Business Objective:

The business problem, following the revenue metrics, is to know which strategies can potentially lead to an increase in sales. This problem is further breakdown into the following specific objectives:

1. Keystone Products
2. Product placement strategy
3. Strategies to sell high margin products
4. Sale on products that can help to sell other products.

Market basket analysis is done applying Apriori algorithm which is based on Association learning. This helped to determine the answers and strategies for the above stated objectives, and are:

- 1. Keystone products:
  'mineral water', 'eggs', 'spaghetti', 'french fries', 'chocolate', 'green tea', 'milk', 'ground beef', 'frozen vegetables', 'pancakes', 'burgers', 'cake', 'cookies', 'escalope', 'low fat yogurt', 'shrimp', 'tomatoes', 'olive oil', 'frozen smoothie', 'turkey'.
- 1. Product placement strategy:
  'spaghetti' is sold more in a combined cart containing 'ground beef', 'tomatoes' or 'tomato sauce', 'olive oil' or 'cooking oil'.
  Selling 'milk' is more favored by the presence of 'bread' and 'frozen vegetables'.
- 1. Strategies to sell high margin products:
  Neighboring placement of 'burgers' and 'milk' alongside 'cake' can help increase sales of 'cake'.
  Similarly, placement of 'light cream' and 'olive oil' alongside 'Chicken' can increase sales of 'Chicken', and
  'spaghetti' and 'milk' can increase sales of 'frozen vegetables'.
- 1. Sale on products that can help to sell other products:
  Discounts on 'fromage blanc' will boost sales of 'honey'.
  Sales on 'Ground beef' and 'spaghetti' will increase selling of 'tomato sauce' and 'olive oil'.

Table of Contents

1. Exploring Dataset

In [1]:
```python
import numpy as np, pandas as pd
import matplotlib.pyplot as plt, seaborn as sns
from wordcloud import WordCloud
import squarify

from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
```

In [2]:
```python
data = pd.read_csv('dataset.csv', header = None)
print(data.shape)
data.head()
```

(7501, 20)

Out[2]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	green tea	honey	salad	mineral water	salmon	antic
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

- The dataset is a sample that contains list of products purchased by 7500 customers.
- Some customers have only purchased one product whereas maximum 20 products have been purchased by one customer.

2. Exploring Itemset

In [3]:
```python
itemset = []
for i in range(data.shape[0]):
    itemset.append([str(data.values[i,j]) for j in range(data.shape[1])])
```

`itemset` contains the list of products separately, but, `'nan'` have been inserted wherever the purchase had less than 20 products.

In [4]:
```python
for product in itemset[4]:
    print(product, end=" | ")
```

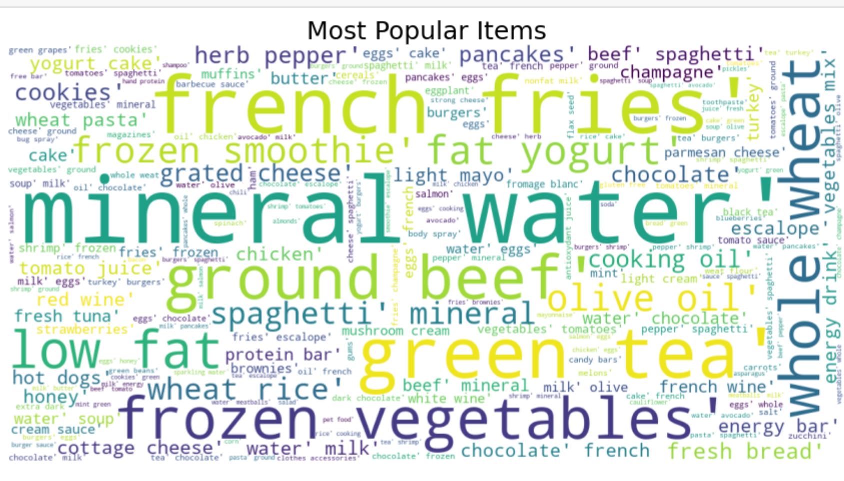
```
mineral water | milk | energy bar | whole wheat rice | green tea | nan | nan | nan | nan | nan | nan
| nan | nan | nan | nan | nan | nan | nan | nan | nan |
```

Let's create a `cleaned_itemset` from `itemset` having only valid entries to **explore itemset through 'Word cloud', 'Frequency plot', and 'Tree Map'**.

