

## Questão 01

Com um **suporte mínimo aceitável de 0.3** e **confiança de 0.8**. O número de *ItensSets* 1, 2, 3 e de regras a partir desta base de dados é:

### ItemSet 1

- ☒ Pão = 5/10 = 0.5
- ☒ Manteiga = 5/10 = 0.5
- ☒ Café = 3/10 = 0.3
- ☐ Leite = 2/10 = 0.2
- ☐ Cerveja = 2/10 = 0.2
- ☐ Arroz = 2/10 = 0.2

### ItemSet 2

- ☒ Café → Pão = 3/3 = 1
- ☒ Café → Manteiga = 3/3 = 1
- ☒ Pão → Manteiga = 4/5 = 0.8
- ☒ Manteiga → Pão = 4/5 = 0.8
- ☐ Pão → Café = 3/5 = 0.6
- ☐ Manteiga → Café = 3/5 = 0.6

### ItemSet 3

- ☒ Café e Pão → Manteiga = 3/3 = 1
- ☒ Café e Manteiga → Pão = 3/3 = 1
- ☐ Pão e Manteiga → Café = 3/4 = 0.75
- ☒ Café → Pão e Manteiga = 3/3 = 1
- ☐ Pão → Café e Manteiga = 3/5 = 0.6
- ☐ Manteiga → Café e Pão = 3/5 = 0.6

### Resultado

Num	Regras	Confiança
1	Se Café → Pão	3/3 = 1
2	Se Café → manteiga	3/3 = 1
3	Pão → manteiga	4/5 = 0.8
4	Manteiga → Pão	4/5 = 0.8
5	Se Café e pão → manteiga	3/3 = 1
6	Se Café e manteiga → pão	3/3 = 1
7	Se café → pão e manteiga	3/3 = 1

▼

## Questão 02

Rodando código do arquivo `Apriori.ipynb` disponibilizado no Canvas, com a base de dados acima:

Instalar dependência e ler arquivo `.csv`:

```
!pip install apyori
```

```
➡ Requirement already satisfied: apyori in /usr/local/lib/python3.11/dist-packages (1.1.2)
```

```
import pandas as pd
from apyori import apriori

df = pd.read_csv('supermercado.csv', sep=';', encoding='utf-8', header=None)
```

df

		0	1	2	3	4	5	6	7
	0	Nº	Leite	Café	Cerveja	Pão	Manteiga	Arroz	Feijão
	1	1	Não	Sim	Não	Sim	Sim	Não	Não
	2	2	Sim	Não	Sim	Sim	Sim	Não	Não
	3	3	Não	Sim	Não	Sim	Sim	Não	Não
	4	4	Sim	Sim	Não	Sim	Sim	Não	Não
	5	5	Não	Não	Sim	Não	Não	Não	Não
	6	6	Não	Não	Não	Não	Sim	Não	Não
	7	7	Não	Não	Não	Sim	Não	Não	Não
	8	8	Não	Não	Não	Não	Não	Não	Sim
	9	9	Não	Não	Não	Não	Não	Sim	Sim
	10	10	Não	Não	Não	Não	Não	Sim	Não

```
df.shape
```

(11, 8)

Separando o cabeçalho dos dados:

```
items = df.iloc[0, 1:].tolist()
transactions = df.iloc[1:].reset_index(drop=True)
```

items

['Leite', 'Café', 'Cerveja', 'Pão', 'Manteiga', 'Arroz', 'Feijão']

transactions

		0	1	2	3	4	5	6	7
	0	1	Não	Sim	Não	Sim	Sim	Não	Não
	1	2	Sim	Não	Sim	Sim	Sim	Não	Não
	2	3	Não	Sim	Não	Sim	Sim	Não	Não
	3	4	Sim	Sim	Não	Sim	Sim	Não	Não
	4	5	Não	Não	Sim	Não	Não	Não	Não
	5	6	Não	Não	Não	Não	Sim	Não	Não
	6	7	Não	Não	Não	Sim	Não	Não	Não
	7	8	Não	Não	Não	Não	Não	Não	Sim
	8	9	Não	Não	Não	Não	Não	Sim	Sim
	9	10	Não	Não	Não	Não	Não	Sim	Não

