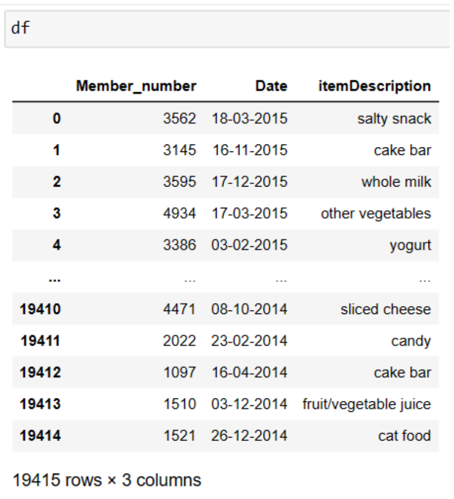
MCSD2123
Assignment 1
NAME: MOHAMMAD IQBAL
MATRIC NO: MCDS221035

SECTION 1: Exploratory Data Analysis (EDA)

In Exploratory Data Analysis, the dataset is examined and analyzed to summaries the main characteristics of the dataset. Often time, EDA is performed at the beginning of any data analysis process to help identify patterns, relationship, and anomalies in the data. In this section, the EDA will be divided into data description, descriptive statistics and visualization.

a. About dataset

The raw dataset and its data type will be shown on this figure below:

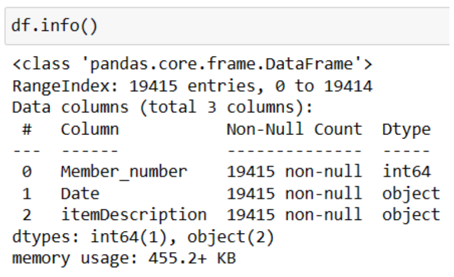


```
df
```

	Member_number	Date	itemDescription
0	3562	18-03-2015	salty snack
1	3145	16-11-2015	cake bar
2	3595	17-12-2015	whole milk
3	4934	17-03-2015	other vegetables
4	3386	03-02-2015	yogurt
...	...	...	...
19410	4471	08-10-2014	sliced cheese
19411	2022	23-02-2014	candy
19412	1097	16-04-2014	cake bar
19413	1510	03-12-2014	fruit/vegetable juice
19414	1521	26-12-2014	cat food

19415 rows × 3 columns

(a)



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19415 entries, 0 to 19414
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Member_number    19415 non-null  int64 
 1   Date             19415 non-null  object
 2   itemDescription  19415 non-null  object
dtypes: int64(1), object(2)
memory usage: 455.2+ KB
```

(b)

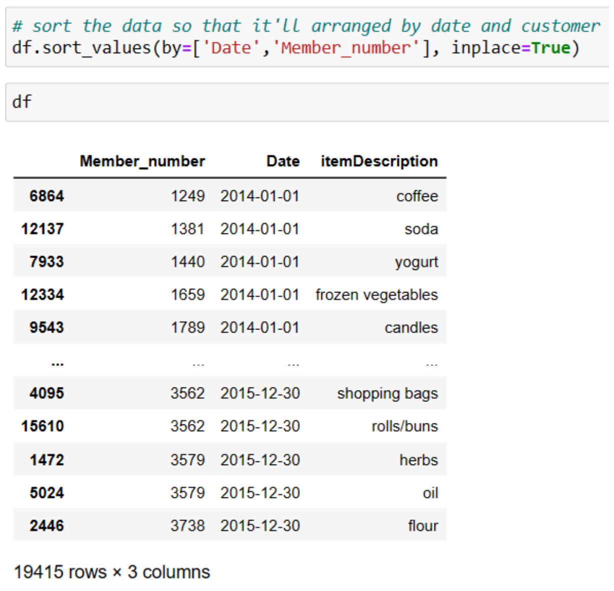
Figure 1. preview of the dataset

Based on figure 1(a), it can be seen that the size of dataset is 19415 rows and 3 columns. The description of the columns are :

Table 1. dataset description

Column name	Data type	Descriptions
Member_number	Int64	Member id
Date	Object	Date of transaction
itemDescription	Object	Item name

Based on table 1, we can see that the date column is not in date format, thus we need to change the format later in the preprocessing step. Figure 1(a) show that the date column is not sort correctly.



```
# sort the data so that it'll arranged by date and customer
df.sort_values(by=['Date','Member_number'], inplace=True)
```

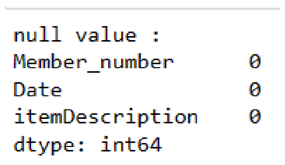
```
df
```

	Member_number	Date	itemDescription
6864	1249	2014-01-01	coffee
12137	1381	2014-01-01	soda
7933	1440	2014-01-01	yogurt
12334	1659	2014-01-01	frozen vegetables
9543	1789	2014-01-01	candles
...	...	...	...
4095	3562	2015-12-30	shopping bags
15610	3562	2015-12-30	rolls/buns
1472	3579	2015-12-30	herbs
5024	3579	2015-12-30	oil
2446	3738	2015-12-30	flour

19415 rows × 3 columns

Figure 2. dataset after sorted

After dataset is sorted by the date, it can be seen that each record consist of individual item that purchased by certain customer. Because itemDescription only consist of one item, thus, the same customer might do the transaction more than once in the same date.



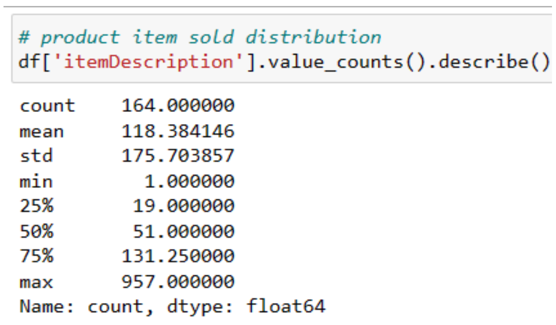
```
null value :
Member_number      0
Date               0
itemDescription    0
dtype: int64
```

Figure 3. null value

The data quality in term of null value, the dataset don’t have any null value, so it can be said that data quality in term of null value is good data.

b. Descriptive statistics

In this section, the descriptive analysis will be perform towards the dataset. Initially, the statistical component of itemDescription will be shown below.

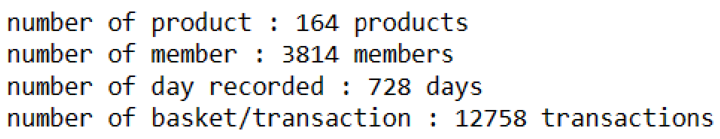


```
# product item sold distribution
df['itemDescription'].value_counts().describe()
```

```
count     164.000000
mean      118.384146
std       175.703857
min         1.000000
25%        19.000000
50%        51.000000
75%       131.250000
max       957.000000
Name: count, dtype: float64
```

Figure 4. distribution of item sold

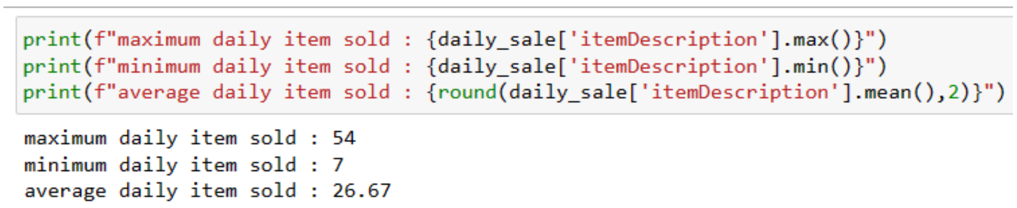
Based on figure above, there is 164 different items in the store with the average of the product being purchased 118.4 times. The standard deviation seems big, the data spread/distribution might be imbalance. Minimum item being purchased is only once and maximum is 957 times. With 25% of the items where purchased 19 times and 75% of items where purchased 131 times.



```
number of product : 164 products
number of member : 3814 members
number of day recorded : 728 days
number of basket/transaction : 12758 transactions
```

Figure 5. total number from dataset

After adding the transaction id column that will be explained in the Preparation of Dataset section, Figure 5 showing the total number towards important features of the data such as total number of product that the store have is 164 products, while total number of member or total number of customer is 3.814 people, total transaction recorded is 728 day, and total of transaction or in this case we can say total of basket is 12.758 transactions.

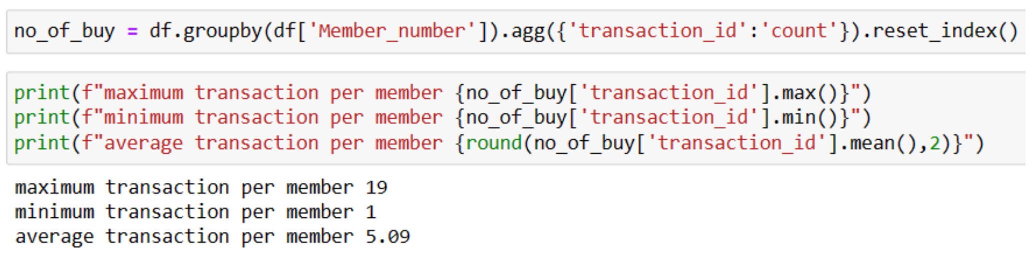


```
print(f"maximum daily item sold : {daily_sale['itemDescription'].max()}")
print(f"minimum daily item sold : {daily_sale['itemDescription'].min()}")
print(f"average daily item sold : {round(daily_sale['itemDescription'].mean(),2)}")
```

```
maximum daily item sold : 54
minimum daily item sold : 7
average daily item sold : 26.67
```

Figure 6. daily item sold statistics

By performing groupby date for every date to the dataset and aggregating the count of each itemDescription we could see the maximum daily transaction is 54 item sola a day, and the minimum daily item sold is 7 transaction a day, while the average transaction is 26.67 item sold a day. From that number, we can infer that the store is not to big.



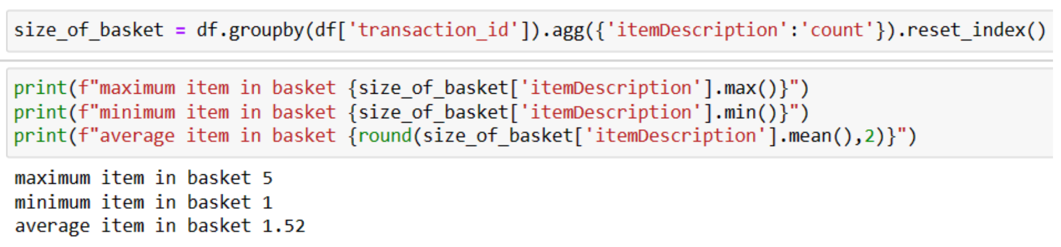
```
no_of_buy = df.groupby(df['Member_number']).agg({'transaction_id':'count'}).reset_index()
```

