Market Basket Analysis Using Apriori Algorithm

In [1]:
```python
import matplotlib.pyplot as plt; plt.rcdefaults()
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```python
df = pd.read_csv('Workshop-5-dataset.zip', sep='\t',dtype=np.str)
```

Printing the first five rows of the dataset

In [3]:
```python
df.head(5)
```

Out[3]:

	transaction_ID	Date	Time	item_0	item_1	item_2	item_3	item_4	
0	536365	01/12/2010	08:26	WHITE HANGING HEART T-LIGHT HOLDER	WHITE METAL LANTERN	CREAM CUPID HEARTS COAT HANGER	KNITTED UNION FLAG HOT WATER BOTTLE	RED WOOLLY HOTTIE WHITE HEART	BA N
1	536366	01/12/2010	08:28	HAND WARMER UNION JACK	HAND WARMER RED POLKA DOT	NaN	NaN	NaN	
2	536367	01/12/2010	08:34	ASSORTED COLOUR BIRD ORNAMENT	POPPY'S PLAYHOUSE BEDROOM	POPPY'S PLAYHOUSE KITCHEN	FELTCRAFT PRINCESS CHARLOTTE DOLL	IVORY KNITTED MUG COSY	B AS (TEAS
3	536368	01/12/2010	08:34	JAM MAKING SET WITH JARS	RED COAT RACK PARIS FASHION	YELLOW COAT RACK PARIS FASHION	BLUE COAT RACK PARIS FASHION	NaN	
4	536369	01/12/2010	08:35	BATH BUILDING BLOCK WORD	NaN	NaN	NaN	NaN	

5 rows × 44 columns

In [4]:
```python
df.head()
```

Out[4]:

	transaction_ID	Date	Time	item_0	item_1	item_2	item_3	item_4	
0	536365	01/12/2010	08:26	WHITE HANGING HEART T-LIGHT HOLDER	WHITE METAL LANTERN	CREAM CUPID HEARTS COAT HANGER	KNITTED UNION FLAG HOT WATER BOTTLE	RED WOOLLY HOTTIE WHITE HEART	BA N
1	536366	01/12/2010	08:28	HAND WARMER UNION JACK	HAND WARMER RED POLKA DOT	NaN	NaN	NaN	
2	536367	01/12/2010	08:34	ASSORTED COLOUR BIRD ORNAMENT	POPPY'S PLAYHOUSE BEDROOM	POPPY'S PLAYHOUSE KITCHEN	FELTCRAFT PRINCESS CHARLOTTE DOLL	IVORY KNITTED MUG COSY	B AS (TEAS
3	536368	01/12/2010	08:34	JAM MAKING SET WITH JARS	RED COAT RACK PARIS FASHION	YELLOW COAT RACK PARIS FASHION	BLUE COAT RACK PARIS FASHION	NaN	
4	536369	01/12/2010	08:35	BATH BUILDING BLOCK WORD	NaN	NaN	NaN	NaN	

5 rows × 44 columns

Printing number of rows and columns in the dataset?

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

Out[5]:
```
(31941, 44)
```

There are 31941 rows and 44 columns in the dataset

Generating my unique Dataset

For this Notebook task, i generated my own version of the dataset.

In [6]:
```
STUDENT_NAME = 'AdaobiEjiasi'
STUDENT_NO = '6317'
```

In [7]:
```
np.random.seed(int(STUDENT_NO))
unique_id = int('2' + STUDENT_NO)
rows = np.random.choice(df.index.values, unique_id)
data = df.loc[rows]
```

In [8]:
```
file_name = STUDENT_NAME + "_" + STUDENT_NO + ".csv"
data.to_csv(file_name)
```

Basic Data Analysis

Let's get familiar with the dataset. In order to gain some first impressions, now we will try getting some counts for different columns.

Getting the number of unique dates in the dataset.

In [9]:
```
data.nunique()
```

Out[9]:
```
transaction_ID    15452
Date                305
Time                743
item_0             2877
item_1             2647
item_2             2633
item_3             2543
item_4             2550
item_5             2515
item_6             2470
item_7             2449
item_8             2408
item_9             2356
item_10            2369
item_11            2317
item_12            2321
item_13            2279
item_14            2262
item_15            2222
item_16            2181
item_17            2189
item_18            2156
item_19            2082
item_20            2089
item_21            2051
item_22            2031
item_23            1997
item_24            2022
item_25            1986
item_26            1915
item_27            1924
item_28            1884
item_29            1857
item_30            1813
item_31            1807
item_32            1752
item_33            1722
item_34            1740
item_35            1752
item_36            1714
item_37            1678
item_38            1702
item_39            1683
item_40            1666
dtype: int64
```

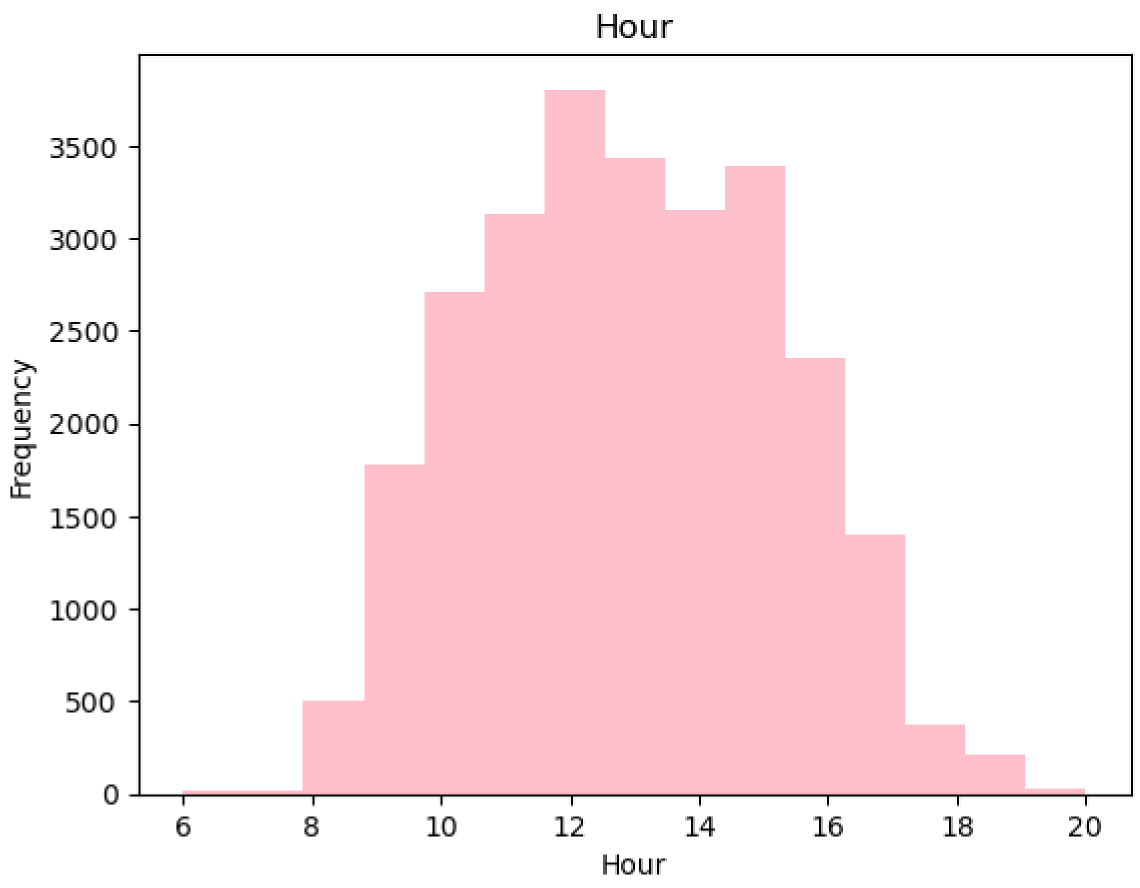
So many unique values to look at, it would be a good to visualise the data for better understanding, find summaries/patterns. We will first visualise the the

