In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

1)

In [65]:
```
df = pd.read_csv('basket.csv')
df.head()
```

Out[65]:

	Basket_ID	Items
0	1	bread, eggs, milk, cheese
1	2	beer, cheese, milk, diapers
2	3	milk, cheese, beer, bread, diapers
3	4	eggs, butter, bread
4	5	bread, diapers

In [66]:
```
from itertools import combinations
from collections import defaultdict

df['Items'] = df['Items'].apply(lambda x: set(x.split(', ')))
transactions = df['Items'].tolist()

# Calculating frequency and support for single-item itemsets
item_counts = defaultdict(int)
num_transactions = len(transactions)
```

In [67]:
```
transactions
```

Out[67]:
```
[{'bread', 'cheese', 'eggs', 'milk'},
 {'beer', 'cheese', 'diapers', 'milk'},
 {'beer', 'bread', 'cheese', 'diapers', 'milk'},
 {'bread', 'butter', 'eggs'},
 {'bread', 'diapers'},
 {'bread', 'diapers', 'milk'},
 {'beer', 'bread', 'cheese', 'diapers', 'eggs'},
 {'butter', 'diapers'},
 {'beer', 'butter', 'cheese'},
 {'bread', 'cheese', 'eggs', 'milk'},
 {'butter', 'diapers', 'milk'},
 {'butter', 'diapers', 'eggs', 'milk'},
 {'beer', 'bread', 'butter', 'diapers', 'milk'},
 {'butter', 'milk'},
 {'bread', 'butter', 'diapers', 'eggs', 'milk'},
 {'bread', 'butter'},
 {'beer', 'bread', 'diapers', 'eggs', 'milk'},
 {'beer', 'milk'},
 {'beer', 'bread', 'butter', 'diapers', 'eggs'},
 {'beer', 'bread', 'butter', 'milk'},
 {'beer', 'bread', 'cheese', 'eggs'},
 {'beer', 'eggs'},
 {'cheese', 'eggs', 'milk'},
 {'bread', 'butter', 'cheese', 'eggs'},
 {'beer', 'bread', 'eggs', 'milk'},
 {'bread', 'diapers', 'eggs', 'milk'},
 {'bread', 'butter', 'cheese'},
 {'butter', 'cheese', 'diapers', 'eggs'},
 {'bread', 'eggs'},
 {'beer', 'bread', 'diapers', 'milk'}]
```

Frequency of each itemset

In [68]:
```
for transaction in transactions:
    for item in transaction:
        item_counts[item] += 1
item_counts
```

Out[68]:
```
defaultdict(int,
            {'eggs': 16,
             'cheese': 11,
             'milk': 17,
             'bread': 20,
             'diapers': 15,
             'beer': 13,
             'butter': 14})
```

Support of each itemset

In [69]:
```
item_support = {item: count / num_transactions for item, count in item_counts.items()}
item_support
```

Out[69]:
```
{'eggs': 0.5333333333333333,
 'cheese': 0.36666666666666664,
 'milk': 0.5666666666666667,
 'bread': 0.6666666666666666,
 'diapers': 0.5,
 'beer': 0.43333333333333335,
 'butter': 0.4666666666666667}
```

In [60]:
```
rules_association = []
for item1, item2 in combinations(item_counts.keys(), 2):

    supportitem_1 = item_support[item1]      # the support of individual items and pair
    supportitem_2 = item_support[item2]

    pair_count = sum(1 for transaction in transactions if {item1, item2}.issubset(transaction))  # the frequency
    support_pair = pair_count / num_transactions

    if pair_count > 0:

        confidence1_to_2 = support_pair / supportitem_1  # Confidence is gotten from pairsupport(item2 | item1)
        confidence2_to_1 = support_pair / supportitem_2

        lift1_to_2 = confidence1_to_2 / supportitem_2
        lift2_to_1 = confidence2_to_1 / supportitem_1


        rules_association.append((item1, item2, pair_count, support_pair, confidence1_to_2, lift1_to_2))
        rules_association.append((item2, item1, pair_count, support_pair, confidence2_to_1, lift2_to_1))
```