Transformando o DataFrame em uma lista de transações:

```
transactions_list = []
for index, row in transactions.iterrows():
    transaction = []
    for i in range(len(items)):
        # Acessa o valor na linha atual, coluna i+1 (para pular a primeira coluna que não é item)
        if row[i + 1] == 'Sim':
            transaction.append(items[i])
    transactions_list.append(transaction)
```

```
# Ordena a lista de transações pelo número de itens em cada uma
transactions_list = sorted(transactions_list, key=lambda x: len(x))
```

```
transactions_list
```

```
[[ 'Cerveja'],
 [ 'Manteiga'],
 [ 'Pão'],
 [ 'Feijão'],
 [ 'Arroz'],
 [ 'Arroz', 'Feijão'],
 [ 'Café', 'Pão', 'Manteiga'],
 [ 'Café', 'Pão', 'Manteiga'],
 [ 'Leite', 'Cerveja', 'Pão', 'Manteiga'],
 [ 'Leite', 'Café', 'Pão', 'Manteiga']]
```

**Executando o Algoritmo Apriori e armazenando as regras obtidas:**

```
regras = apriori(transactions_list, min_support = 0.3, min_confidence = 0.8)
saida = list(regras)
```

```
print(len(saida))
for i in range(len(saida)):
    print(saida[i])
```

```
4
RelationRecord(items=frozenset({'Café', 'Manteiga'}), support=0.3, ordered_statistics=[OrderedStatistic(items_base=frozenset(
RelationRecord(items=frozenset({'Café', 'Pão'}), support=0.3, ordered_statistics=[OrderedStatistic(items_base=frozenset({'Caf
RelationRecord(items=frozenset({'Manteiga', 'Pão'}), support=0.4, ordered_statistics=[OrderedStatistic(items_base=frozenset({
RelationRecord(items=frozenset({'Café', 'Manteiga', 'Pão'}), support=0.3, ordered_statistics=[OrderedStatistic(items_base=fro
```

**Transformando o resultado em um DataFrame para facilitar a vizualização:**

```
antecedente = []
consequente = []
suporte = []
confianca = []
lift = []
regrasFinais = []

for resultado in saida:
    s = resultado[1]
    result_rules = resultado[2]
    for result_rule in result_rules:
        a = list(result_rule[0])
        b = list(result_rule[1])
        c = result_rule[2]
        l = result_rule[3]
        if 'nan' in a or 'nan' in b: continue
        if len(a) == 0 or len(b) == 0: continue
        antecedente.append(a)
        consequente.append(b)
        suporte.append(s)
        confianca.append(c)
        lift.append(l)
    regrasFinais = pd.DataFrame({'Antecedente': antecedente, 'Consequente': consequente, 'suporte': suporte, 'confianca': confian
```

```
regrasFinais
```

	Antecedente	Consequente	suporte	confianca	lift
0	[Café]	[Manteiga]	0.3	1.0	2.0
1	[Café]	[Pão]	0.3	1.0	2.0
2	[Manteiga]	[Pão]	0.4	0.8	1.6
3	[Pão]	[Manteiga]	0.4	0.8	1.6
4	[Café]	[Manteiga, Pão]	0.3	1.0	2.5
5	[Café, Manteiga]	[Pão]	0.3	1.0	2.0
6	[Café, Pão]	[Manteiga]	0.3	1.0	2.0

Ordenando resultados pela métrica *lift*:

```
regrasFinais.sort_values(by='lift', ascending =False)
```