hours where the most and least transactions happened. To do that, we need to extract the value for hour from 'Time' column. We are using to_datetime() function available in Pandas for this purpose. The following code demonstrates this.

In [10]:
```
data['Hour'] = pd.to_datetime(data['Time'], format='%H:%M').dt.hour
```

Let's draw a histogram based on hour.

In [11]:
```
hour_hist = data.hist(column="Hour", bins=15, grid=False, color='pink')
for ax in hour_hist.flatten():
 ax.set_xlabel("Hour")
 ax.set_ylabel("Frequency")
```



We can see that the transactions happen between 06:00 and 20:00, and the shop gets busy around noon. We can also conclude that shop opens at 06:00 and closes at 20:00.

In [12]:
```
df.isnull ().sum ()
```

Out[12]:
```
transaction_ID        0
Date                  0
Time                  0
item_0             1454
item_1             6121
item_2             7864
item_3             9103
item_4            10050
item_5            10955
item_6            11732
item_7            12519
item_8            13258
item_9            14004
item_10           14662
item_11           15351
item_12           15953
item_13           16555
item_14           17162
item_15           17780
item_16           18439
item_17           18979
item_18           19484
item_19           20035
item_20           20546
item_21           21005
item_22           21404
item_23           21832
item_24           22222
item_25           22525
item_26           22830
item_27           23120
item_28           23398
item_29           23720
item_30           23982
item_31           24222
item_32           24462
item_33           24666
item_34           24873
item_35           25072
item_36           25233
item_37           25407
item_38           25560
item_39           25721
item_40           25881
dtype: int64
```

Apriori Algorithm

Now we have a good idea about the dataset. Let's use the Apriori algorithm to mine association rules. For this part we will be using the 'apyori' package that implements the Apriori algorithm. Before we continue, we will install the 'apyori' package on your machine ('pip3 install -U apyori').

In [13]:
```
pip install apyori
```

```
Requirement already satisfied: apyori in c:\users\lenovox260\anaconda3\lib\site-packa
ges (1.1.2)
Note: you may need to restart the kernel to use updated packages.
```

In [14]:
```
# import apyori
from apyori import apriori
```

Data Preprocessing

As we are now familiar with the dataset, we should have noticed that not all columns are of interest to us. Most relevant for this analysis are the invoice number and description columns. The invoice number is essentially an ID for each "basket", and the description column contains the items. Therefore, using these two columns we can infer the items in each basket. In order to use the information in these two columns, we have to prepare them for the algorithm. The package we are using requires the data to be in the form of nested lists, meaning that we need a list of baskets. The following steps carries this out. We first extract the columns where the items are there. i.e., columns 4 to 45.

In [15]:
```
data.head(1)
```

Out[15]:

	transaction_ID	Date	Time	item_0	item_1	item_2	item_3	item_4	item_5
15381	558628	30/06/2011	17:59	FIVE CATS HANGING DECORATION	CHILDRENS CUTLERY SPACEBOY	POSTAGE	NaN	NaN	NaN

1 rows × 45 columns

In [16]:
```
items_df=data[data.columns[3:44]]
```

In [17]:
```
items_df.head()
```

Out[17]:

	item_0	item_1	item_2	item_3	item_4	item_5	item_6	item_7
15381	FIVE CATS HANGING DECORATION	CHILDRENS CUTLERY SPACEBOY	POSTAGE	NaN	NaN	NaN	NaN	NaN
16684	Manual	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3205	REGENCY CAKESTAND 3 TIER	LARGE CAKE STAND HANGING STRAWBERY	RED RETROSPOT ROUND CAKE TINS	CERAMIC CAKE DESIGN SPOTTED MUG	CERAMIC STRAWBERRY DESIGN MUG	PARTY BUNTING	SET OF 3 CAKE TINS PANTRY DESIGN	LARGE HEART MEASURING SPOONS
17794	JAM MAKING SET WITH JARS	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3151	POSTAGE	T-LIGHT HOLDER HANGING LACE	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 41 columns

Then we assign each transaction to a list and finally we will have a list of lists for all the transactions. To do that, we first take the Transpose of the dataframe, using .T and then drop the 'nan' values using the dropna() function.