In [61]:
```
# Prepare data for reporting
frequencies = pd.DataFrame({'Item': item_counts.keys(), 'Frequency': item_counts.values()})
frequencies['Support'] = frequencies['Item'].map(item_support)
rules_df = pd.DataFrame(rules_association, columns=[
    'Item1', 'Item2', 'Pair_Count', 'Support Pair', 'Confidence', 'Lift'
])
frequencies
```

Out[61]:

	Item	Frequency	Support
0	eggs	16	0.533333
1	cheese	11	0.366667
2	milk	17	0.566667
3	bread	20	0.666667
4	diapers	15	0.500000
5	beer	13	0.433333
6	butter	14	0.466667

In [62]:
```
rules_df.head()
```

Out[62]:

	Item1	Item2	Pair_Count	Support Pair	Confidence	Lift
0	eggs	cheese	7	0.233333	0.437500	1.193182
1	cheese	eggs	7	0.233333	0.636364	1.193182
2	eggs	milk	8	0.266667	0.500000	0.882353
3	milk	eggs	8	0.266667	0.470588	0.882353
4	eggs	bread	12	0.400000	0.750000	1.125000

In [ ]:
```
```

2)

Apriori algorithm

In [26]:
```
df1 = pd.read_csv('Market_Basket_Optimisation.csv')
df1.head()
```

Out[26]:

	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	green tea	honey	salad	mineral water
0	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	low fat yogurt	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [23]:
```
transactions = df1.apply(lambda row: row.dropna().tolist(), axis=1).tolist()

def apriori(transactions, min_support):
    def get_support_count(itemsets, transactions):
        support_counts = defaultdict(int)  #Calculating support count for each itemset
        for transaction in transactions:
            for item in itemsets:
                if item.issubset(transaction):
                    support_counts[item] += 1
        return support_counts

    single_items = {frozenset([item]) for transaction in transactions for item in transaction}
    #single_items = {item for transaction in transactions for item in transaction}
    support_counts = get_support_count(single_items, transactions)

    frequent_items = {itemset: count for itemset, count in support_counts.items()
                      if count / len(transactions) >= min_support}   #using the minimum support to filter..

    k = 2
    all_frequent_itemsets = [frequent_items]
    while frequent_items:
        candidates = set(
            [frozenset(item1.union(item2)) for item1 in frequent_items
             for item2 in frequent_items if len(item1.union(item2)) == k]
        )
        support_counts = get_support_count(candidates, transactions)
        frequent_items = {itemset: count for itemset, count in support_counts.items()
                          if count / len(transactions) >= min_support}

        if frequent_itemsets:
            all_frequent_itemsets.append(frequent_items)

        k += 1

    return all_frequent_itemsets
```

In [ ]:
```
```

In [30]:
```
min_support = 0.02
frequent_itemsets = apriori(transactions, min_support)
frequent_itemsets_cleaned

# frequent_itemsets_cleaned = [
#     {tuple(sorted(itemset)): count for itemset, count in level.items()}
#     for level in frequent_itemsets
# ]
```