	Antecedente	Consequente	suporte	confianca	lift
4	[Café]	[Manteiga, Pão]	0.3	1.0	2.5
1	[Café]	[Pão]	0.3	1.0	2.0
0	[Café]	[Manteiga]	0.3	1.0	2.0
6	[Café, Pão]	[Manteiga]	0.3	1.0	2.0
5	[Café, Manteiga]	[Pão]	0.3	1.0	2.0
2	[Manteiga]	[Pão]	0.4	0.8	1.6
3	[Pão]	[Manteiga]	0.4	0.8	1.6

### Questão 03

A partir do código da Questão 02 e com a base de dados acima, imprimindo os *Itemsets* gerados, com os respectivos suportes:

```
from itertools import combinations
from collections import Counter

total_transacoes = len(transactions)

todas_combinacoes = []
# Para cada transação, pega os itens com "Sim" e gera combinações de 1 até N
for _, row in transactions.iterrows():
    itens_presentes = [items[i] for i in range(len(items)) if row[i + 1] == 'Sim']
    # Gera combinações (itemsets) de tamanho 1 até o total presente na transação
    for tamanho in range(1, len(itens_presentes) + 1):
        for combinacao in combinations(sorted(itens_presentes), tamanho):
            todas_combinacoes.append(combinacao)

# Conta quantas vezes cada combinação apareceu
contagem = Counter(todas_combinacoes)

# Imprime itemsets organizados por tamanho e seu suporte
print( "Suporte de cada ItemSets:" )
ultimo_tamanho = 0
for itemset, qtd in sorted(contagem.items(), key=lambda x: (len(x[0]), x[0])):
    tamanho = len(itemset)
    if tamanho != ultimo_tamanho:
        print(f"\nItemset {tamanho}:")
        ultimo_tamanho = tamanho
    suporte = qtd / total_transacoes
    print(f"\t{list(itemset)} -> suporte: {suporte} ({qtd}/{total_transacoes})")
```

```
Itemset 1 (conjuntos com 1 item):
['Arroz'] -> suporte: 20.00% (2/10)
['Café'] -> suporte: 30.00% (3/10)
['Cerveja'] -> suporte: 20.00% (2/10)
['Feijão'] -> suporte: 20.00% (2/10)
['Leite'] -> suporte: 20.00% (2/10)
['Manteiga'] -> suporte: 50.00% (5/10)
['Pão'] -> suporte: 50.00% (5/10)
```

```
Itemset 2 (conjuntos com 2 items):
['Arroz', 'Feijão'] -> suporte: 10.00% (1/10)
['Café', 'Leite'] -> suporte: 10.00% (1/10)
['Café', 'Manteiga'] -> suporte: 30.00% (3/10)
['Café', 'Pão'] -> suporte: 30.00% (3/10)
['Cerveja', 'Leite'] -> suporte: 10.00% (1/10)
['Cerveja', 'Manteiga'] -> suporte: 10.00% (1/10)
['Cerveja', 'Pão'] -> suporte: 10.00% (1/10)
['Leite', 'Manteiga'] -> suporte: 20.00% (2/10)
['Leite', 'Pão'] -> suporte: 20.00% (2/10)
['Manteiga', 'Pão'] -> suporte: 40.00% (4/10)

Itemset 3 (conjuntos com 3 items):
['Café', 'Leite', 'Manteiga'] -> suporte: 10.00% (1/10)
['Café', 'Leite', 'Pão'] -> suporte: 10.00% (1/10)
['Café', 'Manteiga', 'Pão'] -> suporte: 30.00% (3/10)
['Cerveja', 'Leite', 'Manteiga'] -> suporte: 10.00% (1/10)
['Cerveja', 'Leite', 'Pão'] -> suporte: 10.00% (1/10)
['Cerveja', 'Manteiga', 'Pão'] -> suporte: 10.00% (1/10)
['Leite', 'Manteiga', 'Pão'] -> suporte: 20.00% (2/10)

Itemset 4 (conjuntos com 4 items):
['Café', 'Leite', 'Manteiga', 'Pão'] -> suporte: 10.00% (1/10)
['Cerveja', 'Leite', 'Manteiga', 'Pão'] -> suporte: 10.00% (1/10)
```