In [18]:
```
baskets = items_df.T.apply(lambda x: x.dropna().tolist()).tolist()
```

In [19]:
```
for i in baskets[:5]:
    print(i)
```

```
['FIVE CATS HANGING DECORATION', 'CHILDRENS CUTLERY SPACEBOY', 'POSTAGE']
['Manual']
['REGENCY CAKESTAND 3 TIER', 'LARGE CAKE STAND  HANGING STRAWBERY', 'RED RETROSPOT RO
UND CAKE TINS', 'CERAMIC CAKE DESIGN SPOTTED MUG', 'CERAMIC STRAWBERRY DESIGN MUG',
'PARTY BUNTING', 'SET OF 3 CAKE TINS PANTRY DESIGN', 'LARGE HEART MEASURING SPOONS',
'SMALL HEART MEASURING SPOONS', 'FOOD CONTAINER SET 3 LOVE HEART', 'PICNIC BASKET WIC
KER LARGE']
['JAM MAKING SET WITH JARS']
['POSTAGE', 'T-LIGHT HOLDER HANGING LACE']
```

As you can see, the values of the first five baskets are similar to the first five rows of items_df as we saw earlier.

Algorithm Parameters

Now that the data has been put into an acceptable format, we are ready to run the algorithm. In order to do so, we run the 'apriori()' command. However, this command takes 5 parameters that we need to supply. The parameters are described below.

Remember, the calculations are only given as information, we will not have to calculate any parameters for this task!

Dataset

This is the dataset we just compiled in the previous cell. It has to be a list of lists, so in our case a list of lists containing shopping items.

(Minimum) Support

The support value refers to the popularity of an item, i.e., how often an item was bought. With this value, we specify the minimum 'popularity' of items that should be included. It is usually calculated like this:

- Support(Item A) = (Baskets containing Item A)/(Total No. of Baskets)

Due to the limitations of the dataset we are working with, we will be starting with 0.01, and working with values between 0.02 and 0.001.

(Minimum) Confidence

The confidence value denotes the likelihood of a rule, i.e., how likely it is that an Item B will follow an Item A. With the minimum confidence, we specify the minimum likelihood that the rules need to satisfy. It is usually calculated like below:

- Confidence(Item A → Item B) = (Baskets containing both (Item A and Item B))/(Baskets containing Item A)

We will be starting with a minimum confidence of 20% (0.2), meaning that for each rule the minimum likelihood needs to be this value or higher.

(Minimum) Lift

Lift is a measure of the performance of a rule. The lift value is the ratio of the increase of an Item B being bought when Item A is bought. Put more simply, it is the confidence divided by the support. Essentially, if we have a lift of 1 there is no association between two items, a higher lift means an increasing association.

- Lift(Item B → Item A) = (Confidence (Item A → Item B))/(Support (Item A))

We will be starting with a minimum lift of 3.

(Minimum) Length of Rules

This specifies how many items should be in each rule at least. If we set it at 2, then the rules will look like this: ITEM A → ITEM B. If it is set at 4, then we could have a rule like this: ITEM A, ITEM B, ITEM C → ITEM D.

We will be starting with a length of 2.

Now that we know the parameters we can run the command. In the cell below, we run the apriori command with the following parameters (which will serve as our baseline parameters):

- Minimum Support = 0.01
- Minimum Confidence = 0.2
- Minimum Lift = 3
- Minimum Length = 2

In [20]:
```
association_rules = apriori(baskets, min_support=0.01, min_confidence=0.2,
 min_lift=3, min_length=2)
association_results = list(association_rules)
```

After running the apriori algorithm, we may have some rules generated. Now, let's try to answer some association_rules = apriori(baskets, min_support=0.01, min_confidence=0.2, min_lift=3, min_length=2) association_results = list(association_rules) questions about rules.

Finding out how many association rules we were able to generate

In [21]:
```
print('Rules generated: ', len(association_results))
```

```
Rules generated:  90
```

We can also have a look at what an association rule looks like for the algorithm, by printing the first rule from the list.

In [22]:
```
print(association_results[0])
```

```
RelationRecord(items=frozenset({'60 TEATIME FAIRY CAKE CASES', 'PACK OF 72 RETROSPOT
CAKE CASES'}), support=0.010563514078352395, ordered_statistics=[OrderedStatistic(ite
ms_base=frozenset({'60 TEATIME FAIRY CAKE CASES'}), items_add=frozenset({'PACK OF 72
RETROSPOT CAKE CASES'}), confidence=0.3921015514809591, lift=9.172388026955023), Orde
redStatistic(items_base=frozenset({'PACK OF 72 RETROSPOT CAKE CASES'}), items_add=fro
zenset({'60 TEATIME FAIRY CAKE CASES'}), confidence=0.2471111111111111, lift=9.172388
026955021)])
```

Changing the index value [0] of 'association_results[0]' to see a a different rule.

In [23]:
```
print(association_results[75])
```

```
RelationRecord(items=frozenset({'RECIPE BOX PANTRY YELLOW DESIGN', 'SET OF 3 CAKE TIN
S PANTRY DESIGN'}), support=0.011285480867880077, ordered_statistics=[OrderedStatisti
c(items_base=frozenset({'RECIPE BOX PANTRY YELLOW DESIGN'}), items_add=frozenset({'SE
T OF 3 CAKE TINS PANTRY DESIGN'}), confidence=0.3006072874493927, lift=6.463302274351
036), OrderedStatistic(items_base=frozenset({'SET OF 3 CAKE TINS PANTRY DESIGN'}), it
ems_add=frozenset({'RECIPE BOX PANTRY YELLOW DESIGN'}), confidence=0.242647058823529
4, lift=6.463302274351036)])
```

In [24]:
```
print(association_results[56])
```

```
RelationRecord(items=frozenset({'LUNCH BAG SUKI DESIGN', 'LUNCH BAG CARS BLUE'}), sup
port=0.013375384732302315, ordered_statistics=[OrderedStatistic(items_base=frozenset
({'LUNCH BAG CARS BLUE'}), items_add=frozenset({'LUNCH BAG SUKI DESIGN'}), confidence
=0.35881753312945974, lift=9.73505259728659), OrderedStatistic(items_base=frozenset
({'LUNCH BAG SUKI DESIGN'}), items_add=frozenset({'LUNCH BAG CARS BLUE'}), confidence
=0.3628865979381444, lift=9.73505259728659)])
```

In [25]:

```python
print(association_results[48])
```

```
RelationRecord(items=frozenset({'LUNCH BAG  BLACK SKULL', 'RED RETROSPOT CHARLOTTE BA
G'}), support=0.010373522817950374, ordered_statistics=[OrderedStatistic(items_base=f
rozenset({'LUNCH BAG  BLACK SKULL'}), items_add=frozenset({'RED RETROSPOT CHARLOTTE B
AG'}), confidence=0.2353448275862069, lift=6.889399140807793), OrderedStatistic(items
_base=frozenset({'RED RETROSPOT CHARLOTTE BAG'}), items_add=frozenset({'LUNCH BAG  BL
ACK SKULL'}), confidence=0.30367074527252497, lift=6.889399140807793)])
```

