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

In [2]:
```
df = pd.read_csv('Supermarket_sales_prediction.csv')
```

In [3]:
```
sdf = df.copy(deep = True)
```

In [4]:
```
sdf.head()
```

Out[4]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outl
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	

In [5]:
```
sdf['Item_Type'].unique()
```

Out[5]:
```
array(['Dairy', 'Soft Drinks', 'Meat', 'Fruits and Vegetables',
       'Household', 'Baking Goods', 'Snack Foods', 'Frozen Foods',
       'Breakfast', 'Health and Hygiene', 'Hard Drinks', 'Canned',
       'Breads', 'Starchy Foods', 'Others', 'Seafood'], dtype=object)
```

In [46]:
```
sdf[sdf['Item_Type'] == 'Dairy']
```

Out[46]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP
0	FDA15	9.300	Low Fat	0.016047	Dairy	249.8092
11	FDA03	18.500	Regular	0.045464	Dairy	144.1102
19	FDU02	13.350	Low Fat	0.102492	Dairy	230.5352
28	FDE51	5.925	Regular	0.161467	Dairy	45.5086
30	FDV38	19.250	Low Fat	0.170349	Dairy	55.7956
...	...	...	...	...	...	...
8424	FDC39	7.405	Low Fat	0.159165	Dairy	207.1296
8447	FDS26	20.350	Low Fat	0.089975	Dairy	261.6594
8448	FDV50	14.300	Low Fat	0.123071	Dairy	121.1730
8457	FDY50	5.800	Low Fat	0.130931	Dairy	89.9172
8512	FDR26	20.700	Low Fat	0.042801	Dairy	178.3028

682 rows × 12 columns

In [4]:
```
sdf.shape
```

Out[4]: (8523, 12)

In [5]:
```
sdf.head()
```

Out[5]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614

In [6]:
```
sdf.isnull().sum()
```

Out[6]:
```
Item_Identifier                 0
Item_Weight                  1463
Item_Fat_Content                0
Item_Visibility                 0
Item_Type                       0
Item_MRP                        0
Outlet_Identifier               0
Outlet_Establishment_Year       0
Outlet_Size                  2410
Outlet_Location_Type            0
Outlet_Type                     0
Item_Outlet_Sales               0
dtype: int64
```

In [5]:
```
sdf['Item_Weight'].fillna(sdf['Item_Weight'].mean(), inplace=True)
sdf['Outlet_Size'].fillna('Unknown', inplace=True)
```

In [7]:
```
sdf.isnull().sum()
```

Out[7]:
```
Item_Identifier              0
Item_Weight                  0
Item_Fat_Content             0
Item_Visibility              0
Item_Type                    0
Item_MRP                     0
Outlet_Identifier            0
Outlet_Establishment_Year    0
Outlet_Size                  0
Outlet_Location_Type         0
Outlet_Type                  0
Item_Outlet_Sales            0
dtype: int64
```

In [8]:
```
sdf.head()
```

Out[8]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outl
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	

In [6]:
```
from mlxtend.frequent_patterns import apriori , association_rules
```

by using Groupby:

In [7]:
```
sdfg = sdf.groupby(['Outlet_Identifier','Item_Identifier'])['Item_Outle
```

In [8]:
```
sdfg
```

Out[8]:

Item_Identifier Outlet_Identifier	DRA12	DRA24	DRA59	DRB01	DRB13	DRB24	DRB2
OUT010	283.6308	327.5736	185.0924	0.000	948.765	0.0000	214.387
OUT013	2552.6772	4422.2436	555.2772	2466.789	3605.307	0.0000	2036.682
OUT017	2552.6772	1146.5076	2406.2012	0.000	3415.554	1853.5872	2358.263
OUT018	850.8924	0.0000	4442.2176	0.000	0.000	0.0000	1715.10(
OUT019	0.0000	491.3604	555.2772	0.000	0.000	0.0000	0.00(
OUT027	0.0000	4913.6040	7033.5112	569.259	0.000	0.0000	2787.038
OUT035	992.7078	3439.5228	0.0000	0.000	569.259	2162.5184	857.55(
OUT045	3829.0158	0.0000	0.0000	0.000	0.000	4170.5712	0.00(
OUT046	0.0000	0.0000	4442.2176	0.000	0.000	0.0000	0.00(
OUT049	0.0000	982.7208	1295.6468	1518.024	3605.307	4016.1056	0.00(

10 rows × 1559 columns

In [9]:
```
sdfg = sdfg.applymap(lambda x: 1 if x>0 else 0)
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_8392\1445713522.py:1: FutureW
arning: DataFrame.applymap has been deprecated. Use DataFrame.map inst
ead.
  sdfg = sdfg.applymap(lambda x: 1 if x>0 else 0)
```

In [10]:
```
sdfg.shape
```

Out[10]:
```
(10, 1559)
```

In [11]:
```
freq_items = apriori(sdfg , min_support= 0.85 , use_colnames= True)
```

```
C:\Users\hp\anaconda3\Lib\site-packages\mlxtend\frequent_patterns\fpco
mmon.py:109: DeprecationWarning: DataFrames with non-bool types result
in worse computationalperformance and their support might be discontin
ued in the future.Please use a DataFrame with bool type
  warnings.warn(
```

In [12]:
```
freq_items.sort_values('support',ascending=False)
```

Out[12]:

	support	itemsets
63	1.0	(FDG33, FDW13)
6	1.0	(FDG33)
14	1.0	(FDW13)
0	0.9	(DRE49)
405	0.9	(FDW13, NCL31, NCQ06, FDW49)
...	...	...
203	0.9	(FDG33, NCJ30, FDW13)
204	0.9	(NCL31, FDG33, FDW13)
205	0.9	(FDG33, NCQ06, FDW13)
206	0.9	(NCY18, FDG33, FDW13)
602	0.9	(FDT07, FDX04, NCQ06, FDX20, NCL31, FDW49, FDG...