## ❏ Questão 04

Alterando o código feito na [Questão 2](#) para que ele gere regras de associação quando não há presença do produto, isto é, por exemplo, "quem não leva álcool leva detergente":

transactions

		0	1	2	3	4	5	6	7
0	1	Não	Sim	Não	Sim	Sim	Não	Não	
1	2	Sim	Não	Sim	Sim	Sim	Não	Não	
2	3	Não	Sim	Não	Sim	Sim	Não	Não	
3	4	Sim	Sim	Não	Sim	Sim	Não	Não	
4	5	Não	Não	Sim	Não	Não	Não	Não	
5	6	Não	Não	Não	Não	Sim	Não	Não	
6	7	Não	Não	Não	Sim	Não	Não	Não	
7	8	Não	Não	Não	Não	Não	Não	Sim	
8	9	Não	Não	Não	Não	Não	Sim	Sim	
9	10	Não	Não	Não	Não	Não	Sim	Não	

### Tranformando o DataFrame em uma lista de Transações

```
not_transactions_list = []
for index, row in transactions.iterrows():
    not_transaction = []
    for i in range(len(items)):
        # Acessa o valor na linha atual, coluna i+1 (para pular a primeira coluna que não é item)
        if row[i + 1] != 'Sim': # Mudando para "==" 'Não' ou para "!=" 'Sim'
            not_transaction.append(items[i])
    not_transactions_list.append(not_transaction)

# Ordena a lista de transações pelo número de itens em cada uma
not_transactions_list = sorted(not_transactions_list, key=lambda x: len(x))
```

not\_transactions\_list

```
❏ [['Café', 'Arroz', 'Feijão'],
   ['Cerveja', 'Arroz', 'Feijão'],
   ['Leite', 'Cerveja', 'Arroz', 'Feijão'],
   ['Leite', 'Cerveja', 'Arroz', 'Feijão'],
   ['Leite', 'Café', 'Cerveja', 'Pão', 'Manteiga'],
   ['Leite', 'Café', 'Pão', 'Manteiga', 'Arroz', 'Feijão'],
   ['Leite', 'Café', 'Cerveja', 'Pão', 'Arroz', 'Feijão'],
```

```
['Leite', 'Café', 'Cerveja', 'Manteiga', 'Arroz', 'Feijão'],  
['Leite', 'Café', 'Cerveja', 'Pão', 'Manteiga', 'Arroz'],  
['Leite', 'Café', 'Cerveja', 'Pão', 'Manteiga', 'Feijão']]
```

### Executando o Algoritmo Apriori e armazenando as regras obtidas:

```
not_regras = apriori(not_transactions_list, min_support = 0.3, min_confidence = 0.8)  
not_saida = list(not_regras)
```

```
print(len(not_saida))  
for i in range(len(not_saida)):  
    print(not_saida[i])
```

