Problem statement

predicting the house price in USA.To create a model to help him 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("Sales")
```

To display top 10 rows

In [3]:
```
df.head(10)
```

Out[3]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursLease	Sales units	Turnover	Customer	Area (m2)
0	10.2016	1.0	United Kingdom	88253.0	London (I)	1.0	Dry	3184.764	0.0	398560.0	1226244.0	NaN	953.04
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen	1582.941	0.0	82725.0	387810.0	NaN	720.48
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other	47.205	0.0	438400.0	654657.0	NaN	966.72
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	0.0	309425.0	499434.0	NaN	1053.36
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	0.0	165515.0	329397.0	NaN	1053.36
5	10.2016	1.0	United Kingdom	88253.0	London (I)	6.0	Meat	8270.316	0.0	1713310.0	5617137.0	NaN	11735.16
6	10.2016	1.0	United Kingdom	88253.0	London (I)	13.0	Food	16468.251	0.0	3107935.0	8714679.0	NaN	19865.64
7	10.2016	1.0	United Kingdom	88253.0	London (I)	7.0	Clothing	4698.471	0.0	213680.0	1615341.0	NaN	8513.52
8	10.2016	1.0	United Kingdom	88253.0	London (I)	8.0	Household	1183.272	0.0	54915.0	290400.0	NaN	4842.72
9	10.2016	1.0	United Kingdom	88253.0	London (I)	9.0	Hardware	2029.815	0.0	59260.0	450015.0	NaN	5608.8

Data Cleaning And Pre-Processing

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7658 entries, 0 to 7657
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   MonthYear      7658 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  Customer       0 non-null      float64
 12  Area (m2)      7650 non-null   object 
 13  Opening hours  7650 non-null   object 
dtypes: float64(7), object(7)
memory usage: 837.7+ KB
```

```
In [5]: # Display the statistical summary
df.describe()
```

Out[5]:

	Time index	StoreID	Dept_ID	HoursLease	Sales units	Turnover	Customer
count	7650.000000	7650.000000	7650.000000	7650.000000	7.650000e+03	7.650000e+03	0.0
mean	5.000000	61995.220000	9.470588	22.036078	1.076471e+06	3.721393e+06	NaN
std	2.582158	29924.581631	5.337429	133.299513	1.728113e+06	6.003380e+06	NaN
min	1.000000	12227.000000	1.000000	0.000000	0.000000e+00	0.000000e+00	NaN
25%	3.000000	29650.000000	5.000000	0.000000	5.457125e+04	2.726798e+05	NaN
50%	5.000000	75400.500000	9.000000	0.000000	2.932300e+05	9.319575e+05	NaN
75%	7.000000	87703.000000	14.000000	0.000000	9.175075e+05	3.264432e+06	NaN
max	9.000000	98422.000000	18.000000	3984.000000	1.124296e+07	4.271739e+07	NaN

```
In [6]: # To display the col headings
        df.columns
```

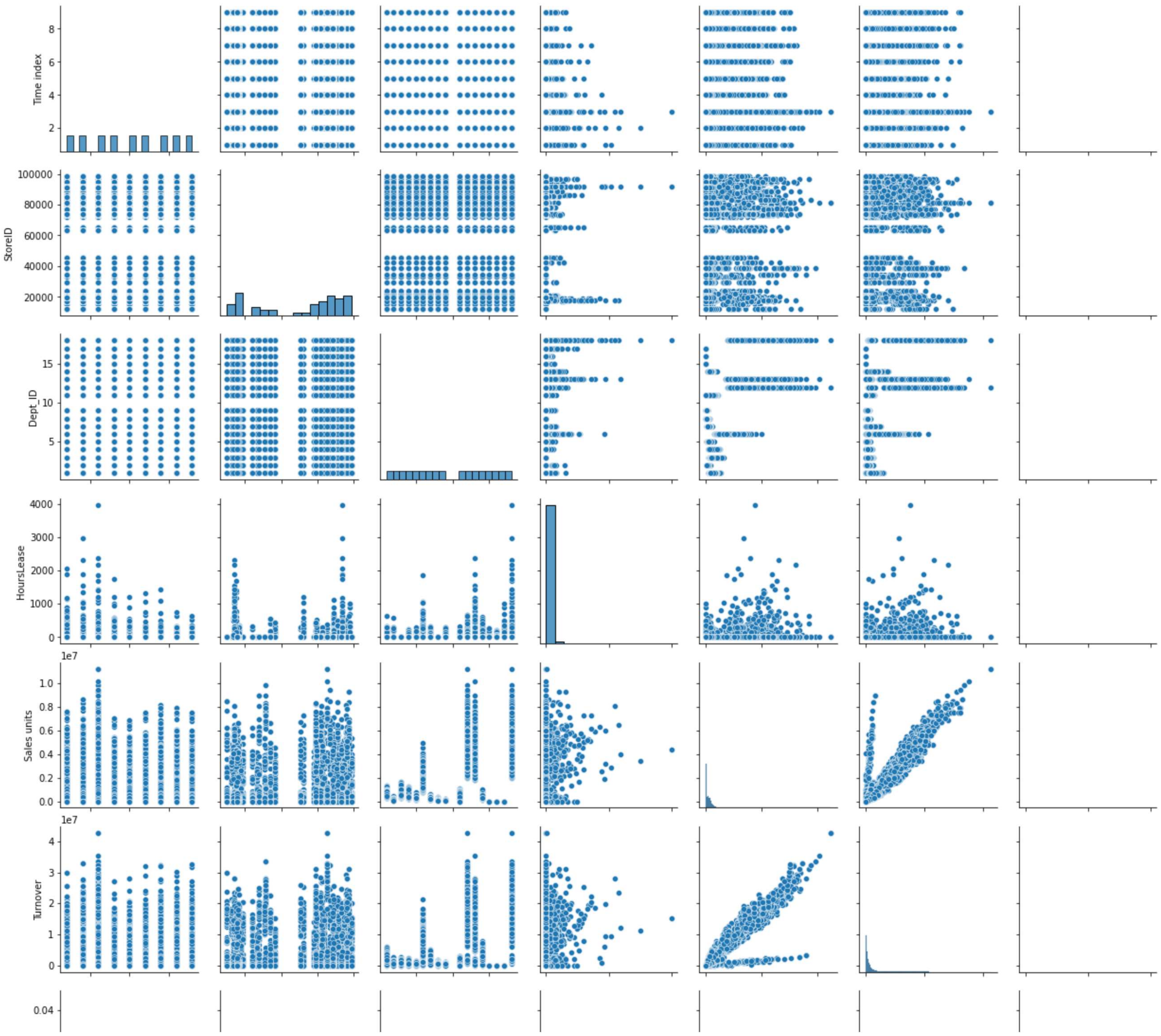
```
Out[6]: Index(['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
               'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
               'Customer', 'Area (m2)', 'Opening hours'],
              dtype='object')
```

```
In [7]: #cols=df.dropna(axis=1)
```

```
In [8]: #cols.columns
```