603 rows × 2 columns

In [13]:
```
rules = association_rules(freq_items , metric= 'lift' , min_threshold= (
```

In [14]:
```
rules
```

Out[14]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
0	(DRN47)	(DRE49)	0.9	0.9	0.9	1.0	1.111111
1	(DRE49)	(DRN47)	0.9	0.9	0.9	1.0	1.111111
2	(FDG09)	(DRE49)	0.9	0.9	0.9	1.0	1.111111
3	(DRE49)	(FDG09)	0.9	0.9	0.9	1.0	1.111111
4	(FDG33)	(DRE49)	1.0	0.9	0.9	0.9	1.000000
...	...	...	...	...	...	...	...
12603	(FDX20)	(FDT07, FDX04, NCQ06, NCL31, FDW49, FDG33, FDW13)	0.9	0.9	0.9	1.0	1.111111
12604	(NCL31)	(FDT07, FDX04, NCQ06, FDX20, FDW49, FDG33, FDW13)	0.9	0.9	0.9	1.0	1.111111
12605	(FDW49)	(FDT07, FDX04, NCQ06, FDX20, NCL31, FDG33, FDW13)	0.9	0.9	0.9	1.0	1.111111
12606	(FDG33)	(FDT07, FDX04, NCQ06, FDX20, NCL31, FDW49, FDW13)	1.0	0.9	0.9	0.9	1.000000
12607	(FDW13)	(FDT07, FDX04, NCQ06, FDX20, NCL31, FDW49, FDG33)	1.0	0.9	0.9	0.9	1.000000

12608 rows × 10 columns

In [30]:
```python
rules[rules['confidence'] >= 1.0]
```

Out[30]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
0	(DRN47)	(DRE49)	0.9	0.9	0.9	1.0	1.111111
1	(DRE49)	(DRN47)	0.9	0.9	0.9	1.0	1.111111
2	(FDG09)	(DRE49)	0.9	0.9	0.9	1.0	1.111111
3	(DRE49)	(FDG09)	0.9	0.9	0.9	1.0	1.111111
5	(DRE49)	(FDG33)	0.9	1.0	0.9	1.0	1.000000
...	...	...	...	...	...	...	...
12602	(FDW49)	(NCQ06, FDW13, FDX04, FDX20, FDT07, NCL31, FDG33)	0.9	0.9	0.9	1.0	1.111111
12603	(FDX04)	(NCQ06, FDW13, FDW49, FDX20, FDT07, NCL31, FDG33)	0.9	0.9	0.9	1.0	1.111111
12604	(FDX20)	(NCQ06, FDW13, FDW49, FDX04, FDT07, NCL31, FDG33)	0.9	0.9	0.9	1.0	1.111111
12605	(FDT07)	(NCQ06, FDW13, FDW49, FDX04, FDX20, NCL31, FDG33)	0.9	0.9	0.9	1.0	1.111111
12606	(NCL31)	(NCQ06, FDW13, FDW49, FDX04, FDX20, FDT07, FDG33)	0.9	0.9	0.9	1.0	1.111111

11858 rows × 10 columns

In [32]:
```
rules[rules['confidence'] >= 0.90]
```

Out[32]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
0	(DRN47)	(DRE49)	0.9	0.9	0.9	1.0	1.111111
1	(DRE49)	(DRN47)	0.9	0.9	0.9	1.0	1.111111
2	(FDG09)	(DRE49)	0.9	0.9	0.9	1.0	1.111111
3	(DRE49)	(FDG09)	0.9	0.9	0.9	1.0	1.111111
4	(FDG33)	(DRE49)	1.0	0.9	0.9	0.9	1.000000
...	...	...	...	...	...	...	...
12603	(FDX04)	(NCQ06, FDW13, FDW49, FDX20, FDT07, NCL31, FDG33)	0.9	0.9	0.9	1.0	1.111111
12604	(FDX20)	(NCQ06, FDW13, FDW49, FDX04, FDT07, NCL31, FDG33)	0.9	0.9	0.9	1.0	1.111111
12605	(FDT07)	(NCQ06, FDW13, FDW49, FDX04, FDX20, NCL31, FDG33)	0.9	0.9	0.9	1.0	1.111111
12606	(NCL31)	(NCQ06, FDW13, FDW49, FDX04, FDX20, FDT07, FDG33)	0.9	0.9	0.9	1.0	1.111111
12607	(FDG33)	(NCQ06, FDW13, FDW49, FDX04, FDX20, FDT07, NCL31)	1.0	0.9	0.9	0.9	1.000000

12608 rows × 10 columns

In [15]:
```
sdf['antecedents'] = rules.antecedents
```

In [54]:
```
sdf
```

Out[54]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP
0	FDA15	9.300	Low Fat	0.016047	Dairy	249.8092
1	DRC01	5.920	Regular	0.019278	Soft Drinks	48.2692
2	FDN15	17.500	Low Fat	0.016760	Meat	141.6180
3	FDX07	19.200	Regular	0.000000	Fruits and Vegetables	182.0950
4	NCD19	8.930	Low Fat	0.000000	Household	53.8614
...	...	...	...	...	...	...
8518	FDF22	6.865	Low Fat	0.056783	Snack Foods	214.5218
8519	FDS36	8.380	Regular	0.046982	Baking Goods	108.1570
8520	NCJ29	10.600	Low Fat	0.035186	Health and Hygiene	85.1224
8521	FDN46	7.210	Regular	0.145221	Snack Foods	103.1332
8522	DRG01	14.800	Low Fat	0.044878	Soft Drinks	75.4670

