Problem Statement:

A real estate agent want to help to predict the house price for regions in USA.He gave us the dataset to work on to use Linear Regression modelCreate a Model that helps him to estimate of what the house would sell for

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

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

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	06.2017	9.0	Sweden	29650.0	Gothenburg	12.0	Checkout	6322.323	
7654	06.2017	9.0	Sweden	29650.0	Gothenburg	16.0	Customer Services	4270.479	
7655	06.2017	9.0	Sweden	29650.0	Gothenburg	11.0	Delivery	0	
7656	06.2017	9.0	Sweden	29650.0	Gothenburg	17.0	others	2224.929	
7657	06.2017	9.0	Sweden	29650.0	Gothenburg	18.0	all	39652.2	

7650 rows × 13 columns

In [3]:
```
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 [4]:
```
df.head()
```

Out[4]:

	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	39
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen	1582.941	0.0	8
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other	47.205	0.0	43
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	0.0	30
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	0.0	16

Data cleaning and Pre-Processing

In [5]:
```
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 [6]:
```
df.describe()
```

Out[6]:

	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 [7]:
```
df.dropna(axis='columns')
```

Out[7]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	Hours
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	06.2017	9.0	Sweden	29650.0	Gothenburg	12.0	Checkout	6322.323	
7654	06.2017	9.0	Sweden	29650.0	Gothenburg	16.0	Customer	4270.479	

In [8]:
```
a = df.dropna(axis='columns')

a.columns
```

Out[8]:
```
Index(['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
       'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
       'Area (m2)', 'Opening hours'],
      dtype='object')
```