```
print(f"maximum transaction per member {no_of_buy['transaction_id'].max()}")
print(f"minimum transaction per member {no_of_buy['transaction_id'].min()}")
print(f"average transaction per member {round(no_of_buy['transaction_id'].mean(),2)}")
```

```
maximum transaction per member 19
minimum transaction per member 1
average transaction per member 5.09
```

Figure 7. member transaction statistics

Same as the previous one, grouping the data by the member id and aggregating the count of itemDescriptions it can be seen that for two years transactions records, the customer that do the most transaction is 19 transaction and the least transaction is only 1 transaction while the average of customer do around 5.09 transactions.



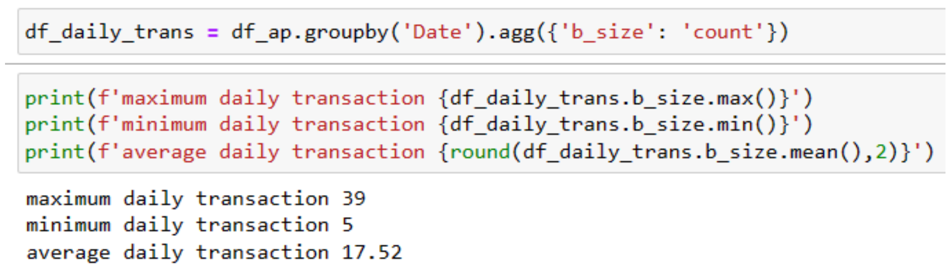
```
size_of_basket = df.groupby(df['transaction_id']).agg({'itemDescription':'count'}).reset_index()
```

```
print(f"maximum item in basket {size_of_basket['itemDescription'].max()}")
print(f"minimum item in basket {size_of_basket['itemDescription'].min()}")
print(f"average item in basket {round(size_of_basket['itemDescription'].mean(),2)}")
```

```
maximum item in basket 5
minimum item in basket 1
average item in basket 1.52
```

Figure 8. basket size statistics

With creating transaction_id column, we can analyze the size of the basket by performing group by towards that column. It shown that the maximum item that customer purchase at the same transaction is 5 items, and the minimum is 1 item, while the average is 1.52 item. We can infer that the customer tend to not buy a lot of stuff in a time considering the store have 164 different product on sale.



```
df_daily_trans = df_ap.groupby('Date').agg({'b_size': 'count'})
```

```
print(f'maximum daily transaction {df_daily_trans.b_size.max()}')
print(f'minimum daily transaction {df_daily_trans.b_size.min()}')
print(f'average daily transaction {round(df_daily_trans.b_size.mean(),2)}')
```