➡ 51  
RelationRecord(items=frozenset({'Arroz'}), support=0.8, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_ad  
RelationRecord(items=frozenset({'Cerveja'}), support=0.8, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_  
RelationRecord(items=frozenset({'Feijão'}), support=0.8, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_a  
RelationRecord(items=frozenset({'Leite'}), support=0.8, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(), items\_ad  
RelationRecord(items=frozenset({'Arroz', 'Feijão'}), support=0.7, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({  
RelationRecord(items=frozenset({'Leite', 'Café'}), support=0.6, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'C  
RelationRecord(items=frozenset({'Café', 'Manteiga'}), support=0.5, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(  
RelationRecord(items=frozenset({'Café', 'Pão'}), support=0.5, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Pão  
RelationRecord(items=frozenset({'Leite', 'Cerveja'}), support=0.7, ordered\_statistics=[OrderedStatistic(items\_base=frozenset(  
RelationRecord(items=frozenset({'Manteiga', 'Cerveja'}), support=0.4, ordered\_statistics=[OrderedStatistic(items\_base=frozens  
RelationRecord(items=frozenset({'Cerveja', 'Pão'}), support=0.4, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'  
RelationRecord(items=frozenset({'Leite', 'Manteiga'}), support=0.5, ordered\_statistics=[OrderedStatistic(items\_base=frozenset  
RelationRecord(items=frozenset({'Leite', 'Pão'}), support=0.5, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({'Pã  
RelationRecord(items=frozenset({'Manteiga', 'Pão'}), support=0.4, ordered\_statistics=[OrderedStatistic(items\_base=frozenset({  
RelationRecord(items=frozenset({'Arroz', 'Café', 'Feijão'}), support=0.4, ordered\_statistics=[OrderedStatistic(items\_base=fro  
RelationRecord(items=frozenset({'Arroz', 'Café', 'Leite'}), support=0.4, ordered\_statistics=[OrderedStatistic(items\_base=froz  
RelationRecord(items=frozenset({'Arroz', 'Café', 'Manteiga'}), support=0.3, ordered\_statistics=[OrderedStatistic(items\_base=f  
RelationRecord(items=frozenset({'Arroz', 'Café', 'Pão'}), support=0.3, ordered\_statistics=[OrderedStatistic(items\_base=frozen  
RelationRecord(items=frozenset({'Arroz', 'Feijão', 'Cerveja'}), support=0.5, ordered\_statistics=[OrderedStatistic(items\_base=  
RelationRecord(items=frozenset({'Arroz', 'Leite', 'Cerveja'}), support=0.5, ordered\_statistics=[OrderedStatistic(items\_base=f  
RelationRecord(items=frozenset({'Arroz', 'Feijão', 'Leite'}), support=0.5, ordered\_statistics=[OrderedStatistic(items\_base=fr  
RelationRecord(items=frozenset({'Arroz', 'Leite', 'Manteiga'}), support=0.3, ordered\_statistics=[OrderedStatistic(items\_base=  
RelationRecord(items=frozenset({'Arroz', 'Leite', 'Pão'}), support=0.3, ordered\_statistics=[OrderedStatistic(items\_base=froze  
RelationRecord(items=frozenset({'Leite', 'Café', 'Cerveja'}), support=0.5, ordered\_statistics=[OrderedStatistic(items\_base=fr  
RelationRecord(items=frozenset({'Manteiga', 'Café', 'Cerveja'}), support=0.4, ordered\_statistics=[OrderedStatistic(items\_base  
RelationRecord(items=frozenset({'Café', 'Cerveja', 'Pão'}), support=0.4, ordered\_statistics=[OrderedStatistic(items\_base=froz  