In [26]:

```python
print(association_results[33])
```

```
RelationRecord(items=frozenset({'JUMBO STORAGE BAG SUKI', 'JUMBO BAG PINK VINTAGE PAI
SLEY'}), support=0.010221529809628756, ordered_statistics=[OrderedStatistic(items_bas
e=frozenset({'JUMBO BAG PINK VINTAGE PAISLEY'}), items_add=frozenset({'JUMBO STORAGE
BAG SUKI'}), confidence=0.4020926756352765, lift=10.7212491840867), OrderedStatistic
(items_base=frozenset({'JUMBO STORAGE BAG SUKI'}), items_add=frozenset({'JUMBO BAG PI
NK VINTAGE PAISLEY'}), confidence=0.2725430597771023, lift=10.721249184086698)])
```

Analysing the Results

Analysing the results may include checking what kind of items are featured in some of the rules. Using the method in the cell below, you can display the results

In [27]:

```python
def display_rules(association_results):
 for item in association_results:
    pair = item[0]
    items = [x for x in pair]
    print("Rule: " + items[0] + " -> " + items[1])
    print("Support: " + str(item[1]))
    print("Confidence: " + str(item[2][0][2]))
    print("Lift: " + str(item[2][0][3]))
    print("=====================================")
```

Using the method above, we can display the rules for the algorithm. Let's display the rules only for 10 results.

In [28]:

```python
display_rules(association_results[:5])
```

```
Rule: 60 TEATIME FAIRY CAKE CASES -> PACK OF 72 RETROSPOT CAKE CASES
Support: 0.010563514078352395
Confidence: 0.3921015514809591
Lift: 9.172388026955023
=====================================
Rule: ALARM CLOCK BAKELIKE GREEN -> ALARM CLOCK BAKELIKE IVORY
Support: 0.010753505338754417
Confidence: 0.34596577017114916
Lift: 18.100956607543008
=====================================
Rule: ALARM CLOCK BAKELIKE GREEN -> ALARM CLOCK BAKELIKE PINK
Support: 0.012159440665729377
Confidence: 0.39119804400978
Lift: 16.44594077349102
=====================================
Rule: ALARM CLOCK BAKELIKE GREEN -> ALARM CLOCK BAKELIKE RED
Support: 0.019417106813086598
Confidence: 0.6246943765281173
Lift: 18.767216788916055
=====================================
Rule: ALARM CLOCK BAKELIKE IVORY -> ALARM CLOCK BAKELIKE RED
Support: 0.011551468632442908
Confidence: 0.6043737574552684
Lift: 18.15673992574235
=====================================
```

Although we have association rules generated, it will be interesting to display frequent items too. Below, you will see a count of the 10 most frequent items in the dataset.

In [29]:
```python
from collections import Counter
counter = Counter(baskets[0])
for i in baskets[1:]:
    if i != 'nan':
        counter.update(i)
del counter['nan']
counter.most_common(10)
```

Out[29]:
```
[('WHITE HANGING HEART T-LIGHT HOLDER', 1978),
 ('REGENCY CAKESTAND 3 TIER', 1798),
 ('JUMBO BAG RED RETROSPOT', 1748),
 ('PARTY BUNTING', 1420),
 ('LUNCH BAG RED RETROSPOT', 1381),
 ('ASSORTED COLOUR BIRD ORNAMENT', 1253),
 ('SET OF 3 CAKE TINS PANTRY DESIGN', 1230),
 ('LUNCH BAG  BLACK SKULL', 1202),
 ('PACK OF 72 RETROSPOT CAKE CASES', 1158),
 ('NATURAL SLATE HEART CHALKBOARD', 1115)]
```

Discovering how many of these items can be found in the rules we have just displayed.

There are 10 items in the rules displayed above

Are all the top 10 items included? Providing an explanation as to why these items may be missing/present in the rules

Yes, all the top 10 items are included in the rules because the code used in the above cell is to display the top 10 most common items from the collection of items. If we wish to display the top 20 most common items we will simply set the rules to display the most_common(20) items on the list.

Example seen in the next column below;

In [30]:
```python
from collections import Counter
counter = Counter(baskets[0])
for i in baskets[1:]:
    if i != 'nan':
        counter.update(i)
del counter['nan']
counter.most_common(20)
```

Out[30]:
```
[('WHITE HANGING HEART T-LIGHT HOLDER', 1978),
 ('REGENCY CAKESTAND 3 TIER', 1798),
 ('JUMBO BAG RED RETROSPOT', 1748),
 ('PARTY BUNTING', 1420),
 ('LUNCH BAG RED RETROSPOT', 1381),
 ('ASSORTED COLOUR BIRD ORNAMENT', 1253),
 ('SET OF 3 CAKE TINS PANTRY DESIGN', 1230),
 ('LUNCH BAG  BLACK SKULL', 1202),
 ('PACK OF 72 RETROSPOT CAKE CASES', 1158),
 ('NATURAL SLATE HEART CHALKBOARD', 1115),
 ('HEART OF WICKER SMALL', 1026),
 ('JUMBO BAG PINK POLKADOT', 1025),
 ('POSTAGE', 1012),
 ('LUNCH BAG CARS BLUE', 1008),
 ('RECIPE BOX PANTRY YELLOW DESIGN', 1002),
 ('JAM MAKING SET WITH JARS', 1001),
 ('LUNCH BAG SUKI DESIGN', 995),
 ('JUMBO STORAGE BAG SUKI', 992),
 ('LUNCH BAG SPACEBOY DESIGN', 990),
 ("PAPER CHAIN KIT 50'S CHRISTMAS", 988)]
```

Report

Runing the apriori algorithm with the following three different settings:

- Setting 1: Min Support = 0.015, Min Confidence = 0.7, Min Lift = 3
- Setting 2: Min Support = 0.009, Min Confidence = 0.5, Min Lift = 3
- Setting 3: Min Support = 0.015, Min Confidence = 0.5, Min Lift = 9

And, Calculating the number of rules we can get for each setting and how the quality of the rules differ in each setting.

