Problem Statement

Linear Regression

Import Libraries

In [1]:

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

In [2]:

```
a=pd.read_csv("Sales.csv")
a
```

Out[2]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursLease	S u
0	10.2016	1.0	United Kingdom	88253.0	London (I)	1.0	Dry	3184.764	0.0	3985
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen	1582.941	0.0	827
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other	47.205	0.0	4384
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	0.0	3094
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	0.0	1655
...	...	...	...	...	...	...	...	...	...	
7653	06.2017	9.0	Sweden	29650.0	Gothenburg	12.0	Checkout	6322.323	0.0	38865

To display top 10 rows

In [3]:

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

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

To find Missing values

In [4]:

```
c.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
```

To display summary of statistics

In [5]:

```
a.describe()
```

Out[5]:

	Time index	StoreID	Dept_ID	HoursLease	Sales units	Turnover	Cu
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	

To display column heading

In [6]:

```
a.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')
```

Pairplot

In [7]:

```
s=a.dropna(axis=1)
s
```

Out[7]:

	MonthYear
0	10.2016
1	10.2016
2	10.2016
3	10.2016
4	10.2016
...	...
7653	06.2017
7654	06.2017
7655	06.2017
7656	06.2017
7657	06.2017

7658 rows × 1 columns

In [8]:

```
s.columns
```

Out[8]:

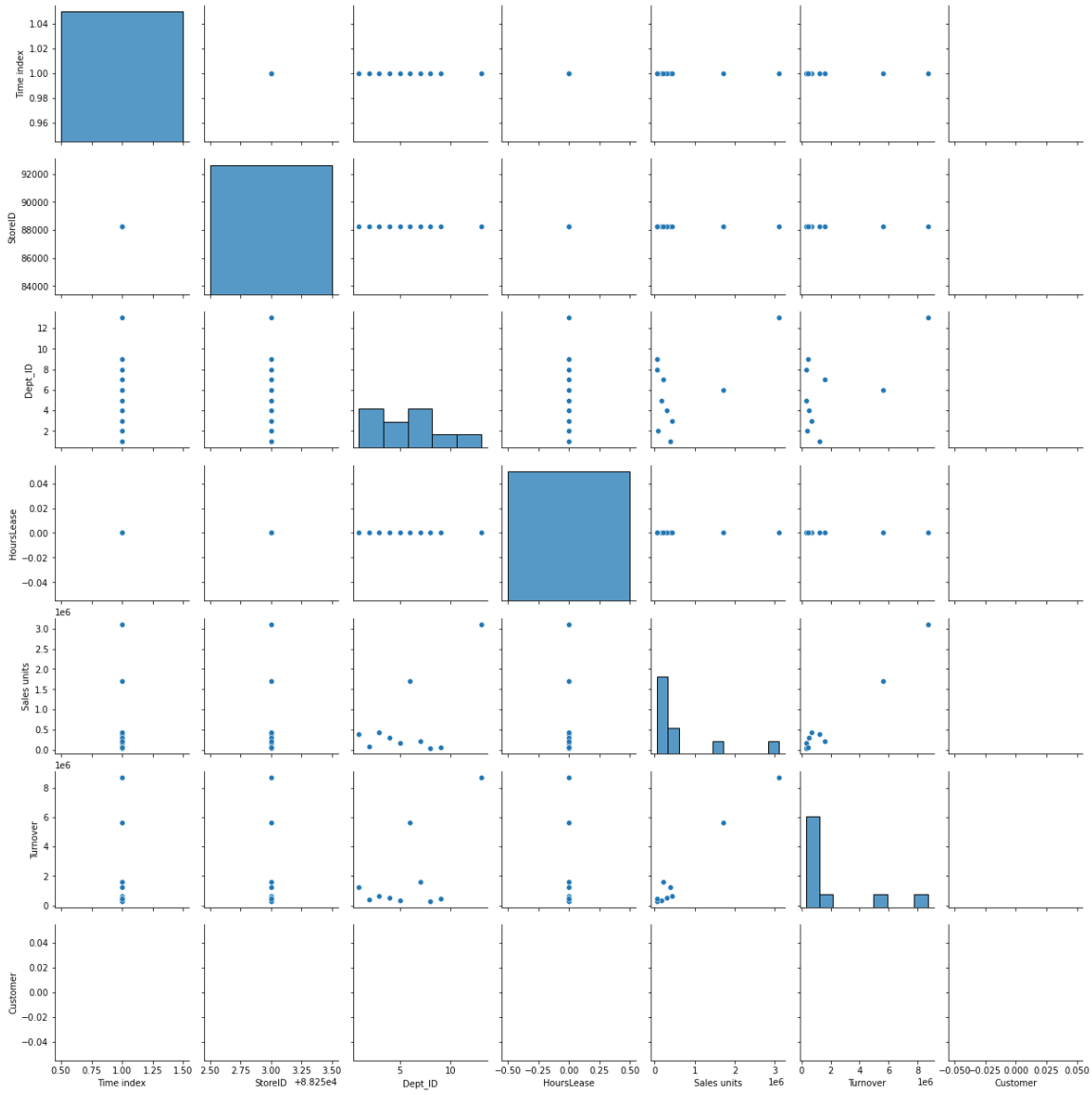
```
Index(['MonthYear'], dtype='object')
```

