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(r'C:\Users\user\Downloads\6_Salesworkload1.csv')
df
```

Out[2]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	Hour
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	

7658 rows × 14 columns

In [3]:

```
df.head(10)
```

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

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]:

```
df.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	

In [6]:

```
df.columns
```

Out[6]:

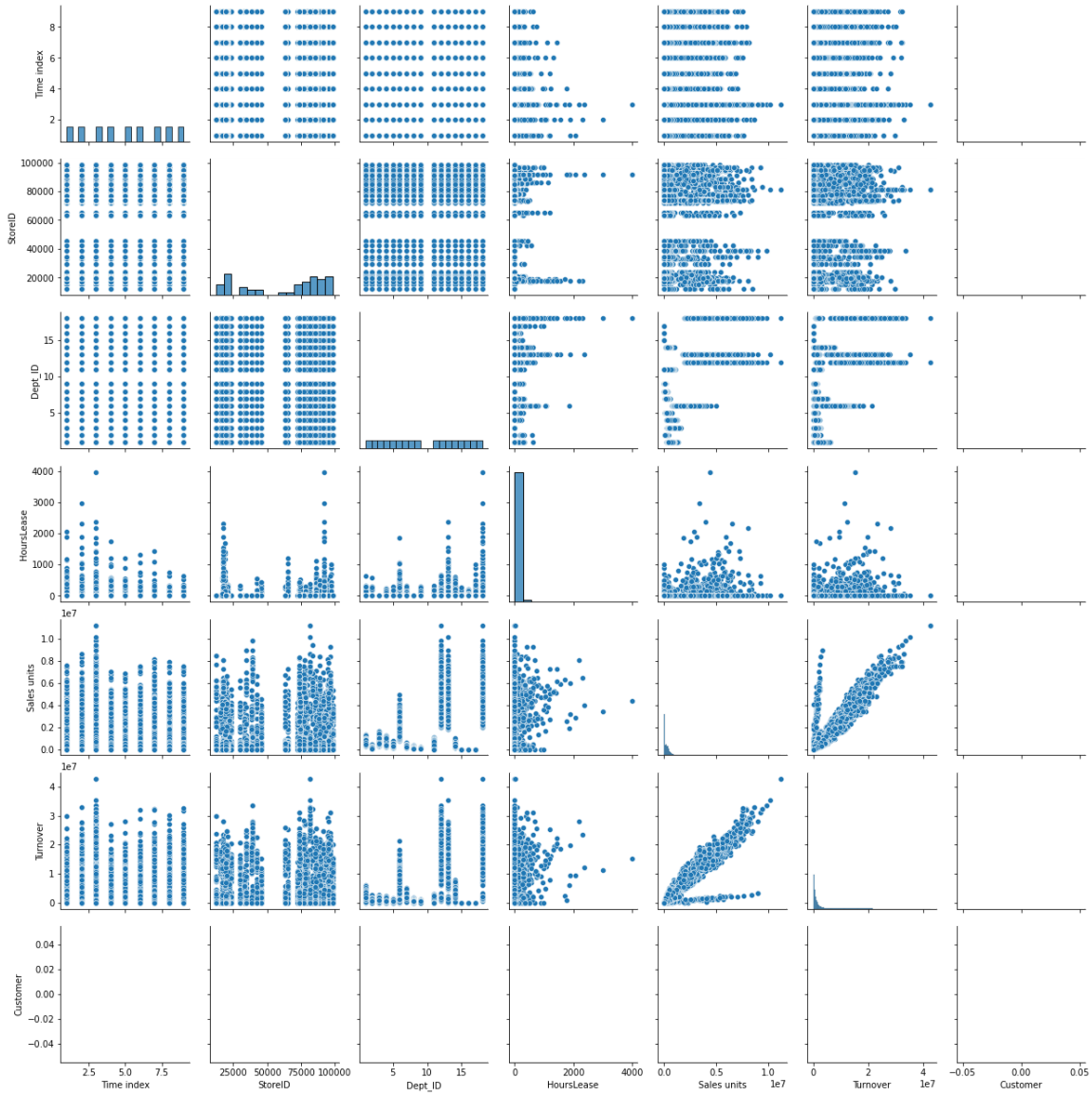
```
Index(['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
       'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
       'Customer', 'Area (m2)', 'Opening hours'],
      dtype='object')
```