Out[30]:
```
[{('meatballs',): 157,
  ('eggs',): 1348,
  ('burgers',): 654,
  ('turkey',): 469,
  ('avocado',): 249,
  ('whole wheat rice',): 439,
  ('mineral water',): 1787,
  ('green tea',): 990,
  ('energy bar',): 203,
  ('milk',): 972,
  ('low fat yogurt',): 573,
  ('french fries',): 1282,
  ('whole wheat pasta',): 221,
  ('light cream',): 117,
  ('soup',): 379,
  ('frozen vegetables',): 715,
  ('spaghetti',): 1306,
  ('cookies',): 603,
  ('cooking oil',): 383,
  ('champagne',): 351,
```

```
 ('salmon',): 318,
 ('shrimp',): 535,
 ('chicken',): 450,
 ('chocolate',): 1229,
 ('honey',): 355,
 ('oil',): 173,
 ('tomatoes',): 513,
 ('extra dark chocolate',): 90,
 ('fresh tuna',): 167,
 ('black tea',): 107,
 ('protein bar',): 139,
 ('pasta',): 118,
 ('pepper',): 199,
 ('red wine',): 211,
 ('rice',): 141,
 ('body spray',): 86,
 ('pancakes',): 713,
 ('ham',): 199,
 ('grated cheese',): 393,
 ('white wine',): 124,
 ('parmesan cheese',): 149,
 ('fresh bread',): 323,
 ('ground beef',): 737,
 ('escalope',): 595,
 ('frozen smoothie',): 474,
 ('yams',): 85,
 ('herb & pepper',): 371,
 ('tomato sauce',): 106,
 ('magazines',): 82,
 ('strawberries',): 160,
 ('cake',): 608,
 ('cottage cheese',): 238,
 ('hot dogs',): 243,
 ('brownies',): 253,
 ('cereals',): 193,
 ('muffins',): 181,
 ('light mayo',): 204,
 ('olive oil',): 493,
 ('energy drink',): 199,
 ('gums',): 101,
 ('cider',): 79,
 ('yogurt cake',): 205,
 ('mint',): 131,
 ('butter',): 226,
 ('french wine',): 169,
 ('almonds',): 152,
 ('barbecue sauce',): 81,
 ('carrots',): 115,
 ('mushroom cream sauce',): 143,
 ('tomato juice',): 227,
 ('nonfat milk',): 78,
 ('vegetables mix',): 192,
 ('eggplant',): 99,
 ('fromage blanc',): 102,
 ('melons',): 90},
{('burgers', 'eggs'): 216,
 ('mineral water', 'whole wheat rice'): 151,
 ('green tea', 'mineral water'): 232,
 ('milk', 'whole wheat rice'): 89,
 ('green tea', 'milk'): 132,
 ('milk', 'mineral water'): 360,
 ('frozen vegetables', 'green tea'): 108,
 ('green tea', 'spaghetti'): 199,
 ('frozen vegetables', 'spaghetti'): 209,
 ('burgers', 'mineral water'): 183,
 ('eggs', 'turkey'): 146,
 ('cooking oil', 'eggs'): 88,
 ('burgers', 'turkey'): 80,
 ('eggs', 'mineral water'): 382,
 ('mineral water', 'turkey'): 144,
 ('cooking oil', 'mineral water'): 151,
 ('mineral water', 'salmon'): 127,
 ('chicken', 'chocolate'): 110,
 ('chocolate', 'shrimp'): 135,
 ('chocolate', 'low fat yogurt'): 111,
 ('chocolate', 'cooking oil'): 102,
 ('chicken', 'spaghetti'): 129,
 ('spaghetti', 'tomatoes'): 157,
 ('chicken', 'eggs'): 108,
 ('chicken', 'mineral water'): 171,
 ('salmon', 'spaghetti'): 101,
 ('mineral water', 'tomatoes'): 183,
 ('eggs', 'spaghetti'): 274,
```

```
('eggs', 'tomatoes'): 92,
('spaghetti', 'turkey'): 124,
