Project:

Problem Statement:

A Grocery Store shared the transactional data with you. Your job is to identify the most popular combos that can be suggested to the Grocery Store chain after a thorough analysis of the most commonly occurring sets of items in the customer orders. The Store doesn't have any combo offers. Can you suggest the best combos & offers?

Q1: Exploratory Analysis --> Exploratory Analysis of data & an executive summary (in PPT) of your top findings, supported by graphs. --> Are there trends across months/years/quarters/days etc. that you are able to notice?

In [ ]:

```

```

We will be performing Exploratory Data Analysis to understand the Given Data and based on that We will be doing Market Basket Analysis to identify the Popular combos and offers for the Grocery Store.

In [2]:

```
# importing required libraries.

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use ('ggplot')
import warnings
warnings.filterwarnings ('ignore')
```

executed in 16ms, finished 16:47:40 2021-08-28

In [4]:

```
# pulling the data:

df = pd.read_csv (r'E:\Great Learning\Capstone\Market and Retail Analytics (MRA)\Marke
df.head()
```

executed in 518ms, finished 16:48:42 2021-08-28

Out[4]:

	Date	Order_id	Product
0	2018-01-01	1	yogurt
1	2018-01-01	1	pork
2	2018-01-01	1	sandwich bags
3	2018-01-01	1	lunch meat
4	2018-01-01	1	all- purpose

In [5]:

```
df.info()
```

executed in 168ms, finished 16:49:19 2021-08-28

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20641 entries, 0 to 20640
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Date      20641 non-null  object
 1   Order_id  20641 non-null  int64 
 2   Product   20641 non-null  object
dtypes: int64(1), object(2)
memory usage: 483.9+ KB
```

In [6]:

```
df.shape
```

executed in 8ms, finished 16:49:26 2021-08-28

Out[6]:

```
(20641, 3)
```

In [7]:

```
df.isnull().sum()
```

executed in 19ms, finished 16:49:53 2021-08-28

Out[7]:

```
Date        0
Order_id    0
Product     0
dtype: int64
```

In [9]:

```
dups = df.duplicated()
print(dups.sum())
```

executed in 21ms, finished 16:50:22 2021-08-28

```
4730
```

In [12]:

```
df.value_counts('Product')
```

executed in 24ms, finished 16:51:23 2021-08-28

Out[12]:

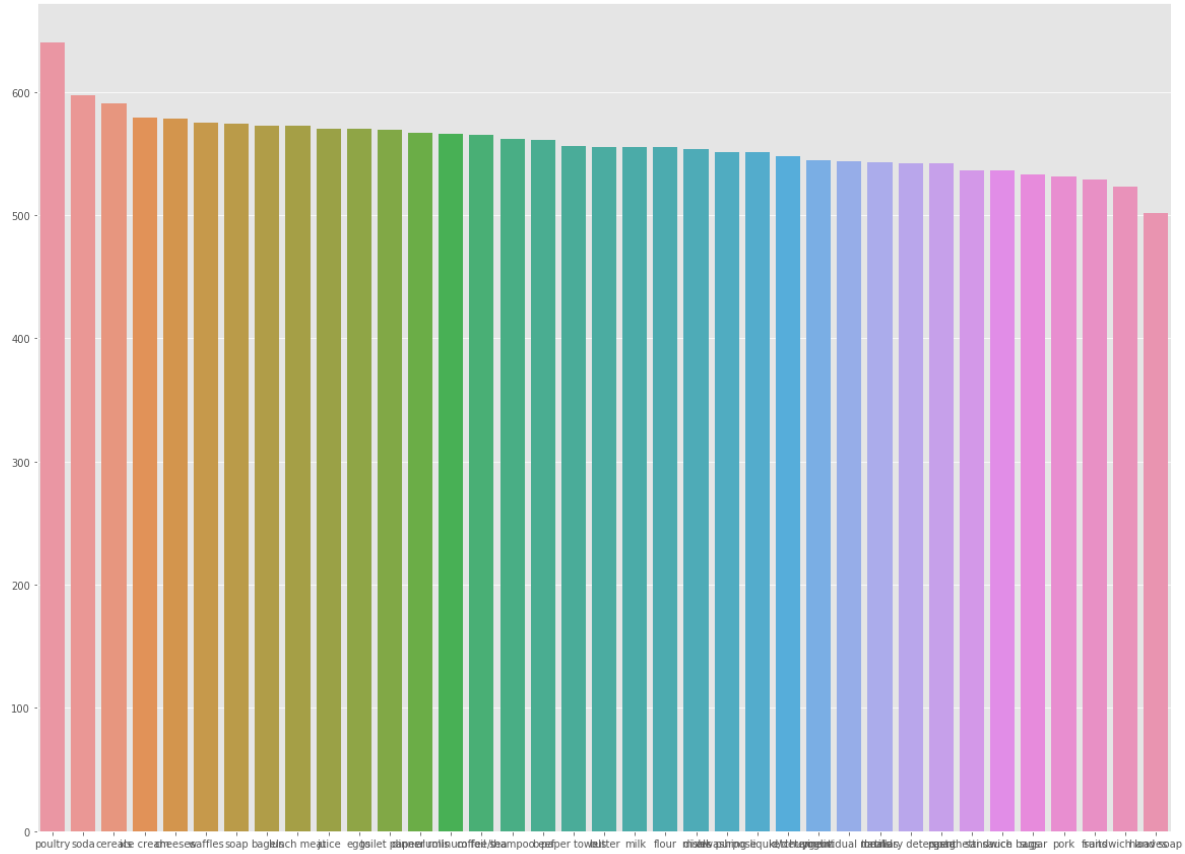
```
Product
poultry                          640
soda                             597
cereals                          591
ice cream                        579
cheeses                          578
waffles                          575
soap                             574
bagels                           573
lunch meat                       573
eggs                             570
juice                            570
toilet paper                     569
dinner rolls                     567
aluminum foil                    566
coffee/tea                       565
shampoo                          562
beef                             561
paper towels                     556
flour                            555
butter                           555
milk                             555
mixes                            554
dishwashing liquid/detergent     551
all- purpose                     551
ketchup                          548
yogurt                           545
individual meals                 544
tortillas                        543
pasta                            542
laundry detergent                542
sandwich bags                    536
spaghetti sauce                  536
sugar                            533
pork                             531
fruits                           529
sandwich loaves                  523
hand soap                        502
dtype: int64
```

In [14]:

```
plt.figure(figsize=(20,15))
sns.barplot (df.Product.value_counts().index, df.Product.value_counts().values);