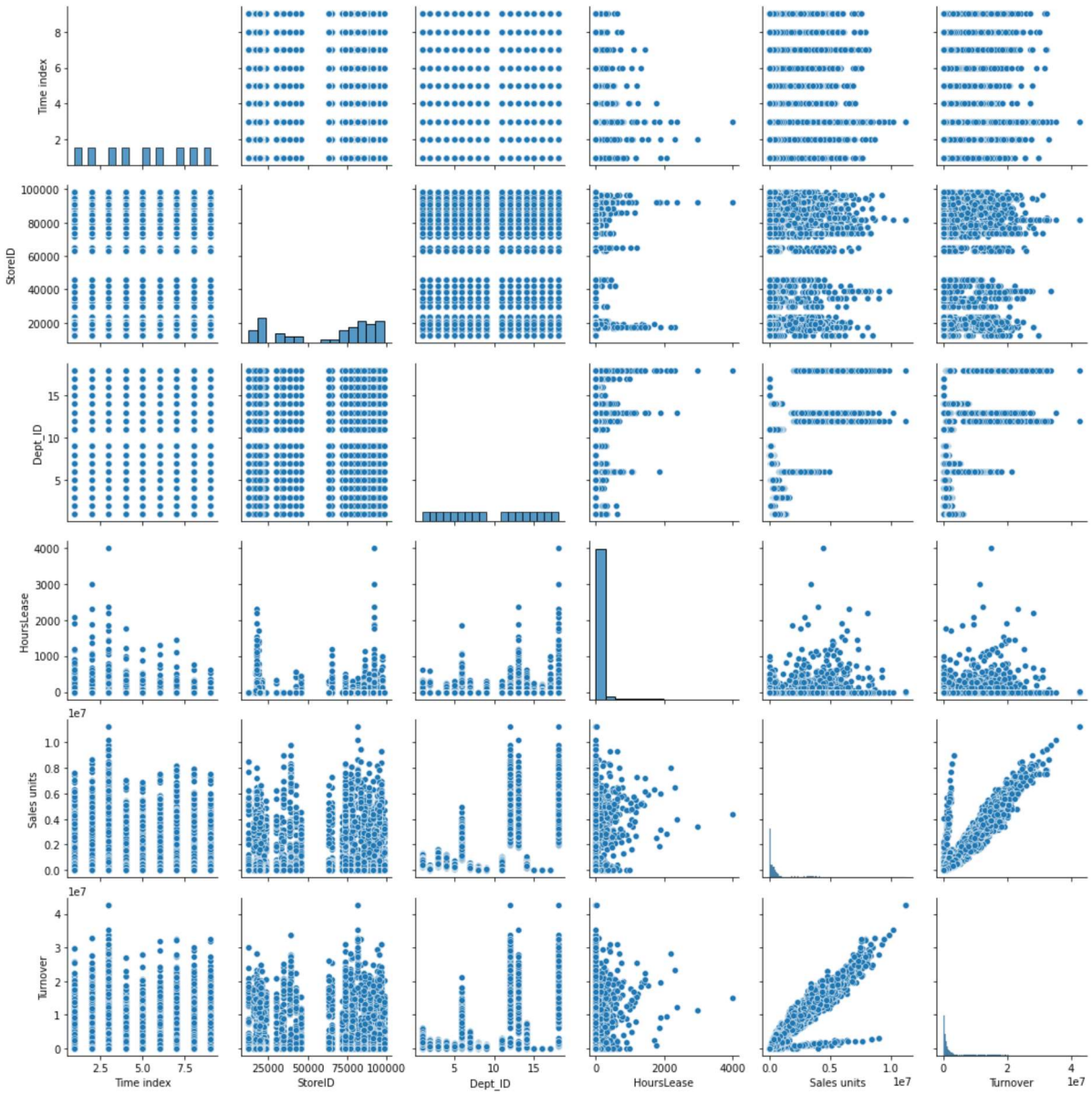
In [9]:
```
df.columns
```

Out[9]:
```
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 [10]:
```
sns.pairplot(df)
```

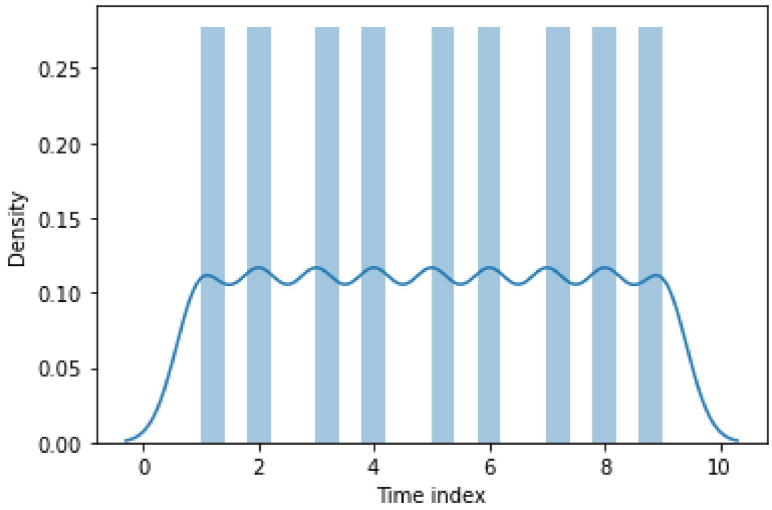
Out[10]:
```
<seaborn.axisgrid.PairGrid at 0x2695e974130>
```



In [11]:
```
sns.distplot(df['Time index'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

Out[11]:
```
<AxesSubplot:xlabel='Time index', ylabel='Density'>
```

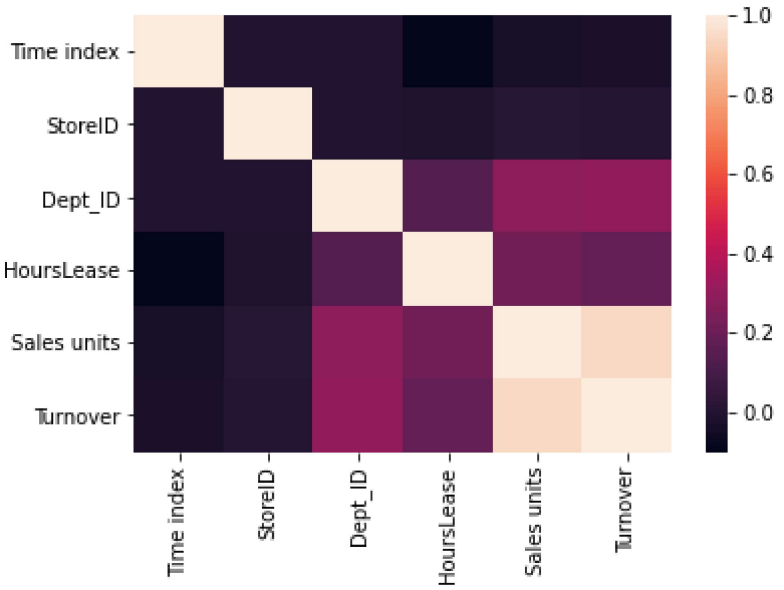


In [12]:
```
df1=df[['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
       'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
       'Area (m2)', 'Opening hours']]
```