Setting 1:

- Min Support = 0.015,
- Min Confidence = 0.7,
- Min Lift = 3

In the cell below, we run the apriori command with the following parameters in Setting 1 which will serve as our baseline parameters

Applying the Apriori Algorithm

In [31]:
```
association_rules = apriori(baskets, min_support=0.015, min_confidence=0.7,
 min_lift=3, min_length=2)
association_results = list(association_rules)
```

Now lets calculate number of rules generated

In [32]:
```
print('Rules generated: ', len(association_results))
```

```
Rules generated:  4
```

Analysing the Results

Here we will check what type of items are featured in some of the rules using the code in the cell below to display the results.

In [33]:
```
def display_rules(association_results):
    for item in association_results:
        pair = item[0]
        items = [x for x in pair]
        print("Rule: " + items[0] + " -> " + items[1])
        print("Support: " + str(item[1]))
        print("Confidence: " + str(item[2][0][2]))
        print("Lift: " + str(item[2][0][3]))
        print("=====================================")
```

The code displays the rules for the algorithm. We will display the rules for all 4 and discuss all below.

In [34]:
```
display_rules(association_results[:4])
```

```
Rule: GREEN REGENCY TEACUP AND SAUCER -> PINK REGENCY TEACUP AND SAUCER
Support: 0.018847133031880535
Confidence: 0.7811023622047245
Lift: 24.070574784709294
=====================================
Rule: ROSES REGENCY TEACUP AND SAUCER -> GREEN REGENCY TEACUP AND SAUCER
Support: 0.02314093551696622
Confidence: 0.7131147540983608
Lift: 20.760001088060353
=====================================
Rule: ROSES REGENCY TEACUP AND SAUCER -> PINK REGENCY TEACUP AND SAUCER
Support: 0.017517194209066382
Confidence: 0.7259842519685039
Lift: 21.13465437948575
=====================================
Rule: ROSES REGENCY TEACUP AND SAUCER -> GREEN REGENCY TEACUP AND SAUCER
Support: 0.015617281605046168
Confidence: 0.8286290322580645
Lift: 24.122821064087923
=====================================
```

Furher Analysis of the Result in Setting 1

Setting 1:

- Min Support = 0.015,
- Min Confidence = 0.7,
- Min Lift = 3

Setting 1 above acts as a threshold values and parameters in analyzing the outcome of the association rules.

In Setting 1 we generated 4 rules and we will be discussing the outcome of these rules below.

Rule 1 -

- Rule: PINK REGENCY TEACUP AND SAUCER -> GREEN REGENCY TEACUP AND SAUCER
- Support: 0.018847133031880535
- Confidence: 0.7811023622047245
- Lift: 24.070574784709294

This rule shows that people who buy 'GREEN REGENCY TEACUP AND SAUCER' also buy 'PINK REGENCY TEACUP AND SAUCER'

With the below Algorithm Parameters;

- support=0.018847133031880535
- confidence=0.7811023622047245
- lift=24.070574784709294

The values in Rule 1 is above the threshold algorithm values in Setting 1 for Min Support, Min Confidence and Min Lift

Rule 2:

Rule: ROSES REGENCY TEACUP AND SAUCER -> GREEN REGENCY TEACUP AND SAUCER Support: 0.02314093551696622 Confidence: 0.7131147540983608 Lift: 20.760001088060353

The below shows that people who buy 'ROSES REGENCY TEACUP AND SAUCER' are more likely to buy 'GREEN REGENCY TEACUP AND SAUCER'. With the below Algorithm Parameters.

- support=0.02314093551696622
- confidence=0.7131147540983608
- lift=20.760001088060353

The values in Rule 2 are above the threshold algorithm values in Setting 1 for Min Support, Min Confidence and Min Lift

Rule 3:

Rule: PINK REGENCY TEACUP AND SAUCER -> ROSES REGENCY TEACUP AND SAUCER Support: 0.017517194209066382 Confidence: 0.7259842519685039 Lift: 21.13465437948575

This rule shows that people who buy PINK REGENCY TEACUP AND SAUCER also buy ROSES REGENCY TEACUP AND SAUCER With the below parameters

- Support: 0.017517194209066382
- Confidence: 0.7259842519685039
- Lift: 21.13465437948575

The values in Rule 3 are above the threshold algorithm values in Setting 1 for Min Support, Min Confidence and Min Lift.

Rule 4:

Rule: ROSES REGENCY TEACUP AND SAUCER -> PINK REGENCY TEACUP AND SAUCER Support: 0.015617281605046168 Confidence: 0.8286290322580645 Lift: 24.122821064087923

Association rule 4 shows that people who buy 'ROSES REGENCY TEACUP AND SAUCER' are mostly like to buy 'PINK REGENCY TEACUP AND SAUCER' with;

- Support: 0.015617281605046168
- Confidence: 0.8286290322580645
- Lift: 24.122821064087923

The values in Rule 4 are above the threshold algorithm values in Setting 1 for Min Support, Min Confidence and Min Lift

The support, confidence and lift in Rule 1, 2 and 3 are above the algorithm setting and very strong and appropriate for analysis and business decisions.

Setting 2:

- Min Support = 0.009,
- Min Confidence = 0.5,
- Min Lift = 3

In the cell below, we run the apriori command with the following parameters in Setting 2. This will serve as our baseline parameters in analyzing the results

Applying the Apriori Algorithm

In [35]:
```
association_rules = apriori(baskets, min_support=0.009, min_confidence=0.5,
 min_lift=3, min_length=2)
association_results = list(association_rules)
```

Now lets calculate the number of rules generated

In [36]:
```
print('Rules generated: ', len(association_results))
```

```
Rules generated:  53
```

Here I would be printing 3 association rule 1,10 & 20, to see what the association rule looks like for algorithm

In [37]:
```
print(association_results[1])
```

```
RelationRecord(items=frozenset({'ALARM CLOCK BAKELIKE GREEN', 'ALARM CLOCK BAKELIKE P
INK'}), support=0.012159440665729377, ordered_statistics=[OrderedStatistic(items_base
=frozenset({'ALARM CLOCK BAKELIKE PINK'}), items_add=frozenset({'ALARM CLOCK BAKELIKE
GREEN'}), confidence=0.5111821086261981, lift=16.44594077349102)])
```