plt.show()
print (df.Product.value_counts(normalize=True))
```

executed in 718ms, finished 16:55:28 2021-08-28



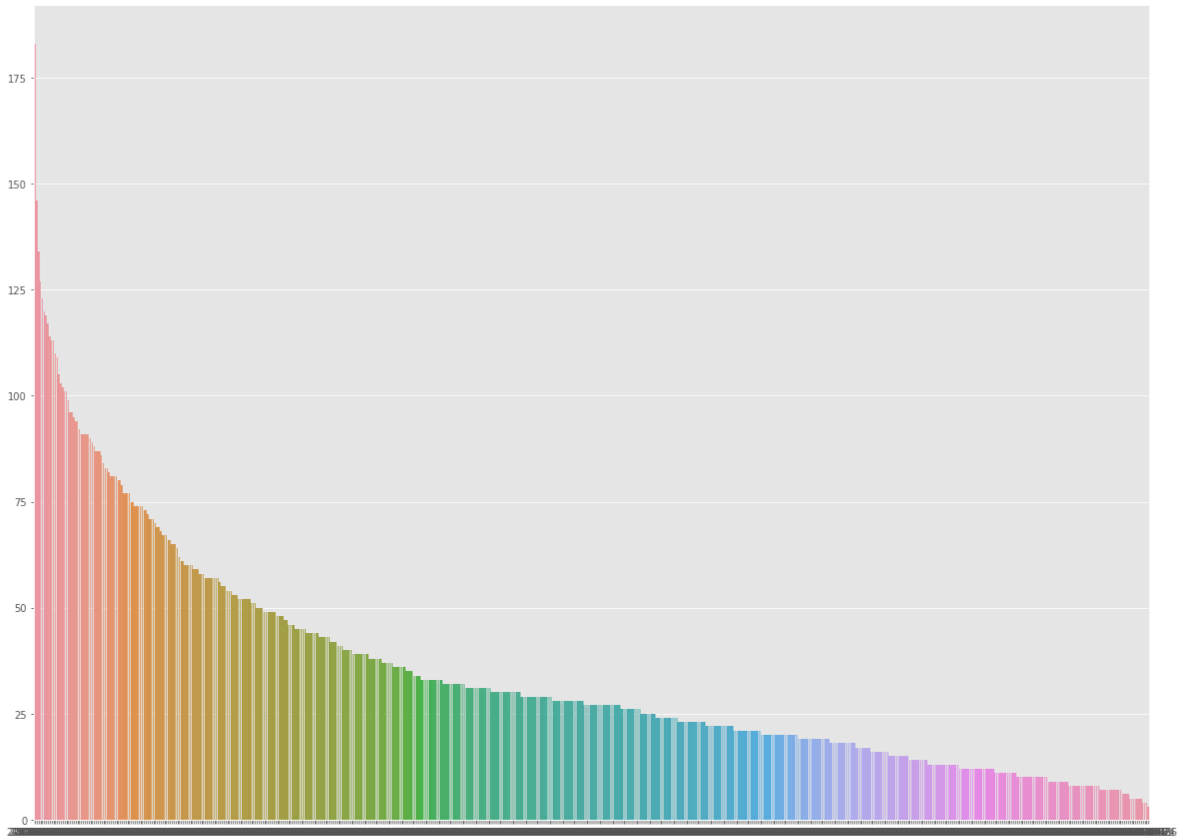

```
poultry                         0.031006
soda                            0.028923
cereals                         0.028632
ice cream                       0.028051
cheeses                         0.028003
waffles                         0.027857
soap                            0.027809
bagels                          0.027760
lunch meat                      0.027760
juice                           0.027615
eggs                            0.027615
toilet paper                    0.027566
```

```
dinner rolls                   0.027470
aluminum foil                  0.027421
coffee/tea                     0.027373
shampoo                        0.027227
beef                           0.027179
paper towels                   0.026937
butter                         0.026888
milk                           0.026888
flour                          0.026888
mixes                          0.026840
all- purpose                   0.026694
dishwashing liquid/detergent   0.026694
ketchup                        0.026549
yogurt                         0.026404
individual meals               0.026355
tortillas                      0.026307
laundry detergent              0.026258
pasta                          0.026258
spaghetti sauce                0.025968
sandwich bags                  0.025968
sugar                          0.025822
pork                           0.025725
fruits                         0.025629
sandwich loaves                0.025338
hand soap                      0.024321
Name: Product, dtype: float64
```

In [16]:

```
plt.figure(figsize=(20,15))
sns.barplot (df.Date.value_counts().index, df.Date.value_counts().values);
```

executed in 19.2s, finished 17:03:48 2021-08-28



In [18]:

```
print (df.Date.value_counts(normalize=True))
```

executed in 23ms, finished 17:06:43 2021-08-28

```
2019-02-08    0.008866
2019-02-20    0.007073
2018-03-06    0.006492
2018-03-01    0.006153
2018-05-17    0.005959
                ...
2019-04-02    0.000242
2019-09-05    0.000194
2019-03-11    0.000194
2018-09-24    0.000194
2020-02-26    0.000145
Name: Date, Length: 603, dtype: float64
```

In [24]:

```
df = df.drop (df [df.Product == 'none'].index)
```

executed in 43ms, finished 17:50:22 2021-08-28

In [25]:

```
df.info()
```

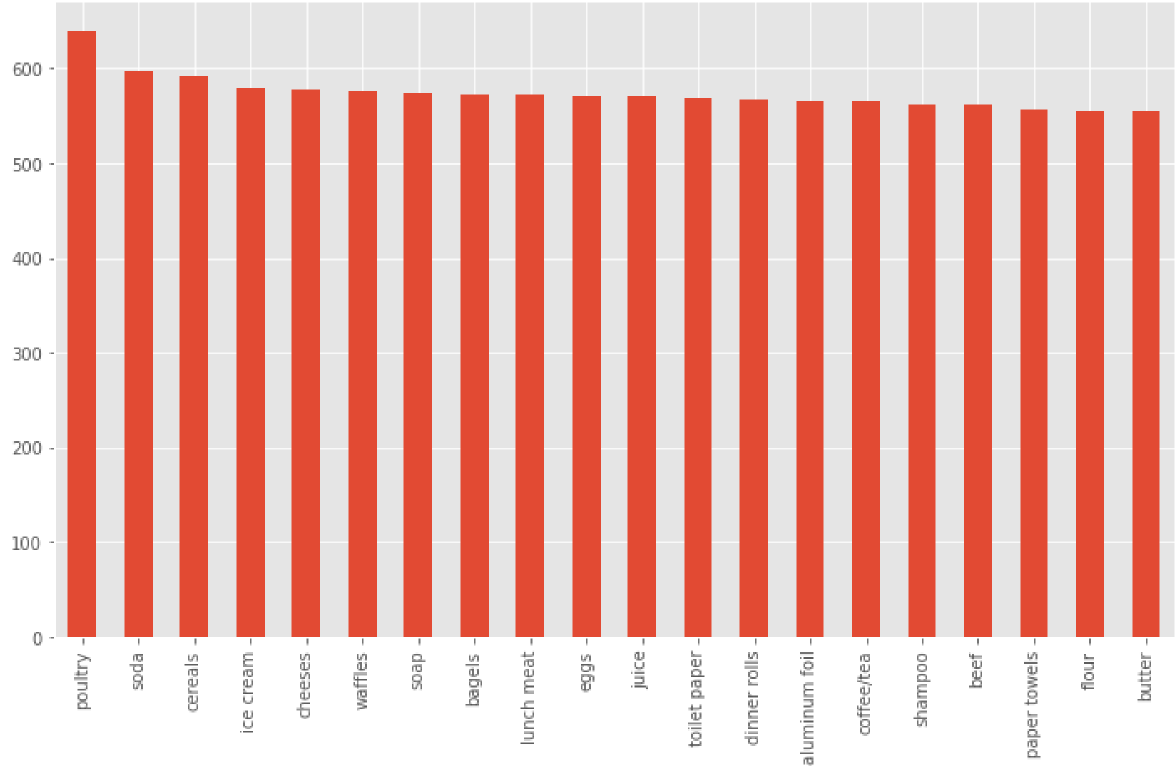
executed in 35ms, finished 17:50:26 2021-08-28

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20641 entries, 0 to 20640
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Date      20641 non-null  object
 1   Order_id  20641 non-null  int64 
 2   Product   20641 non-null  object
dtypes: int64(1), object(2)
memory usage: 645.0+ KB
```

In [26]:

```
1  df['Product'].value_counts().sort_values(ascending=False).head(20).plot(kind='bar',fig
2
```

executed in 417ms, finished 17:51:13 2021-08-28



Summary on EDA.

Given Data is having 20641 Rows and 3 Columns which are (Date, Order_id and Product).

Date : It is the time when product purchased by customer.

Order_id : It is the unique Id which defines the number of purchases by every single customer.

Product : Name of Product.

There are no Missing Value.

There are 4730 Duplicates which we do understand that this numbers are basically the Order_id and Products also. So we will consider this duplicates for our analysis.

Products: All products are equally counted but for Poultry there are maximum numbers which is of 0.03 % of entire Product range. So we can see that Poultry products are high in range. Other products too have 0.02 to 0.028 %.

Date Counts: When we count the Date maximum purchase is from the starting of the data which is of 2019-02-08 and then there is slightly negative trend in dates.

