A real estate agent want help to predict the house price for regions in Usa.he gave us the dataset to work on to use linear Regression model.Create a model that helps him to estimate

Data Collection

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

In [28]:
```
#import the dataset
data=pd.read_csv(r"C:\Users\user\Desktop\Vicky\6_Salesworkload1.csv")[0:500]
```

In [29]:
```
#to display top 10 rows
data.head()
```

Out[29]:

	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

In [30]:
```
#to display null values
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   MonthYear      500 non-null    object 
 1   Time index     500 non-null    float64
 2   Country        500 non-null    object 
 3   StoreID        500 non-null    float64
 4   City           500 non-null    object 
 5   Dept_ID        500 non-null    float64
 6   Dept. Name     500 non-null    object 
 7   HoursOwn       500 non-null    object 
 8   HoursLease     500 non-null    float64
 9   Sales units    500 non-null    float64
 10  Turnover       500 non-null    float64
 11  Customer       0 non-null      float64
 12  Area (m2)      500 non-null    object 
 13  Opening hours  500 non-null    object 
dtypes: float64(7), object(7)
memory usage: 54.8+ KB
```

In [31]:
```
data.shape
```

Out[31]:
```
(500, 14)
```

In [32]:
```
#to display summary of statistics
data.describe()
```

Out[32]:

	Time index	StoreID	Dept_ID	HoursLease	Sales units	Turnover	Customer
count	500.0	500.000000	500.000000	500.000000	5.000000e+02	5.000000e+02	0.0
mean	1.0	57412.764000	9.406000	31.520000	9.397837e+05	3.153113e+06	NaN
std	0.0	32104.273482	5.350366	142.134408	1.486945e+06	5.165524e+06	NaN
min	1.0	15552.000000	1.000000	0.000000	0.000000e+00	0.000000e+00	NaN
25%	1.0	20891.000000	5.000000	0.000000	5.200250e+04	2.345122e+05	NaN
50%	1.0	71991.000000	9.000000	0.000000	2.555375e+05	7.053345e+05	NaN
75%	1.0	88253.000000	14.000000	0.000000	8.903900e+05	2.542147e+06	NaN
max	1.0	96857.000000	18.000000	1896.000000	7.476680e+06	2.571973e+07	NaN