8523 rows × 13 columns

In [16]:
```
sdf['consequents'] = rules.consequents
```

In [57]:
```
sdf.head()
```

Out[57]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outl
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	

In [67]:
```
sdf['Item_Type'].values
```

Out[67]:
```
array(['Dairy', 'Soft Drinks', 'Meat', ..., 'Health and Hygiene',
       'Snack Foods', 'Soft Drinks'], dtype=object)
```

In [66]:

```
Item_name = 'Dairy'
Item_id = Item_name
if Item_id in sdf['Item_Type'].values:
        Item_data = sdf[sdf['Item_Type'] == Item_id].iloc[0:]

print(Item_data)
```

```
     Item_Identifier  Item_Weight Item_Fat_Content  Item_Visibility It
em_Type  \
0              FDA15        9.300          Low Fat         0.016047
Dairy
11             FDA03       18.500          Regular         0.045464
Dairy
19             FDU02       13.350          Low Fat         0.102492
Dairy
28             FDE51        5.925          Regular         0.161467
Dairy
30             FDV38       19.250          Low Fat         0.170349
Dairy
...              ...          ...              ...              ...
...
8424           FDC39        7.405          Low Fat         0.159165
Dairy
8447           FDS26       20.350          Low Fat         0.089975
Dairy
8448           FDV50       14.300          Low Fat         0.123071
Dairy
8457           FDY50        5.800          Low Fat         0.130931
Dairy
8512           FDR26       20.700          Low Fat         0.042801
Dairy

      Item_MRP Outlet_Identifier  Outlet_Establishment_Year Outlet_Siz
e  \
0     249.8092            OUT049                       1999      Mediu
m
11    144.1102            OUT046                       1997       Smal
l
19    230.5352            OUT035                       2004       Smal
l
28     45.5086            OUT010                       1998         Na
N
30     55.7956            OUT010                       1998         Na
N
...        ...               ...                        ...
...
8424  207.1296            OUT035                       2004       Smal
l
8447  261.6594            OUT017                       2007         Na
N
8448  121.1730            OUT018                       2009      Mediu
m
8457   89.9172            OUT035                       2004       Smal
l
8512  178.3028            OUT013                       1987        Hig
h

     Outlet_Location_Type        Outlet_Type  Item_Outlet_Sales  \
0                  Tier 1  Supermarket Type1          3735.1380
11                 Tier 1  Supermarket Type1          2187.1530
19                 Tier 2  Supermarket Type1          2748.4224
28                 Tier 3      Grocery Store           178.4344
30                 Tier 3      Grocery Store           163.7868
...                   ...                ...                ...
```

```
8424                Tier 2  Supermarket Type1          3739.1328
8447                Tier 2  Supermarket Type1          7588.1226
8448                Tier 3  Supermarket Type2          2093.9410
8457                Tier 2  Supermarket Type1          1516.6924
8512                Tier 3  Supermarket Type1          2479.4392

                         antecedents                    consequents
0                            (DRN47)                        (DRE49)
11                           (DRE49)                        (NCJ30)
19                           (FDO19)                        (DRN47)
28                           (FDG33)                        (FDD38)
30                           (FDW13)                        (FDD38)
...                              ...                            ...
8424  (FDG33, FDW13, FDX04, NCL31)                 (FDW49, FDT07)
8447         (FDT07, NCL31, FDX04)          (FDW49, FDW13, FDG33)
8448         (FDG33, FDT07, FDX04)          (FDW49, FDW13, NCL31)
8457                (FDW49, FDT07)   (FDG33, FDW13, FDX04, NCL31)
8512         (FDT07, FDW13, FDG33)          (FDW49, NCQ06, FDX04)

[682 rows x 14 columns]
```

In [17]:

```python
def check_product_availability(Item_name):

    Item_id = Item_name
    if Item_id in sdf['Item_Type'].values:
        Item_data = sdf[sdf['Item_Type'] == Item_id].iloc[0:]
        return {
            'available': True,
            'details': {
                'anticidents': Item_data['antecedents'],
                'consequents': Item_data['consequents'],
                'Weight': Item_data['Item_Weight'],
                'Fat_Content': Item_data['Item_Fat_Content'],
                'Visibility': Item_data['Item_Visibility'],
                'Type': Item_data['Item_Type'],
                'MRP': Item_data['Item_MRP'],
                'Sales': Item_data['Item_Outlet_Sales']
            }
        }
    else:
        return {'available': False}
```