```
maximum daily transaction 39
minimum daily transaction 5
average daily transaction 17.52
```

Figure 9. daily transaction statistics

After preparation of data done, we can observe daily transaction. Statistically, maximum number of transaction in a day is 39 transaction, minimum is 5 transaction, and average daily transaction is 17.52 transaction.

c. Visualization

Since the time series data especially transaction data is time bound, so the EDA will perform to see the pattern in certain period of the time. Using aggregated daily transaction data, the quantity of daily transaction is shown in the figure below :

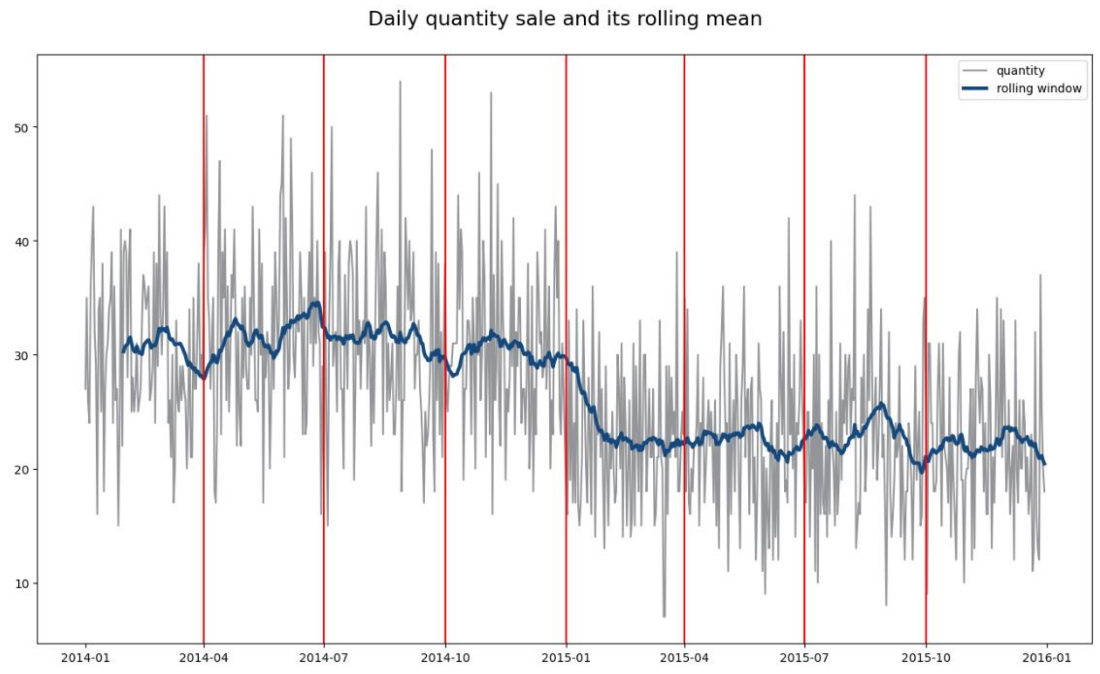


Figure 10. daily quantity sales

Based on quantity of daily sales throughout the transaction recorded and with the aid of rolling mean with 30 windows to see the pattern, there is no notable trend that seen from the figure except that the overall quantity of sales in 2015 is lower than overall sales in 2014.

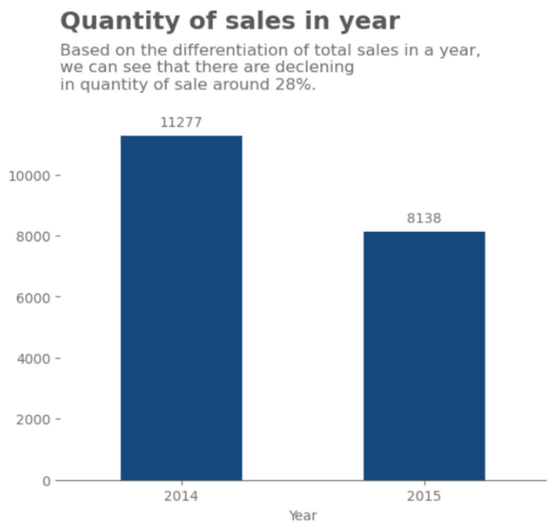


Figure 11. total quantity of sales in a year 2014 vs 2015

As it seen in the daily quantity sales, the sales of 2015 is 28% lower than 2014 sales which is 8138 items and 11277 items respectively.

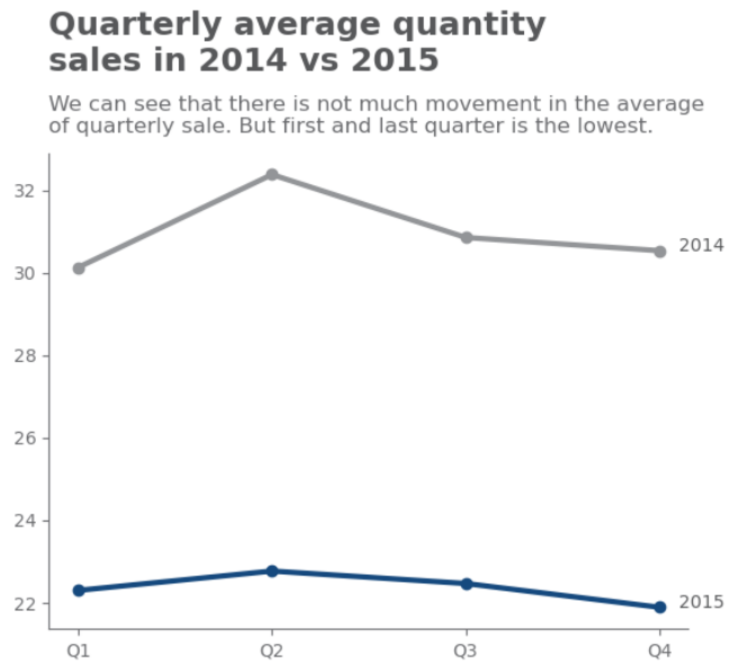


Figure 12. quantity of average sales in a quarter 2014 vs 2015

When it dig deeper into time frame of quarter, the pattern that can be seen is that the first and last quarter is among the lowest.

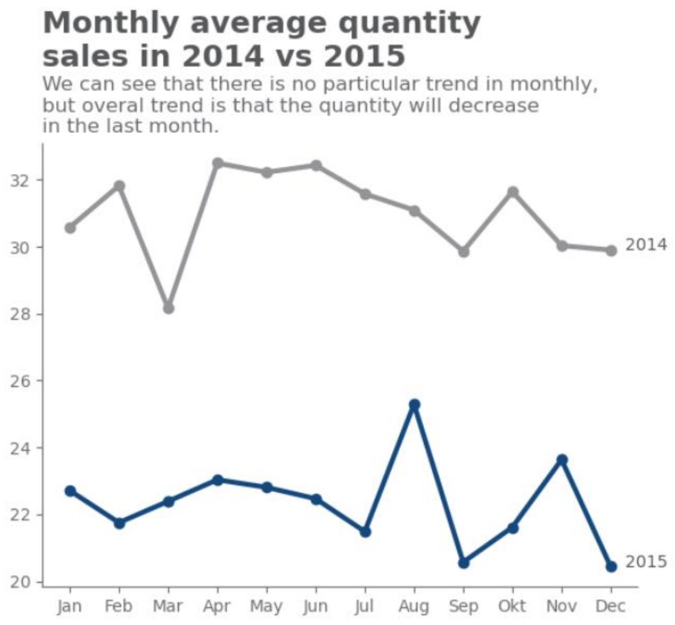


Figure 13. quantity of average sales in each month 2014 vs 2015

Based on figure 11, there is no particular trend than can be infer except that the last month transaction tend to decrease throughout the time.

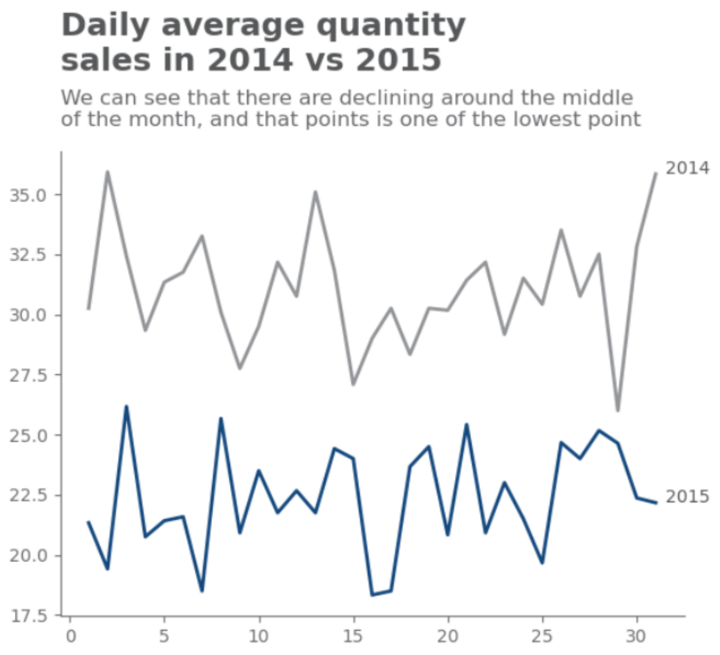


Figure 14. quantity of average sales in the day of the month 2014 vs 2015

The day of the month trend is that at around the middle of the month, the quantity sales is one of the lowest sales for the entire day in a month.

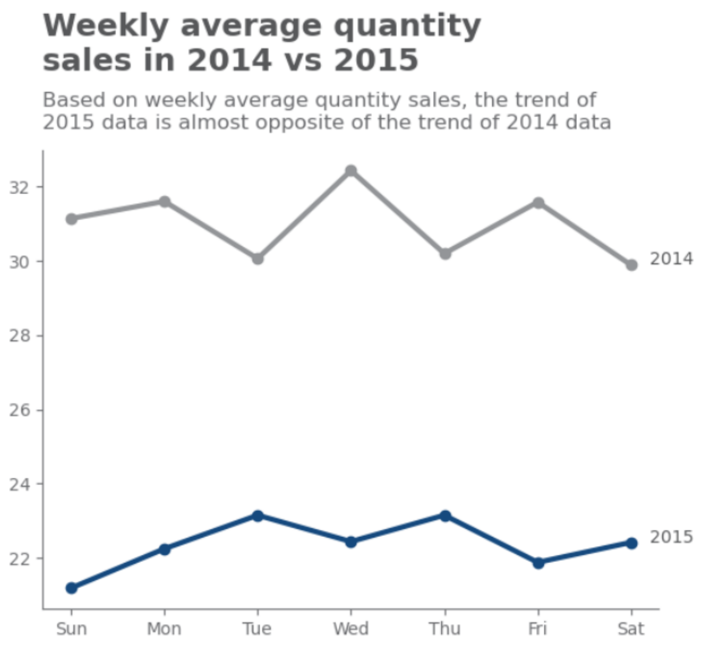


Figure 15. quantity of average sales in the day of the week 2014 vs 2015

The common transaction pattern in the day of week transaction data is that the spend in the weekend tend to be highest and usually the transaction will rise with the weekend as its peak. But that pattern cannot be seen on the dataset.

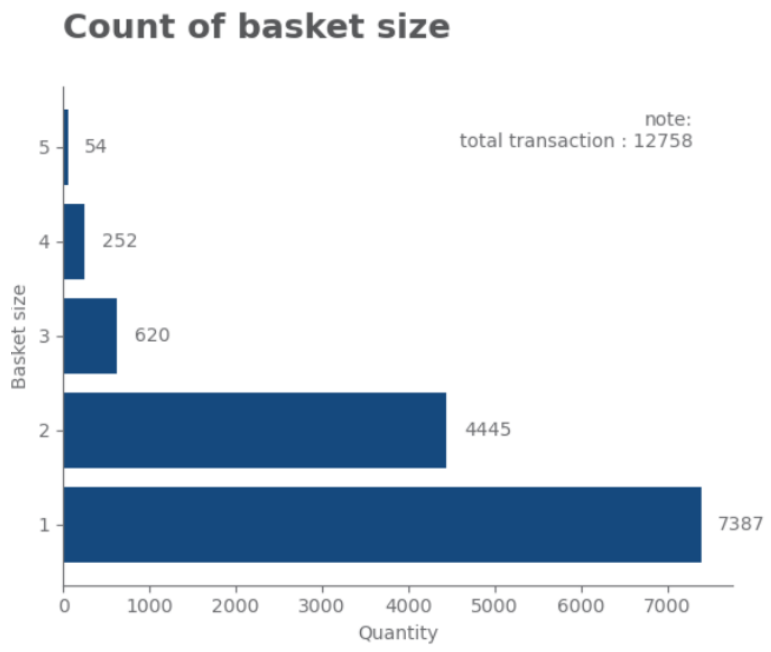
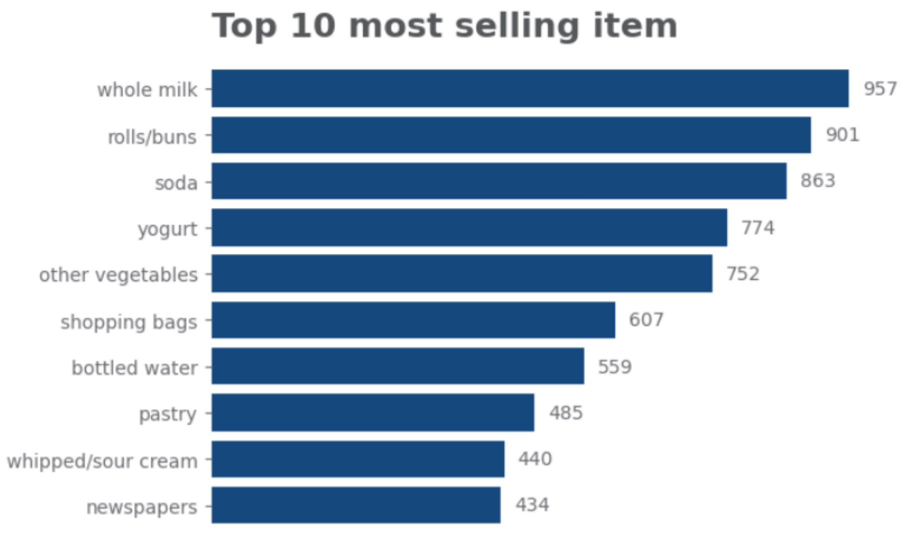
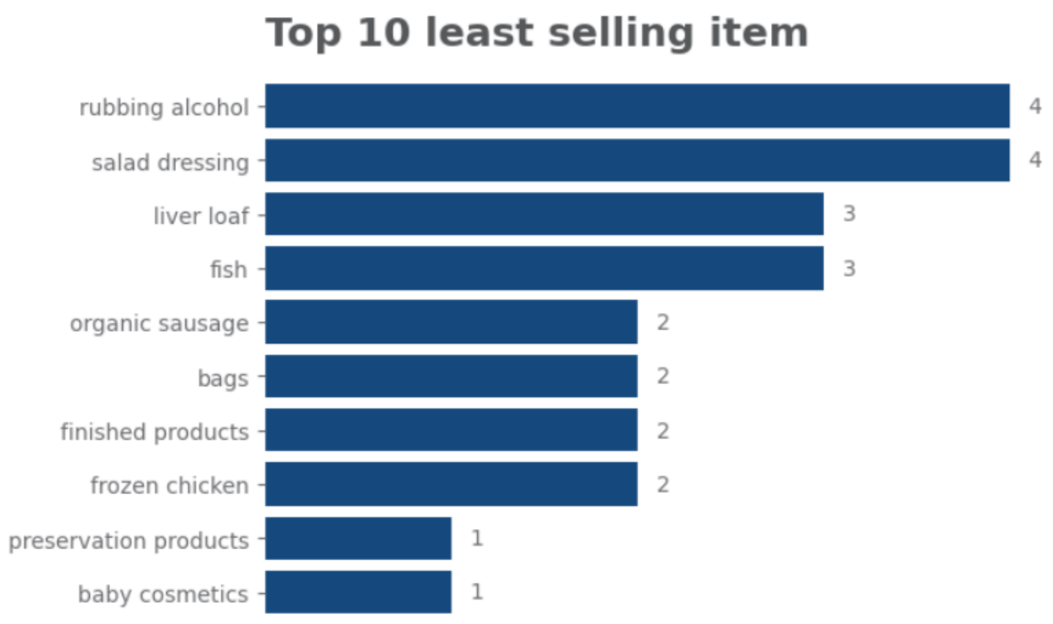


Figure 16. basket size distribution

Figure above showing that majority of transaction only consist of one item and only small of number that do the transaction with up to five items.



(a)



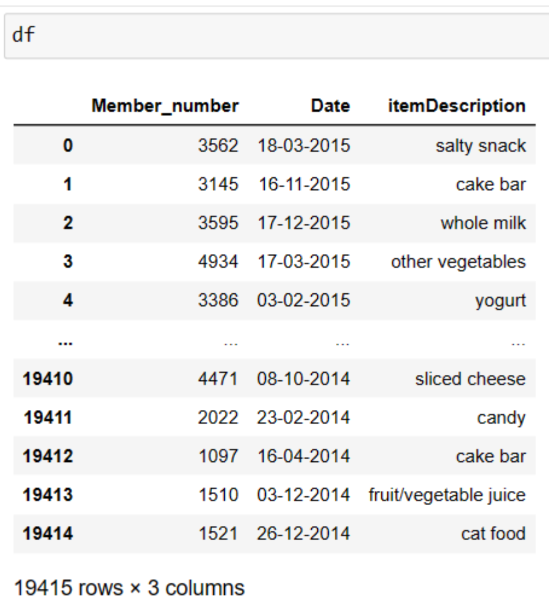
(b)

Figure 17 (a) top 10 most selling item and, (b) top 10 least selling item

Figure above showing the highest and lowest selling product that can be justification to stop selling certain product or add more stock of certain product.

SECTION 2: Preparation of Dataset

Considering this as a raw dataset :