RelationRecord(items=frozenset({'Leite', 'Café', 'Feijão'}), support=0.4, ordered\_statistics=[OrderedStatistic(items\_base=fro  
RelationRecord(items=frozenset({'Café', 'Feijão', 'Manteiga'}), support=0.3, ordered\_statistics=[OrderedStatistic(items\_base=  
RelationRecord(items=frozenset({'Café', 'Feijão', 'Pão'}), support=0.3, ordered\_statistics=[OrderedStatistic(items\_base=froze  
RelationRecord(items=frozenset({'Leite', 'Café', 'Manteiga'}), support=0.5, ordered\_statistics=[OrderedStatistic(items\_base=f  
RelationRecord(items=frozenset({'Leite', 'Café', 'Pão'}), support=0.5, ordered\_statistics=[OrderedStatistic(items\_base=frozen  
RelationRecord(items=frozenset({'Café', 'Manteiga', 'Pão'}), support=0.4, ordered\_statistics=[OrderedStatistic(items\_base=fro  
RelationRecord(items=frozenset({'Leite', 'Feijão', 'Cerveja'}), support=0.5, ordered\_statistics=[OrderedStatistic(items\_base=  
RelationRecord(items=frozenset({'Leite', 'Manteiga', 'Cerveja'}), support=0.4, ordered\_statistics=[OrderedStatistic(items\_bas  
RelationRecord(items=frozenset({'Leite', 'Cerveja', 'Pão'}), support=0.4, ordered\_statistics=[OrderedStatistic(items\_base=fro  
RelationRecord(items=frozenset({'Leite', 'Feijão', 'Manteiga'}), support=0.3, ordered\_statistics=[OrderedStatistic(items\_base  
RelationRecord(items=frozenset({'Leite', 'Feijão', 'Pão'}), support=0.3, ordered\_statistics=[OrderedStatistic(items\_base=froz  
RelationRecord(items=frozenset({'Leite', 'Manteiga', 'Pão'}), support=0.4, ordered\_statistics=[OrderedStatistic(items\_base=fr  
RelationRecord(items=frozenset({'Arroz', 'Café', 'Leite', 'Cerveja'}), support=0.3, ordered\_statistics=[OrderedStatistic(item  
RelationRecord(items=frozenset({'Arroz', 'Café', 'Leite', 'Manteiga'}), support=0.3, ordered\_statistics=[OrderedStatistic(ite  
RelationRecord(items=frozenset({'Arroz', 'Café', 'Leite', 'Pão'}), support=0.3, ordered\_statistics=[OrderedStatistic(items\_ba  
RelationRecord(items=frozenset({'Arroz', 'Feijão', 'Cerveja', 'Leite'}), support=0.4, ordered\_statistics=[OrderedStatistic(it  
RelationRecord(items=frozenset({'Leite', 'Café', 'Feijão', 'Cerveja'}), support=0.3, ordered\_statistics=[OrderedStatistic(ite  
RelationRecord(items=frozenset({'Leite', 'Manteiga', 'Café', 'Cerveja'}), support=0.4, ordered\_statistics=[OrderedStatistic(i  
RelationRecord(items=frozenset({'Leite', 'Café', 'Cerveja', 'Pão'}), support=0.4, ordered\_statistics=[OrderedStatistic(items\_  
RelationRecord(items=frozenset({'Manteiga', 'Café', 'Cerveja', 'Pão'}), support=0.3, ordered\_statistics=[OrderedStatistic(ite  
RelationRecord(items=frozenset({'Leite', 'Café', 'Feijão', 'Manteiga'}), support=0.3, ordered\_statistics=[OrderedStatistic(it  
RelationRecord(items=frozenset({'Leite', 'Café', 'Feijão', 'Pão'}), support=0.3, ordered\_statistics=[OrderedStatistic(items\_b  
RelationRecord(items=frozenset({'Leite', 'Café', 'Manteiga', 'Pão'}), support=0.4, ordered\_statistics=[OrderedStatistic(items  
RelationRecord(items=frozenset({'Leite', 'Manteiga', 'Cerveja', 'Pão'}), support=0.3, ordered\_statistics=[OrderedStatistic(it  
RelationRecord(items=frozenset({'Manteiga', 'Leite', 'Café', 'Cerveja', 'Pão'}), support=0.3, ordered\_statistics=[OrderedStat