In [19]:

```
Item_name = input("enter product name")
Item_status = check_product_availability(Item_name)
if Item_status['available'] == True:
    print(f"Yes, {Item_name} is available.")
    print(f"Details: {Item_status['details']}")
else:
    print(f"Sorry, {Item_name} is currently not available.")
```

```
enter product name Dairy
```

```
Yes, Dairy is available.
Details: {'anticidents': 0                            (DRE49)
11                             (NCJ30)
19                             (FDO19)
28                             (FDG33)
30                             (FDW13)
                    ...
8424    (FDT07, FDG33, NCL31, FDW49)
8447           (FDT07, FDW49, FDX04)
8448           (FDG33, FDW49, FDX04)
8457                  (FDW13, FDX04)
8512           (FDT07, FDG33, FDX04)
Name: antecedents, Length: 682, dtype: object, 'consequents': 0
(DRN47)
11                             (DRE49)
19                             (DRN47)
28                             (FDD38)
30                             (FDD38)
                    ...
8424                  (FDW13, FDX04)
8447           (FDW13, FDG33, NCL31)
8448           (FDW13, FDT07, NCL31)
8457    (FDT07, FDG33, NCL31, FDW49)
8512           (NCQ06, FDW13, FDW49)
Name: consequents, Length: 682, dtype: object, 'Weight': 0        9.30
0
11      18.500
19      13.350
28       5.925
30      19.250
         ...
8424     7.405
8447    20.350
8448    14.300
8457     5.800
8512    20.700
Name: Item_Weight, Length: 682, dtype: float64, 'Fat_Content': 0
Low Fat
11      Regular
19      Low Fat
28      Regular
30      Low Fat
         ...
8424    Low Fat
8447    Low Fat
8448    Low Fat
8457    Low Fat
8512    Low Fat
Name: Item_Fat_Content, Length: 682, dtype: object, 'Visibility': 0
0.016047
11      0.045464
19      0.102492
28      0.161467
30      0.170349
          ...
8424    0.159165
8447    0.089975
```

```
8448    0.123071
8457    0.130931
8512    0.042801
Name: Item_Visibility, Length: 682, dtype: float64, 'Type': 0       Da
iry
11      Dairy
19      Dairy
28      Dairy
30      Dairy
        ...
8424    Dairy
8447    Dairy
8448    Dairy
8457    Dairy
8512    Dairy
Name: Item_Type, Length: 682, dtype: object, 'MRP': 0       249.8092
11      144.1102
19      230.5352
28       45.5086
30       55.7956
          ...
8424    207.1296
8447    261.6594
8448    121.1730
8457     89.9172
8512    178.3028
Name: Item_MRP, Length: 682, dtype: float64, 'Sales': 0       3735.138
0
11      2187.1530
19      2748.4224
28       178.4344
30       163.7868
          ...
8424    3739.1328
8447    7588.1226
8448    2093.9410
8457    1516.6924
8512    2479.4392
Name: Item_Outlet_Sales, Length: 682, dtype: float64}
```

Performing ML model on big superstore market after adding antecedents and consiquents:

In [18]:
```
sdf.head()
```

Out[18]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outl
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	

In [26]:
```
sdf.shape
```

Out[26]:
```
(8523, 14)
```

In [27]:
```
sdf.isnull().sum()
```

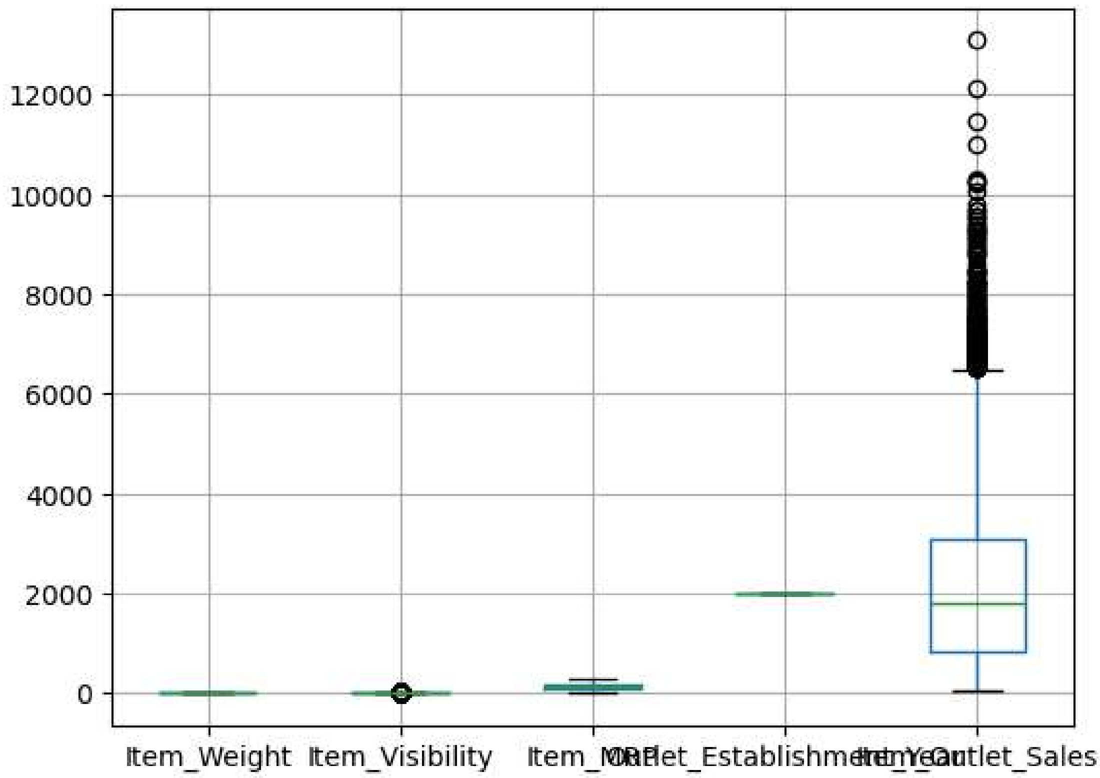
Out[27]:
```
Item_Identifier              0
Item_Weight                  0
Item_Fat_Content             0
Item_Visibility              0
Item_Type                    0
Item_MRP                     0
Outlet_Identifier            0
Outlet_Establishment_Year    0
Outlet_Size                  0
Outlet_Location_Type         0
Outlet_Type                  0
Item_Outlet_Sales            0
antecedents                  0
consequents                  0
dtype: int64
```

In [28]:
```
sdf[sdf.duplicated()]
```

Out[28]:

Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outle

In [29]:
```
sdf.boxplot()
plt.show()
```



In [19]:
```
sdf.drop(['Outlet_Establishment_Year'],axis = 1, inplace= True)
```

In [36]:
```
sdf.head()
```

Out[36]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outl
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	