```
df
```

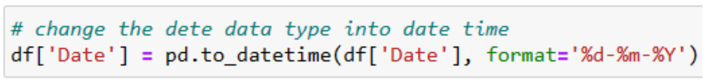
	Member_number	Date	itemDescription
0	3562	18-03-2015	salty snack
1	3145	16-11-2015	cake bar
2	3595	17-12-2015	whole milk
3	4934	17-03-2015	other vegetables
4	3386	03-02-2015	yogurt
...	...	...	...
19410	4471	08-10-2014	sliced cheese
19411	2022	23-02-2014	candy
19412	1097	16-04-2014	cake bar
19413	1510	03-12-2014	fruit/vegetable juice
19414	1521	26-12-2014	cat food

19415 rows × 3 columns

Figure 18. raw dataset

The step that taken to prepare the dataset to the format that usable for association rule mining are as follows:

1. Change date data type into datetime format

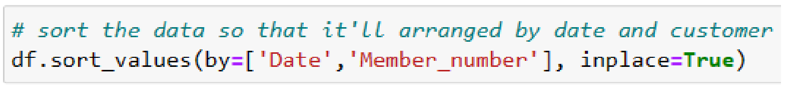


```
# change the dete data type into date time
df['Date'] = pd.to_datetime(df['Date'], format='%d-%m-%Y')
```

Figure 19. change date format

Converting date data type using built in pandas function to_datetime, so that the EDA can be done

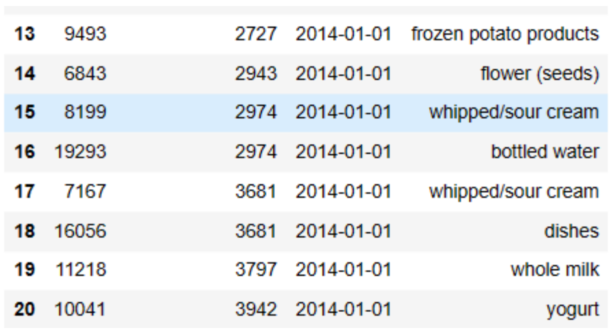
2. Sort the data based on date and member_number



```
# sort the data so that it'll arranged by date and customer
df.sort_values(by=['Date','Member_number'], inplace=True)
```

Figure 20. sorting the data by date and member number

Sorting the data using sort_value by date and member_number so that the data will arranged based on the same date after another and in one date, the same member_number will stack one after another if the customer do the transaction more than once in a day, like shown in the figure below



13	9493	2727	2014-01-01	frozen potato products
14	6843	2943	2014-01-01	flower (seeds)
15	8199	2974	2014-01-01	whipped/sour cream
16	19293	2974	2014-01-01	bottled water
17	7167	3681	2014-01-01	whipped/sour cream
18	16056	3681	2014-01-01	dishes
19	11218	3797	2014-01-01	whole milk
20	10041	3942	2014-01-01	yogurt

Figure 21. dataset after sorted

The member id 2974 and 3681 do the transaction more than once on that date.

3. Create transaction_id column

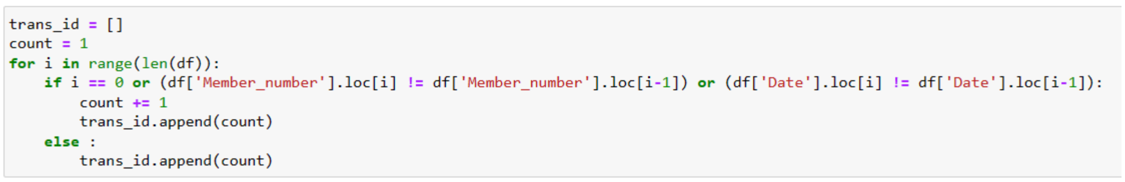
In creating transaction_id column, the process is based on assumption that :

a. The transaction that done in one day is assumed as one transaction

b. The transaction by the same member_number in the same date is considering as one basket/transaction.

After we have sorted data by date and member_number and index is being reset we can easily identified one basket or one transaction. By **observing weather the records and previous records is not in the same day, and weather the records and previous records done by the different member_number**.

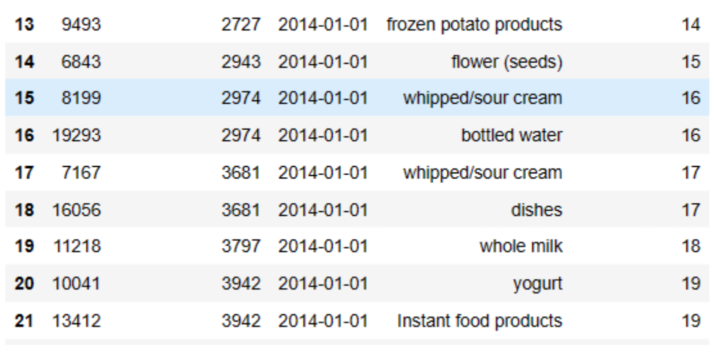
- If satisfying that condition, mean that it is different basket, then transaction_id will increase
- if not, it means that it is in the one basket so the transaction_id will be the same eith previous records



```
trans_id = []
count = 1
for i in range(len(df)):
    if i == 0 or (df['Member_number'].loc[i] != df['Member_number'].loc[i-1]) or (df['Date'].loc[i] != df['Date'].loc[i-1]):
        count += 1
        trans_id.append(count)
    else :
        trans_id.append(count)
```

Figure 22. adding transaction id

The result of that operation is in a form of list, then that list is added as a new column that will produce this dataset:



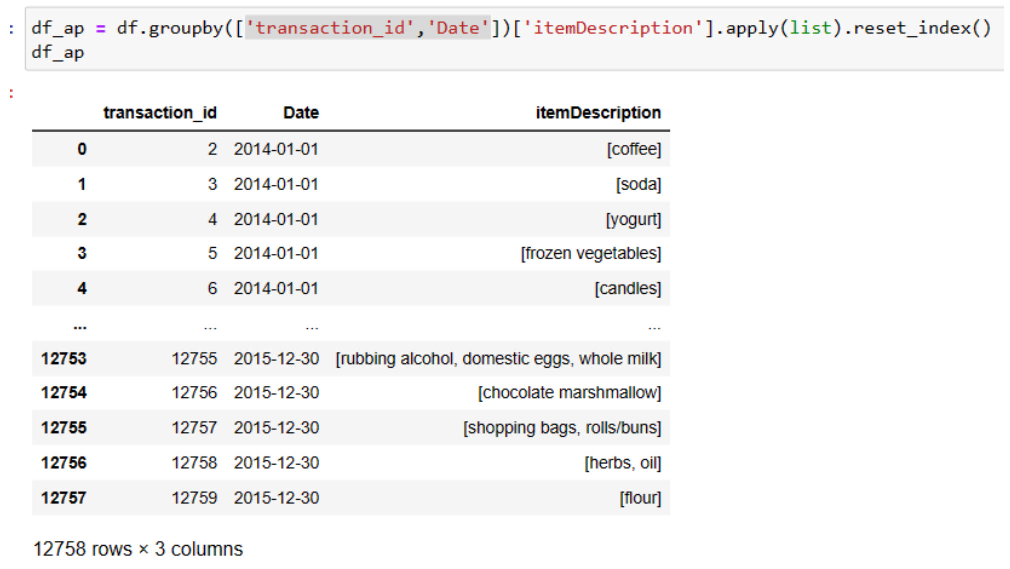
13	9493	2727	2014-01-01	frozen potato products	14
14	6843	2943	2014-01-01	flower (seeds)	15
15	8199	2974	2014-01-01	whipped/sour cream	16
16	19293	2974	2014-01-01	bottled water	16
17	7167	3681	2014-01-01	whipped/sour cream	17
18	16056	3681	2014-01-01	dishes	17
19	11218	3797	2014-01-01	whole milk	18
20	10041	3942	2014-01-01	yogurt	19
21	13412	3942	2014-01-01	Instant food products	19

Figure 23. result of generating transaction id

As we can see in figure above, record that have same date and same member_number will have a same transaction id, it indicating that on that date, it is considering as one transaction. There is another way to prepare the data so it can be used to do association mining. Rather than create transaction id and group by transaction id later, it can just group by the date and member id, the result will be the same. But this method is chosen because with the transaction id, we can do EDA with more variable.

4. Group the data by transaction_id

After transaction id column is generated, then the data is grouped by transaction id and do apply(list) so that the same item in the same transaction id grouped in a list.