In [9]:

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

Out[9]:

```
<seaborn.axisgrid.PairGrid at 0x1ed3addbf70>
```



Distribution Plot

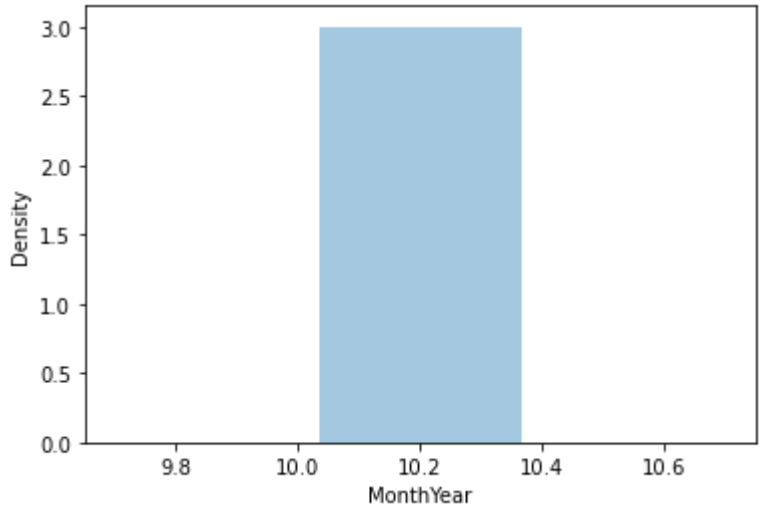
In [10]:

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

```
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)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:306: U
serWarning: Dataset has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
```

Out[10]:

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



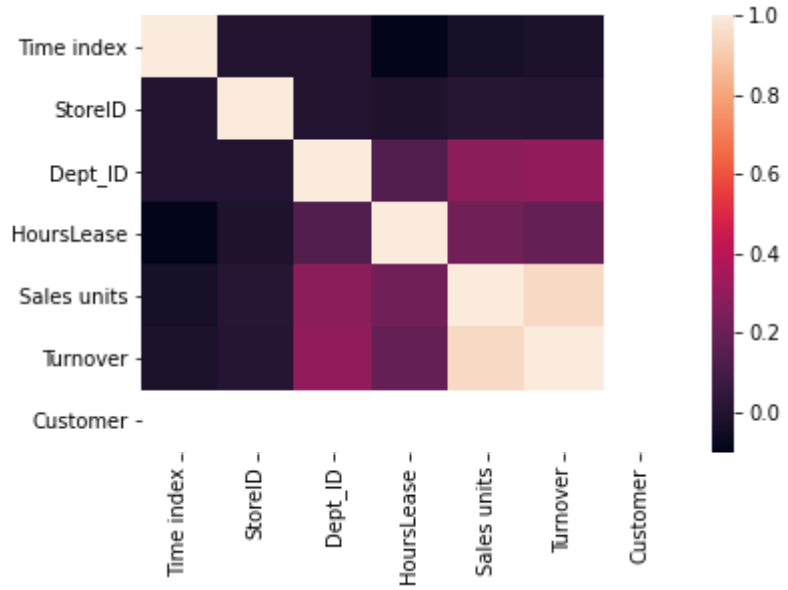
Correlation

In [11]:

```
b=a[['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
       'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
       'Customer', 'Area (m2)', 'Opening hours']]
sns.heatmap(b.corr())
```