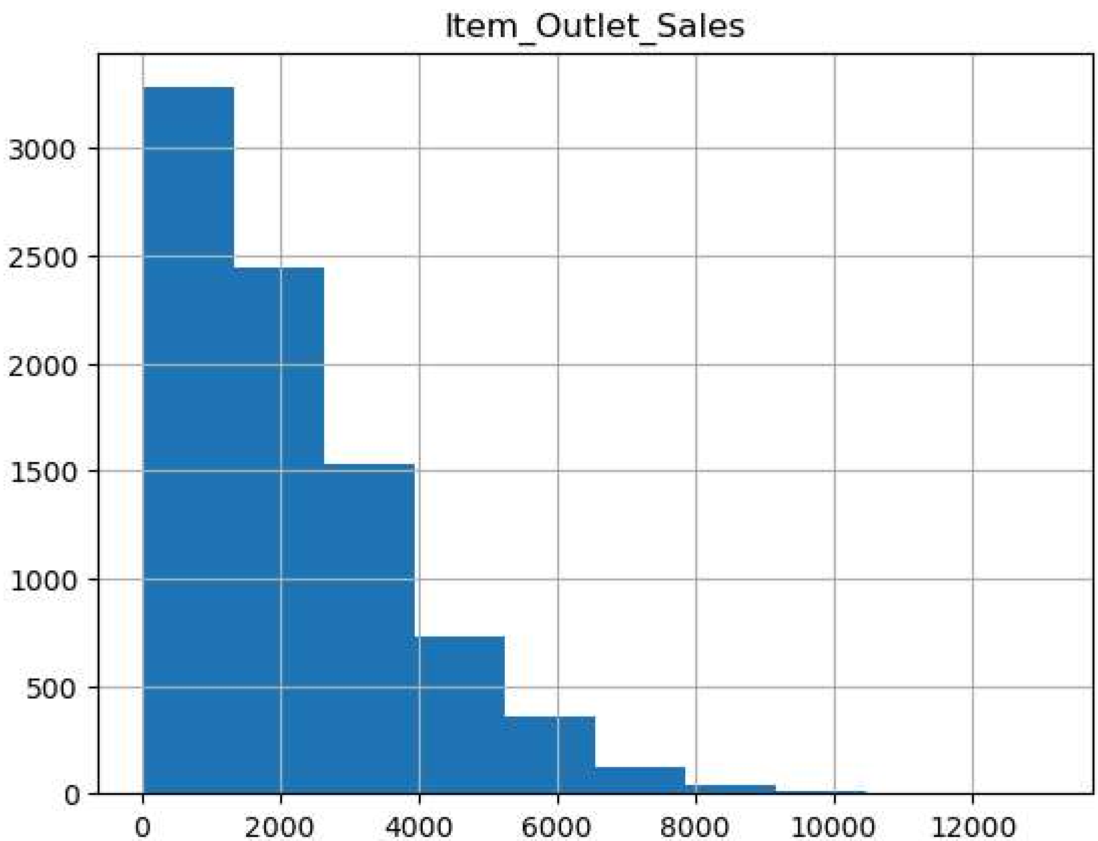
In [20]:
```
from sklearn.preprocessing import LabelEncoder
import warnings
warnings.filterwarnings('ignore')
```

In [21]:
```
for col in sdf.columns:
    if sdf[col].dtypes == 'object':
        sdf[col] = LabelEncoder().fit_transform(sdf[col])
```

In [22]:
```
from sklearn.model_selection import train_test_split , GridSearchCV
from sklearn.linear_model import LinearRegression
```

In [23]:
```
X = sdf.iloc[: , sdf.columns != 'Item_Outlet_Sales']
y = sdf[['Item_Outlet_Sales']]
```