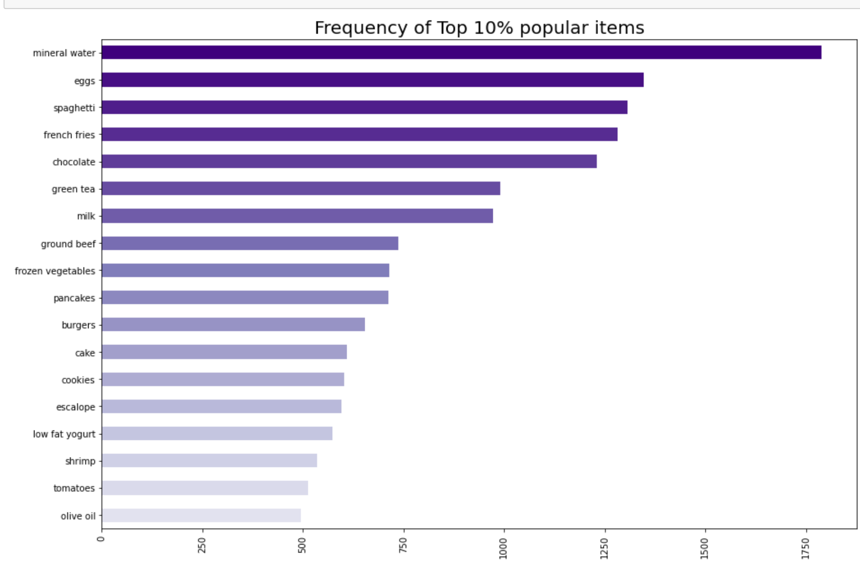
In [5]:
```python
cleaned_itemset = []
for each in itemset:
    for item in each:
        if str(item).lower() != 'nan':
            cleaned_itemset.append(item)

print("There are {} unique items.".format(len(set(cleaned_itemset))))
```

There are 120 unique items.

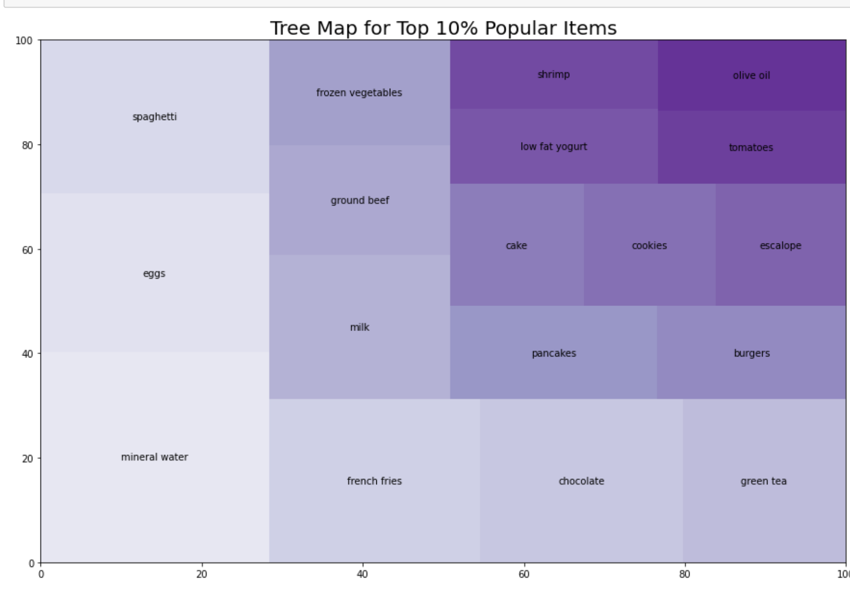
In [6]:
```python
plt.rcParams['figure.figsize'] = (15, 10)
wordcloud = WordCloud(background_color = 'white', width = 1200,  height = 600)\
                      .generate(str(cleaned_itemset))
plt.imshow(wordcloud)
plt.axis('off')
plt.title('Most Popular Items',fontsize = 25)
plt.show()
```



In [7]:
```python
# looking at the frequency of Top 20% popular items
ten = int(0.15*len(set(cleaned_itemset)))
plt.rcParams['figure.figsize'] = (15, 10)
color = plt.cm.Purples(np.linspace(0.2, 1, ten))
pd.DataFrame(cleaned_itemset)[0].value_counts().head(ten).sort_values().plot.barh(color = color)
plt.title('Frequency of Top 10% popular items', fontsize = 20)
plt.xticks(rotation = 90 )
plt.show()
```



In [8]:
```python
tree = pd.DataFrame(cleaned_itemset)[0].value_counts().head(ten).to_frame()
plt.rcParams['figure.figsize'] = (15, 10)
color = plt.cm.Purples(np.linspace(0.2, 1, ten))
squarify.plot(sizes = tree.values, label = tree.index, alpha=.8, color = color)
plt.title('Tree Map for Top 10% Popular Items', fontsize = 20)
plt.axis()
plt.show()
```



3. Apriori algorithm theory

Association Rule finds an association between different products in a set and finds frequent patterns in a transaction database such as purchasing behavior analysis. The applications of Association Rule are in Marketing, Basket Data Analysis (or Market Basket Analysis) in retailing, clustering, and classification. **The association rule learning has three popular algorithms – Apriori, Eclat, and FP-Growth.**

This repository is dedicated to the ***Apriori algorithm for Market Basket Analysis. The approach is based on the theory that customers who buy a specific product (or group of products) are more likely to buy another particular product (or group of products).***

Market Basket Analysis aims to find relationships and establish patterns across purchases. The relationship is modeled in the form of a conditional algorithm: `IF { itemset 'A' } THEN { itemset 'C' }`

`itemset'` : A collection of products/items purchased by a customer
`antecedent` : The set of products on the *Left*-hand side of the rule
`consequent` : The set of products on the *Right*-hand side of the rule.

4. Preparation

Removing NaNs

In [9]:
```python
new = {'items': []}

for i in range(data.shape[0]):
    product_list = []
    for j in range(data.shape[1]):
        product = data.iloc[i,j]
        if str(product).lower() != 'nan':
            product_list.append(product)
    new['items'].append(product_list)

new_df = pd.DataFrame(new)
new_df.head(3)
```

Out[9]:

	items
0	[shrimp, almonds, avocado, vegetables mix, gre...
1	[burgers, meatballs, eggs]
2	[chutney]

One-hot encoding

In [10]:
```python
def do_one_hot(dataframe):
    df_new = pd.DataFrame()
    #For every row in the dataframe, iterate through the list of items and place a 1 into the correspon
ding column
    for index, row in dataframe.iterrows():
        for each in row['items']:
            df_new.at[index, each] = 1
```

[illegible]

5. Rules resulted from the Apriori algorithm

[illegible]

Explanation for the above resulted rule:

[illegible]

6. Finding answers to the business objectives

1. Keystone Products:

[illegible]

2. Product placement strategy:

[illegible]