Out[11]:

```
<AxesSubplot:>
```



Train the model - Model Building

In [12]:

```
g=c[['MonthYear']]
h=c['MonthYear']
```

To split dataset into training end test

In [13]:

```
from sklearn.model_selection import train_test_split
g_train,g_test,h_train,h_test=train_test_split(g,h,test_size=0.6)
```

To run the model

In [14]:

```
from sklearn.linear_model import LinearRegression
```

In [15]:

```
lr=LinearRegression()
lr.fit(g_train,h_train)
```

Out[15]:

```
LinearRegression()
```

In [16]:

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

```
10.2016
```

Coeffecient

In [17]:

```
coeff=pd.DataFrame(lr.coef_,g.columns,columns=['Co-effecient'])
coeff
```

Out[17]:

	Co-effecient
MonthYear	0.0

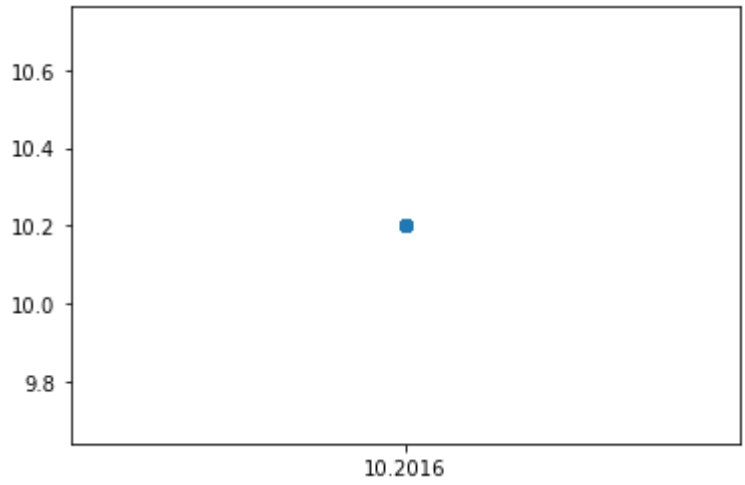
Best Fit line

In [18]:

```
prediction=lr.predict(g_test)
plt.scatter(h_test,prediction)
```

Out[18]:

```
<matplotlib.collections.PathCollection at 0x1ed3fb862e0>
```



To find score

In [19]:

```
print(lr.score(g_test,h_test))
```

```
1.0
```

Import Lasso and ridge

In [20]:

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

Ridge

In [21]:

```
ri=Ridge(alpha=5)
ri.fit(g_train,h_train)
```

Out[21]:

```
Ridge(alpha=5)
```

In [22]:

```
ri.score(g_test,h_test)
```

Out[22]:

```
1.0
```

In [23]:

```
ri.score(g_train,h_train)
```

Out[23]:

```
1.0
```

Lasso

In [24]:

```
l=Lasso(alpha=6)
l.fit(g_train,h_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: 0.0, tolerance:
0.0
  model = cd_fast.enet_coordinate_descent(
```

Out[24]:

```
Lasso(alpha=6)
```

In [25]:

```
l.score(g_test,h_test)
```

Out[25]:

```
1.0
```

In [26]:

```
ri.score(g_train,h_train)
```

Out[26]:

```
1.0
```

ElasticNet

In [27]:

```
from sklearn.linear_model import ElasticNet
e=ElasticNet()
e.fit(g_train,h_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: 0.0, tolerance:
0.0
  model = cd_fast.enet_coordinate_descent(
```

Out[27]:

```
ElasticNet()
```

Coeffecient,intercept

In [28]:

```
print(e.coef_)
```

```
[0.]
```

In [29]:

```
print(e.intercept_)
```

```
10.2016
```

Prediction

In [30]:

```
d=e.predict(g_test)
d
```

Out[30]:

```
array([10.2016, 10.2016, 10.2016, 10.2016, 10.2016, 10.2016])
```

In [31]:

```
print(e.score(g_test,h_test))
```

```
1.0
```

Evaluation

In [34]:

```
from sklearn import metrics
print("Mean Absolute error:",metrics.mean_absolute_error(h_test,d))
```

```
Mean Absolute error: 0.0
```

In [35]:

```
print("Mean Squared error:",metrics.mean_squared_error(h_test,d))
```

```
Mean Squared error: 0.0
```

In [36]:

```
print("Mean Squared error:",np.sqrt(metrics.mean_squared_error(h_test,d)))
```

```
Mean Squared error: 0.0
```

In [ ]:

```

```