In [2]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
```

In [4]:
```
data=pd.read_csv('Groceries_dataset.csv')
data.head()
```

Out[4]:

	Member_number	Date	itemDescription
0	1808	21-07-2015	tropical fruit
1	2552	05-01-2015	whole milk
2	2300	19-09-2015	pip fruit
3	1187	12-12-2015	other vegetables
4	3037	01-02-2015	whole milk

In [5]:
```
data.shape
```

Out[5]:
```
(38765, 3)
```

In [6]:
```
x=data['itemDescription'].value_counts().sort_values(ascending=False)[:10]
#Arranging data in ascending order with Top 10 Item sold
```

In [7]:
```
x
```

Out[7]:
```
whole milk          2502
other vegetables    1898
rolls/buns          1716
soda                1514
yogurt              1334
root vegetables     1071
tropical fruit      1032
bottled water        933
sausage              924
citrus fruit         812
Name: itemDescription, dtype: int64
```

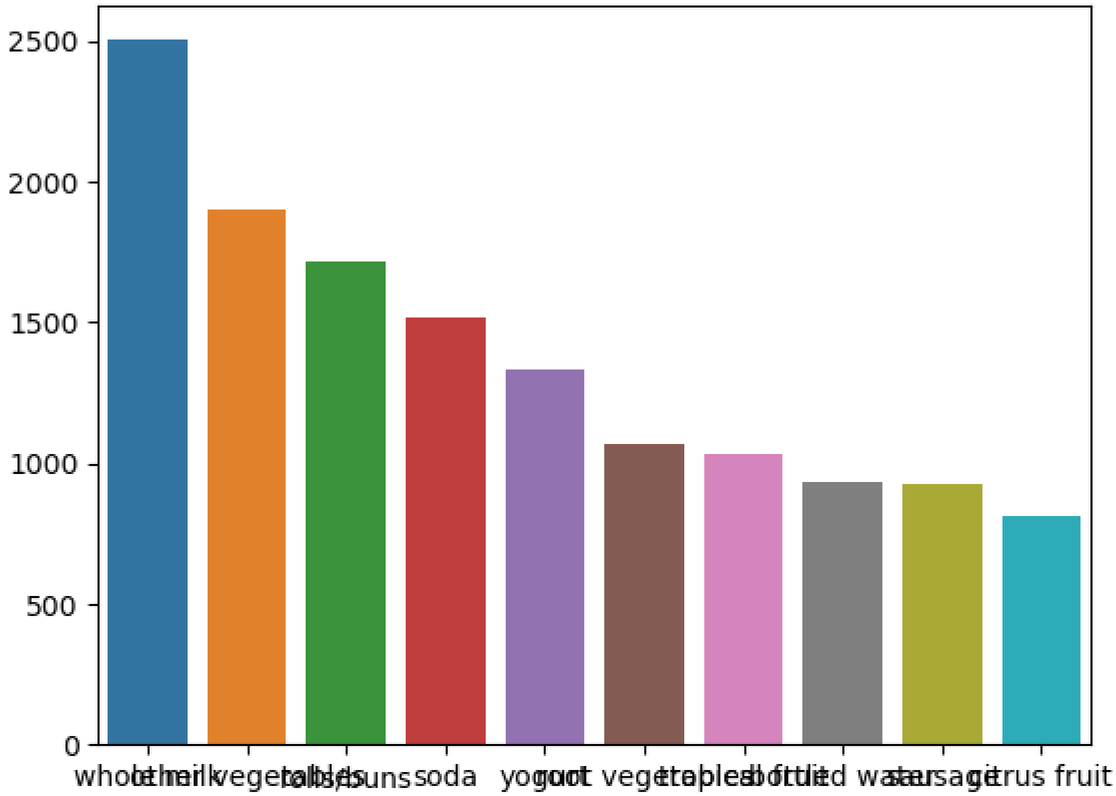
In [10]:
```
plt.figure(figsize=(15,10))
```

Out[10]:
```
<Figure size 1500x1000 with 0 Axes>
```

```
<Figure size 1500x1000 with 0 Axes>
```

In [13]:
```
sns.barplot(x=x.index, y=x.values)
```

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



In [14]:
```
data['Quantity']=1
```

In [15]:
```
trans=data.groupby(['Member_number','itemDescription'])['Quantity'].sum().unstack()
# making Pivot table and grouping valuable data
```

In [16]:
```
trans=trans.fillna(0) # replacing NaN values with 0
```

In [17]:
```
trans
```

Out[17]:

itemDescription Member_number	Instant food products	UHT-milk	abrasive cleaner	artif. sweetener	baby cosmetics	bags	baking powder	bathroo clea
1000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1001	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1002	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1003	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1004	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	
4996	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4997	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4998	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

4999	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5000	0.0	0.0	0.0	0.0	0.0	0.0	0.0

3898 rows × 167 columns

In [18]:
```
def encode(x):
    if x<=0:
        return 0
    elif x>0:
        return 1
basket=trans.applymap(encode) # Apriori Algo only works on 0 and 1, we are modifyin
```

In [21]:
```
freq=apriori(basket,min_support=0.06,use_colnames=True)
rules=association_rules(freq,metric='lift',min_threshold=1)
```

```
C:\Users\User\anaconda3\lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:110:
DeprecationWarning: DataFrames with non-bool types result in worse computationalperf
ormance and their support might be discontinued in the future.Please use a DataFrame
with bool type
  warnings.warn(
```

In [22]:
```
rules.head()
```

Out[22]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	lev
0	(beef)	(whole milk)	0.119548	0.458184	0.064135	0.536481	1.170886	0.0
1	(whole milk)	(beef)	0.458184	0.119548	0.064135	0.139978	1.170886	0.0
2	(bottled beer)	(other vegetables)	0.158799	0.376603	0.068497	0.431341	1.145345	0.0
3	(other vegetables)	(bottled beer)	0.376603	0.158799	0.068497	0.181880	1.145345	0.0
4	(bottled beer)	(rolls/buns)	0.158799	0.349666	0.063109	0.397415	1.136555	0.0

In [23]:
```
rules[(rules['confidence']>0.4)&(rules['lift']>1.0)]
```

Out[23]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift
0	(beef)	(whole milk)	0.119548	0.458184	0.064135	0.536481	1.170886
2	(bottled beer)	(other vegetables)	0.158799	0.376603	0.068497	0.431341	1.145345
6	(bottled beer)	(whole milk)	0.158799	0.458184	0.085428	0.537964	1.174124
8	(bottled water)	(other vegetables)	0.213699	0.376603	0.093894	0.439376	1.166680
14	(bottled water)	(whole milk)	0.213699	0.458184	0.112365	0.525810	1.147597

18	(brown bread)	(whole milk)	0.135967	0.458184	0.069779	0.513208	1.120091
20	(butter)	(whole milk)	0.126475	0.458184	0.066188	0.523327	1.142176
23	(canned beer)	(other vegetables)	0.165213	0.376603	0.067214	0.406832	1.080267
25	(canned beer)	(rolls/buns)	0.165213	0.349666	0.066701	0.403727	1.154605
26	(canned beer)	(whole milk)	0.165213	0.458184	0.087224	0.527950	1.152268
29	(citrus fruit)	(other vegetables)	0.185480	0.376603	0.077476	0.417704	1.109135
35	(citrus fruit)	(whole milk)	0.185480	0.458184	0.092355	0.497925	1.086737
36	(curd)	(whole milk)	0.120831	0.458184	0.063622	0.526539	1.149188
38	(domestic eggs)	(whole milk)	0.133145	0.458184	0.070292	0.527938	1.152242
40	(frankfurter)	(other vegetables)	0.137506	0.376603	0.061057	0.444030	1.179038
42	(frankfurter)	(whole milk)	0.137506	0.458184	0.067984	0.494403	1.079050
44	(fruit/vegetable juice)	(whole milk)	0.124936	0.458184	0.062340	0.498973	1.089025
47	(newspapers)	(whole milk)	0.139815	0.458184	0.072345	0.517431	1.129310
49	(pastry)	(other vegetables)	0.177527	0.376603	0.071575	0.403179	1.070567
51	(pip fruit)	(other vegetables)	0.170600	0.376603	0.072345	0.424060	1.126013
52	(rolls/buns)	(other vegetables)	0.349666	0.376603	0.146742	0.419663	1.114335
54	(root vegetables)	(other vegetables)	0.230631	0.376603	0.094151	0.408231	1.083982