('mineral water', 'spaghetti'): 448,
('french fries', 'milk'): 178,
('eggs', 'shrimp'): 106,
('chocolate', 'eggs'): 249,
('pancakes', 'spaghetti'): 189,
('green tea', 'pancakes'): 123,
('mineral water', 'pancakes'): 253,
('milk', 'soup'): 114,
('soup', 'spaghetti'): 107,
('milk', 'spaghetti'): 266,
('frozen smoothie', 'spaghetti'): 117,
('frozen smoothie', 'milk'): 107,
('ground beef', 'spaghetti'): 294,
('ground beef', 'milk'): 165,
('escalope', 'mineral water'): 128,
('frozen smoothie', 'mineral water'): 151,
('ground beef', 'mineral water'): 307,
('escalope', 'spaghetti'): 105,
('chicken', 'french fries'): 83,
('chocolate', 'mineral water'): 395,
('chocolate', 'french fries'): 258,
('eggs', 'french fries'): 273,
('french fries', 'mineral water'): 253,
('frozen vegetables', 'mineral water'): 268,
('avocado', 'mineral water'): 86,
('french fries', 'turkey'): 80,
('chocolate', 'frozen vegetables'): 172,
('mineral water', 'red wine'): 82,
('cake', 'mineral water'): 206,
('frozen smoothie', 'frozen vegetables'): 75,
('spaghetti', 'whole wheat rice'): 106,
('honey', 'spaghetti'): 89,
('honey', 'mineral water'): 112,
('french fries', 'green tea'): 214,
('french fries', 'pancakes'): 151,
('cereals', 'mineral water'): 77,
('burgers', 'green tea'): 131,
('chocolate', 'green tea'): 176,
('chocolate', 'spaghetti'): 294,
('escalope', 'french fries'): 123,
('frozen vegetables', 'ground beef'): 127,
('olive oil', 'spaghetti'): 172,
('frozen vegetables', 'olive oil'): 85,
('chocolate', 'ground beef'): 173,
('ground beef', 'tomatoes'): 88,
('chocolate', 'tomatoes'): 105,
('milk', 'olive oil'): 128,
('chocolate', 'soup'): 76,
('chocolate', 'milk'): 241,
('milk', 'tomatoes'): 105,
('frozen vegetables', 'tomatoes'): 121,
('chocolate', 'turkey'): 85,
('mineral water', 'soup'): 173,
('frozen vegetables', 'milk'): 177,
('chocolate', 'olive oil'): 123,
('ground beef', 'olive oil'): 106,
('milk', 'turkey'): 85,
('mineral water', 'olive oil'): 206,
('cookies', 'eggs'): 79,
('chicken', 'green tea'): 89,
('green tea', 'turkey'): 89,
('ground beef', 'herb & pepper'): 120,
('herb & pepper', 'mineral water'): 128,
('low fat yogurt', 'mineral water'): 179,
('grated cheese', 'spaghetti'): 124,
('grated cheese', 'mineral water'): 131,
('grated cheese', 'ground beef'): 85,
('cake', 'french fries'): 134,
('cake', 'eggs'): 143,
('burgers', 'milk'): 134,
('burgers', 'spaghetti'): 161,
('burgers', 'french fries'): 165,
('french fries', 'spaghetti'): 207,
('eggs', 'herb & pepper'): 94,
('chocolate', 'frozen smoothie'): 112,
('french fries', 'low fat yogurt'): 100,
('shrimp', 'spaghetti'): 159,
('herb & pepper', 'spaghetti'): 122,
('eggs', 'olive oil'): 90,
('mineral water', 'shrimp'): 176,
('cookies', 'green tea'): 90,
