Name: Janhavi Vijay Pawar **Roll No.**TI56

In [1]:

```
import pandas as pd
import numpy as np
!pip install wordcloud
!pip install mlxtend
#Mlxtend is a Python library of useful tools for the day-to-day data science tasks
```

```
Requirement already satisfied: wordcloud in c:\users\91988\anaconda3\lib\s
ite-packages (1.8.1)
Requirement already satisfied: numpy>=1.6.1 in c:\users\91988\anaconda3\li
b\site-packages (from wordcloud) (1.18.5)
Requirement already satisfied: matplotlib in c:\users\91988\anaconda3\lib
\site-packages (from wordcloud) (3.2.2)
Requirement already satisfied: pillow in c:\users\91988\anaconda3\lib\site
-packages (from wordcloud) (7.2.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\91988\anac
onda3\lib\site-packages (from matplotlib->wordcloud) (2.8.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
c:\users\91988\anaconda3\lib\site-packages (from matplotlib->wordcloud)
(2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\91988\anacond
a3\lib\site-packages (from matplotlib->wordcloud) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\91988\anaconda3\li
b\site-packages (from matplotlib->wordcloud) (0.10.0)
Requirement already satisfied: six>=1.5 in c:\users\91988\anaconda3\lib\si
te-packages (from python-dateutil>=2.1->matplotlib->wordcloud) (1.15.0)
Requirement already satisfied: mlxtend in c:\users\91988\anaconda3\lib\sit
e-packages (0.19.0)
Requirement already satisfied: pandas>=0.24.2 in c:\users\91988\anaconda3
\lib\site-packages (from mlxtend) (1.0.5)
Requirement already satisfied: joblib>=0.13.2 in c:\users\91988\anaconda3
\lib\site-packages (from mlxtend) (0.16.0)
Requirement already satisfied: matplotlib>=3.0.0 in c:\users\91988\anacond
a3\lib\site-packages (from mlxtend) (3.2.2)
Requirement already satisfied: scipy>=1.2.1 in c:\users\91988\anaconda3\li
b\site-packages (from mlxtend) (1.5.0)
Requirement already satisfied: scikit-learn>=0.20.3 in c:\users\91988\anac
onda3\lib\site-packages (from mlxtend) (0.23.1)
Requirement already satisfied: setuptools in c:\users\91988\anaconda3\lib
\site-packages (from mlxtend) (49.2.0.post20200714)
Requirement already satisfied: numpy>=1.16.2 in c:\users\91988\anaconda3\l
ib\site-packages (from mlxtend) (1.18.5)
Requirement already satisfied: python-dateutil>=2.6.1 in c:\users\91988\an
aconda3\lib\site-packages (from pandas>=0.24.2->mlxtend) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in c:\users\91988\anaconda3\li
b\site-packages (from pandas>=0.24.2->mlxtend) (2020.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
c:\users\91988\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxten
d) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\91988\anacond
a3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\91988\anaconda3\li
b\site-packages (from matplotlib>=3.0.0->mlxtend) (0.10.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\91988\anac
onda3\lib\site-packages (from scikit-learn>=0.20.3->mlxtend) (2.1.0)
```

```
Requirement already satisfied: six>=1.5 in c:\users\91988\anaconda3\lib\si
te-packages (from python-dateutil>=2.6.1->pandas>=0.24.2->mlxtend) (1.15.
0)
```

In [2]:

```
import matplotlib.pyplot as plt

import seaborn as sns
plt.style.use('fivethirtyeight')
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
```

In [3]: data =

```
pd.read_csv("D:/Users/Janhavi/TE/TE1/ML/Market_Basket_Optimisation.csv") In
```

[4]:

```
data.shape
```

Out[4]:

```
(7501, 20)
```

In [5]:

```
data.head()
```

Out[5]:

	0	1	2	3	4	5	6	7	8	9	1
0	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	lo f yogu
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
3	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
4	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	Na

In [6]:

```
data.tail()
```

Out[6]:

	0	1	2	3	4	5	6	7	8	9	10	1
7496	butter	light mayo	fresh bread	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