```
df_ap = df.groupby(['transaction_id','Date'])['itemDescription'].apply(list).reset_index()
df_ap
```

	transaction_id	Date	itemDescription
0	2	2014-01-01	[coffee]
1	3	2014-01-01	[soda]
2	4	2014-01-01	[yogurt]
3	5	2014-01-01	[frozen vegetables]
4	6	2014-01-01	[candles]
...	...	...	...
12753	12755	2015-12-30	[rubbing alcohol, domestic eggs, whole milk]
12754	12756	2015-12-30	[chocolate marshmallow]
12755	12757	2015-12-30	[shopping bags, rolls/buns]
12756	12758	2015-12-30	[herbs, oil]
12757	12759	2015-12-30	[flour]

12758 rows × 3 columns

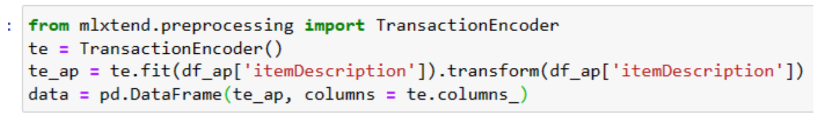
Figure 24. grouping by date and transaction id

The final format is that each rows is represent one transaction/basket, with the itemDescription is the itemset inside the basket. There is another way to prepare the data so it can be used to do association mining.

Rather than create transaction id and group by transaction id later, it can just group by the date and member id, which resulting similar form of dataset. But I choose the first methods because with the transaction id, we can do EDA with more variable.

5. Perform transaction encoding

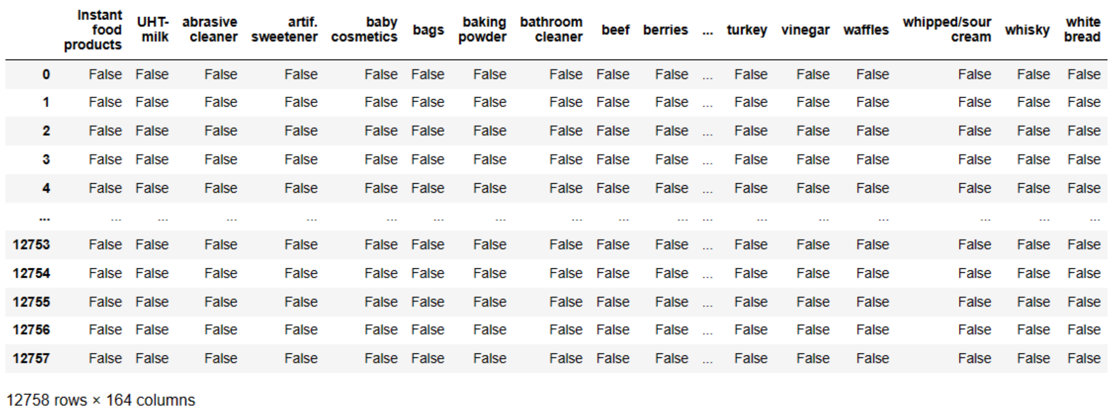
   Because mlxtend.frequent_patterns.apriori will be used in the next chapter, the dataset need to reform so that will works with that library. The process is called encoding, using mlxtend.preprocessing.TransactionEncoder.



```
from mlxtend.preprocessing import TransactionEncoder
te = TransactionEncoder()
te_ap = te.fit(df_ap['itemDescription']).transform(df_ap['itemDescription'])
data = pd.DataFrame(te_ap, columns = te.columns_)
```

Figure 25. encoding the dataset

After dataset is encoded, the result is in a form of a list. Then the list is converted into data frame that the result will be the same if we use one-hot-encoder but, rather than returning class of 0 and 1, it will return the Boolean.



	Instant food products	UHT-milk	abrasive cleaner	artif. sweetener	baby cosmetics	bags	baking powder	bathroom cleaner	beef	berries	...	turkey	vinegar	waffles	whipped/sour cream	whisky	white bread
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
12753	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
12754	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
12755	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
12756	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
12757	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False

12758 rows × 164 columns

Figure 26. result of encoding

As it can be seen in the figure above, there are 12758 rows that represent the total basket/transaction and 164 columns that represent the number of unique item. Figure showing the data format that useable for association rule mining.

SECTION 3: Methodology

Apriori

In this works, the algorithm that used in association rule mining is apriori algorithm. Generally speaking apriori work in three steps. The preliminary step is set the minimum support which is minimum occurrence of item throughout entire basket. The first step is forming the table that containing occurrence of each item in dataset with only one item set, we can call it c-1. After that the support count is compared with minimum_support. If the itemset support count is less than the threshold, then the pruning is perform by deleting that itemset. This process will repeatedly run with increasing number of itemset until there is no itemset left. In this work, mxltend library will be used to perform apriori.

Determining minimum support

There is no specific ways to choose the minimum support value. In this work, considering the size of dataset that not too big, we will choose the least support of the item. based on figure the minimum occurrence of the item is one, so the minimum support is :

$$(miminimum\ occurance)/(total\ number\ of\ basket)$$

Choosing this small number may resulting in too many itemset and rules. To mitigate that, later we will use filter in certain evaluation parameter like lift, conviction, leverage and zhangs metric to only focus on certain range based on the statistical data of evaluation metrics.

Evaluating result

In order to interpreting the association rule, there are some metrics that helps to evaluate the rule of apriori that can interpret the rule.

1. Support is showing how many occurrence are the itemset divided by all the transaction.
2. Confidence measure how many often items in y appear in transaction that contain x. for instance, if we have confidence 0.15, it means that 15% of item x purchased, item y is also purchased.
3. Likelihood of item of items being bought together, compared to their likelihood of being bought individually is called lift. Let say we have lift equal to 4, it means that the presence of item a in a transaction increases the probability of having item b in the same transaction by 4 times compared to transaction where item a is not present.

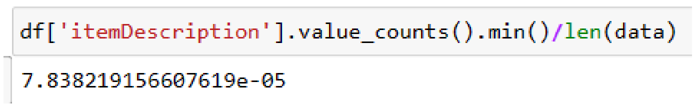
4. Leverage computes the difference between the observed frequency of A and C appearing together and the frequency that would be expected if A and C were independent1.
5. Zhangs metric measure the relationship between products, weather it is positively or negatively correlated.

Discussion

In this work, two scenarios will be undertaken. The first one involves using the entire dataset to perform market basket analysis, while the second scenario utilizes only transactions from a specific time period. By reducing the dataset, we can improve the results of the Apriori algorithm. Let's take the example of Walmart in 2004, which successfully applied association rules to predict demand when hurricanes hit Florida. In that case, Walmart only used transactions from around the hurricane time, and that slice of data during that period provided actionable insights that led Walmart to stock up on strawberry pop tarts, ultimately selling all of them.
The idea behind the second scenario is that customer preferences might vary during certain times due to specific circumstances. Slicing the data for only a certain period might capture patterns unique to that time. Additionally, arranging products in a store can be done regularly, whether monthly, quarterly, or based on events such as when a storm is predicted to hit the city.
Even though the time series analysis in the previous section did not reveal any notable trends, we will take the first quarter of 2014 as an example for the second scenario, assuming that this analysis will be implemented for the first quarter of 2016.

a. **Scenario 1**

   In the first scenario, dataset that will be used is the entire dataset as mentioned in previous sub-section. After dataset is preprocessed, the apriori will be done using mlxtend library. As mentioned above, for initial minimum support, I will use the least support of the item list using by dividing least product occurrence by total transaction.

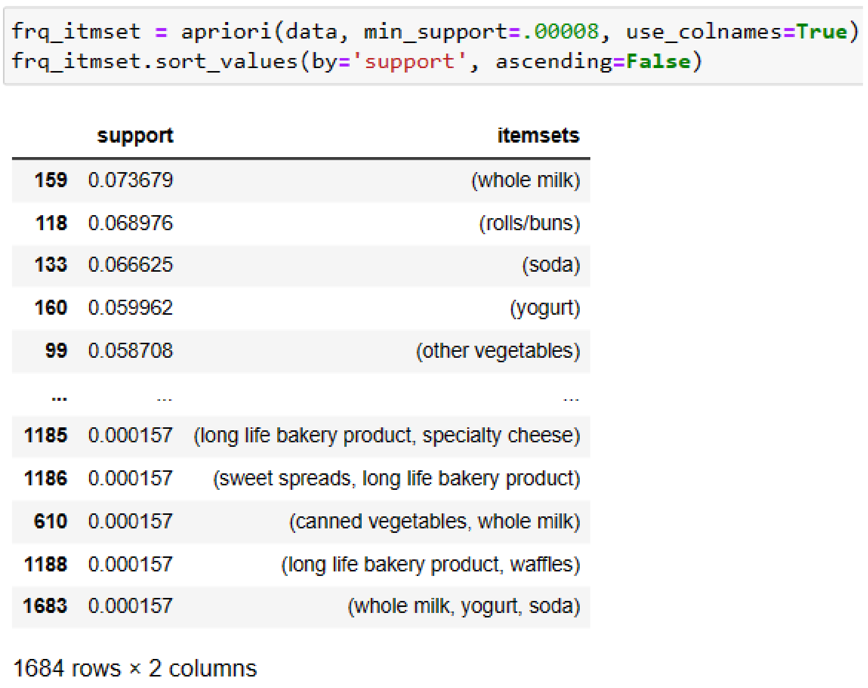


```
df['itemDescription'].value_counts().min()/len(data)
```

```
7.838219156607619e-05
```

Figure 27. initial minimum support

Based on that calculation, we get 0.00008 as minimal support that then included in apriori using mlxtend.



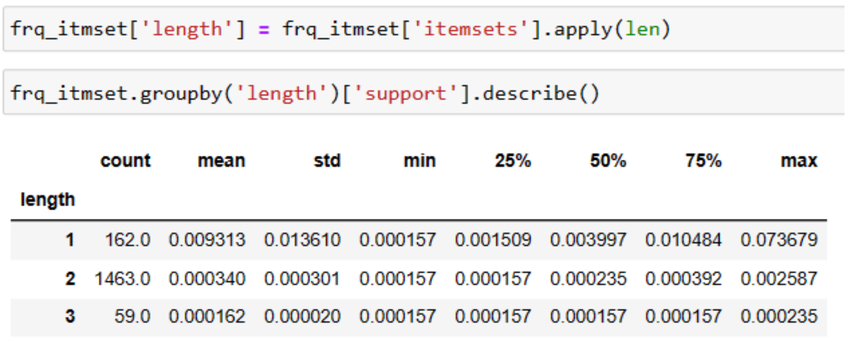
```
frq_itmset = apriori(data, min_support=.00008, use_colnames=True)
frq_itmset.sort_values(by='support', ascending=False)
```

	support	itemsets
159	0.073679	(whole milk)
118	0.068976	(rolls/buns)
133	0.066625	(soda)
160	0.059962	(yogurt)
99	0.058708	(other vegetables)
...	...	...
1185	0.000157	(long life bakery product, specialty cheese)
1186	0.000157	(sweet spreads, long life bakery product)
610	0.000157	(canned vegetables, whole milk)
1188	0.000157	(long life bakery product, waffles)
1683	0.000157	(whole milk, yogurt, soda)

1684 rows × 2 columns

Figure 28. apriori result for scenario 1

Based on the result above, it produce 1684 combination of itemset. To determine the threshold for creating the rule, the statistical descriptive of the support will be checked.



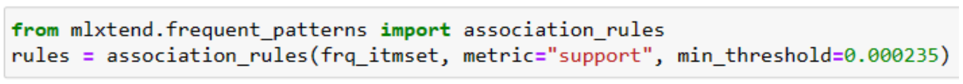
```
frq_itmset['length'] = frq_itmset['itemsets'].apply(len)
```