In [19]:

```
# Lets check the Trends.

df2 = pd.read_excel (r'E:\Great Learning\Capstone\Market and Retail Analytics (MRA)\Mar
df2.head()
```

executed in 8.75s, finished 17:11:03 2021-08-28

Out[19]:

	Order_id
Date	
2018-01-01	1
2018-01-01	1
2018-01-01	1
2018-01-01	1
2018-01-01	1

In [20]:

```
df2.info()
```

executed in 24ms, finished 17:11:11 2021-08-28

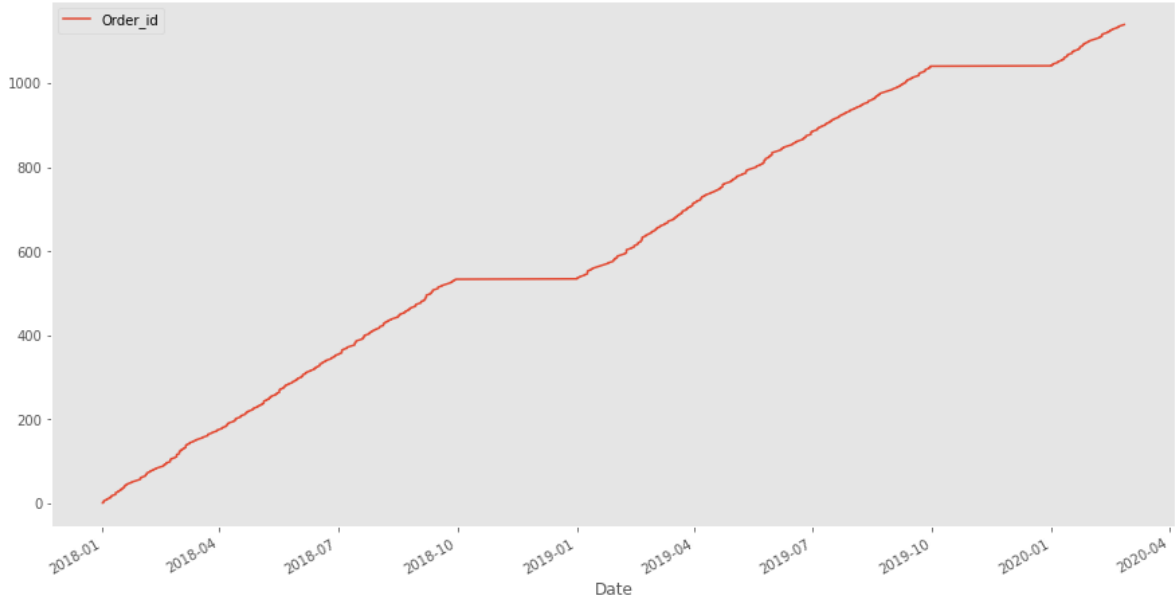
```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 20641 entries, 2018-01-01 to 2020-02-26
Data columns (total 1 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Order_id  20641 non-null  int64
dtypes: int64(1)
memory usage: 838.6 KB
```

In [21]:

```
from pylab import rcParams
from IPython.display import display

rcParams['figure.figsize'] = 15,8
df2.plot();
plt.grid();
```

executed in 1.10s, finished 17:11:48 2021-08-28



Summary on Trends:

We can clearly see that there is High increase in Order trends and overall the Trend is on Positive/Increasing.

Every October Month we can see the Highest Orders and it is Increasing too.

After October, November to January are the Flat line Sales which is constant in this 3 months there is no Trend, Need to understand the reason behind it, weather is it Natural/Season or No discounts.

So February to October is having Trend of increasing Orders (October is the Highest)

November to January is having the Flat sales (lower than previous months) which is constant thorugh out all the 3 years.

In [ ]:

```

```

Q:2 Use of Market Basket Analysis (Association Rules) -->Write Something about the association rules and its relevance in this case -->Add KNIME workflow Image or Python package used -->Write about threshold values of Support and Confidence

Association Rules:

There are 3 Rules in Market Basket Analysis:

1: Support: It identifies the number of times one product has been baught in one basket irrespective of number of baksets. It should always be miniumum as we will use the support of (0.1) means 10 %.

2: Confidence: By using Conditional probability it calculates the chances of Support given (which is of 10% as per the above) that 10% of chance that Product is present in every Basket,.