In [38]:
```
print(association_results[10])
```

```
RelationRecord(items=frozenset({'CHRISTMAS CRAFT LITTLE FRIENDS', 'CHRISTMAS CRAFT TR
EE TOP ANGEL'}), support=0.009233575255538246, ordered_statistics=[OrderedStatistic(i
tems_base=frozenset({'CHRISTMAS CRAFT LITTLE FRIENDS'}), items_add=frozenset({'CHRIST
MAS CRAFT TREE TOP ANGEL'}), confidence=0.5214592274678111, lift=33.80108987505021),
OrderedStatistic(items_base=frozenset({'CHRISTMAS CRAFT TREE TOP ANGEL'}), items_add=
frozenset({'CHRISTMAS CRAFT LITTLE FRIENDS'}), confidence=0.5985221674876847, lift=3
3.80108987505021)])
```

In [39]:
```
print(association_results[20])
```

```
RelationRecord(items=frozenset({'JUMBO BAG PEARS', 'JUMBO BAG APPLES'}), support=0.01
307139871565908, ordered_statistics=[OrderedStatistic(items_base=frozenset({'JUMBO BA
G PEARS'}), items_add=frozenset({'JUMBO BAG APPLES'}), confidence=0.6838966202783301,
lift=20.879474890794448)])
```

Analysing the Results

This involves checking the kind of items are featured in some rules using the code below to display results.

In [40]:
```
def display_rules(association_results):
    for item in association_results:
        pair = item[0]
        items = [x for x in pair]
        print("Rule: " + items[0] + " -> " + items[1])
        print("Support: " + str(item[1]))
        print("Confidence: " + str(item[2][0][2]))
        print("Lift: " + str(item[2][0][3]))
        print("=====================================")
```

The code below displays the rules for the algorithm. Here we will display the rules for only 5 results and analyze each outcome

In [41]:
```
display_rules(association_results[:5])
```

```
Rule: ALARM CLOCK BAKELIKE GREEN -> ALARM CLOCK BAKELIKE IVORY
Support: 0.010753505338754417
Confidence: 0.562624254473161
Lift: 18.100956607543004
=====================================
Rule: ALARM CLOCK BAKELIKE GREEN -> ALARM CLOCK BAKELIKE PINK
Support: 0.012159440665729377
Confidence: 0.5111821086261981
Lift: 16.44594077349102
=====================================
Rule: ALARM CLOCK BAKELIKE GREEN -> ALARM CLOCK BAKELIKE RED
Support: 0.019417106813086598
Confidence: 0.6246943765281173
Lift: 18.767216788916055
=====================================
Rule: ALARM CLOCK BAKELIKE IVORY -> ALARM CLOCK BAKELIKE RED
Support: 0.011551468632442908
Confidence: 0.6043737574552684
Lift: 18.15673992574235
=====================================
Rule: ALARM CLOCK BAKELIKE ORANGE -> ALARM CLOCK BAKELIKE RED
Support: 0.009955542045065926
Confidence: 0.6787564766839378
Lift: 20.39136323846026
=====================================
```

Further Analysis of the result in Setting 2

Setting 2:

- Min Support = 0.009,
- Min Confidence = 0.5,
- Min Lift = 3

Setting 2 displays the threshold parameters in analyzing our association results.

in Setting 2 we generated 53 rules and we printed only 3 rules just to see how these rules look like but for the purpose of this analysis we will be discussing the outcome on only 5 results.

Rule 1

- Rule: ALARM CLOCK BAKELIKE IVORY -> ALARM CLOCK BAKELIKE GREEN
- Support: 0.010753505338754417
- Confidence: 0.562624254473161
- Lift: 18.100956607543004

The above indicates that people who puchase 'ALARM CLOCK BAKELIKE IVORY' are likely to purchase 'ALARM CLOCK BAKELIKE GREEN' at the same time. With Minimum Suport of 0.010753505338754417, Minimum Confidence of 0.5 and Minimum Lift of 3.

This shows the parameters in Rule 1 is above the algorithm threshold set in Setting 2. Which is very suitable for business decisions for the store keepers ad product manufacturers

Rule 2

- Rule: ALARM CLOCK BAKELIKE GREEN -> ALARM CLOCK BAKELIKE PINK
- Support: 0.012159440665729377
- Confidence: 0.5111821086261981
- Lift: 16.44594077349102

Rule 2 shows that customers who buy ' ALARM CLOCK BAKELIKE GREEN' also buys 'ALARM CLOCK BAKELIKE PINK' together. With the above parameters for Minimum Support, Minimum Confidence and Minimum Lift.

These parameters in Rule 2 are above the algorithm threshold in Setting 2 used for this analysis and suitable for further analysis and decison making.

Rule 3

- Rule: ALARM CLOCK BAKELIKE GREEN -> ALARM CLOCK BAKELIKE RED
- Support: 0.019417106813086598
- Confidence: 0.6246943765281173
- Lift: 18.767216788916055

Rule 3 indicates that customers who buy 'ALARM CLOCK BAKELIKE GREEN' also buys 'ALARM CLOCK BAKELIKE RED' or vice-versa with the above parameters for Minimum Support, Minimum Confidence and Minimum Lift. This also indicates that results in Rule 3 is higher than the algorithm values in Setting 2 which is serves as a threshold for analysis, meaning min support, min confidence and lift shouldn't fall below these parameters in setting 2 cause they will be considered as weak and not suitable for decision making and further analysis.

Rule 4

- Rule: ALARM CLOCK BAKELIKE IVORY -> ALARM CLOCK BAKELIKE RED
- Support: 0.011551468632442908
- Confidence: 0.6043737574552684
- Lift: 18.15673992574235

The above rule shows that people who buy 'ALARM CLOCK BAKELIKE IVORY' mostly buys it with 'ALARM CLOCK BAKELIKE RED' or vice-versa. The above parameters in Rule 4 shows that the rule is a strong one and also shows that the algorithm parameters applied in Setting 2 is strong and appropriate for business decision making.

Rule 5

- Rule: ALARM CLOCK BAKELIKE ORANGE -> ALARM CLOCK BAKELIKE RED
- Support: 0.009955542045065926
- Confidence: 0.6787564766839378
- Lift: 20.39136323846026

Lastly, Rule 5 indicates that customers who buy 'ALARM CLOCK BAKELIKE ORANGE' also buys 'ALARM CLOCK BAKELIKE RED' or vice versa. With a minimum support of 0.009955542045065926, minimum Confidence of 0.6787564766839378 and Lift of 20.39136323846026. These are above the parameters in setting 2 and deemed appropriate for further analysis.