In [7]:

```
sns.pairplot(df)
```

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x1f5b1ca83a0>
```



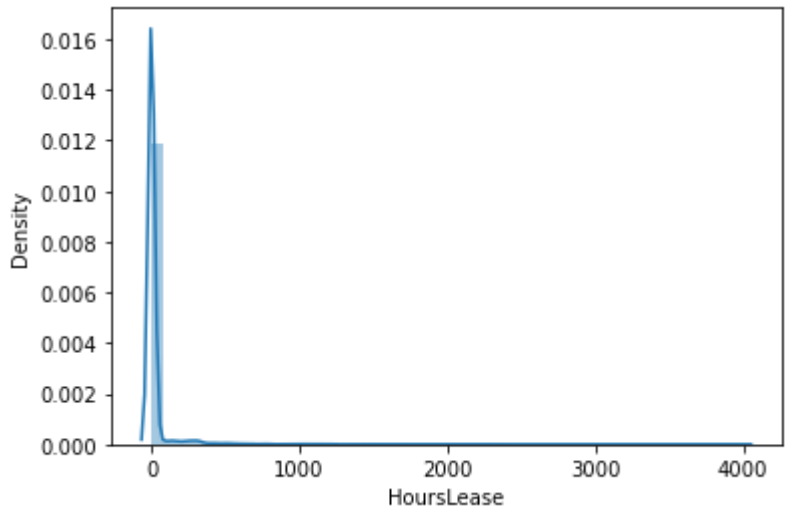
In [8]:

```
sns.distplot(df['HoursLease'])
```

```
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)
```

Out[8]:

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

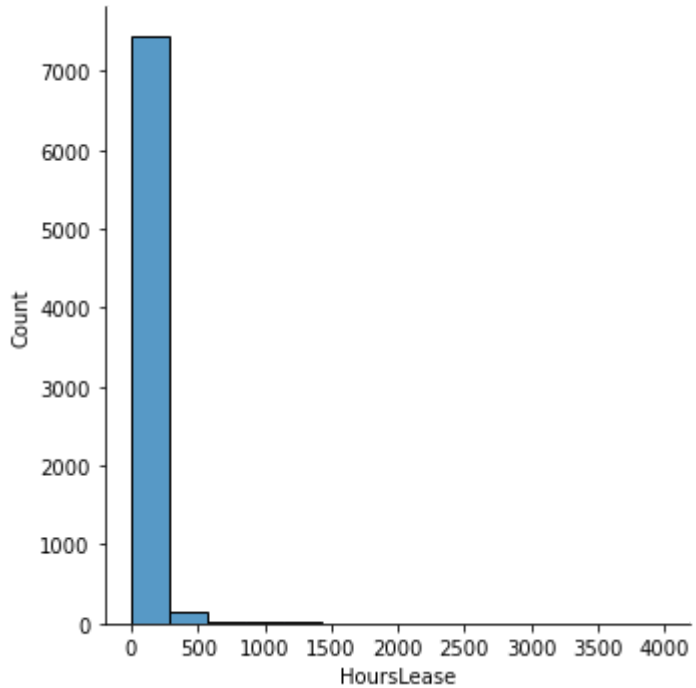


In [9]:

```
sns.displot(df["HoursLease"])
```

Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x1f5acef15e0>
```



In [10]:

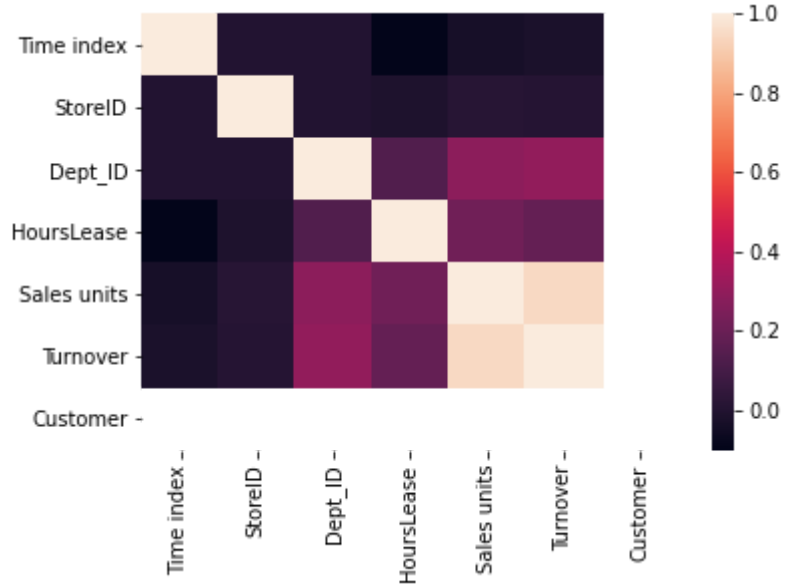
```
df1=df[['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
       'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
       'Customer', 'Area (m2)', 'Opening hours']]
```