### Transformando o resultado em um DataFrame para facilitar a visualização:

```
not_antecedente = []  
not_consequente = []  
not_suporte = []  
not_confianca = []  
not_lift = []  
not_regrasFinais = []  
  
for not_resultado in not_saida:  
    s = not_resultado[1]  
    result_rules = not_resultado[2]
```



```
for result_rule in result_rules:
    a = list(result_rule[0])
    b = list(result_rule[1])
    c = result_rule[2]
    l = result_rule[3]
    if 'nan' in a or 'nan' in b: continue
    if len(a) == 0 or len(b) == 0: continue
    not_antecedente.append(a)
    not_consequente.append(b)
    not_suporte.append(s)
    not_confianca.append(c)
    not_lift.append(l)
not_regrasFinais = pd.DataFrame({'Antecedente': not_antecedente, 'Consequente': not_consequente, 'suporte': not_suporte, 'con
```

not\_regrasFinais

	Antecedente	Consequente	suporte	confianca	lift
0	[Arroz]	[Feijão]	0.7	0.875000	1.093750
1	[Feijão]	[Arroz]	0.7	0.875000	1.093750
2	[Café]	[Leite]	0.6	0.857143	1.071429
3	[Manteiga]	[Café]	0.5	1.000000	1.428571
4	[Pão]	[Café]	0.5	1.000000	1.428571
...	...	...	...	...	...
114	[Leite, Manteiga, Pão]	[Café]	0.4	1.000000	1.428571
115	[Manteiga, Cerveja, Pão]	[Leite]	0.3	1.000000	1.250000
116	[Manteiga, Cerveja, Pão]	[Leite, Café]	0.3	1.000000	1.666667
117	[Manteiga, Café, Cerveja, Pão]	[Leite]	0.3	1.000000	1.250000
118	[Leite, Manteiga, Cerveja, Pão]	[Café]	0.3	1.000000	1.428571

119 rows × 5 columns

Ordenando resultados pela métrica lift:

```
not_regrasFinais.sort_values(by='lift', ascending =False)
```

	Antecedente	Consequente	suporte	confianca	lift
40	[Leite, Café]	[Manteiga]	0.5	0.833333	1.666667
44	[Leite, Café]	[Pão]	0.5	0.833333	1.666667
43	[Pão]	[Leite, Café]	0.5	1.000000	1.666667
39	[Manteiga]	[Leite, Café]	0.5	1.000000	1.666667
82	[Manteiga, Cerveja]	[Leite, Café]	0.4	1.000000	1.666667
...	...	...	...	...	...
77	[Leite, Feijão, Cerveja]	[Arroz]	0.4	0.800000	1.000000
74	[Arroz, Feijão, Cerveja]	[Leite]	0.4	0.800000	1.000000
56	[Leite, Manteiga]	[Cerveja]	0.4	0.800000	1.000000
59	[Leite, Pão]	[Cerveja]	0.4	0.800000	1.000000
95	[Leite, Café, Pão]	[Cerveja]	0.4	0.800000	1.000000

119 rows × 5 columns

❏ Questão 05

Funcionamento da biblioteca `mlxtend` para geração de regras de associação:

Exemplo de Como Usar o Apriori do `mlxtend`:


- [https://rasbt.github.io/mlxtend/user\\_guide/frequent\\_patterns/apriori/](https://rasbt.github.io/mlxtend/user_guide/frequent_patterns/apriori/)

Exemplo de Como Usar o Association Rules do mlxtend:

- [https://rasbt.github.io/mlxtend/user\\_guide/frequent\\_patterns/association\\_rules/](https://rasbt.github.io/mlxtend/user_guide/frequent_patterns/association_rules/)

Instalar dependência e ler arquivo .csv :

```
!pip install mlxtend
```




```
Requirement already satisfied: mlxtend in /usr/local/lib/python3.11/dist-packages (0.23.4)
Requirement already satisfied: scipy>=1.2.1 in /usr/local/lib/python3.11/dist-packages (from mlxtend) (1.14.1)
Requirement already satisfied: numpy>=1.16.2 in /usr/local/lib/python3.11/dist-packages (from mlxtend) (2.0.2)
Requirement already satisfied: pandas>=0.24.2 in /usr/local/lib/python3.11/dist-packages (from mlxtend) (2.2.2)
Requirement already satisfied: scikit-learn>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from mlxtend) (1.6.1)
Requirement already satisfied: matplotlib>=3.0.0 in /usr/local/lib/python3.11/dist-packages (from mlxtend) (3.10.0)
Requirement already satisfied: joblib>=0.13.2 in /usr/local/lib/python3.11/dist-packages (from mlxtend) (1.4.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.0.0->mlxtend)
Requirement already satisfied: cyclor>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.0.0->mlxtend) (0.1
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.0.0->mlxtend)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.0.0->mlxtend)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.0.0->mlxtend) (
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.0.0->mlxtend) (11.1.0
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.0.0->mlxtend)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib>=3.0.0->mlxe
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=0.24.2->mlxtend) (2025.2
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=0.24.2->mlxtend) (2025
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.3.1->mlx
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib>=3.
```