Setting 3:

- Min Support = 0.015
- Min Confidence = 0.5
- Min Lift = 9

The above will serve as a baseline parameters for our apriori algorithm and analysing the result for minimum support, minimum confidence and minimum lift.

Applying the Apriori Algorithm

In the cell below we will run the apriori algorithm based on the parameters in Setting 3

In [42]:
```
association_rules = apriori(baskets, min_support=0.015, min_confidence=0.5,
 min_lift=9, min_length=2)
```

Now lets calculate the number of rules generated using the code below

In [43]:
```
print('Rules generated: ', len(association_results))
```

```
Rules generated:  53
```

Analysing the Results

Here it involves checking the kind of items are featured in some rules using the code below to display results.

In [44]:
```
def display_rules(association_results):
 for item in association_results:
    pair = item[0]
    items = [x for x in pair]
    print("Rule: " + items[0] + " -> " + items[1])
    print("Support: " + str(item[1]))
    print("Confidence: " + str(item[2][0][2]))
    print("Lift: " + str(item[2][0][3]))
    print("=====================================")
```

The code below displays the rules for the algorithm. Here we will display the rules for only 3 results and analyze each outcome

In [45]:
```
display_rules(association_results[:3])
```

```
Rule: ALARM CLOCK BAKELIKE GREEN -> ALARM CLOCK BAKELIKE IVORY
Support: 0.010753505338754417
Confidence: 0.562624254473161
Lift: 18.100956607543004
=====================================
Rule: ALARM CLOCK BAKELIKE GREEN -> ALARM CLOCK BAKELIKE PINK
Support: 0.012159440665729377
Confidence: 0.5111821086261981
Lift: 16.44594077349102
=====================================
Rule: ALARM CLOCK BAKELIKE GREEN -> ALARM CLOCK BAKELIKE RED
Support: 0.019417106813086598
Confidence: 0.6246943765281173
Lift: 18.767216788916055
=====================================
```

Further Analysis of the result in Setting 3

Setting 3:

- Min Support = 0.015
- Min Confidence = 0.5
- Min Lift = 9

Setting 3 shows the parameters to be used in applying our apriori algorithm.

In Setting 3 we generated 53 rules and we printed results for only 3 rules which we will be discussing their association below.

Rule 1:

- Rule: ALARM CLOCK BAKELIKE IVORY -> ALARM CLOCK BAKELIKE GREEN
- Support: 0.010753505338754417
- Confidence: 0.562624254473161
- Lift: 18.100956607543004

Rule 1 confidence and lift falls within the threadhold of the paramters in Setting 3 but the minimum support is weak/low which is not fit for analysis but If increased by 0.5 we will get better results for analysis and result will also aid decision making for the store owners and manufacturers.

Rule 2:

- Rule: ALARM CLOCK BAKELIKE PINK -> ALARM CLOCK BAKELIKE GREEN
- Support: 0.012159440665729377
- Confidence: 0.5111821086261981
- Lift: 16.44594077349102

Rule 2 confidence and lift falls within the threadhold of the paramters in Setting 3 but the minimum support is weak/low which is not fit for analysis but If increased by 0.5 then we will get better results to run our analysis

Rule 3:

- Rule: ALARM CLOCK BAKELIKE RED -> ALARM CLOCK BAKELIKE GREEN
- Support: 0.019417106813086598
- Confidence: 0.6246943765281173
- Lift: 18.767216788916055

Rule 3 indicates that customers who buy 'ALARM CLOCK BAKELIKE RED' also buys 'ALARM CLOCK BAKELIKE GREEN' or vice-versa with the above parameters for Minimum Support, Minimum Confidence and Minimum Lift. This also indicates that results in Rule 3 is higher than the algorithm values in Setting 3 which serves as a threshold for analysis which is suitable for decision making and further analysis

Lets run further analysis to determine:

- To determine 15 most frequent items bought in the store.

15 most frequent items bought in the store

In [46]:
```python
from collections import Counter
counter = Counter(baskets[0])
for i in baskets[1:]:
 if i != 'nan':
    counter.update(i)
del counter['nan']
counter.most_common(15)
```

```
Out[46]: [('WHITE HANGING HEART T-LIGHT HOLDER', 1978),
          ('REGENCY CAKESTAND 3 TIER', 1798),
          ('JUMBO BAG RED RETROSPOT', 1748),
          ('PARTY BUNTING', 1420),
          ('LUNCH BAG RED RETROSPOT', 1381),
          ('ASSORTED COLOUR BIRD ORNAMENT', 1253),
          ('SET OF 3 CAKE TINS PANTRY DESIGN', 1230),
          ('LUNCH BAG  BLACK SKULL', 1202),
          ('PACK OF 72 RETROSPOT CAKE CASES', 1158),
          ('NATURAL SLATE HEART CHALKBOARD', 1115),
          ('HEART OF WICKER SMALL', 1026),
          ('JUMBO BAG PINK POLKADOT', 1025),
          ('POSTAGE', 1012),
          ('LUNCH BAG CARS BLUE', 1008),
          ('RECIPE BOX PANTRY YELLOW DESIGN', 1002)]
```

Discussion of Market Basket Analysis Above.

In setting 1,2 and 3 we used the Apriori Algorithm to find out the association rules between items in the basket. This simply means how two items are associated and related to each other. In simple words, the apriori algorithm is an association rule learning that analyzes that “People who bought item X also bought item Y.

The objective of the apriori algorithm is to generate the association rule between objects. The association rule tells us how two or three objects are correlated to each other. It is also known as frequent pattern mining.

The apriori algorithm settings in 1,2 and 3 acts as a threshold value to filter out strong rules from weak rules and analyse the results. For business decisions, only strong rules are considereed.

The result from this analysis will be used by different business stalkholders. It would be very helpful to shop/store owners who would like to place both items together on the shelf so can customers can reach them or place frequently bought items apart so that customers who are likely to buy these products gets the time to explore the entire store in search of the product and end up buying more products hereby increasing sales.The shop/store owners would alos use this information to ensure the availability of both frequently bought product for customers purchase. Lastly, lets not forget the manufacturers of these frequently bought product, they will also use this Market Basket Analysis to guage the appetite of the consumers of their product in producing and making these products readily available for the shop owners.

Further Observation. Just exploring other reasons why these product were bought together

- Maybe these products were on sale
- Maybe products were on a discount
- And maybe products were placed on the same location or shelf in the store

Filtering the transactions on the 'day' of the week or on the 'month' to perform analysis on either of them on two durations.