In [11]:

```
sns.heatmap(df1.corr())
```

Out[11]:

<AxesSubplot:>



In [16]:

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

In [17]:

```
from sklearn.model_selection import train_test_split
```

In [18]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [20]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)#ValueError: Input contains NaN, infinity or a value too large for
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
<ipython-input-20-58bc3c346312> in <module>
      2
      3 lr=LinearRegression()
----> 4 lr.fit(x_train,y_train)#ValueError: Input contains NaN, infinity o
r a value too large for dtype('float64')

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py i
n 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_spar
se,
    519                                    y_numeric=True, multi_output=Tr
ue)
    520

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py in _validate_da
ta(self, X, y, reset, validate_separately, **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, accept_large_sparse, dtype, order, copy, fo
rce_all_finite, ensure_2d, allow_nd, multi_output, ensure_min_samples, ens
ure_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_min_feat
ures, estimator)
    661
    662         if force_all_finite:
```

```
--> 663             _assert_all_finite(array,
    664                                allow_nan=force_all_finite == 'allo
w-nan')
    665

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

ValueError: Input contains NaN, infinity or a value too large for dtype('f
loat64').
```

In [21]:

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

```
---------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call las
t)
<ipython-input-21-182bb45ab960> in <module>
----> 1 print(lr.intercept_)

AttributeError: 'LinearRegression' object has no attribute 'intercept_'
```

In [22]:

```
coef= pd.DataFrame(lr.coef_)
coef
```

```
---------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call las
t)
<ipython-input-22-0ff321f0a2a5> in <module>
----> 1 coef= pd.DataFrame(lr.coef_)
      2 coef

AttributeError: 'LinearRegression' object has no attribute 'coef_'
```

In [23]:

```
print(lr.score(x_test,y_test))
```

```
---------------------------------------------------------------------------
-
NotFittedError                            Traceback (most recent call las
t)
<ipython-input-23-6bc23016a4ce> in <module>
----> 1 print(lr.score(x_test,y_test))

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py in score(self,
X, y, sample_weight)
    551
    552         from .metrics import r2_score
--> 553         y_pred = self.predict(X)
    554         return r2_score(y, y_pred, sample_weight=sample_weight)
    555

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py i
n predict(self, X)
    236             Returns predicted values.
    237         """
--> 238         return self._decision_function(X)
    239
    240     _preprocess_data = staticmethod(_preprocess_data)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py i
n _decision_function(self, X)
    216
    217     def _decision_function(self, X):
--> 218         check_is_fitted(self)
    219
    220         X = check_array(X, accept_sparse=['csr', 'csc', 'coo'])

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_is_fitted(estimator, attributes, msg, all_or_any)
   1039
   1040     if not attrs:
-> 1041         raise NotFittedError(msg % {'name': type(estimator).__name
__})
   1042
   1043

NotFittedError: This LinearRegression instance is not fitted yet. Call 'fi
t' with appropriate arguments before using this estimator.
```

In [24]:

```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

```
---------------------------------------------------------------------------
-
NotFittedError                            Traceback (most recent call las
t)
<ipython-input-24-10d398fd7dc3> in <module>
----> 1 prediction = lr.predict(x_test)
      2 plt.scatter(y_test,prediction)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py i
n predict(self, X)
    236             Returns predicted values.
    237         """
--> 238         return self._decision_function(X)
    239
    240     _preprocess_data = staticmethod(_preprocess_data)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py i
n _decision_function(self, X)
    216
    217     def _decision_function(self, X):
--> 218         check_is_fitted(self)
    219
    220         X = check_array(X, accept_sparse=['csr', 'csc', 'coo'])

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_is_fitted(estimator, attributes, msg, all_or_any)
   1039
   1040     if not attrs:
-> 1041         raise NotFittedError(msg % {'name': type(estimator).__name
__})
   1042
   1043

NotFittedError: This LinearRegression instance is not fitted yet. Call 'fi
t' with appropriate arguments before using this estimator.
```