

kaviyadevi 20106064

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

```
In [81]: #to import dataset
data=pd.read_csv(r"C:\Users\user\Downloads\6_Salesworkload1 - 6_Salesworkload1.csv")
data
```

Out[81]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursLeas
0	10.2016	1.0	United Kingdom	88253.0	London (I)	1.0	Dry	3184.764	0.
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen	1582.941	0.
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other	47.205	0.
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	0.
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	0.
...
7653	6.2017	9.0	Sweden	29650.0	Gothenburg	12.0	Checkout	6322.323	0.
7654	6.2017	9.0	Sweden	29650.0	Gothenburg	16.0	Customer Services	4270.479	0.
7655	6.2017	9.0	Sweden	29650.0	Gothenburg	11.0	Delivery	0	0.
7656	6.2017	9.0	Sweden	29650.0	Gothenburg	17.0	others	2224.929	0.
7657	6.2017	9.0	Sweden	29650.0	Gothenburg	18.0	all	39652.2	0.

7658 rows × 14 columns

```
In [82]: #to display top 5 rows
data.head()
```

Out[82]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursLease	S
0	10.2016	1.0	United Kingdom	88253.0	London (l)	1.0	Dry	3184.764	0.0	398
1	10.2016	1.0	United Kingdom	88253.0	London (l)	2.0	Frozen	1582.941	0.0	82
2	10.2016	1.0	United Kingdom	88253.0	London (l)	3.0	other	47.205	0.0	438
3	10.2016	1.0	United Kingdom	88253.0	London (l)	4.0	Fish	1623.852	0.0	309
4	10.2016	1.0	United Kingdom	88253.0	London (l)	5.0	Fruits & Vegetables	1759.173	0.0	165



DATA CLEANING AND PREPROCESSING

```
In [83]: data.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 [84]: #to display summary of statistics
data.describe()
```

```
Out[84]:
```

	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.
mean	5.000000	61995.220000	9.470588	22.036078	1.076471e+06	3.721393e+06	Na
std	2.582158	29924.581631	5.337429	133.299513	1.728113e+06	6.003380e+06	Na
min	1.000000	12227.000000	1.000000	0.000000	0.000000e+00	0.000000e+00	Na
25%	3.000000	29650.000000	5.000000	0.000000	5.457125e+04	2.726798e+05	Na
50%	5.000000	75400.500000	9.000000	0.000000	2.932300e+05	9.319575e+05	Na
75%	7.000000	87703.000000	14.000000	0.000000	9.175075e+05	3.264432e+06	Na
max	9.000000	98422.000000	18.000000	3984.000000	1.124296e+07	4.271739e+07	Na

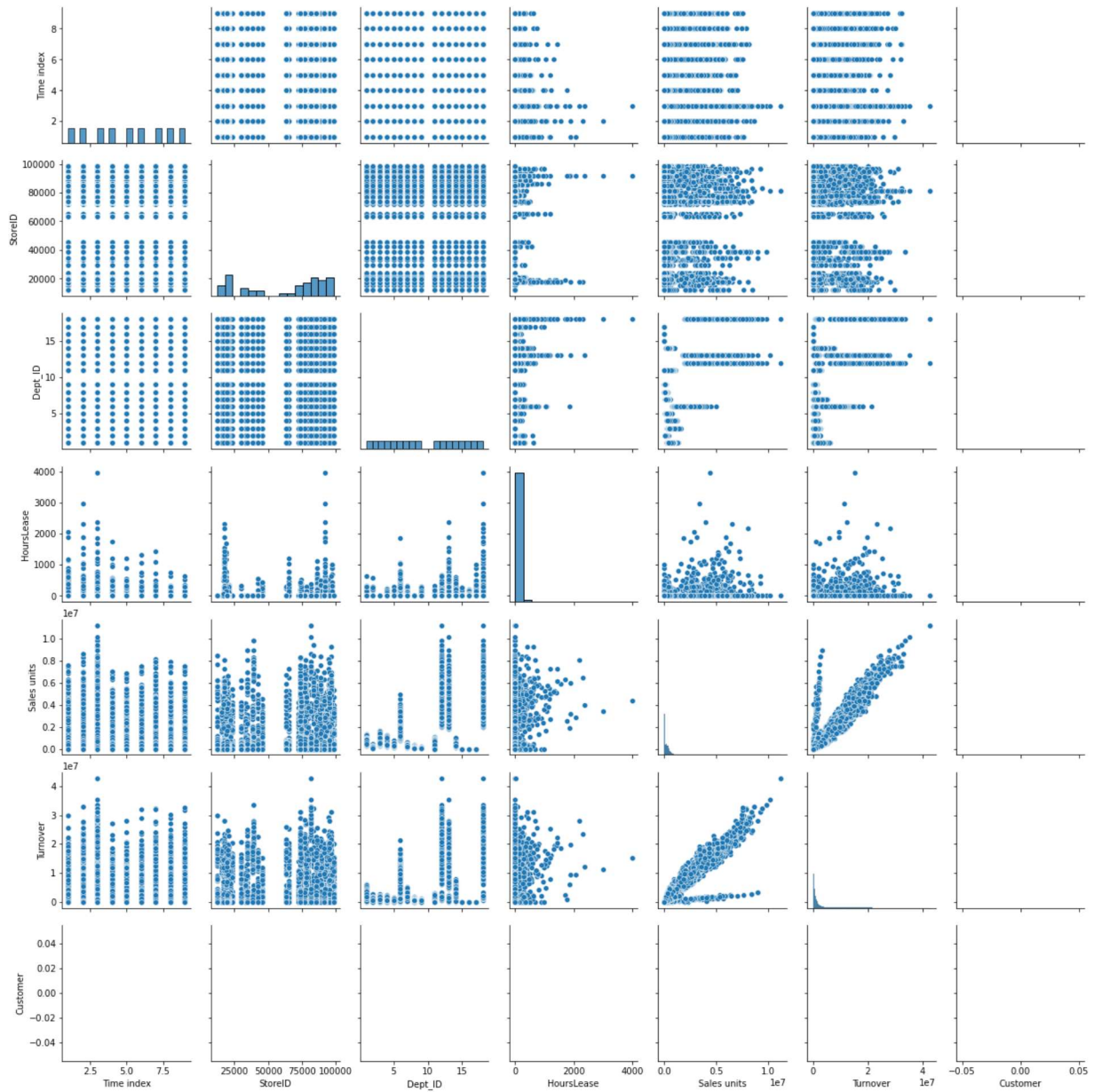
```
In [85]: #to display the column heading
data.columns
```

```
Out[85]: Index(['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
                'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
                'Customer', 'Area (m2)', 'Opening hours'],
                dtype='object')
```

EDA and DATA VISUALIZATION

```
In [86]: sns.pairplot(data)
```

```
Out[86]: <seaborn.axisgrid.PairGrid at 0x1e5b2a7ed30>
```

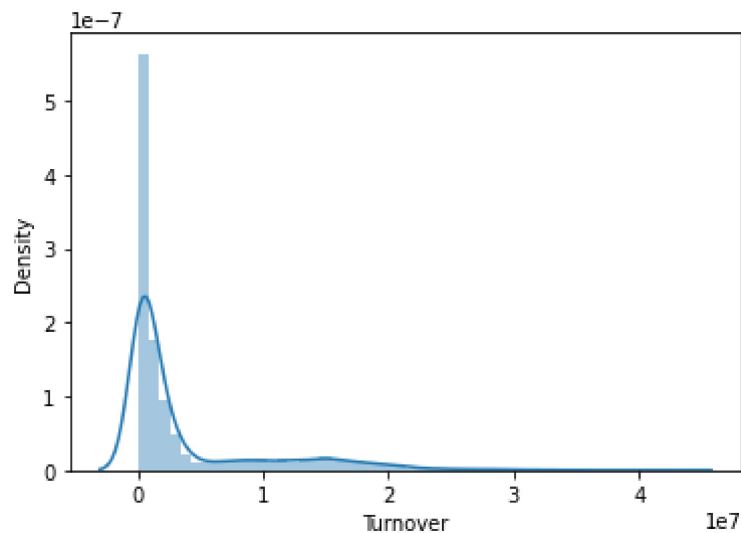


```
In [88]: sns.distplot(data['Turnover'])
```

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 function for histograms).

```
warnings.warn(msg, FutureWarning)
```

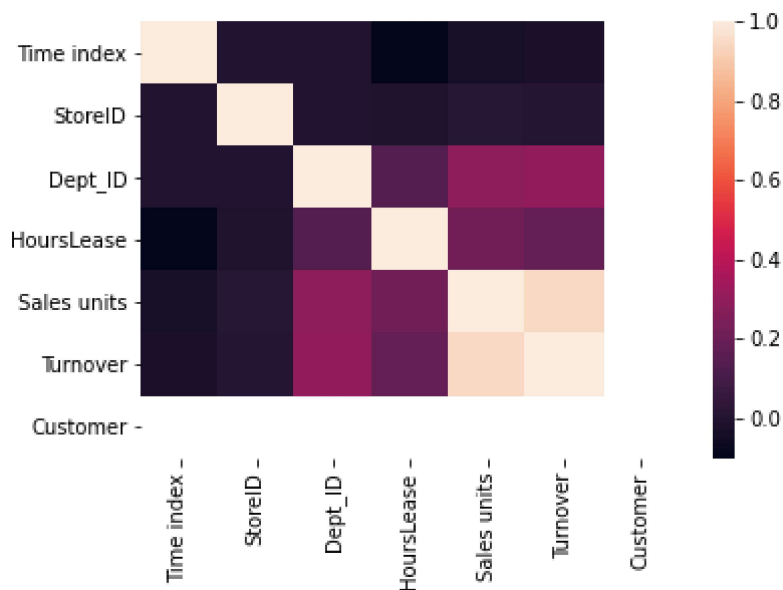
```
Out[88]: <AxesSubplot:xlabel='Turnover', ylabel='Density'>
```



```
In [95]: df=data[['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',  
                'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',  
                'Customer', 'Area (m2)', 'Opening hours']]
```

```
In [96]: sns.heatmap(df.corr())
```

```
Out[96]: <AxesSubplot:>
```



MODEL TRAINING

```
In [100]: x=df[['MonthYear', 'Time index', 'StoreID', 'Dept_ID', 'HoursOwn', 'HoursLease', 'Customer']]
           y=df[['Turnover']]
```

```
In [101]: #to split my dataset into training and test
```

```
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 [102]: from sklearn.linear_model import LinearRegression
```

```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

```
-----
ValueError                                Traceback (most recent call last)
<ipython-input-102-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', 'co
o']
    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(se
lf, 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, force_all_fi
nite, ensure_2d, allow_nd, multi_output, ensure_min_samples, ensure_min_feature
s, 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_features, estimato
r)
    614         array = array.astype(dtype, casting="unsafe", copy=
False)
    615         else:
--> 616         array = np.asarray(array, order=order, dtype=dtype)
    617     except ComplexWarning as complex_warning:
    618         raise ValueError("Complex data not supported\n"

```

C:\ProgramData\Anaconda3\lib\site-packages\numpy\core_asarray.py in asarray(a, dtype, order, like)

```

    100         return _asarray_with_like(a, dtype=dtype, order=order, like=lik
e)
    101
--> 102     return array(a, dtype, copy=False, order=order)
    103
    104

```

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in __array__(self, dtype)

```

    1897
    1898     def __array__(self, dtype=None) -> np.ndarray:
-> 1899         return np.asarray(self._values, dtype=dtype)
    1900
    1901     def __array_wrap__(

```

C:\ProgramData\Anaconda3\lib\site-packages\numpy\core_asarray.py in asarray(a, dtype, order, like)

```

    100         return _asarray_with_like(a, dtype=dtype, order=order, like=lik
e)
    101
--> 102     return array(a, dtype, copy=False, order=order)
    103
    104

```

ValueError: could not convert string to float: '- - - -'

In [77]: *#to find intercept*
print(lr.intercept_)

[18.04819595]

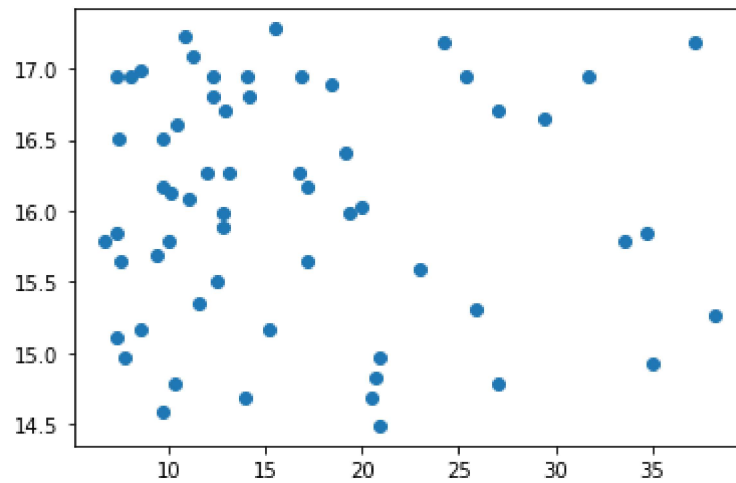
In [78]: coeff = pd.DataFrame(lr.coef_, x.columns, columns=['Co-efficient'])
coeff

Out[78]:

	Co-efficient
Age	-0.048056


```
In [79]: prediction = lr.predict(x_test)
plt.scatter(y_test, prediction)
```

Out[79]: <matplotlib.collections.PathCollection at 0x1e5b2a50910>



```
In [80]: print(lr.score(x_test, y_test))
```

-0.022059454830758662