7497	burgers	frozen vegetables	eggs	french fries	magazines	green tea	NaN	NaN	NaN	NaN	NaN	Na
7498	chicken	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
7499	escalope	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
7500	eggs	frozen smoothie	yogurt cake	low fat yogurt	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

In [7]:

```
#Pandas sample() is used to generate a sample random row or column
data.sample(10)
```

Out[7]:

	0	1	2	3	4	5	6	7	8
6716	burgers	shrimp	frozen vegetables	ground beef	meatballs	whole wheat rice	barbecue sauce	sparkling water	green tea
6287	pepper	spaghetti	mineral water	milk	salmon	chocolate	frozen smoothie	escalope	hot dogs
1611	red wine	mineral water	soup	avocado	NaN	NaN	NaN	NaN	NaN
4431	mineral water	chicken	chocolate	escalope	NaN	NaN	NaN	NaN	NaN
1944	grated cheese	cereals	NaN	NaN	NaN	NaN	NaN	NaN	NaN
203	turkey	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN
485	cookies	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2371	eggs	chocolate	escalope	NaN	NaN	NaN	NaN	NaN	NaN
2679	mineral water	vegetables mix	yogurt cake	NaN	NaN	NaN	NaN	NaN	NaN
3297	rice	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [8]:

```
#  Data Visualizations
```

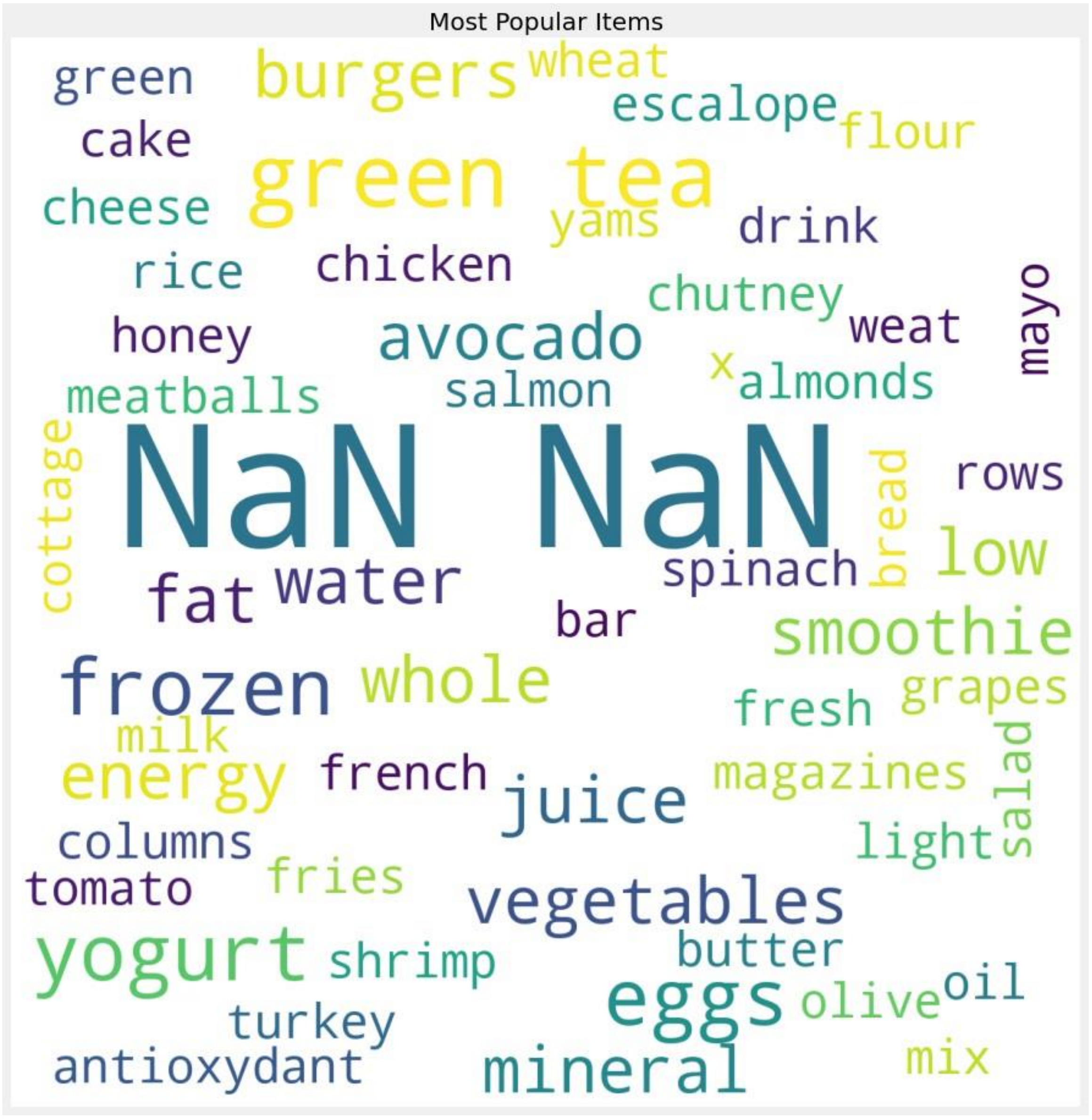
In [9]:

```
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
```

