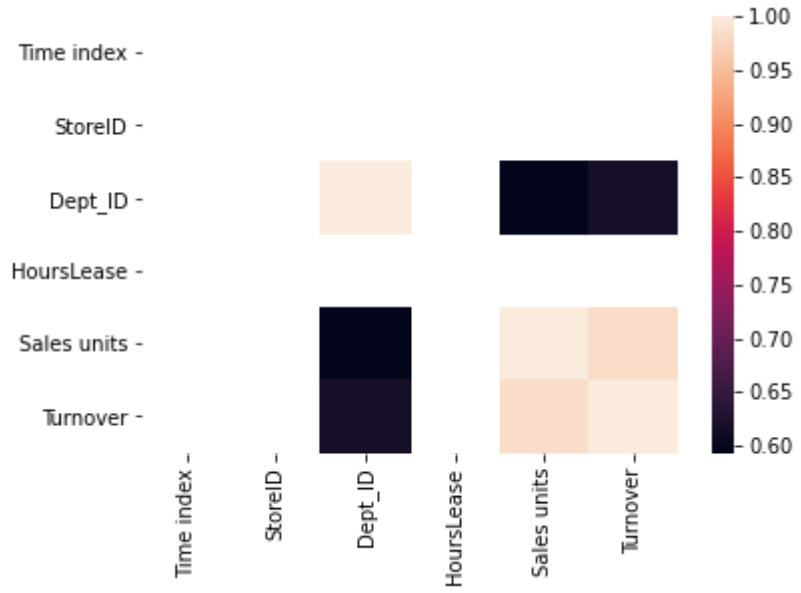
```
f=c[['MonthYear', 'Time index','StoreID','Dept_ID',
      'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
       'Area (m2)']]
```

In [11]:

```
sns.heatmap(f.corr())
```

Out[11]:

<AxesSubplot:>



In [12]:

```
x=f[['MonthYear', 'Time index','StoreID',
      'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
       'Area (m2)']]
y=f['Dept_ID']
```

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=0.5)
```

In [14]:

```
from sklearn.linear_model import LinearRegression

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

Out[14]:

```
LinearRegression()
```

In [15]:

```
print(lr.intercept_)
```

```
2.4463291806789282
```

In [16]:

```
r=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
r
```

Out[16]:

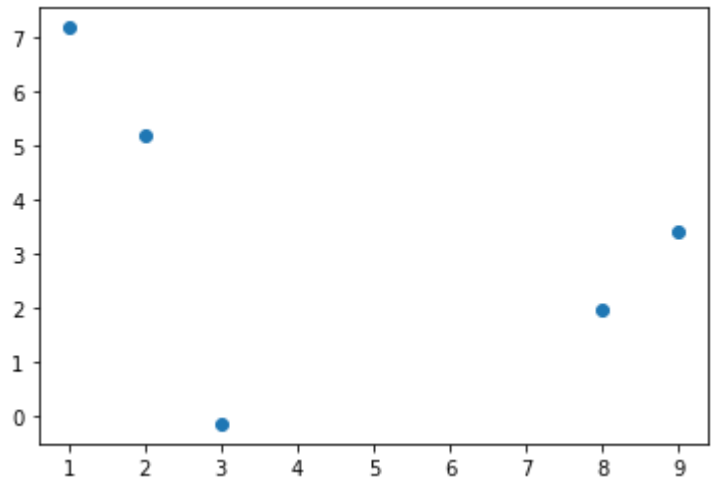
	Co-efficient
MonthYear	-4.625630e-16
Time index	5.605664e-17
StoreID	-6.505213e-19
HoursOwn	2.260395e-03
HoursLease	0.000000e+00
Sales units	-4.868182e-06
Turnover	3.798578e-08
Area (m2)	-5.976206e-04

In [17]:

```
u=lr.predict(x_test)
plt.scatter(y_test,u)
```

Out[17]:

```
<matplotlib.collections.PathCollection at 0x290e8e942e0>
```



In [18]:

```
print(lr.score(x_test,y_test))
```

```
-1.3666258799319078
```

In [19]:

```
lr.score(x_train,y_train)
```

Out[19]:

```
1.0
```

RIDGE REGRESSION

In [20]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [21]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[21]:

```
Ridge(alpha=10)
```

In [22]:

```
rr.score(x_test,y_test)
```

Out[22]:

```
-1.3663925611021308
```

LASSO REGRESSION

In [23]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_coordinat
e_descent.py:530: ConvergenceWarning: Objective did not converge. You migh
t want to increase the number of iterations. Duality gap: 0.12751548581240
268, tolerance: 0.005
  model = cd_fast.enet_coordinate_descent(
```

Out[23]:

```
Lasso(alpha=10)
```

In [24]:

```
la.score(x_test,y_test)
```

Out[24]:

```
-0.424935445863019
```

In [25]:

```
from sklearn.linear_model import ElasticNet
p=ElasticNet()
p.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_coordinat
e_descent.py:530: ConvergenceWarning: Objective did not converge. You migh
t want to increase the number of iterations. Duality gap: 0.02325436975414
4858, tolerance: 0.005
  model = cd_fast.enet_coordinate_descent(
```

Out[25]:

```
ElasticNet()
```

In [26]:

```
print(p.coef_)
```

```
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.85473317e-03
  0.00000000e+00 -2.48275464e-06 -7.42808798e-07 -2.98259748e-04]
```

In [27]:

```
print(p.intercept_)
```

```
2.5722743426939
```

In [28]:

```
print(p.predict(x_test))
```

```
[4.79986278 2.97048758 6.29451677 0.79676874 4.18275713]
```

In [29]:

```
prediction=p.predict(x_test)
print(p.score(x_test,y_test))
```

```
-0.677202847410312
```

In [30]:

```
from sklearn import metrics
```

In [31]:

```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolytre Error: 4.028873219902137
```

In [32]:

```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 17.84543829644572
```

In [33]:

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

```
Root Mean Squared Error: 4.22438614433455
```

In [ ]:

```

```