```
frq_itmset.groupby('length')['support'].describe()
```

	count	mean	std	min	25%	50%	75%	max
length								
1	162.0	0.009313	0.013610	0.000157	0.001509	0.003997	0.010484	0.073679
2	1463.0	0.000340	0.000301	0.000157	0.000157	0.000235	0.000392	0.002587
3	59.0	0.000162	0.000020	0.000157	0.000157	0.000157	0.000157	0.000235

Figure 29. statistical component of apriori

Based on figure above, majority of itemset is consist of two items. Thus, the support of 50% quartile of the two itemset will be chosen as a threshold to determine rules.



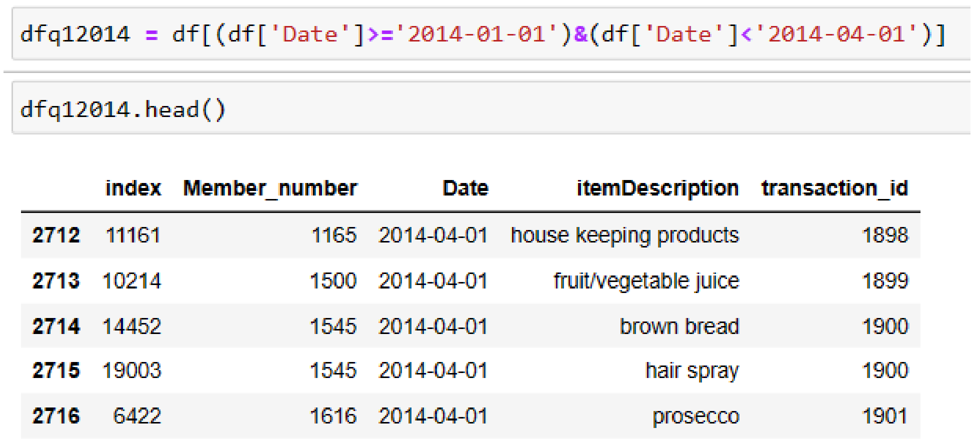
```
from mlxtend.frequent_patterns import association_rules
rules = association_rules(frq_itmset, metric="support", min_threshold=0.000235)
```

Figure 30. generating rule

The result and discussion of the rule will be discussed in the net section.

b. Scenario 2

Scenario two is based on the hypothesis that customer preferences will vary depending on certain factors such as holidays, special events, the day of the week, the day of the month, and other factors. The only notable pattern that shows in exploratory data analysis is that sales touch their lowest point around the middle of the month. But for the sake of simplicity, scenario 2 will use Q1 in 2014 as it gives better selling performance compared to 2015, assuming that this analysis will be used to support the business in Q1 of 2016.



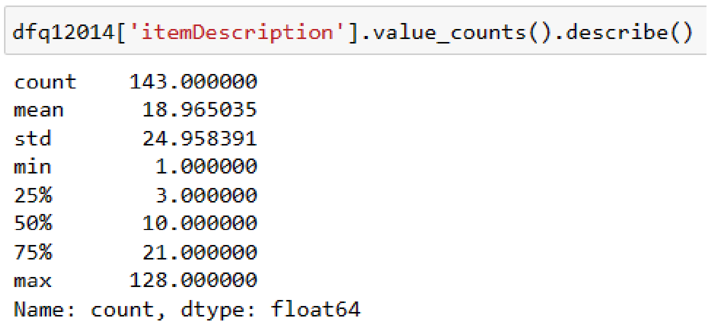
```
dfq12014 = df[(df['Date']>='2014-01-01')&(df['Date']<'2014-04-01')]
```

```
dfq12014.head()
```

	index	Member_number	Date	itemDescription	transaction_id
2712	11161	1165	2014-04-01	house keeping products	1898
2713	10214	1500	2014-04-01	fruit/vegetable juice	1899
2714	14452	1545	2014-04-01	brown bread	1900
2715	19003	1545	2014-04-01	hair spray	1900
2716	6422	1616	2014-04-01	prosecco	1901

Figure 31. second scenario dataset

After dataset is slicing, then we check the distribution of the itemDescription to determine minimum support.

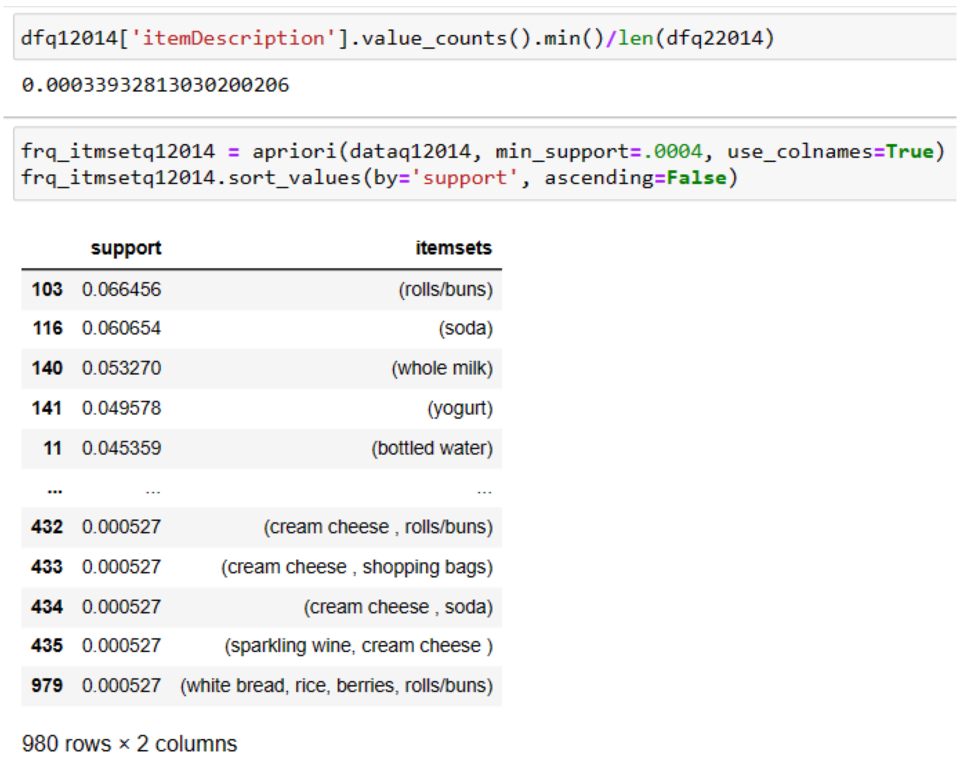


```
dfq12014['itemDescription'].value_counts().describe()
```

```
count    143.000000
mean      18.965035
std       24.958391
min        1.000000
25%        3.000000
50%       10.000000
75%       21.000000
max      128.000000
Name: count, dtype: float64
```

Figure 32. descriptive statistic for second scenario dataset

We will use a same methodology to determine the minimum support which is dividing the least occurrence with the total basket. In this case the least occurrence is one and the size of basket is 1896, so minimum support will be :



```
dfq12014['itemDescription'].value_counts().min()/len(dfq22014)
```

```
0.00033932813030200206
```

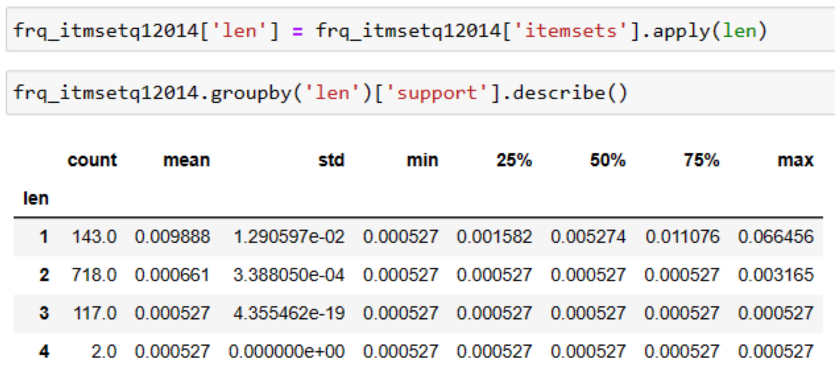
```
frq_itmsetq12014 = apriori(dataq12014, min_support=.0004, use_colnames=True)
frq_itmsetq12014.sort_values(by='support', ascending=False)
```

	support	itemsets
103	0.066456	(rolls/buns)
116	0.060654	(soda)
140	0.053270	(whole milk)
141	0.049578	(yogurt)
11	0.045359	(bottled water)
...	...	...
432	0.000527	(cream cheese , rolls/buns)
433	0.000527	(cream cheese , shopping bags)
434	0.000527	(cream cheese , soda)
435	0.000527	(sparkling wine, cream cheese)
979	0.000527	(white bread, rice, berries, rolls/buns)

980 rows × 2 columns

Figure 33. grouping second scenario dataset

Using that configuration resulting 980 itemset combination. After that, the descriptive statistic of support will be check



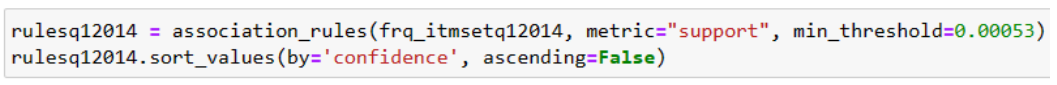
```
frq_itmsetq12014['len'] = frq_itmsetq12014['itemsets'].apply(len)
```

```
frq_itmsetq12014.groupby('len')['support'].describe()
```

len	count	mean	std	min	25%	50%	75%	max
1	143.0	0.009888	1.290597e-02	0.000527	0.001582	0.005274	0.011076	0.066456
2	718.0	0.000661	3.388050e-04	0.000527	0.000527	0.000527	0.000527	0.003165
3	117.0	0.000527	4.355462e-19	0.000527	0.000527	0.000527	0.000527	0.000527
4	2.0	0.000527	0.000000e+00	0.000527	0.000527	0.000527	0.000527	0.000527

Figure 34. statistical component of aprioiri from second scenario dataset

Based on the descriptive statistics, 50% of itemset 2 will be choosen for minimum threshold for generating rules, which is 0.000527.