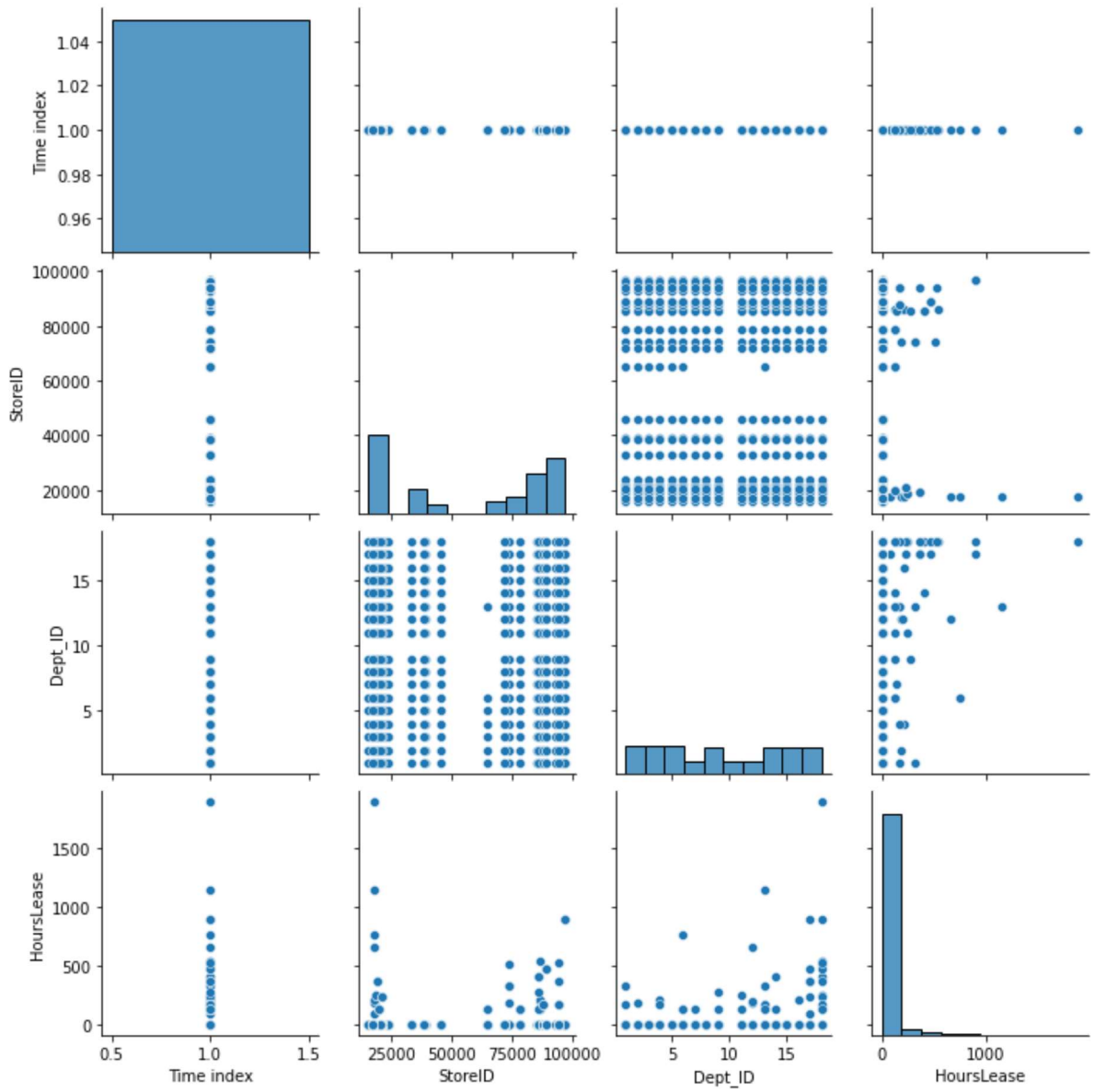
In [33]:
```
#to display columns name
data.columns
```

Out[33]:
```
Index(['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
       'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
       'Customer', 'Area (m2)', 'Opening hours'],
      dtype='object')
```