Generating association rules to discover if there are significant differences in the buying behaviour between chosen durations, and, discuss if the rules are useful.

Selcting two durations either from day or month. For example: Picking two durations from 'day' - a weekday (Wednesday) and a weekend (Saturday, Sunday). Or, pick two durations in 'month' (Easter and Christmas periods/ summer and winter).

Finally, discussing whether the rules change in different days or different months. If there are different patterns in the buying behaviour at different days or months? and, what are the possible reasons for it and how the rules generated would help in increasing sales.

In [ ]:
```
Month_hist = data.hist(column="Month", bins=12, grid=False)
for ax in Month_hist.flatten():
    ax.set_xlabel("Month")
    ax.set_ylabel("Frequency")
```

Comparison for 3 months January, February and November

In [ ]:
```
#January
month_jan= data.loc[data['Month']==1]
```

In [ ]:
```
#February
month_feb= data.loc[data['Month']==4]
```

In [ ]:
```
#November
month_nov= data.loc[data['Month']==11]
```

In [ ]:
```
month_nov.head()
```

In [ ]:
```
items_df1=month_jan[month_jan.columns[3:44]]
```

In [ ]:
```
items_df2=month_feb[month_feb.columns[3:44]]
```

In [ ]:
```
items_df11=month_nov[month_nov.columns[3:44]]
```

In [ ]:
```
baskets1 = items_df1.T.apply(lambda x: x.dropna().tolist()).tolist()
```

In [ ]:
```
baskets2 = items_df2.T.apply(lambda x: x.dropna().tolist()).tolist()
```

In [ ]:
```
baskets11 = items_df11.T.apply(lambda x: x.dropna().tolist()).tolist()
```

In [ ]:
```
print(len(baskets1))
```

In [ ]:
```
print(len(baskets2))
```

In [ ]:
```
print(len(baskets11))
```

In [ ]:
```
for i in baskets1[:5]:
 print(i)
```

In [ ]:
```
for i in baskets2[:5]:
 print(i)
```

In [ ]:
```
for i in baskets11[:5]:
 print(i)
```

Algorithm Parameters

We would generate association rules using the parameters below;

- Minimum Support = 0.01
- Minimum Confidence = 0.2
- Minimum Lift = 3
- Minimum Length = 2

In [ ]:
```
association_rules1 = apriori(baskets1, min_support=0.01, min_confidence=0.2,
 min_lift=3, min_length=2)
association_results1 = list(association_rules1)
```

In [ ]:
```
association_rules2 = apriori(baskets2, min_support=0.01, min_confidence=0.2,
 min_lift=3, min_length=2)
association_results2 = list(association_rules2)
```

In [ ]:
```
association_rules11 = apriori(baskets11, min_support=0.01, min_confidence=0.2,
 min_lift=3, min_length=2)
association_results11 = list(association_rules11)
```

Number of Rules Generated

In [ ]:
```
print('Rules generated: ', len(association_results1))
```

In [ ]:
```
print('Rules generated: ', len(association_results2))
```

In [ ]:
```
print('Rules generated: ', len(association_results11))
```

Now Lets Print the Association Rules

In [ ]:
```
print(association_results1[0])
```

In [ ]:
```
print(association_results2[0])
```

In [ ]:
```
print(association_results11[0])
```

Analysing the Results

Here we will check what kind of items are featured in some of the rules

In [ ]:
```
def display_rules(association_results1):
    for item in association_results1:
        pair = item[0]
        items = [x for x in pair]
        print("Rule: " + items[0] + " -> " + items[1])
        print("Support: " + str(item[1]))
        print("Confidence: " + str(item[2][0][2]))
        print("Lift: " + str(item[2][0][3]))
        print("=====================================")
```

In [ ]:
```
def display_rules(association_results2):
    for item in association_results1:
        pair = item[0]
        items = [x for x in pair]
        print("Rule: " + items[0] + " -> " + items[1])
        print("Support: " + str(item[1]))
        print("Confidence: " + str(item[2][0][2]))
        print("Lift: " + str(item[2][0][3]))
        print("=====================================")
```

In [ ]:
```
def display_rules(association_results11):
    for item in association_results1:
        pair = item[0]
        items = [x for x in pair]
        print("Rule: " + items[0] + " -> " + items[1])
        print("Support: " + str(item[1]))
        print("Confidence: " + str(item[2][0][2]))
        print("Lift: " + str(item[2][0][3]))
        print("=====================================")
```

In [ ]:
```
display_rules(association_results1[:5])
```

In [ ]:
```
display_rules(association_results2[:5])
```

In [ ]:
```
display_rules(association_results11[:5])
```

Lets display the 5 most frequent items in the dataset

In [ ]:
```
from collections import Counter
counter = Counter(baskets1[0])
for i in baskets1[1:]:
    if i != 'nan':
        counter.update(i)
del counter['nan']
counter.most_common(5)
```

In [ ]:
```
from collections import Counter
counter = Counter(baskets2[0])
for i in baskets2[1:]:
    if i != 'nan':
        counter.update(i)
del counter['nan']
counter.most_common(5)
```

In [ ]:
```
from collections import Counter
counter = Counter(baskets11[0])
for i in baskets11[1:]:
    if i != 'nan':
        counter.update(i)
```

```
del counter['nan']
counter.most_common(5)
```

Discussing whether the rules changed in different days or different months. If there are different patterns in the buying behaviour at different days or months and, what are the possible reasons for it and how the rules generated would help in increasing sales?

The rules generated in different month are all the same across the 3months Jan, Feb and Nov. This shows a steady buying behavioural pattern of customers frequenting the store. In the 3months customers maintained the same pattern of pair purchase of product. It further shows that customers who buy maintained same purchase pattern across the 3months with apriori algorithm strong rules.

This information will be usedful to:

- Store owners to make informed decision on product stocking
- The business stalkholders can use knowledge of these patterns to improve the placement of these items in the store or the layout of mail-order catalog pages and Web pages.
- Inform the manufacturers of these product on consumer apetite inorder to make these products readily available for customer purchase

All these above will inturn increase Sales and customer participation in the business.

Further reason why these items were bought concurrently.

- Maybe items were placed on sales together
- Maybe items were placed in the same shelf location in the store
- Maybe these items were discounted