In [24]:
```
y.hist()
plt.show()
```

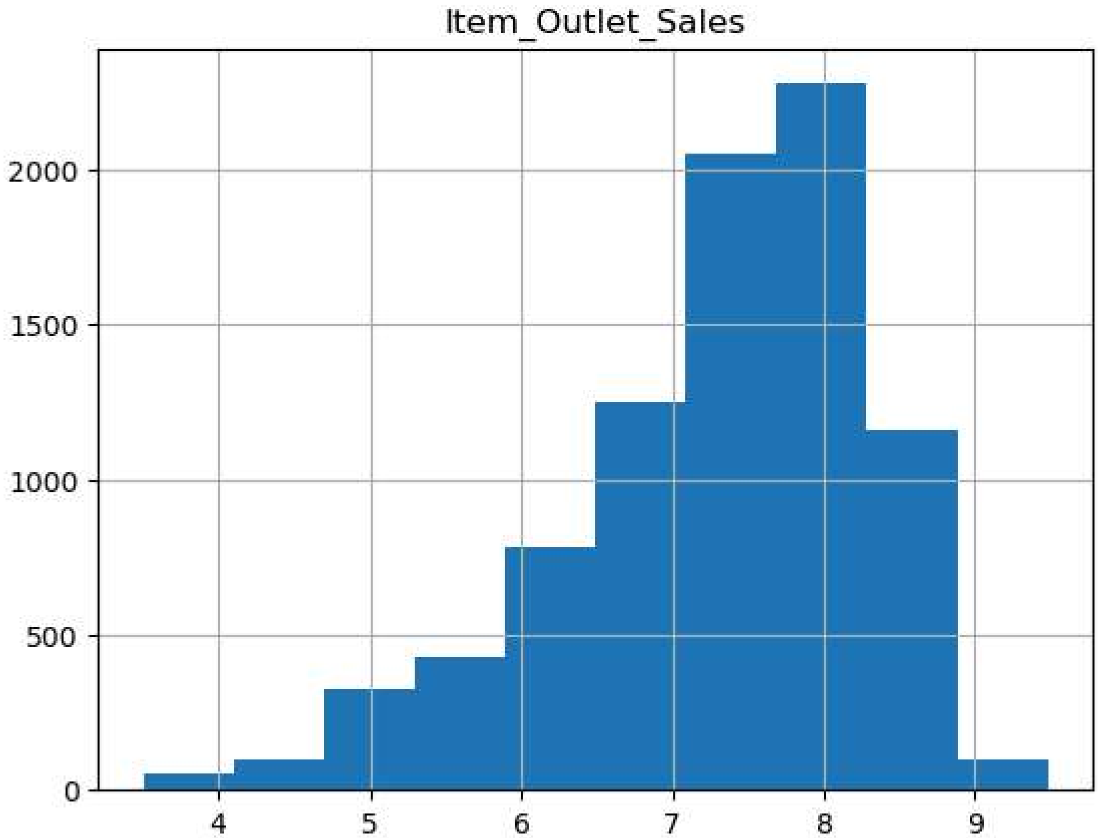


In [25]:
```
y.skew()
```

Out[25]:
```
Item_Outlet_Sales    1.177531
dtype: float64
```

In [26]:
```
yl = np.log(y)
```

In [27]:
```
yl.hist()
plt.show()
```



by applying log to dependent variable:

In [28]:
```
X_train , X_test , y_train , y_test = train_test_split(X , yl , test_si
```

In [29]:
```
model = LinearRegression()
model.fit(X_train , y_train)
```

Out[29]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [30]:
```
ypred = model.predict(X_test)
```

In [31]:
```
ypred1 = np.exp(ypred)
```

In [32]:
```
from sklearn.metrics import mean_squared_error,r2_score
```

In [33]:
```
mse = mean_squared_error(y_test ,ypred1)
rmse = np.sqrt(mse)
print(f"RMSE is {rmse}")
r2 = r2_score(y_test , ypred1)
print(f"R2_Score is {r2}")
```

```
RMSE is 2610.226233504838
R2_Score is -6919760.7700457955
```

By not applying log to dependent variable:

In [34]:
```
X_train , X_test , y_train , y_test = train_test_split(X , y , test_size
```

In [35]:
```
modell = LinearRegression()
modell.fit(X_train , y_train)
```

Out[35]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [36]:
```
ypredd= modell.predict(X_test)
```

In [37]:
```
mse1 = mean_squared_error(y_test ,ypredd)
rmse1 = np.sqrt(mse1)
print(f"RMSE is {rmse1}")
r21 = r2_score(y_test , ypredd)
print(f"R2_Score is {r21}")
```

```
RMSE is 1183.0714023882292
R2_Score is 0.5050564775076779
```

By performing ensemble techniques:

In [50]:
```
from sklearn.ensemble import GradientBoostingRegressor
```

In [43]:
```
gbr = GradientBoostingRegressor()
gbr.fit(X_train , y_train)
```

Out[43]:
```
GradientBoostingRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [46]:
```
yp = gbr.predict(X_test)
```

In [47]:
```
mse1g = mean_squared_error(y_test ,yp)
rmse1g = np.sqrt(mse1g)
print(f"RMSE is {rmse1g}")
r21g = r2_score(y_test , yp)
print(f"R2_Score is {r21g}")
```

```
RMSE is 1053.1229773931188
R2_Score is 0.6078141605817093
```

by performing intial df (not added antecedents and consiquents):

In [38]:
```
dff = df.copy()
```

In [39]:
```
dff['Item_Weight'].fillna(dff['Item_Weight'].mean(), inplace=True)
dff['Outlet_Size'].fillna('Unknown', inplace=True)
```

In [40]:
```
for col in dff.columns:
    if dff[col].dtypes == 'object':
        dff[col] = LabelEncoder().fit_transform(dff[col])
```

In [41]:
```
Xf = dff.iloc[: , dff.columns != 'Item_Outlet_Sales']
yf = dff[['Item_Outlet_Sales']]
```

In [46]:
```
yfl = np.log(yf)
```

In [47]:
```
X_train , X_test , y_train , y_test = train_test_split(Xf , yfl , test_s
```

In [48]:
```
modelf = LinearRegression()
modelf.fit(X_train , y_train)
y_predf= modelf.predict(X_test)
ypredf = np.exp(y_predf)
mse1f = mean_squared_error(y_test ,ypredf)
rmse1f = np.sqrt(mse1f)
print(f"RMSE is {rmse1f}")
r21f = r2_score(y_test , ypredf)
print(f"R2_Score is {r21f}")
```

```
RMSE is 2503.7446323574254
R2_Score is -6366706.607669659
```

In [52]:
```
X_train , X_test , y_train , y_test = train_test_split(Xf , yf , test_s
```

In [54]:
```
modelf = LinearRegression()
modelf.fit(X_train , y_train)
ypredf= modelf.predict(X_test)
mse1f = mean_squared_error(y_test ,ypredf)
rmse1f = np.sqrt(mse1f)
print(f"RMSE is {rmse1f}")
r21f = r2_score(y_test , ypredf)
print(f"R2_Score is {r21f}")
```

```
RMSE is 1182.5556712461164
R2_Score is 0.5054879005830992
```

perform Gradientboostingregressor:

In [55]:
```
modelfg = GradientBoostingRegressor()
modelfg.fit(X_train , y_train)
ypredfg= modelfg.predict(X_test)
mse1fg = mean_squared_error(y_test ,ypredfg)
rmse1fg = np.sqrt(mse1fg)
print(f"RMSE is {rmse1fg}")
r21fg = r2_score(y_test , ypredfg)
print(f"R2_Score is {r21fg}")
```