In [35]:
```
data1=data[['Time index',"StoreID","Dept_ID","HoursLease"]]
```

In [36]:
```
sns.pairplot(data1)
```

Out[36]:
```
<seaborn.axisgrid.PairGrid at 0x23885d4e400>
```

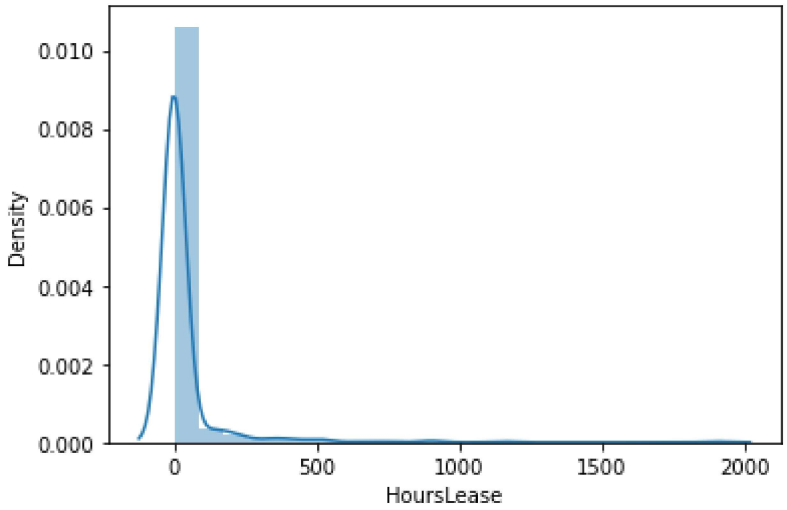


EDA and Visualization

In [37]:
```
sns.distplot(data['HoursLease'])
```

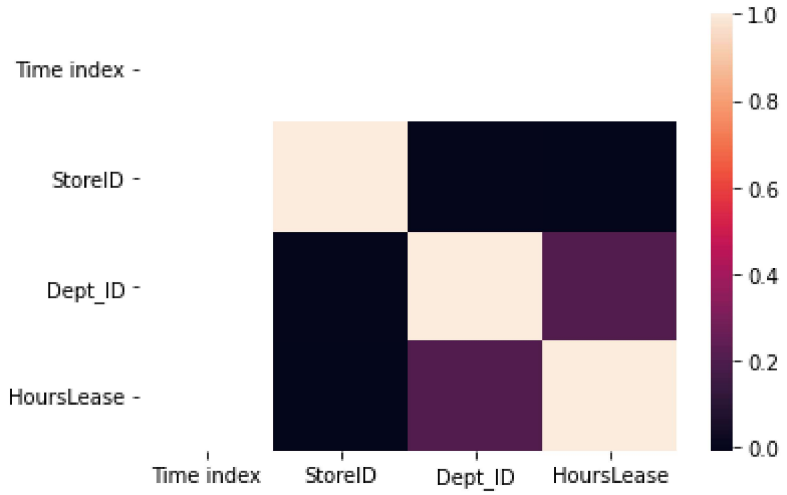
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

Out[37]: <AxesSubplot:xlabel='HoursLease', ylabel='Density'>



In [38]:
```
sns.heatmap(data1.corr())
```

Out[38]: <AxesSubplot:>



To train the model

we are going to train the linear regression model ;We need to split the two variable x and y where x in independent variable (input) and y is dependent of x(output) so we could ignore address columns as it is not requires for our model

In [39]:
```
x=data1[[ "StoreID","Dept_ID"]]
y=data1["HoursLease"]
```

In [40]:
```
#To split test and train data
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.6)
```

In [41]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[41]:
```
LinearRegression()
```

In [42]:
```
lr.intercept_
```

Out[42]:
```
-26.53459678223367
```

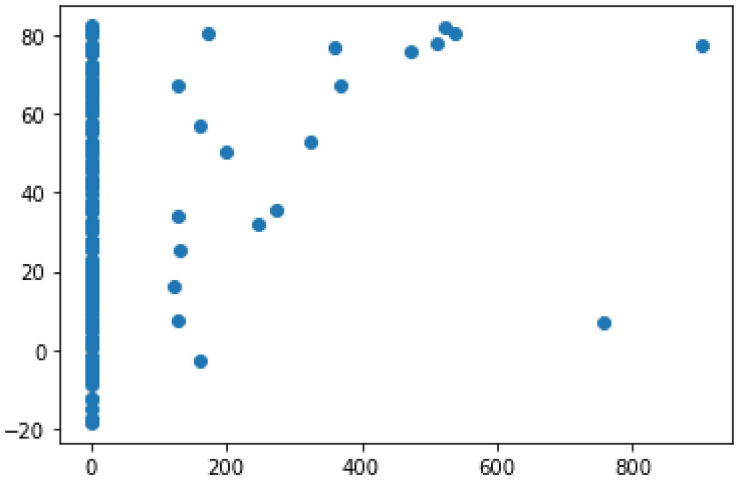
In [43]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=["Co-efficient"])
coeff
```

Out[43]:

	Co-efficient
StoreID	0.000199
Dept_ID	4.984790

In [44]:
```
prediction = lr.predict(x_train)
plt.scatter(y_train,prediction)
```

Out[44]:
```
<matplotlib.collections.PathCollection at 0x238877b4550>
```



In [45]:
```
lr.score(x_test,y_test)
```

Out[45]:
```
0.03263513258852835
```

In [46]:
```
lr.score(x_train,y_train)
```

Out[46]: 0.053687181684918595

In [47]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [48]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[48]: 0.03261629431960267

In [49]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
la.score(x_test,y_test)
```

Out[49]: 0.031739070143901094

In [ ]:

In [ ]:

In [ ]:

In [ ]: