Problem statement

Data collection ¶

In [1]:

```
#to import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [73]:

```
df=pd.read_csv(r"E:\Dataset\6_Salesworkload1.csv")[0:500]
df
```

Out[73]:

	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	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
...	...	...	...	...	...	...	...	...	
495	10.2016	1.0	Italy	64983.0	Milano	3.0	other	47.205	0
496	10.2016	1.0	Italy	64983.0	Milano	4.0	Fish	2451.513	0
497	10.2016	1.0	Italy	64983.0	Milano	5.0	Fruits & Vegetables	1944.846	0
498	10.2016	1.0	Italy	64983.0	Milano	6.0	Meat	11980.629	122
499	10.2016	1.0	Italy	64983.0	Milano	13.0	Food	23665.44	122

500 rows × 14 columns

In [74]:

```
df.head()
```

Out[74]:

	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

In [75]:

```
df.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 [76]:

```
#to display summary of statistics
df.describe()
```

Out[76]:

	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 [77]:

```
#to display cloumn heading
df.columns
```

Out[77]:

```
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 VISUALIZATION

In [78]:

```
df1=df[['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID','Dept. Name']]
df1
```

Out[78]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name
0	10.2016	1.0	United Kingdom	88253.0	London (I)	1.0	Dry
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables
...	...	...	...	...	...	...	...
495	10.2016	1.0	Italy	64983.0	Milano	3.0	other
496	10.2016	1.0	Italy	64983.0	Milano	4.0	Fish
497	10.2016	1.0	Italy	64983.0	Milano	5.0	Fruits & Vegetables
498	10.2016	1.0	Italy	64983.0	Milano	6.0	Meat
499	10.2016	1.0	Italy	64983.0	Milano	13.0	Food

500 rows × 7 columns

In [79]:

```
df1.fillna(1)
```

Out[79]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name
0	10.2016	1.0	United Kingdom	88253.0	London (I)	1.0	Dry
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables
...	...	...	...	...	...	...	...
495	10.2016	1.0	Italy	64983.0	Milano	3.0	other
496	10.2016	1.0	Italy	64983.0	Milano	4.0	Fish
497	10.2016	1.0	Italy	64983.0	Milano	5.0	Fruits & Vegetables
498	10.2016	1.0	Italy	64983.0	Milano	6.0	Meat
499	10.2016	1.0	Italy	64983.0	Milano	13.0	Food

500 rows × 7 columns

In [80]:

```
df1.info()
```

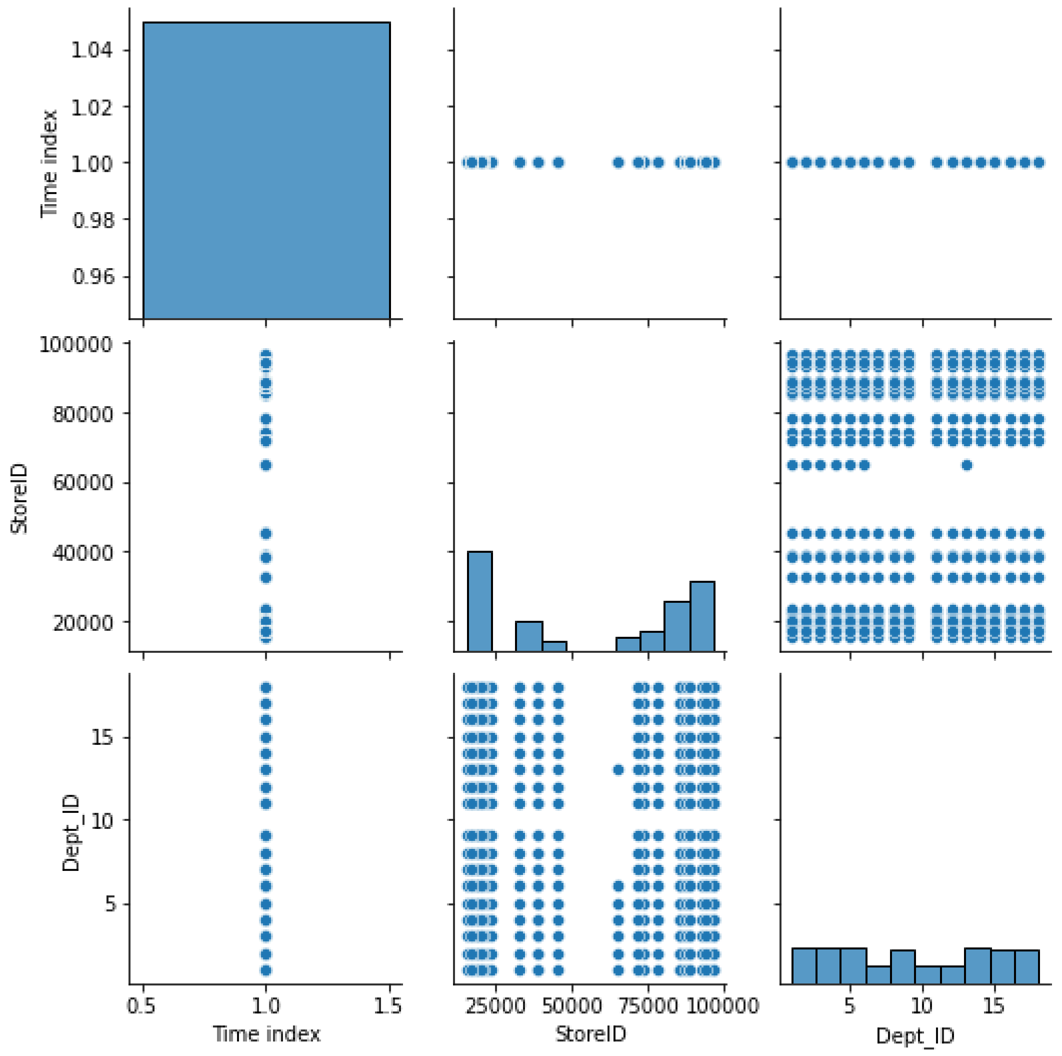
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   MonthYear    500 non-null    object 
 1   Time index   500 non-null    float64
 2   Country      500 non-null    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 
dtypes: float64(3), object(4)
memory usage: 27.5+ KB
```

In [81]:

```
sns.pairplot(df1)
```

Out[81]:

```
<seaborn.axisgrid.PairGrid at 0x165be45dac0>
```



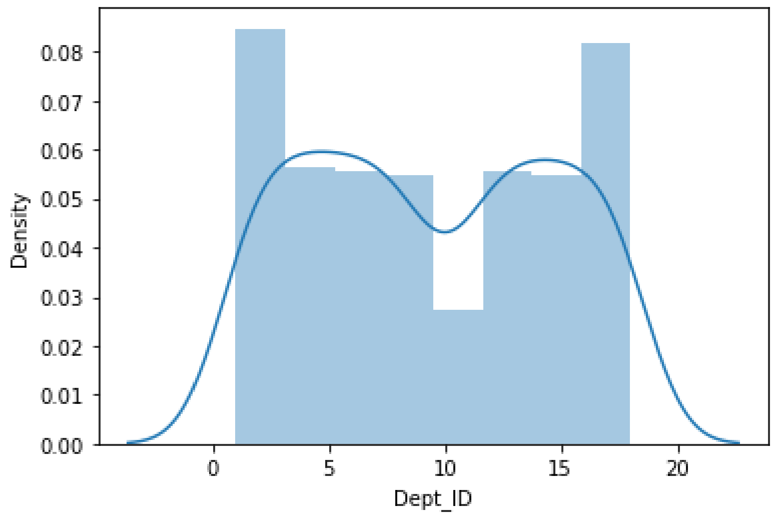
In [83]:

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

```
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[83]:

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

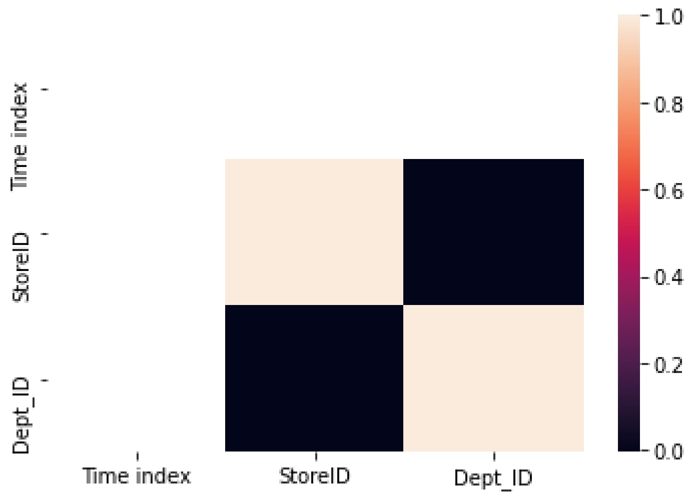


In [84]:

```
data=df1[[ 'Time index', 'StoreID', 'Dept_ID']]
sns.heatmap(data.corr())
```

Out[84]:

```
<AxesSubplot:>
```



to Train the model-Model buliding

we are going to split our data into two variable where x is a independent and y is dependent on x

In [85]:

```
x=data[['Time index', 'StoreID', 'Dept_ID']]
y=data['Dept_ID']
```

In [86]:

```
# to split my dataset into 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=0.3)
```

In [87]:

```
from sklearn.linear_model import LinearRegression

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

Out[87]:

```
LinearRegression()
```

In [88]:

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

```
1.0658141036401503e-14
```

In [89]:

```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-effecient'])
coeff
```

Out[89]:

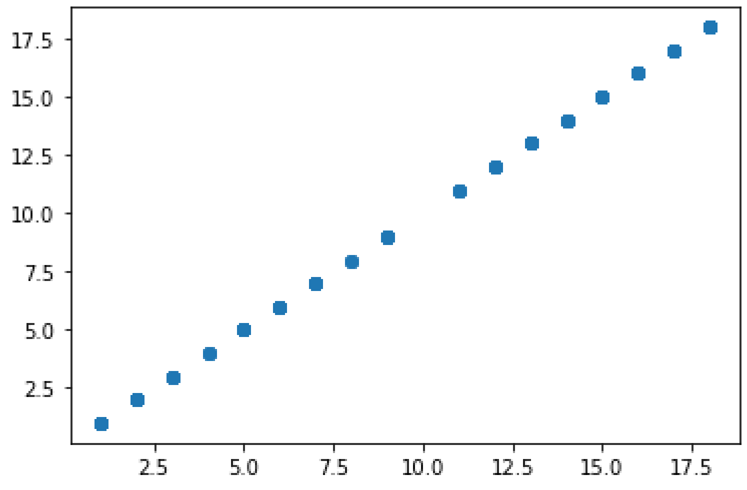
	Co-effecient
Time index	0.000000e+00
StoreID	-1.247199e-19
Dept_ID	1.000000e+00

In [90]:

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

Out[90]:

```
<matplotlib.collections.PathCollection at 0x165bec49730>
```



In [91]:

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

```
1.0
```

In [ ]:

```

```