EDA and Visualization

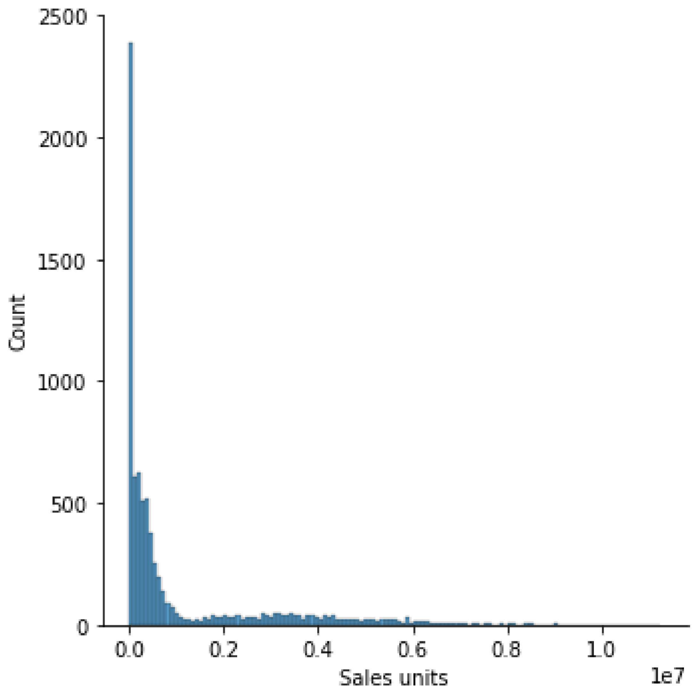
```
In [9]: sns.pairplot(df)

Out[9]: <seaborn.axisgrid.PairGrid at 0x16135d858e0>
```



Time index
StoreID
Dept_ID
HoursLease
Sales units
Turnover
1e7
1e7

```
In [10]: sns.displot(df['Sales units'])
```

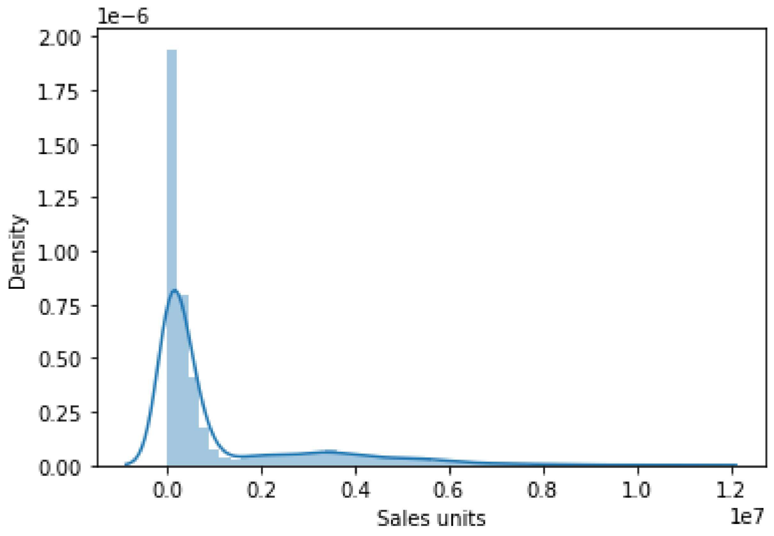
```
Out[10]: <seaborn.axisgrid.FacetGrid at 0x16139f2dbb0>
```



```
In [11]: # We use displot in older version we get distplot use displot
         sns.distplot(df['Sales units'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a dep
recated 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[11]: <AxesSubplot:xlabel='Sales units', ylabel='Density'>
```