```
rulesq12014 = association_rules(frq_itmsetq12014, metric="support", min_threshold=0.00053)
rulesq12014.sort_values(by='confidence', ascending=False)
```

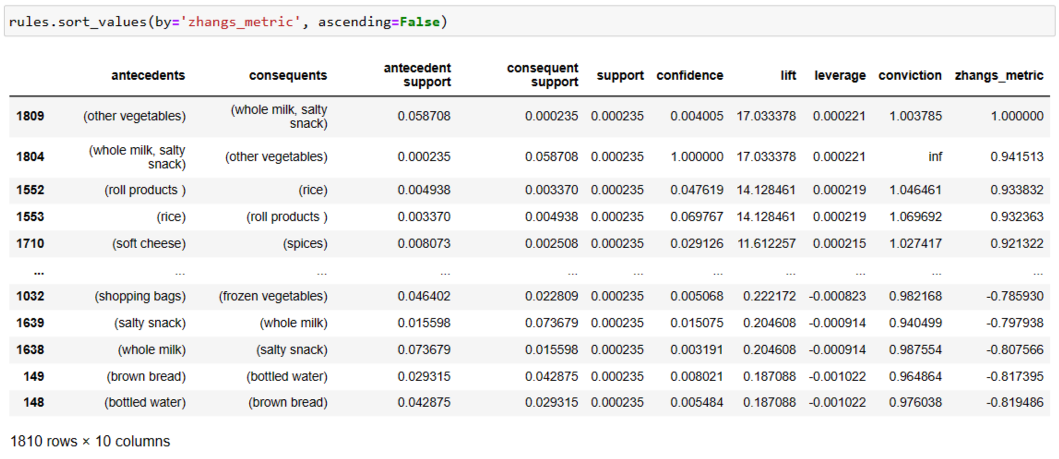
Figure 35. generating rule for second scenario dataset

SECTION 4: Result and Interpretation

For result interpretation, we will focus on Zhang's metric because it provides insights into the correlation between item sets in both directions, indicating positive or negative correlations. Based on Zhang's metric, we can enhance sales performance by applying it through item placement, cross-selling, or targeted promotions. Therefore, we need a meaningful relation between item sets. Thus, we will filter the data, taking only the upper and bottom 25% values. Because the range is from -1 to 1, the 25% data will be 0.5 and -0.5.

a. Scenario 1

The rule that generated by scenario 1 is shown below :



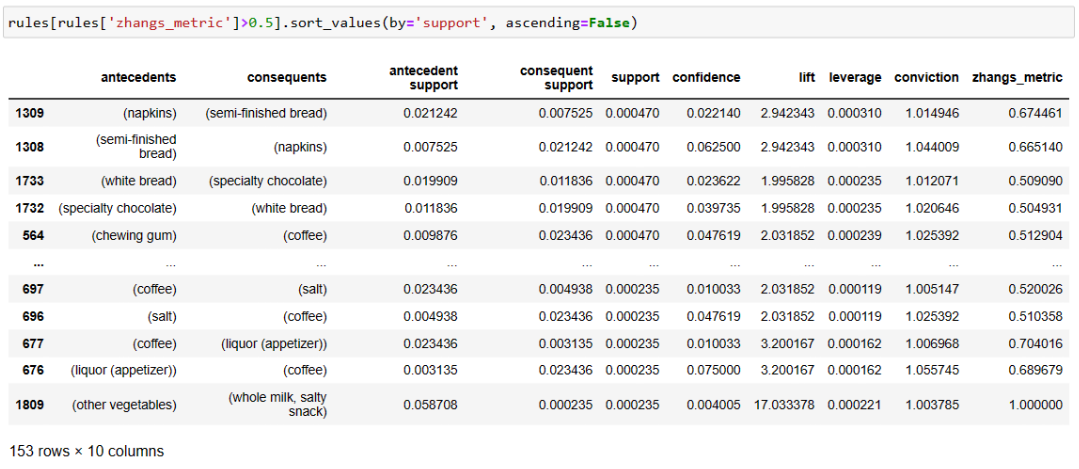
```
rules.sort_values(by='zhangs_metric', ascending=False)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
1809	(other vegetables)	(whole milk, salty snack)	0.058708	0.000235	0.000235	0.004005	17.033378	0.000221	1.003785	1.000000
1804	(whole milk, salty snack)	(other vegetables)	0.000235	0.058708	0.000235	1.000000	17.033378	0.000221	inf	0.941513
1552	(roll products)	(rice)	0.004938	0.003370	0.000235	0.047619	14.128461	0.000219	1.046461	0.933832
1553	(rice)	(roll products)	0.003370	0.004938	0.000235	0.069767	14.128461	0.000219	1.069692	0.932363
1710	(soft cheese)	(spices)	0.008073	0.002508	0.000235	0.029126	11.612257	0.000215	1.027417	0.921322
...	...	...	...	...	...	...	...	...	...	...
1032	(shopping bags)	(frozen vegetables)	0.046402	0.022809	0.000235	0.005068	0.222172	-0.000823	0.982168	-0.785930
1639	(salty snack)	(whole milk)	0.015598	0.073679	0.000235	0.015075	0.204608	-0.000914	0.940499	-0.797938
1638	(whole milk)	(salty snack)	0.073679	0.015598	0.000235	0.003191	0.204608	-0.000914	0.987554	-0.807566
149	(brown bread)	(bottled water)	0.029315	0.042875	0.000235	0.008021	0.187088	-0.001022	0.964864	-0.817395
148	(bottled water)	(brown bread)	0.042875	0.029315	0.000235	0.005484	0.187088	-0.001022	0.976038	-0.819486

1810 rows × 10 columns

Figure 36. rules result

The are 1810 combination. in maximizing the result, then the rules will be filtered.



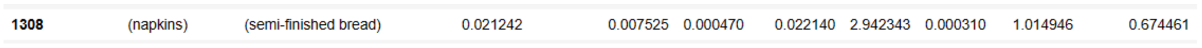
```
rules[rules['zhangs_metric']>0.5].sort_values(by='support', ascending=False)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
1309	(napkins)	(semi-finished bread)	0.021242	0.007525	0.000470	0.022140	2.942343	0.000310	1.014946	0.674461
1308	(semi-finished bread)	(napkins)	0.007525	0.021242	0.000470	0.062500	2.942343	0.000310	1.044009	0.665140
1733	(white bread)	(specialty chocolate)	0.019909	0.011836	0.000470	0.023622	1.995828	0.000235	1.012071	0.509090
1732	(specialty chocolate)	(white bread)	0.011836	0.019909	0.000470	0.039735	1.995828	0.000235	1.020646	0.504931
564	(chewing gum)	(coffee)	0.009876	0.023436	0.000470	0.047619	2.031852	0.000239	1.025392	0.512904
...	...	...	...	...	...	...	...	...	...	...
697	(coffee)	(salt)	0.023436	0.004938	0.000235	0.010033	2.031852	0.000119	1.005147	0.520026
696	(salt)	(coffee)	0.004938	0.023436	0.000235	0.047619	2.031852	0.000119	1.025392	0.510358
677	(coffee)	(liquor (appetizer))	0.023436	0.003135	0.000235	0.010033	3.200167	0.000162	1.006968	0.704016
676	(liquor (appetizer))	(coffee)	0.003135	0.023436	0.000235	0.075000	3.200167	0.000162	1.055745	0.689679
1809	(other vegetables)	(whole milk, salty snack)	0.058708	0.000235	0.000235	0.004005	17.033378	0.000221	1.003785	1.000000

153 rows × 10 columns

Figure 37. filtered rules

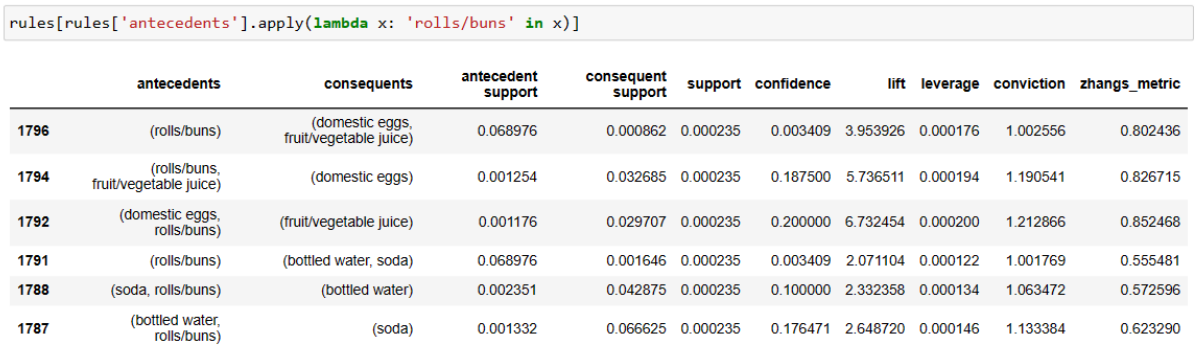
Filtered data resulting 153 rules that focusing on the rule that have high correlation



1308	(napkins)	(semi-finished bread)	0.021242	0.007525	0.000470	0.022140	2.942343	0.000310	1.014946	0.674461

Figure 38. napkins rule

For example, seller can place the napkins near the semi-finished bread since it has high correlation showing in its zhangs metric value.



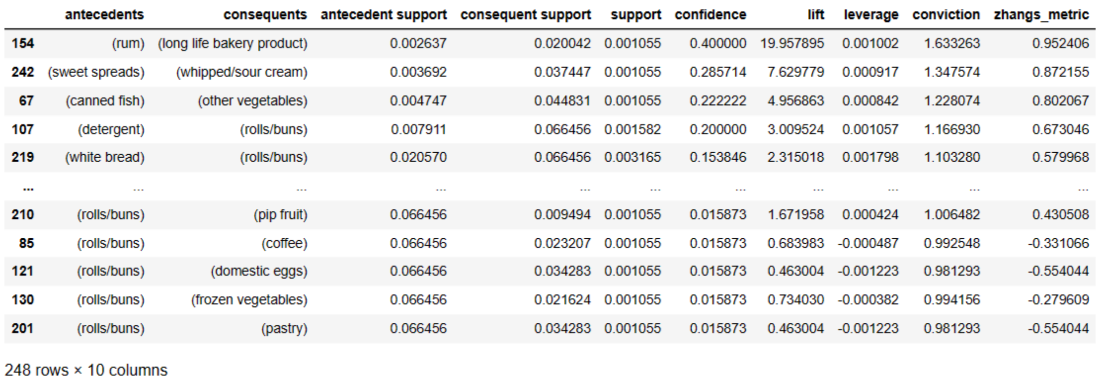
```
rules[rules['antecedents'].apply(lambda x: 'rolls/buns' in x)]
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
1796	(rolls/buns)	(domestic eggs, fruit/vegetable juice)	0.068976	0.000862	0.000235	0.003409	3.953926	0.000176	1.002556	0.802436
1794	(rolls/buns, fruit/vegetable juice)	(domestic eggs)	0.001254	0.032685	0.000235	0.187500	5.736511	0.000194	1.190541	0.826715
1792	(domestic eggs, rolls/buns)	(fruit/vegetable juice)	0.001176	0.029707	0.000235	0.200000	6.732454	0.000200	1.212866	0.852468
1791	(rolls/buns)	(bottled water, soda)	0.068976	0.001646	0.000235	0.003409	2.071104	0.000122	1.001769	0.555481
1788	(soda, rolls/buns)	(bottled water)	0.002351	0.042875	0.000235	0.100000	2.332358	0.000134	1.063472	0.572596
1787	(bottled water, rolls/buns)	(soda)	0.001332	0.066625	0.000235	0.176471	2.648720	0.000146	1.133384	0.623290