In [10]:

```
plt.rcParams['figure.figsize']=(15,15)
wc=WordCloud(background_color = 'white',width=1200,height=1200,max_words=121).generate(
str(data))

#plot the WordCloud image
plt.imshow(wc)
plt.axis('off')
#defining title and fontsize
plt.title('Most Popular Items',fontsize=20)
plt.show()
```



In [11]:

```
#making each customers shopping an identical list
trans=[]
for i in range(0,7501):
    trans.append([str(data.values[i,j]) for j in range(1,19)])

# converting it into an numpy array
trans = np.array(trans)

# checking the shape of the array
print(trans.shape)
```

```
(7501, 18)
```
In [12]:

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder

te = TransactionEncoder()
data = te.fit_transform(trans)
data = pd.DataFrame(data, columns=te.columns_)

data.shape
```

Out[12]:

```
(7501, 120)
```

In [13]:

```
import warnings
warnings.filterwarnings('ignore')

# getting correlations for 121 items would be messy
# so let&#39;s reduce the items from 121 to 50
```

In [14]:

```
# checking the shape
data.shape
```

Out[14]:

```
(7501, 120)
```
In [15]:

```
data.columns
```

Out[15]:

```
Index(['almonds', 'antioxydant juice', 'asparagus', 'avocado', 'babies foo
d',
       'bacon', 'barbecue sauce', 'black tea', 'blueberries', 'body spra
y',
       ...
       'turkey', 'vegetables mix', 'water spray', 'white wine',
       'whole weat flour', 'whole wheat pasta', 'whole wheat rice', 'yam
s',
       'yogurt cake', 'zucchini'],
 dtype='object', length=120)
```

In [16]:

```
data.head()
```

Out[16]:

	almonds	antioxydant juice	asparagus	avocado	babies food	bacon	barbecue sauce	black tea	blueberries
0	True	True	False	True	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	True	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False

5 rows × 120 columns

In [17]:

```
from mlxtend.frequent_patterns import apriori
#Now, let us return the items and itemsets with at least 5% support:
apriori(data,min_support = 0.01, use_colnames = True)
```

Out[17]:

	support	itemsets
0	0.018931	(almonds)
1	0.025730	(avocado)
2	0.010399	(barbecue sauce)
3	0.013065	(black tea)
4	0.011332	(body spray)
...	...	...
338	0.011065	(nan, spaghetti, whole wheat rice)
339	0.011199	(eggs, nan, mineral water, milk)
340	0.011865	(eggs, nan, spaghetti, mineral water)
341	0.012532	(nan, ground beef, spaghetti, mineral water)
342	0.012665	(nan, spaghetti, mineral water, milk)

343rows × 2 columns

In [18]:

```
#output for extracting frequent itemsets
frequent_itemsets = apriori(data, min_support = 0.05, use_colnames=True)
frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x))
frequent_itemsets
```

Out[18]:

	support	itemsets	length
0	0.067991	(cake)	1
1	0.054126	(chicken)	1
2	0.111718	(chocolate)	1

3	0.142514	(eggs)	1
4	0.060259	(escalope)	1
5	0.138382	(french fries)	1
6	0.059059	(frozen smoothie)	1
7	0.119184	(green tea)	1
8	0.069191	(ground beef)	1
9	0.070391	(low fat yogurt)	1
10	0.105453	(milk)	1
11	0.161445	(mineral water)	1
12	0.999600	(nan)	1
13	0.056792	(olive oil)	1
14	0.084389	(pancakes)	1
15	0.126916	(spaghetti)	1
16	0.052260	(whole wheat rice)	1
17	0.067858	(nan, cake)	2
18	0.054126	(nan, chicken)	2
19	0.111585	(nan, chocolate)	2
20	0.142514	(eggs, nan)	2
21	0.060259	(nan, escalope)	2
22	0.138248	(french fries, nan)	2
23	0.058659	(nan, frozen smoothie)	2
24	0.118917	(green tea, nan)	2
25	0.069191	(nan, ground beef)	2
26	0.070124	(nan, low fat yogurt)	2
27	0.105319	(nan, milk)	2
28	0.161179	(nan, mineral water)	2
29	0.056659	(olive oil, nan)	2
30	0.084389	(pancakes, nan)	2
31	0.126783	(nan, spaghetti)	2
32	0.052260	(nan, whole wheat rice)	2

In [19]:

```
frequent_itemsets[(frequent_itemsets['length'] == 2) & (frequent_itemsets['support']>=
0.01)]
```

Out[19]:

	support	itemsets	length
17	0.067858	(nan, cake)	2

18	0.054126	(nan, chicken)	2
19	0.111585	(nan, chocolate)	2
20	0.142514	(eggs, nan)	2
21	0.060259	(nan, escalope)	2
22	0.138248	(french fries, nan)	2
23	0.058659	(nan, frozen smoothie)	2
24	0.118917	(green tea, nan)	2
25	0.069191	(nan, ground beef)	2
26	0.070124	(nan, low fat yogurt)	2
27	0.105319	(nan, milk)	2
28	0.161179	(nan, mineral water)	2
29	0.056659	(olive oil, nan)	2
30	0.084389	(pancakes, nan)	2
31	0.126783	(nan, spaghetti)	2
32	0.052260	(nan, whole wheat rice)	2

In [20]:

```
frequent_itemsets[(frequent_itemsets['length'] == 1) & (frequent_itemsets['support']>=
0.01)]
```

Out[20]:

	support	itemsets	length
0	0.067991	(cake)	1
1	0.054126	(chicken)	1
2	0.111718	(chocolate)	1
3	0.142514	(eggs)	1
4	0.060259	(escalope)	1
5	0.138382	(french fries)	1
6	0.059059	(frozen smoothie)	1
7	0.119184	(green tea)	1
8	0.069191	(ground beef)	1
9	0.070391	(low fat yogurt)	1
10	0.105453	(milk)	1
11	0.161445	(mineral water)	1
12	0.999600	(nan)	1
13	0.056792	(olive oil)	1
14	0.084389	(pancakes)	1
15	0.126916	(spaghetti)	1

16	0.052260	(whole wheat rice)	1

In [21]:
```
frequent_itemsets[frequent_itemsets['itemsets']=={'eggs','mineral water'}]
```

Out[21]: **support itemsets length**

In [22]:
```
frequent_itemsets[frequent_itemsets['itemsets']=={'mineral water'}]
```

Out[22]:

	support	itemsets	length
11	0.161445	(mineral water)	1

In [23]:
```
frequent_itemsets[frequent_itemsets['itemsets']=={'milk'}]
```

Out[23]:

	support	itemsets	length
10	0.105453	(milk)	1

In [24]:
```
frequent_itemsets[frequent_itemsets['itemsets']=={'chicken'}]
```

Out[24]:

	support	itemsets	length
1	0.054126	(chicken)	1

In [25]:
```
frequent_itemsets[frequent_itemsets['itemsets']=={'frozen vegetables'}]
```

Out[25]: **support itemsets length**