kaviyadevi 20106064

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

In [2]:
```
#to import dataset
data1=pd.read_csv(r"C:\Users\user\Downloads\6_Salesworkload1 - 6_Salesworkload1.c
data1
```

Out[2]:

	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.
...	...	...	...	...	...	...	...	...	.
7653	6.2017	9.0	Sweden	29650.0	Gothenburg	12.0	Checkout	6322.323	0.
7654	6.2017	9.0	Sweden	29650.0	Gothenburg	16.0	Customer Services	4270.479	0.
7655	6.2017	9.0	Sweden	29650.0	Gothenburg	11.0	Delivery	0	0.
7656	6.2017	9.0	Sweden	29650.0	Gothenburg	17.0	others	2224.929	0.
7657	6.2017	9.0	Sweden	29650.0	Gothenburg	18.0	all	39652.2	0.

7658 rows × 14 columns

In [3]:
```
#to display top 5 rows
data=data1.head(200)
data
```

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
...	...	...	...	...	...	...	...	...	...
195	10.2016	1.0	The Netherlands	95434.0	Den Haag	8.0	Household	2127.372	0.0
196	10.2016	1.0	The Netherlands	95434.0	Den Haag	9.0	Hardware	2158.842	0.0
197	10.2016	1.0	The Netherlands	95434.0	Den Haag	14.0	Non Food	9887.874	0.0
198	10.2016	1.0	The Netherlands	95434.0	Den Haag	15.0	Admin	5589.072	0.0
199	10.2016	1.0	The Netherlands	95434.0	Den Haag	12.0	Checkout	6781.785	0.0

200 rows × 14 columns

DATA CLEANING AND PREPROCESSING

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   MonthYear      200 non-null    object 
 1   Time index     200 non-null    float64
 2   Country        200 non-null    object 
 3   StoreID        200 non-null    float64
 4   City           200 non-null    object 
 5   Dept_ID        200 non-null    float64
 6   Dept. Name     200 non-null    object 
 7   HoursOwn       200 non-null    object 
 8   HoursLease     200 non-null    float64
 9   Sales units    200 non-null    float64
 10  Turnover       200 non-null    float64
 11  Customer       0 non-null      float64
 12  Area (m2)      200 non-null    object 
 13  Opening hours  200 non-null    object 
dtypes: float64(7), object(7)
memory usage: 22.0+ KB
```

In [5]:
```
#to display summary of statistics
data.describe()
```

Out[5]:

	Time index	StoreID	Dept_ID	HoursLease	Sales units	Turnover	Customer
count	200.0	200.000000	200.000000	200.00000	2.000000e+02	2.000000e+02	0.0
mean	1.0	46739.115000	9.350000	41.23000	9.317313e+05	3.000231e+06	NaN
std	0.0	30654.343517	5.320625	184.09236	1.521370e+06	5.188606e+06	NaN
min	1.0	15552.000000	1.000000	0.00000	0.000000e+00	0.000000e+00	NaN
25%	1.0	18808.000000	5.000000	0.00000	5.200250e+04	2.084858e+05	NaN
50%	1.0	23623.000000	9.000000	0.00000	2.429175e+05	5.771910e+05	NaN
75%	1.0	73949.000000	14.000000	0.00000	9.019388e+05	2.358503e+06	NaN
max	1.0	95434.000000	18.000000	1896.00000	7.476680e+06	2.571973e+07	NaN

In [6]:
```
#to display the column heading
data.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 DATA VISUALIZATION

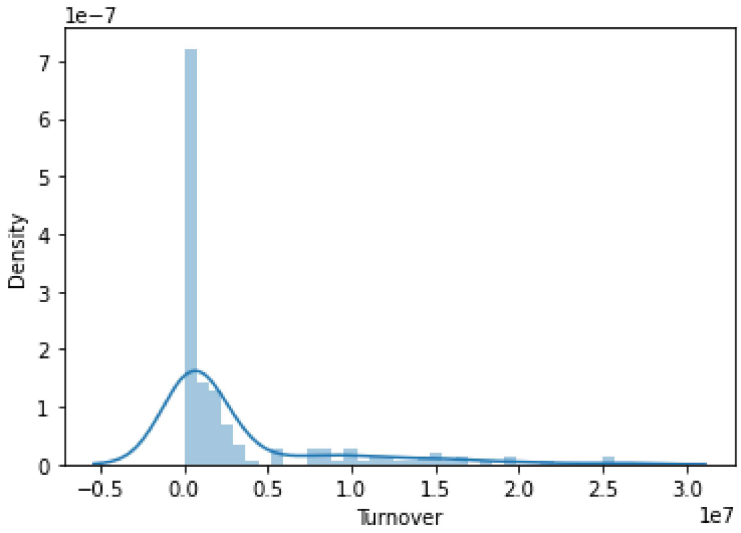
In [7]:
```
sns.pairplot(data)
```

Out[7]: <seaborn.axisgrid.PairGrid at 0x26860e5bd00>

In [8]:
```
sns.distplot(data['Turnover'])
```

```
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)
```