So combination of Support and Confidence gives us the probability that one Specific Product can be recommended.

3: Lift: It gives us the weightage of every single product present in the Basket which helps us to determine the Best Product to be recommend.

In [ ]:

```

```

In [27]:

```
basket=df.groupby(['Order_id', 'Product'])['Product'].count().unstack().reset_index().

```

executed in 297ms, finished 17:52:59 2021-08-28

In [28]:

```
basket.head()
```

executed in 72ms, finished 17:53:09 2021-08-28

Out[28]:

Product Order_id	all-purpose	aluminum foil	bagels	beef	butter	cereals	cheeses	coffee/tea	dinner rolls	dis liquid/
1	3.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	2.0	
2	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	
3	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	1.0	
4	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	
5	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	

5 rows × 37 columns

In [29]:

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

executed in 1.15s, finished 17:57:58 2021-08-28

In [30]:

```
def encode_zero_one(x):
    if x <= 0:
        return 0
    if x >= 1:
        return 1
```

executed in 17ms, finished 17:58:13 2021-08-28

In [31]:

```
basket=basket.applymap(encode_zero_one)
```

executed in 66ms, finished 17:59:10 2021-08-28

In [35]:

```
#### Find the support for itemsets using Apriori

### With Support value = 0.1

itemsets = apriori(basket, min_support = 0.1, use_colnames = True, low_memory=True)
itemsets
```

executed in 91ms, finished 18:12:36 2021-08-28

Out[35]:

	support	itemsets
0	0.374890	(all- purpose)
1	0.384548	(aluminum foil)
2	0.385426	(bagels)
3	0.374890	(beef)
4	0.367867	(butter)
...	...	...
698	0.172959	(waffles, toilet paper)
699	0.162423	(yogurt, toilet paper)
700	0.149254	(waffles, tortillas)
701	0.152766	(yogurt, tortillas)
702	0.173837	(waffles, yogurt)

703 rows × 2 columns

Support is 0.1 (10%) which means that 10 % of time One product cannot be bought in every customer Basket, which means that 10 % of chance of buying the same one product in every customer's basket, and 90 % of chance that every basket will have different products.