56	(sausage)	(other vegetables)	0.206003	0.376603	0.092868	0.450809	1.197040
58	(shopping bags)	(other vegetables)	0.168291	0.376603	0.073114	0.434451	1.153604
64	(whipped/sour cream)	(other vegetables)	0.154695	0.376603	0.066957	0.432836	1.149315
66	(other vegetables)	(whole milk)	0.376603	0.458184	0.191380	0.508174	1.109106
67	(whole milk)	(other vegetables)	0.458184	0.376603	0.191380	0.417693	1.109106
69	(yogurt)	(other vegetables)	0.282966	0.376603	0.120318	0.425204	1.129050
74	(pastry)	(whole milk)	0.177527	0.458184	0.091072	0.513006	1.119651

80	(pip fruit)	(whole milk)	0.170600	0.458184	0.086968	0.509774	1.112598
82	(pork)	(whole milk)	0.132376	0.458184	0.066957	0.505814	1.103955
88	(shopping bags)	(rolls/buns)	0.168291	0.349666	0.068753	0.408537	1.168361
94	(rolls/buns)	(whole milk)	0.349666	0.458184	0.178553	0.510638	1.114484
100	(root vegetables)	(whole milk)	0.230631	0.458184	0.113135	0.490545	1.070630
106	(sausage)	(whole milk)	0.206003	0.458184	0.106978	0.519303	1.133394
112	(shopping bags)	(whole milk)	0.168291	0.458184	0.091329	0.542683	1.184422
116	(soda)	(whole milk)	0.313494	0.458184	0.151103	0.481997	1.051973
120	(tropical fruit)	(whole milk)	0.233710	0.458184	0.116470	0.498353	1.087672
124	(whipped/sour cream)	(whole milk)	0.154695	0.458184	0.079785	0.515755	1.125650
126	(yogurt)	(whole milk)	0.282966	0.458184	0.150590	0.532185	1.161510
128	(rolls/buns, other vegetables)	(whole milk)	0.146742	0.458184	0.082093	0.559441	1.220996
129	(rolls/buns, whole milk)	(other vegetables)	0.178553	0.376603	0.082093	0.459770	1.220834
130	(other vegetables, whole milk)	(rolls/buns)	0.191380	0.349666	0.082093	0.428954	1.226753
134	(soda, other vegetables)	(whole milk)	0.124166	0.458184	0.069266	0.557851	1.217528
135	(soda, whole milk)	(other vegetables)	0.151103	0.376603	0.069266	0.458404	1.217206
140	(other vegetables, yogurt)	(whole milk)	0.120318	0.458184	0.071832	0.597015	1.303003
142	(yogurt, whole milk)	(other vegetables)	0.150590	0.376603	0.071832	0.477002	1.266589
146	(soda, rolls/buns)	(whole milk)	0.119805	0.458184	0.065162	0.543897	1.187072
147	(soda, whole milk)	(rolls/buns)	0.151103	0.349666	0.065162	0.431239	1.233288
152	(rolls/buns, yogurt)	(whole milk)	0.111339	0.458184	0.065931	0.592166	1.292420
154	(yogurt, whole milk)	(rolls/buns)	0.150590	0.349666	0.065931	0.437819	1.252106

In [25]:
```
filtered_rules = rules[(rules['confidence'] > 0.4) & (rules['lift'] > 1.0)]

# Store the filtered results in a CSV file
filtered_rules.to_csv('filtered_rules.csv', index=False)
```

In [ ]:
```
```