Out[14]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
3002	(cereals, ground beef)	(spaghetti)	0.004533	0.174110	0.003066	0.676471	3.885303	0.002277	2.552751
2086	(tomatoes, olive oil)	(spaghetti)	0.007199	0.174110	0.004399	0.611111	3.509912	0.003146	2.123717
3265	(soup, frozen vegetables, mineral water)	(milk)	0.005066	0.129583	0.003066	0.605263	4.670863	0.002410	2.205057
2992	(ground beef, tomato sauce)	(spaghetti)	0.005333	0.174110	0.003066	0.575000	3.302508	0.002138	1.943270
2874	(ground beef, cooking oil)	(spaghetti)	0.008399	0.174110	0.004799	0.571429	3.281995	0.003337	1.927076
3461	(mineral water, tomatoes, ground beef)	(spaghetti)	0.005466	0.174110	0.003066	0.560976	3.221959	0.002115	1.881194
3531	(chocolate, frozen vegetables, ground beef)	(spaghetti)	0.005733	0.174110	0.003066	0.534884	3.072100	0.002068	1.775663
3517	(frozen vegetables, ground beef, milk)	(spaghetti)	0.005733	0.174110	0.003066	0.534884	3.072100	0.002068	1.775663
2404	(red wine, eggs)	(spaghetti)	0.007066	0.174110	0.003733	0.528302	3.034297	0.002503	1.750887
822	(shrimp, ground beef)	(spaghetti)	0.011465	0.174110	0.005999	0.523256	3.005315	0.004003	1.732354

3. Strategy to sell high margin products:

Placing high margin items near associated high confidence (driver) items (if one of the Keystone products than better) can increase the overall margin on purchases. Let's say that 'spaghetti' and 'ground beef' are the high margin products.

In [15]:
```python
bottom = rules.sort_values(by="antecedent support")[:75]['antecedents'].to_list()
least_sold = []
for each in bottom:
    for product in each:
        if product not in least_sold:
            least_sold.append(product) if product not in keystone else None
print(len(least_sold))
for product in least_sold:
    print(product, end=" | ")
```

```
15
soup | cereals | tomato sauce | light cream | chicken | herb & pepper | whole wheat rice | salmon | g
rated cheese | pepper | fresh bread | cottage cheese | red wine | cooking oil | avocado |
```

In [16]:
```python
rules[rules['consequents'] == {'cake'}].sort_values(by='confidence', ascending=False).head(2)
```

Out[16]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
2204	(burgers, milk)	(cake)	0.017864	0.081056	0.003733	0.208955	2.577916	0.002285	1.161684
1326	(burgers, mineral water)	(cake)	0.024397	0.081056	0.004799	0.196721	2.426984	0.002822	1.143992

In [17]:
```python
rules[rules['consequents'] == {'chicken'}].sort_values(by='confidence', ascending=False).head(2)
```

Out[17]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
468	(light cream)	(chicken)	0.015598	0.059992	0.004533	0.290598	4.843951	0.003597	1.325072
2030	(olive oil, milk)	(chicken)	0.017064	0.059992	0.003600	0.210938	3.516094	0.002576	1.191297

In [18]:
```python
rules[rules['consequents'] == {'frozen smoothie'}].sort_values(by='confidence', ascending=False).head(2
)
```

Out[18]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
3076	(spaghetti, milk, mineral water)	(frozen smoothie)	0.015731	0.063325	0.003200	0.203390	3.211847	0.002203	1.175826
1953	(spaghetti, milk)	(frozen smoothie)	0.035462	0.063325	0.005599	0.157895	2.493407	0.003354	1.112302

4. Sale on products that can help to sell other products:

Providing discounts or sale on the products which provide greater lift to the consequents can help improve the overall sales metrics.

In [22]:
```python
rules[rules.lift>=5].sort_values(by=["lift","antecedent support"], ascending=False)
```