In [12]:
```python
df1=df[['HoursLease', 'Sales units', 'Turnover',
        'Customer']]
df1
```

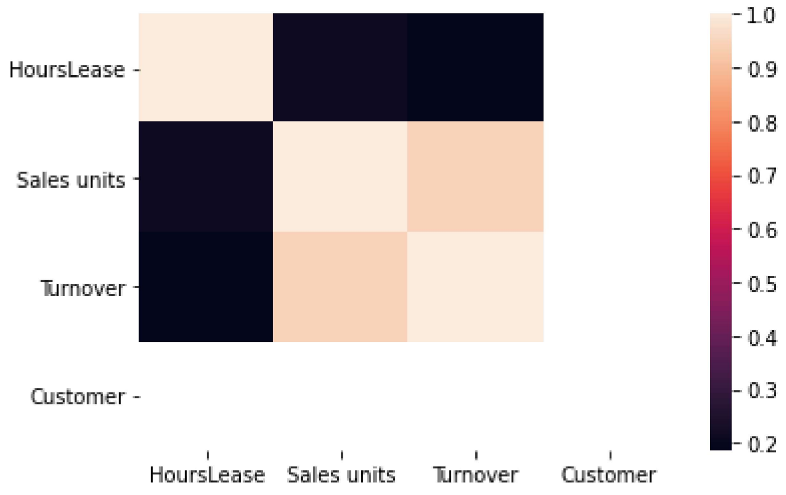
Out[12]:

	HoursLease	Sales units	Turnover	Customer
0	0.0	398560.0	1226244.0	NaN
1	0.0	82725.0	387810.0	NaN
2	0.0	438400.0	654657.0	NaN
3	0.0	309425.0	499434.0	NaN
4	0.0	165515.0	329397.0	NaN
...	...	...	...	...
7653	0.0	3886530.0	14538825.0	NaN
7654	0.0	245.0	0.0	NaN
7655	0.0	0.0	0.0	NaN
7656	0.0	245.0	0.0	NaN
7657	0.0	3886530.0	15056214.0	NaN

7658 rows × 4 columns

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

```
Out[13]: <AxesSubplot:>
```



To train the model - MODEL BUILD

Going to train linear regression model;We split our data into 2 variables x and y where x is independent var(input) and y is dependent on x(output), we could ignore address col as it is not required for our model

```
In [14]: x=df1[['HoursLease', 'Sales units', 'Turnover',
                'Customer']]
         y=df1[['Sales units']]
```

To split the dataset into test data

```
In [15]: # importing lib for splitting test data
         from sklearn.model_selection import train_test_split
```

```
In [16]: 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)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-17-b0fd2c20cba9> in <module>
      2 
      3 lr=LinearRegression()
----> 4 lr.fit(x_train,y_train)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py in fit(self, X, y, sample_weight)
    516         accept_sparse = False if self.positive else ['csr', 'csc', 'coo']
    517 
--> 518         X, y = self._validate_data(X, y, accept_sparse=accept_sparse,
    519                                    y_numeric=True, multi_output=True)
    520 

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py in _validate_data(self, X, y, reset, validate_sep
arately, **check_params)
    431                 y = check_array(y, **check_y_params)
    432             else:
--> 433                 X, y = check_X_y(X, y, **check_params)
    434             out = X, y
    435 

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64 
     65             # extra_args > 0

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in check_X_y(X, y, accept_sparse, acc
ept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, multi_output, ensure_min_sample
s, ensure_min_features, y_numeric, estimator)
    812         raise ValueError("y cannot be None")
    813 
--> 814     X = check_array(X, accept_sparse=accept_sparse,
    815                     accept_large_sparse=accept_large_sparse,
    816                     dtype=dtype, order=order, copy=copy,

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64 
     65             # extra_args > 0

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in check_array(array, accept_sparse,
accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_m
in_features, estimator)
    661
    662         if force_all_finite:
--> 663             _assert_all_finite(array,
    664                                allow_nan=force_all_finite == 'allow-nan')
    665

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in _assert_all_finite(X, allow_nan, m
sg_dtype)
    101                 not allow_nan and not np.isfinite(X).all()):
    102             type_err = 'infinity' if allow_nan else 'NaN, infinity'
--> 103             raise ValueError(
    104                     msg_err.format
    105                     (type_err,

ValueError: Input contains NaN, infinity or a value too large for dtype('float64').
```

In [ ]:
```
print(lr.intercept_)
```

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

In [ ]:
```
coeff=pd.DataFrame(lr.coef_)
coeff
```

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

In [ ]:
```
cols=df.dropna()
cols
```

In [ ]:
```

```