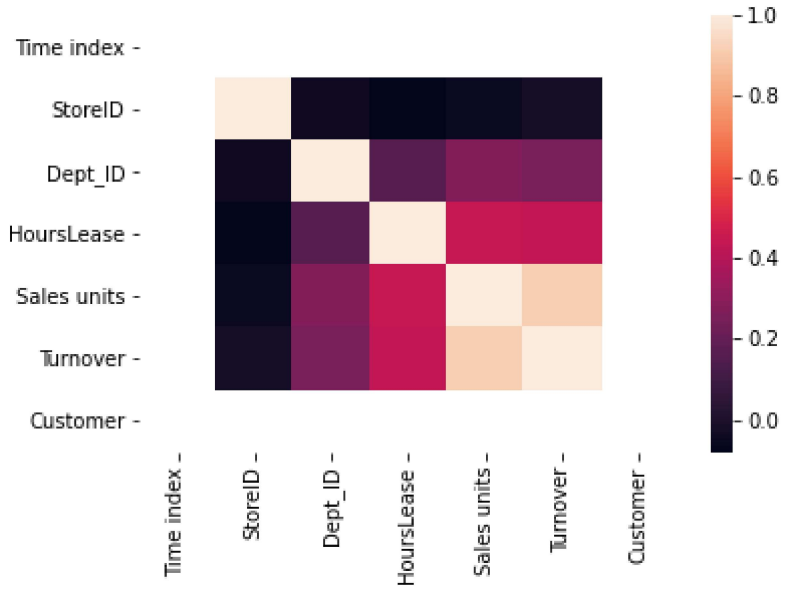
Out[8]: <AxesSubplot:xlabel='Turnover', ylabel='Density'>



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

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

Out[10]: <AxesSubplot:>



TRAINING MODEL

In [11]:
```
x=df[['MonthYear','Time index', 'Dept_ID','HoursOwn', 'HoursLease', 'Sales units'
y=df[['Turnover']]
```

In [12]:
```
#to split my dataset into trainning and test
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 [13]:
```
from sklearn.linear_model import LinearRegression

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

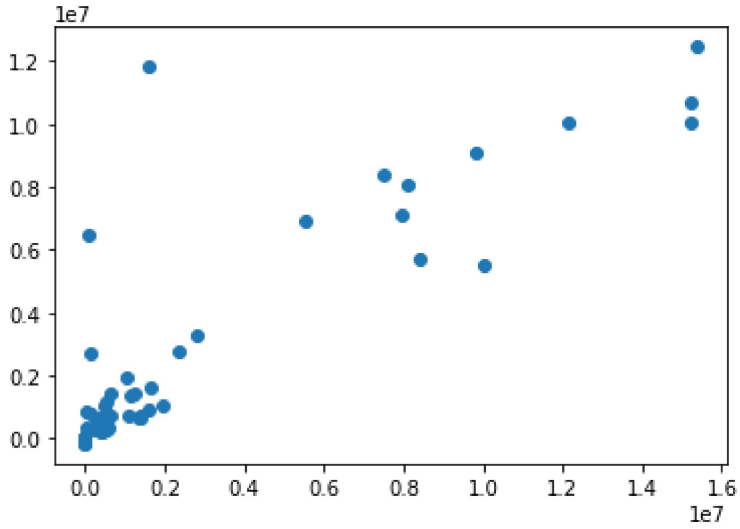
Out[13]: LinearRegression()

In [14]:
```
#to find intercept
print(lr.intercept_)
```

```
[193392.90362753]
```

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

Out[15]:
```
<matplotlib.collections.PathCollection at 0x26864cacca0>
```



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

```
0.7538574505843971
```

RIDGE AND LASSO REGRESSION

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

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

Out[18]:
```
Ridge(alpha=10)
```

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

Out[19]:
```
0.7538576529256205
```

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

Out[20]:
```
Lasso(alpha=10)
```

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

Out[21]:
```
0.7538574516018293
```

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

Out[22]:
```
ElasticNet()
```

In [23]:
```
print(en.coef_)
```

```
[ 0.00000000e+00  0.00000000e+00 -2.03699608e+04  2.50037064e+01
  7.10236193e+02  2.87402141e+00  1.68873956e+01]
```

In [24]:
```
print(en.predict(x_test))
```

```
[ 2.73498723e+05  7.41666662e+05 -3.14103845e+04  3.05042148e+05
  1.41039097e+06  8.65258382e+05  2.78447876e+06  3.30379937e+06
  5.50720797e+06  6.52858151e+05  6.49213581e+06  1.00174014e+07
  9.33609231e+03  9.68916975e+02  5.82429232e+04  9.07355618e+06
  6.56923058e+05 -1.55836185e+05  1.18490637e+07  3.56070481e+05
  1.00405784e+07  7.07795389e+05 -7.72479785e+04  1.40593248e+06
  9.22347229e+05  1.38589371e+06 -9.91893212e+04 -1.13910564e+05
  5.68836417e+06  7.48761410e+05  6.43077926e+05  3.45703486e+05
  2.69908100e+05  1.05577075e+06  3.13329466e+05  4.97679389e+05
 -1.55793075e+05  3.16116480e+05  2.72470200e+06  7.74045494e+03
  7.72514132e+05  1.15171485e+06  1.06905109e+07  1.61465432e+06
  2.60657062e+05 -5.83820340e+04  6.90783490e+06  1.24587620e+07
 -5.33392514e+04  4.86962739e+05  1.05060171e+06  8.09525311e+06
  4.71505148e+05  2.39217524e+05  7.14108569e+06 -2.08727506e+04
  8.36247283e+06  2.29049271e+05  2.39218654e+04  1.93851705e+06]
```

In [25]:
```
print(en.score(x_test,y_test))
```

```
0.7538585060063644
```

In [26]:
```
from sklearn import metrics
```

In [27]:
```
print("Mean Absolute error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute error 962708.2696065212
```

In [28]:
```
print("Mean Squared error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared error 4204379222938.081
```

In [29]:
```
print("Root Mean Absolute error",np.sqrt(metrics.mean_squared_error(y_test,predic
```

```
Root Mean Absolute error 2050458.2958300032
```