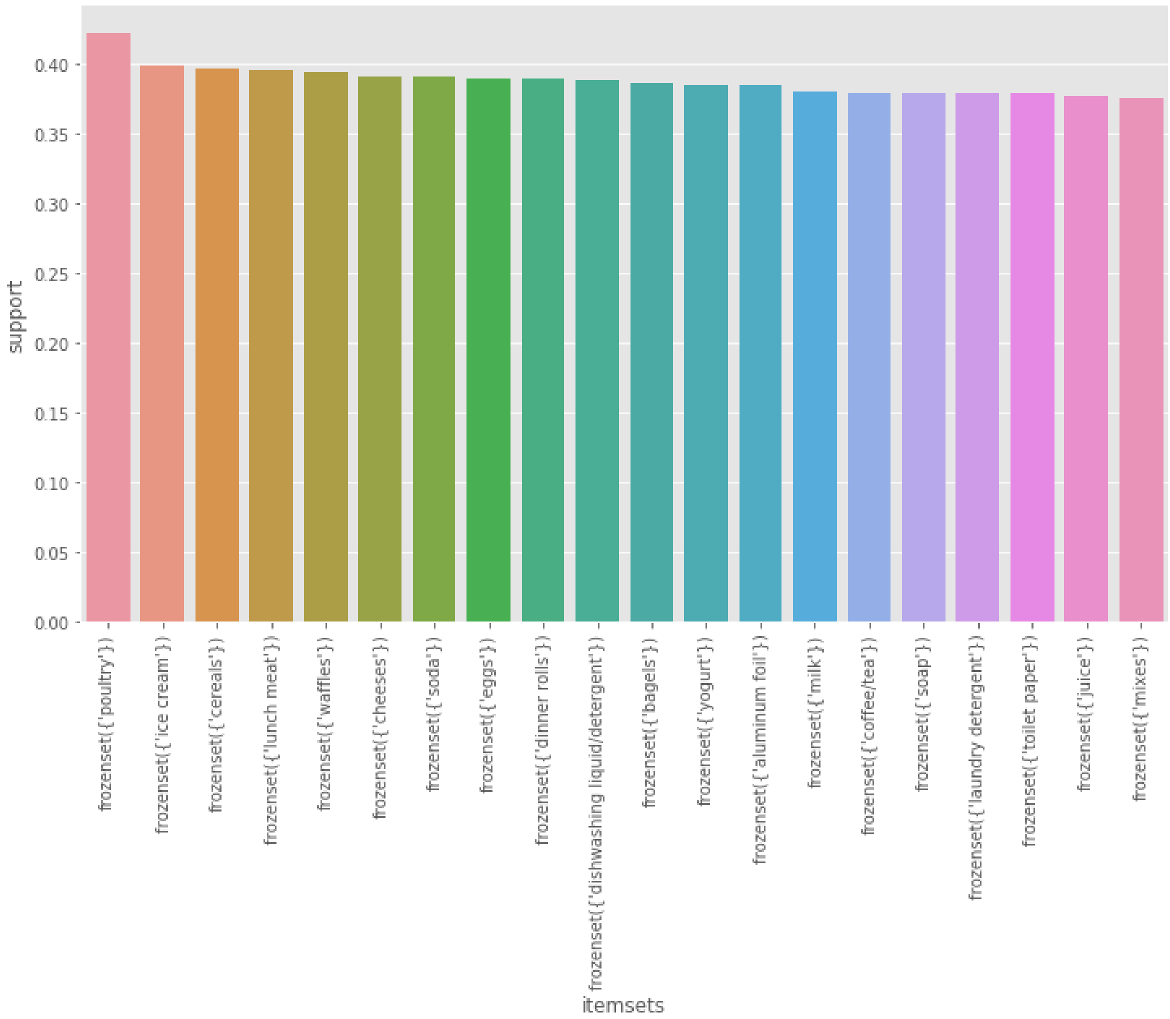
Support is 0.08 (8%) which means that 8 % of time One product cannot be bought in every customer Basket, which means that 8 % of chance of buying the same one product in every customer's basket, and 92 % of chance that every basket will have different products.

In [47]:

```
plt.figure(figsize=(12,7))
sns.barplot(itemsets.sort_values('support',ascending=False).iloc[0:20,1],
            itemsets.sort_values('support',ascending=False).iloc[0:20,0])
plt.xticks(rotation=90)
plt.show()

```

executed in 512ms, finished 18:37:45 2021-08-28



As we know if we increase the support value our Rows will get decrease, so we will be using the 0.1 % (10%) of Suport value.

Threshold Values:

For Support : 0.1 (10%)

For Confidence : 0.5 (50%)

In [ ]:

```
1
```

Q3: Associations Identified --> Put the associations in a tabular manner --> Explain about support, confidence, & lift values that are calculated

In [48]:

```
basket = association_rules(itemsets, metric ="lift")
basket = basket.sort_values(['lift','confidence'], ascending =[False, False])
```

executed in 55ms, finished 18:37:51 2021-08-28

In [49]:

```
basket.head()
```

executed in 45ms, finished 18:37:56 2021-08-28

Out[49]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leve
3254	(lunch meat, sandwich loaves)	(individual meals)	0.146620	0.375768	0.082529	0.562874	1.497929	0.02
3255	(individual meals)	(lunch meat, sandwich loaves)	0.375768	0.146620	0.082529	0.219626	1.497929	0.02
2652	(spaghetti sauce, poultry)	(dinner rolls)	0.171203	0.388938	0.099210	0.579487	1.489923	0.03
2657	(dinner rolls)	(spaghetti sauce, poultry)	0.388938	0.171203	0.099210	0.255079	1.489923	0.03
2244	(ketchup, cheeses)	(sandwich loaves)	0.160667	0.349429	0.082529	0.513661	1.470000	0.02