Out[22]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
3273	(milk, mineral water)	(soup, frozen vegetables)	0.047994	0.007999	0.003066	0.063889	7.987176	0.002682	1.059704
3268	(soup, frozen vegetables)	(milk, mineral water)	0.007999	0.047994	0.003066	0.383333	7.987176	0.002682	1.543794
3092	(olive oil, frozen vegetables)	(milk, mineral water)	0.011332	0.047994	0.003333	0.294118	6.128268	0.002789	1.348676
3097	(milk, mineral water)	(olive oil, frozen vegetables)	0.047994	0.011332	0.003333	0.069444	6.128268	0.002789	1.062449
1224	(whole wheat pasta, mineral water)	(olive oil)	0.009599	0.065858	0.003866	0.402778	6.115863	0.003234	1.564145
1225	(olive oil)	(whole wheat pasta, mineral water)	0.065858	0.009599	0.003866	0.058704	6.115863	0.003234	1.052168
3050	(chocolate, frozen vegetables)	(shrimp, mineral water)	0.022930	0.023597	0.003200	0.139535	5.913283	0.002658	1.134739
3055	(shrimp, mineral water)	(chocolate, frozen vegetables)	0.023597	0.022930	0.003200	0.135593	5.913283	0.002658	1.130336
3270	(frozen vegetables, mineral water)	(soup, milk)	0.035729	0.015198	0.003066	0.085821	5.646864	0.002523	1.077253
3271	(soup, milk)	(frozen vegetables, mineral water)	0.015198	0.035729	0.003066	0.201754	5.646864	0.002523	1.207988
3272	(soup, mineral water)	(frozen vegetables, milk)	0.023064	0.023597	0.003066	0.132948	5.634140	0.002522	1.126118
3269	(frozen vegetables, milk)	(soup, mineral water)	0.023597	0.023064	0.003066	0.129944	5.634140	0.002522	1.122842
2990	(spaghetti, ground beef)	(tomato sauce)	0.039195	0.014131	0.003066	0.078231	5.535971	0.002512	1.069540
2995	(tomato sauce)	(spaghetti, ground beef)	0.014131	0.039195	0.003066	0.216981	5.535971	0.002512	1.227052
3275	(soup)	(frozen vegetables, mineral water, milk)	0.050527	0.011065	0.003066	0.060686	5.484407	0.002507	1.052827
3266	(frozen vegetables, mineral water, milk)	(soup)	0.011065	0.050527	0.003066	0.277108	5.484407	0.002507	1.313438
3096	(frozen vegetables, mineral water)	(olive oil, milk)	0.035729	0.017064	0.003333	0.093284	5.466564	0.002723	1.084061
3093	(olive oil, milk)	(frozen vegetables, mineral water)	0.017064	0.035729	0.003333	0.195313	5.466564	0.002723	1.198318
171	(honey)	(fromage blanc)	0.047460	0.013598	0.003333	0.070225	5.164271	0.002688	1.060903
170	(fromage blanc)	(honey)	0.013598	0.047460	0.003333	0.245098	5.164271	0.002688	1.261806
3094	(olive oil, mineral water)	(frozen vegetables, milk)	0.027596	0.023597	0.003333	0.120773	5.118180	0.002682	1.110524
3095	(frozen vegetables, milk)	(olive oil, mineral water)	0.023597	0.027596	0.003333	0.141243	5.118180	0.002682	1.132338
3519	(spaghetti, milk)	(frozen vegetables, ground beef)	0.035462	0.016931	0.003066	0.086466	5.106950	0.002466	1.076117
3522	(frozen vegetables, ground beef)	(spaghetti, milk)	0.016931	0.035462	0.003066	0.181102	5.106950	0.002466	1.177849
3036	(spaghetti, frozen vegetables)	(shrimp, mineral water)	0.027863	0.023597	0.003333	0.119617	5.069202	0.002675	1.109067
3041	(shrimp, mineral water)	(spaghetti, frozen vegetables)	0.023597	0.027863	0.003333	0.141243	5.069202	0.002675	1.132028
3518	(spaghetti, frozen vegetables)	(milk, ground beef)	0.027863	0.021997	0.003066	0.110048	5.002842	0.002453	1.098939
3523	(milk, ground beef)	(spaghetti, frozen vegetables)	0.021997	0.027863	0.003066	0.139394	5.002842	0.002453	1.129596

7. Conclusion

Market Basket Analysis or the rules resulted applying **The Apriori algorithm** which is another style of the association learning can help the business find interesting insights to find the Keystone products, their placement with poorly performing products to increase sales and strategies to improve revenue by enabling sell of high margin products.