Figure 39. rules that contain rolls/buns

Since the result is in dataframe, it can be filtered based on seller interest for instance whole milk, then we can see the rules that contains with rolls/buns.

b. Scenario 2

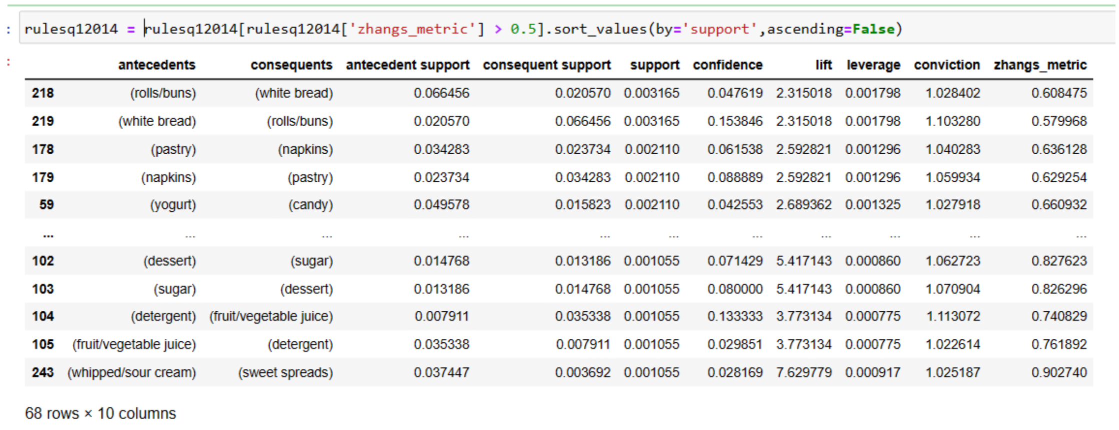


	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
154	(rum)	(long life bakery product)	0.002637	0.020042	0.001055	0.400000	19.957895	0.001002	1.633263	0.952406
242	(sweet spreads)	(whipped/sour cream)	0.003692	0.037447	0.001055	0.285714	7.629779	0.000917	1.347574	0.872155
67	(canned fish)	(other vegetables)	0.004747	0.044831	0.001055	0.222222	4.956863	0.000842	1.228074	0.802067
107	(detergent)	(rolls/buns)	0.007911	0.066456	0.001582	0.200000	3.009524	0.001057	1.166930	0.673046
219	(white bread)	(rolls/buns)	0.020570	0.066456	0.003165	0.153846	2.315018	0.001798	1.103280	0.579968
...	...	...	...	...	...	...	...	...	...	...
210	(rolls/buns)	(pip fruit)	0.066456	0.009494	0.001055	0.015873	1.671958	0.000424	1.006482	0.430508
85	(rolls/buns)	(coffee)	0.066456	0.023207	0.001055	0.015873	0.683983	-0.000487	0.992548	-0.331066
121	(rolls/buns)	(domestic eggs)	0.066456	0.034283	0.001055	0.015873	0.463004	-0.001223	0.981293	-0.554044
130	(rolls/buns)	(frozen vegetables)	0.066456	0.021624	0.001055	0.015873	0.734030	-0.000382	0.994156	-0.279609
201	(rolls/buns)	(pastry)	0.066456	0.034283	0.001055	0.015873	0.463004	-0.001223	0.981293	-0.554044

248 rows × 10 columns

Figure 40. rules for second scenario dataset

The are 248 combination. in maximizing the result, then the rules will be filtered.



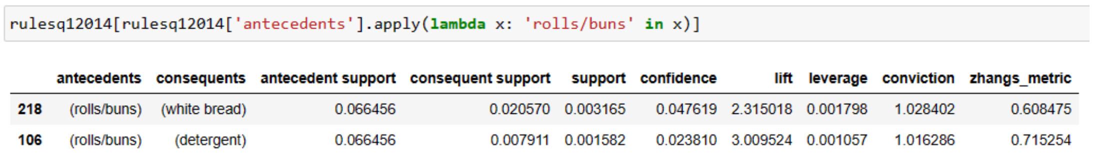
```
rulesq12014 = rulesq12014[rulesq12014['zhangs_metric'] > 0.5].sort_values(by='support',ascending=False)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
218	(rolls/buns)	(white bread)	0.066456	0.020570	0.003165	0.047619	2.315018	0.001798	1.028402	0.608475
219	(white bread)	(rolls/buns)	0.020570	0.066456	0.003165	0.153846	2.315018	0.001798	1.103280	0.579968
178	(pastry)	(napkins)	0.034283	0.023734	0.002110	0.061538	2.592821	0.001296	1.040283	0.636128
179	(napkins)	(pastry)	0.023734	0.034283	0.002110	0.088889	2.592821	0.001296	1.059934	0.629254
59	(yogurt)	(candy)	0.049578	0.015823	0.002110	0.042553	2.689362	0.001325	1.027918	0.660932
...	...	...	...	...	...	...	...	...	...	...
102	(dessert)	(sugar)	0.014768	0.013186	0.001055	0.071429	5.417143	0.000860	1.062723	0.827623
103	(sugar)	(dessert)	0.013186	0.014768	0.001055	0.080000	5.417143	0.000860	1.070904	0.826296
104	(detergent)	(fruit/vegetable juice)	0.007911	0.035338	0.001055	0.133333	3.773134	0.000775	1.113072	0.740829
105	(fruit/vegetable juice)	(detergent)	0.035338	0.007911	0.001055	0.029851	3.773134	0.000775	1.022614	0.761892
243	(whipped/sour cream)	(sweet spreads)	0.037447	0.003692	0.001055	0.028169	7.629779	0.000917	1.025187	0.902740

68 rows × 10 columns

Figure 41. filtered rules for second scenario dataset

Filtered data resulting 68 rules that focusing on the rule that have high correlation



```
rulesq12014[rulesq12014['antecedents'].apply(lambda x: 'rolls/buns' in x)]
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
218	(rolls/buns)	(white bread)	0.066456	0.020570	0.003165	0.047619	2.315018	0.001798	1.028402	0.608475
106	(rolls/buns)	(detergent)	0.066456	0.007911	0.001582	0.023810	3.009524	0.001057	1.016286	0.715254

Figure 42. rules that contains rolls/buns in second scenario dataset

Since the result is in dataframe, it can be filtered based on seller interest for instance whole milk, then we can see the rules that contains with rolls/buns.

Discussion

In interpreting the results, we will focus on the Zhang's metric. The highest Zhang's metric indicates a high positive correlation within rules, while a negative Zhang's metric indicates a high negative correlation.

In terms of product placement, products need to be grouped into certain categories first, such as bread products, drink products, and hygiene products, because sellers can't just place rolls/buns beside detergent. However, if those two items have a higher Zhang's metric, we can take advantage of it by implementing cross-selling strategies, such as offering rolls/buns at a discounted price to customers who buy detergent. Additionally, targeted marketing can be employed by promoting new rolls/buns to customers who buy detergent.

When comparing the results of the first and second scenarios, we can observe from Figures 39 and 42, which depict the rules for rolls/buns, that the rules differ. This suggests that customer preferences may vary at different times. Scenario two also displays data with the highest support, aiding in choosing the minimum support. Other evaluation metrics may show good indicators, but when the support is small, it implies that the case occurred only a few times. Therefore, reducing the dataset can lead to more reliable results. In this work, the writer suggests that association rules be generated regularly, either monthly, quarterly, or based on events, as customer behavior is always changing.