In [50]:

```
basket.tail()
```

executed in 45ms, finished 18:37:58 2021-08-28

Out[50]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	lev
1033	(lunch meat)	(pasta)	0.395083	0.371378	0.142230	0.360000	0.969362	-0.0(
1089	(milk)	(waffles)	0.380158	0.394205	0.143108	0.376443	0.954942	-0.0(
1088	(waffles)	(milk)	0.394205	0.380158	0.143108	0.363029	0.954942	-0.0(
211	(butter)	(beef)	0.367867	0.374890	0.128183	0.348449	0.929469	-0.0(
210	(beef)	(butter)	0.374890	0.367867	0.128183	0.341920	0.929469	-0.0(

Support Value Calculated:

As our Threshold Value for Support is 0.1 (10%), Hence we could see that our Support Value is ranging from 0.08 to 0.12 which means 10% of time every product each product is present in every Basket.

Confidence Value Calculated:

As our Confidence Value is ranging from 34 % to 56 % which means that there is chance of 34% to 56% that every product is available or present in each basket based on our Support value which is of (10% of times every basket has same product).

So Confidence value gives us Probability/Chance how much our Support Value (10%) which we assumed about 10 percent of chances that every basket has same product is True or not.

How much our assumtion on Support value is possible, Confidence gives us the Probability.

Lift:

It calculates which Product/Item can be first recommend to the Basket based on the Items present in the existing basket, Suppose if one basket has 2 products and other basket also has same 2 products so depending upon the LIFT value whichever product will have the HIGHER LIFT value system will receommend that product first.

So its, depending on the Acending order of LIFT our system calculates and put the item/product.

In [ ]:

```

```

Q4: Suggestion of Possible Combos with Lucrative Offers --> Write recommendations --> Make discount offers or combos (or buy two get one free) based on the associations and your experience

In [51]:

```
# Lets see the Possible combos top 50.

basket.head(50)
```

executed in 209ms, finished 18:49:40 2021-08-28

Out[51]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	coı
3254	(lunch meat, sandwich loaves)	(individual meals)	0.146620	0.375768	0.082529	0.562874	1.497929	0.027433	1
3255	(individual meals)	(lunch meat, sandwich loaves)	0.375768	0.146620	0.082529	0.219626	1.497929	0.027433	1
2652	(spaghetti sauce, poultry)	(dinner rolls)	0.171203	0.388938	0.099210	0.579487	1.489923	0.032623	1
2657	(dinner rolls)	(spaghetti sauce, poultry)	0.388938	0.171203	0.099210	0.255079	1.489923	0.032623	1
2244	(ketchup, cheeses)	(sandwich loaves)	0.160667	0.349429	0.082529	0.513661	1.470000	0.026387	1

Possible Combos: Which have good Lift Value

Product	Combo with	Lift Value
Lunch Meat & Sandwich Loaves	Individual Meals	1.49
Spaghetti Sauce & Poultry	Dinner Rolls	1.48
Ketchup & Cheeses	Sandwich Loaves	1.47
Juice & Dinner Rolls	Spaghetti Sauce	1.46
Poulty & Aluminium Foil	Juice	1.44
Beef & Soda	Eggs	1.43

In [ ]:

```

```

Discount offers/Combos:

Buy 3 and get 1 Free Sandwich Loaves.

20 % Discount on any Poultry Item.

Buy 3 Dozens of Eggs and get 1 Spaghetti Sauce Free.

Buy 2 Dinner Rolls and get 50 % discount on Soda.

Buy 3 Ketcheup at Price of 2.

Purchase any item above 1000 (INR) and get one Aluminium coil absolutely FREE.

In [ ]:

```
1
```

Recommendations:

People are highly active towards the Poultry products, hence store can put this products in one dedicated section.

Soda with Bakery and Eatable items and try to see which brand of soda is highly active with different bakery and eatable foods.

Check whether Ice-Cream section can be adjusted at the Entrance location or in Billing Counter.

Make one day for flat price on those products which are sitting in Inventory.

Have one Display at the Entrance for the above Combos/Offers.

Send all the Combo/Offers to the Loyal customers which have past history of buying those products via mail,message or by calling.

See if you can get in Bulk for those products which are selling fast.

In [ ]:

```
1
```