Plot Using Heat Map

In [13]:
```
sns.heatmap(df1.corr())
```

Out[13]: <AxesSubplot:>



In [14]:
```
df1.fillna(1)
```

Out[14]:

	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	06.2017	9.0	Sweden	29650.0	Gothenburg	12.0	Checkout	6322.323	
7654	06.2017	9.0	Sweden	29650.0	Gothenburg	16.0	Customer Services	4270.479	
7655	06.2017	9.0	Sweden	29650.0	Gothenburg	11.0	Delivery	0	
7656	06.2017	9.0	Sweden	29650.0	Gothenburg	17.0	others	2224.929	
7657	06.2017	9.0	Sweden	29650.0	Gothenburg	18.0	all	39652.2	

7650 rows × 13 columns

To Train The Model-Model Building

we are going to train Linera Regression Model;We need to split out data into two variables x and y where x is independent variable(input) and y is dependent on x(output) we could ignore address column as it required for our model

In [15]:
```
x=df1[[ 'Time index', 'StoreID', 'Dept_ID',
        'HoursLease', 'Sales units']]
y=df1['Turnover']
```

To Split my dataset into training and test data

In [16]:
```
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 [17]:
```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

Out[17]:
```
LinearRegression()
```

In [18]:
```
lr.intercept_
```

Out[18]:
```
-281388.2319653798
```

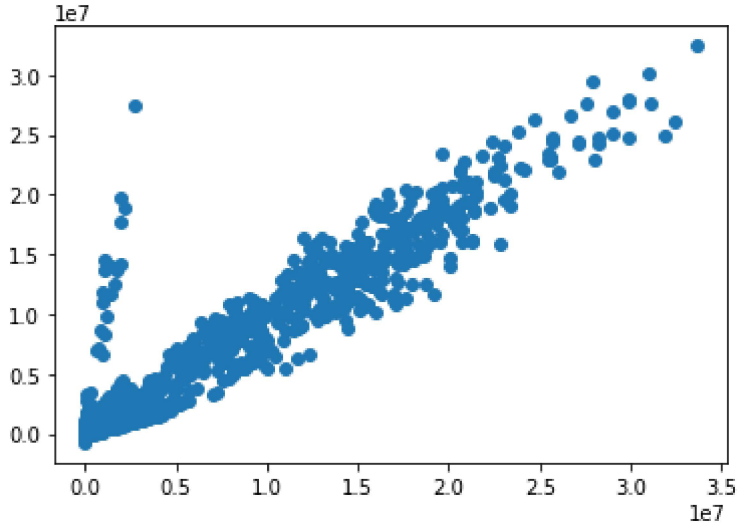
In [19]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[19]:

	Co-efficient
Time index	33037.385975
StoreID	-0.070475
Dept_ID	35535.512910
HoursLease	-1105.595348
Sales units	3.254017

In [20]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[20]: <matplotlib.collections.PathCollection at 0x269622b66a0>



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

Out[21]: 0.921686140736523

ACCURACY

In [22]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [23]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
rr.score(x_train,y_train)
```

Out[23]: 0.8883227129650566

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

Out[24]: 0.9216862170543277

In [25]:
```
la = Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[25]: Lasso(alpha=10)

In [26]:
```
la.score(x_test,y_test)
```

Out[26]: 0.9216861539339466

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

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

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

```
[ 3.07213026e+04 -6.99318013e-02  3.48793771e+04 -1.10863061e+03
  3.25454619e+00]
```

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

```
-264040.1513742958
```

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

```
[  414554.72191403   441721.62303423  2153399.44651863 ...
   358035.92945212 11977032.76391358  1444481.30742936]
```

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

```
0.921703674703704
```

In [32]:
```
# Evaluation Metrics
from sklearn import metrics
```

In [33]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 942123.2733312075
```

In [34]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 2935304509574.557
```

In [35]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prec
```

```
Root Mean Squared Error: 1713273.0399952475
```