```
RMSE is 1051.7828350813074
R2_Score is 0.6088116707198594
```

In [ ]:
```
#param = {'learning_rate':[0.1 , 0.5 , 0.001 , 10,1,100 ,1000] ,'n_esti
#gscv = GridSearchCV(modelfg , param , scoring= 'r2' , cv=5)
#gscv.fit(X_train , y_train)
```

perform Adaboostregressor:

In [25]:
```
from sklearn.ensemble import AdaBoostRegressor
```

In [28]:
```
modelfa = AdaBoostRegressor()
modelfa.fit(X_train , y_train)
ypredfa= modelfa.predict(X_test)
mse1fa = mean_squared_error(y_test ,ypredfa)
rmse1fa = np.sqrt(mse1fa)
print(f"RMSE is {rmse1fa}")
r21fa = r2_score(y_test , ypredfa)
print(f"R2_Score is {r21fa}")
```

```
RMSE is 1303.106974431279
R2_Score is 0.39952646657736646
```

In [97]:

```
!pip install xgboost
```

```
WARNING: Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<pip._vendor.urllib3.connection.HTTPSConnection object at 0x0000023FE7CDBB10>: Failed to establish a new connection: [Errno 11001] getaddrinfo failed')': /simple/xgboost/
WARNING: Retrying (Retry(total=3, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<pip._vendor.urllib3.connection.HTTPSConnection object at 0x0000023FE489F250>: Failed to establish a new connection: [Errno 11001] getaddrinfo failed')': /simple/xgboost/
WARNING: Retrying (Retry(total=2, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<pip._vendor.urllib3.connection.HTTPSConnection object at 0x0000023FE7CA0C10>: Failed to establish a new connection: [Errno 11001] getaddrinfo failed')': /simple/xgboost/
WARNING: Retrying (Retry(total=1, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<pip._vendor.urllib3.connection.HTTPSConnection object at 0x0000023FE7CAB410>: Failed to establish a new connection: [Errno 11001] getaddrinfo failed')': /simple/xgboost/
WARNING: Retrying (Retry(total=0, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<pip._vendor.urllib3.connection.HTTPSConnection object at 0x0000023FE7CAF690>: Failed to establish a new connection: [Errno 11001] getaddrinfo failed')': /simple/xgboost/
ERROR: Could not find a version that satisfies the requirement xgboost (from versions: none)
ERROR: No matching distribution found for xgboost
```

In [29]:

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
```

In [100]:

```
modelfd = DecisionTreeRegressor()
modelfd.fit(X_train , y_train)
ypredfd= modelfd.predict(X_test)
mse1fd = mean_squared_error(y_test ,ypredfd)
rmse1fd = np.sqrt(mse1fd)
print(f"RMSE is {rmse1fd}")
r21fd = r2_score(y_test , ypredfd)
print(f"R2_Score is {r21fd}")
```

```
RMSE is 1538.0623023639262
R2_Score is 0.16346987114060219
```

In [102]:
```
modelfr = RandomForestRegressor()
modelfr.fit(X_train , y_train)
ypredfr= modelfr.predict(X_test)
mse1fr= mean_squared_error(y_test ,ypredfr)
rmse1fr = np.sqrt(mse1fr)
print(f"RMSE is {rmse1fr}")
r21fr = r2_score(y_test , ypredfr)
print(f"R2_Score is {r21fr}")
```

```
RMSE is 1095.8768993257659
R2_Score is 0.5753244320813895
```

In [30]:
```
from sklearn.model_selection import GridSearchCV
```

In [106]:
```
param = {'n_estimators':[20,40,70] ,
        'max_depth':[5 , 7, 9] ,
        'min_samples_split':[15,20] ,
        'ccp_alpha': [0.01 , 0.1,10,100]}
gscv = GridSearchCV(modelfr , param , scoring= 'r2' , cv=5)
gscv.fit(X_train , y_train)
```

Out[106]:
```
GridSearchCV(cv=5, estimator=RandomForestRegressor(),
             param_grid={'ccp_alpha': [0.01, 0.1, 10, 100],
                         'max_depth': [5, 7, 9], 'min_samples_split':
[15, 20],
                         'n_estimators': [20, 40, 70]},
             scoring='r2')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [107]:
```
gscv.best_params_
```

Out[107]:
```
{'ccp_alpha': 0.01,
 'max_depth': 5,
 'min_samples_split': 15,
 'n_estimators': 70}
```

In [108]:
```
modelfr = RandomForestRegressor(70 , max_depth=5,min_samples_split=15)
modelfr.fit(X_train , y_train)
ypredfr= modelfr.predict(X_test)
mse1fr= mean_squared_error(y_test ,ypredfr)
rmse1fr = np.sqrt(mse1fr)
print(f"RMSE is {rmse1fr}")
r21fr = r2_score(y_test , ypredfr)
print(f"R2_Score is {r21fr}")
```