```

```
 ('cookies', 'french fries'): 100,
 ('eggs', 'green tea'): 191,
 ('green tea', 'ground beef'): 111,
 ('burgers', 'ground beef'): 90,
 ('burgers', 'chocolate'): 128,
 ('eggs', 'ground beef'): 150,
 ('eggs', 'milk'): 231,
 ('milk', 'shrimp'): 132,
 ('champagne', 'chocolate'): 87,
 ('chocolate', 'escalope'): 132,
 ('fresh bread', 'mineral water'): 100,
 ('eggs', 'low fat yogurt'): 126,
 ('eggs', 'pancakes'): 163,
 ('french fries', 'grated cheese'): 78,
 ('frozen vegetables', 'shrimp'): 125,
 ('low fat yogurt', 'milk'): 99,
 ('frozen vegetables', 'low fat yogurt'): 76,
 ('burgers', 'frozen vegetables'): 79,
 ('milk', 'pancakes'): 124,
 ('french fries', 'shrimp'): 75,
 ('chocolate', 'pancakes'): 149,
 ('french fries', 'frozen vegetables'): 143,
 ('low fat yogurt', 'spaghetti'): 114,
 ('burgers', 'pancakes'): 79,
 ('frozen vegetables', 'pancakes'): 101,
 ('pancakes', 'shrimp'): 79,
 ('eggs', 'frozen vegetables'): 163,
 ('cake', 'spaghetti'): 136,
 ('french fries', 'tomatoes'): 90,
 ('eggs', 'frozen smoothie'): 83,
 ('cooking oil', 'spaghetti'): 119,
 ('cooking oil', 'milk'): 86,
 ('cake', 'milk'): 100,
 ('red wine', 'spaghetti'): 77,
 ('cake', 'chocolate'): 102,
 ('ground beef', 'shrimp'): 86,
 ('chocolate', 'whole wheat rice'): 90,
 ('eggs', 'whole wheat rice'): 82,
 ('green tea', 'shrimp'): 85,
 ('olive oil', 'pancakes'): 81,
 ('shrimp', 'tomatoes'): 84,
 ('french fries', 'frozen smoothie'): 109,
 ('ground beef', 'pancakes'): 109,
 ('eggs', 'escalope'): 83,
 ('chocolate', 'grated cheese'): 82,
 ('burgers', 'cake'): 86,
 ('cake', 'green tea'): 106,
 ('chocolate', 'salmon'): 80,
 ('frozen smoothie', 'green tea'): 83,
 ('chocolate', 'cookies'): 78,
 ('french fries', 'ground beef'): 104,
 ('french fries', 'whole wheat rice'): 79,
 ('green tea', 'tomatoes'): 92,
 ('cake', 'frozen vegetables'): 77,
 ('cake', 'pancakes'): 89,
 ('chicken', 'milk'): 111},
{('eggs', 'mineral water', 'spaghetti'): 107,
 ('mineral water', 'pancakes', 'spaghetti'): 86,
 ('milk', 'mineral water', 'spaghetti'): 118,
 ('ground beef', 'milk', 'mineral water'): 83,
 ('ground beef', 'mineral water', 'spaghetti'): 128,
 ('chocolate', 'eggs', 'mineral water'): 101,
 ('frozen vegetables', 'mineral water', 'spaghetti'): 90,
 ('frozen vegetables', 'milk', 'mineral water'): 83,
 ('chocolate', 'ground beef', 'mineral water'): 82,
 ('chocolate', 'milk', 'spaghetti'): 82,
 ('mineral water', 'olive oil', 'spaghetti'): 77,
 ('chocolate', 'mineral water', 'spaghetti'): 119,
 ('chocolate', 'milk', 'mineral water'): 105,
 ('chocolate', 'eggs', 'spaghetti'): 79,
 ('eggs', 'ground beef', 'mineral water'): 76,
 ('eggs', 'milk', 'mineral water'): 98,
 ('french fries', 'mineral water', 'spaghetti'): 76},
{}]
```