```
RMSE is 1048.392089450375
R2_Score is 0.6113298370418313
```

In [56]:
```
Report =pd.DataFrame({'Model':['Linreg mdl after adding A&C with log of
'RMSE':[2610.22],
'R-Squared': [-6919760.770]})
```

In [57]:
```
Report
```

Out[57]:

	Model	RMSE	R-Squared
0	Linreg mdl after adding A&C with log of y	2610.22	-6919760.77

In [58]:
```
Rep1 =pd.DataFrame({'Model':['Linreg mdl after adding A&C without log o
'RMSE':[1183.071],
'R-Squared': [0.50505]})
```

In [59]:
```
Report = pd.concat([Report , Rep1] , ignore_index= True)
```

In [60]:
```
Report
```

Out[60]:

	Model	RMSE	R-Squared
0	Linreg mdl after adding A&C with log of y	2610.220	-6.919761e+06
1	Linreg mdl after adding A&C without log of y	1183.071	5.050500e-01

In [61]:
```
Rep2 =pd.DataFrame({'Model':['GradientBoostingReg ensemble after adding
'RMSE':[1053.12297],
'R-Squared': [0.6078]})
```

In [62]:
```
Report = pd.concat([Report , Rep2] , ignore_index= True)
```

In [63]:
```
Report
```

Out[63]:

	Model	RMSE	R-Squared
0	Linreg mdl after adding A&C with log of y	2610.22000	-6.919761e+06
1	Linreg mdl after adding A&C without log of y	1183.07100	5.050500e-01
2	GradientBoostingReg ensemble after adding A&C ...	1053.12297	6.078000e-01

In [64]:
```
Rep3 =pd.DataFrame({'Model':['Linreg mdl Before adding A&C with log of y
'RMSE':[2503.74],
'R-Squared': [-6366706.607]})
```

In [65]:
```
Report = pd.concat([Report , Rep3] , ignore_index= True)
```

In [66]:
```
Report
```

Out[66]:

	Model	RMSE	R-Squared
0	Linreg mdl after adding A&C with log of y	2610.22000	-6.919761e+06
1	Linreg mdl after adding A&C without log of y	1183.07100	5.050500e-01
2	GradientBoostingReg ensemble after adding A&C ...	1053.12297	6.078000e-01
3	Linreg mdl Before adding A&C with log of y	2503.74000	-6.366707e+06

In [68]:
```
Rep4 =pd.DataFrame({'Model':['Linreg mdl Before adding A&C without log o
'RMSE':[1183.071],
'R-Squared': [0.50505]})
```

In [69]:
```
Report = pd.concat([Report , Rep4] , ignore_index= True)
```

In [70]:
```
Report
```

Out[70]:

	Model	RMSE	R-Squared
0	Linreg mdl after adding A&C with log of y	2610.22000	-6.919761e+06
1	Linreg mdl after adding A&C without log of y	1183.07100	5.050500e-01
2	GradientBoostingReg ensemble after adding A&C ...	1053.12297	6.078000e-01
3	Linreg mdl Before adding A&C with log of y	2503.74000	-6.366707e+06
4	Linreg mdl Before adding A&C without log of y	1183.07100	5.050500e-01

In [71]:
```
Rep5 =pd.DataFrame({'Model':['GradientBoostingReg ensemble Before addin
'RMSE':[1051.78],
'R-Squared': [0.6078]})
```

In [72]:
```
Report = pd.concat([Report , Rep5] , ignore_index= True)
```

In [73]:
```
Report
```

Out[73]:

	Model	RMSE	R-Squared
0	Linreg mdl after adding A&C with log of y	2610.22000	-6.919761e+06
1	Linreg mdl after adding A&C without log of y	1183.07100	5.050500e-01
2	GradientBoostingReg ensemble after adding A&C ...	1053.12297	6.078000e-01
3	Linreg mdl Before adding A&C with log of y	2503.74000	-6.366707e+06
4	Linreg mdl Before adding A&C without log of y	1183.07100	5.050500e-01
5	GradientBoostingReg ensemble Before adding A&C...	1051.78000	6.078000e-01

In [74]:
```
Rep6 =pd.DataFrame({'Model':['RandomforestReg Before adding A&C without
'RMSE':[1095.87689],
'R-Squared': [0.5753]})
```

In [75]:
```
Report = pd.concat([Report , Rep6] , ignore_index= True)
```

In [76]:
```
Report
```

Out[76]:

	Model	RMSE	R-Squared
0	Linreg mdl after adding A&C with log of y	2610.22000	-6.919761e+06
1	Linreg mdl after adding A&C without log of y	1183.07100	5.050500e-01
2	GradientBoostingReg ensemble after adding A&C ...	1053.12297	6.078000e-01
3	Linreg mdl Before adding A&C with log of y	2503.74000	-6.366707e+06
4	Linreg mdl Before adding A&C without log of y	1183.07100	5.050500e-01
5	GradientBoostingReg ensemble Before adding A&C...	1051.78000	6.078000e-01
6	RandomforestReg Before adding A&C without log ...	1095.87689	5.753000e-01

In [77]:
```
Rep7 =pd.DataFrame({'Model':['RandomforestReg with pruned hyper paramet
'RMSE':[1048.39],
'R-Squared': [0.611329]})
```

In [78]:
```
Report = pd.concat([Report , Rep7] , ignore_index= True)
```

In [79]:
```
Report
```

Out[79]:

	Model	RMSE	R-Squared
0	Linreg mdl after adding A&C with log of y	2610.22000	-6.919761e+06
1	Linreg mdl after adding A&C without log of y	1183.07100	5.050500e-01
2	GradientBoostingReg ensemble after adding A&C ...	1053.12297	6.078000e-01
3	Linreg mdl Before adding A&C with log of y	2503.74000	-6.366707e+06
4	Linreg mdl Before adding A&C without log of y	1183.07100	5.050500e-01
5	GradientBoostingReg ensemble Before adding A&C...	1051.78000	6.078000e-01
6	RandomforestReg Before adding A&C without log ...	1095.87689	5.753000e-01
7	RandomforestReg with pruned hyper parameters	1048.39000	6.113290e-01

In [ ]:
```
Rep8 =pd.DataFrame({'Model':['DecisionTreeReg without log of y'],
'RMSE':[1538.062],
'R-Squared': [0.16353]})
```