In [ ]:

Improved Apriori algorithm

In [32]:
```
def improved_apriori(transactions, min_support):

    item_support = defaultdict(int)
    total_transactions = len(transactions)
```

```
    min_support_count = min_support * total_transactions

    for transaction in transactions:
        for item in transaction:
            item_support[frozenset([item])] += 1

    current_itemsets = {itemset: count for itemset, count in item_support.items() if count >= min_support_count}
    frequent_itemsets = list(current_itemsets.items())

    k = 2
    while current_itemsets:
        candidate_support = defaultdict(int)
        current_items = list(current_itemsets.keys())

        for i in range(len(current_items)):
            for j in range(i + 1, len(current_items)):
                candidate = current_items[i] | current_items[j]
                if len(candidate) == k:
                    for transaction in transactions:
                        if candidate.issubset(transaction):
                            candidate_support[candidate] += 1

        # remove the non-frequent candidates
        current_itemsets = {itemset: count for itemset, count in candidate_support.items() if count >= min_suppo
        frequent_itemsets.extend(current_itemsets.items())

        k += 1

    return frequent_itemsets


frequent_itemsets_improved = improved_apriori(transactions, min_support)
frequent_itemsets_improved[:10]
```

Out[32]:
```
[(frozenset({'burgers'}), 654),
 (frozenset({'meatballs'}), 157),
 (frozenset({'eggs'}), 1348),
 (frozenset({'turkey'}), 469),
 (frozenset({'avocado'}), 249),
 (frozenset({'mineral water'}), 1787),
 (frozenset({'milk'}), 972),
 (frozenset({'energy bar'}), 203),
 (frozenset({'whole wheat rice'}), 439),
 (frozenset({'green tea'}), 990)]
```

In [33]:
```
def generate_association_rules(frequent_itemsets, min_confidence):
    rules = []
    itemset_support = {itemset: support for itemset, support in frequent_itemsets}

    for itemset, support in frequent_itemsets:
        if len(itemset) > 1:
            for item in map(frozenset, combinations(itemset, len(itemset) - 1)):
                consequent = itemset - item
                if consequent and item in itemset_support:
                    confidence = support / itemset_support[item]
                    if confidence >= min_confidence:
                        rules.append((item, consequent, confidence))
    return rules


min_conf = 0.3
association_rules = generate_association_rules(frequent_itemsets_improved, min_conf)
association_rules[:10]
```

Out[33]:
```
[(frozenset({'burgers'}), frozenset({'eggs'}), 0.3302752293577982),
 (frozenset({'milk'}), frozenset({'mineral water'}), 0.37037037037037035),
 (frozenset({'whole wheat rice'}),
  frozenset({'mineral water'}),
  0.3439635535307517),
 (frozenset({'low fat yogurt'}),
  frozenset({'mineral water'}),
  0.31239092495637),
 (frozenset({'soup'}), frozenset({'mineral water'}), 0.45646437994722955),
 (frozenset({'frozen vegetables'}),
  frozenset({'mineral water'}),
  0.3748251748251748),
 (frozenset({'spaghetti'}), frozenset({'mineral water'}), 0.3430321592649311),
 (frozenset({'cooking oil'}),
  frozenset({'mineral water'}),
  0.39425587467362927),
 (frozenset({'shrimp'}), frozenset({'mineral water'}), 0.32897196261682243),
 (frozenset({'chocolate'}), frozenset({'mineral water'}), 0.32113821138211385)]
```

In [ ]:
```

```

3) Eclat

In [52]:
```
def build_vert_data(transactions):
    vertical_data = defaultdict(set)
    for tid, transaction in enumerate(transactions):
        for item in transaction:
            vertical_data[item].add(tid)
    return vertical_data


def eclat(prefix, items, vertical_data, min_support_count, frequent_itemsets):
    while items:
        item = items.pop()
        new_prefix = prefix | {item}
        tid_set = vertical_data[item]

        support = len(tid_set)
        if support >= min_support_count:
            frequent_itemsets.append((new_prefix, support))
            new_items = [i for i in items if len(tid_set & vertical_data[i]) >= min_support_count]  # Generating
            new_vertical_data = {i: tid_set & vertical_data[i] for i in new_items}
            eclat(new_prefix, new_items, new_vertical_data, min_support_count, frequent_itemsets)
```

In [53]:
```
def eclat_run(transactions, min_support):
    total_transactions = len(transactions)
    min_support_count = min_support * total_transactions
    vertical_data = build_vert_data(transactions)
    frequent_itemsets = []
    J = set(vertical_data.keys())
    eclat(set(), J, vertical_data, min_support_count, frequent_itemsets)
    return frequent_itemsets

frequent_itemsets = eclat_run(transactions, min_support)
frequent_itemsets[:10]
```

Out[53]:
```
[({'ham'}, 199),
 ({'champagne'}, 351),
 ({'milk'}, 972),
 ({'ground beef', 'milk'}, 165),
 ({'milk', 'mineral water'}, 360),
 ({'milk', 'spaghetti'}, 266),
 ({'french fries', 'milk'}, 178),
 ({'eggs', 'milk'}, 231),
 ({'chocolate', 'milk'}, 241),
 ({'frozen vegetables', 'milk'}, 177)]
```

In [ ]:
```
association_rules_eclat = generate_association_rules(frequent_itemsets_eclat, min_confidence)
association_rules_eclat[:10]
```

In [ ]:
```

```