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules


mlx_df = pd.read_csv('supermercado.csv', sep=';', encoding='utf-8')
```

mlx\_df



	Nº	Leite	Café	Cerveja	Pão	Manteiga	Arroz	Feijão
0	1	Não	Sim	Não	Sim	Sim	Não	Não
1	2	Sim	Não	Sim	Sim	Sim	Não	Não
2	3	Não	Sim	Não	Sim	Sim	Não	Não
3	4	Sim	Sim	Não	Sim	Sim	Não	Não
4	5	Não	Não	Sim	Não	Não	Não	Não
5	6	Não	Não	Não	Não	Sim	Não	Não
6	7	Não	Não	Não	Sim	Não	Não	Não
7	8	Não	Não	Não	Não	Não	Não	Sim
8	9	Não	Não	Não	Não	Não	Sim	Sim
9	10	Não	Não	Não	Não	Não	Sim	Não

```
mlx_df.shape
```



```
(10, 8)
```

Transformando o DataFrame em uma lista de transações:

```
mlx_items = list(mlx_df.columns.values)
mlx_transactions_list = []
for index, row in mlx_df.iterrows():
    transaction = []
    for i in range(len(mlx_items)):
        if row.iloc[i] == 'Sim':
            transaction.append(mlx_items[i])
    mlx_transactions_list.append(transaction)
```

```
mlx_transactions_list
```



```
➦ [['Café', 'Pão', 'Manteiga'],
   ['Leite', 'Cerveja', 'Pão', 'Manteiga'],
   ['Café', 'Pão', 'Manteiga'],
   ['Leite', 'Café', 'Pão', 'Manteiga'],
   ['Cerveja'],
   ['Manteiga'],
   ['Pão'],
   ['Feijão'],
   ['Arroz', 'Feijão'],
   ['Arroz']]
```

Codificando para o formato esperado:

```
te = TransactionEncoder()
te_ary = te.fit(mlx_transactions_list).transform(mlx_transactions_list)
mlx_df = pd.DataFrame(te_ary, columns=te.columns_)
```

mlx\_df

➦		Arroz	Café	Cerveja	Feijão	Leite	Manteiga	Pão
	0	False	True	False	False	False	True	True
	1	False	False	True	False	True	True	True
	2	False	True	False	False	False	True	True
	3	False	True	False	False	True	True	True
	4	False	False	True	False	False	False	False
	5	False	False	False	False	False	True	False
	6	False	False	False	False	False	False	True
	7	False	False	False	True	False	False	False
	8	True	False	False	True	False	False	False
	9	True	False	False	False	False	False	False

Selecionando e Filtrando Resultados:

ItemSets

```
mlx_itemsets = apriori(mlx_df, min_support=0.3, use_colnames=True)
mlx_itemsets['length'] = mlx_itemsets['itemsets'].apply(lambda x: len(x))
```

mlx\_itemsets

➦		support	itemsets	length
	0	0.3	(Café)	1
	1	0.5	(Manteiga)	1
	2	0.5	(Pão)	1
	3	0.3	(Café, Manteiga)	2
	4	0.3	(Café, Pão)	2
	5	0.4	(Manteiga, Pão)	2
	6	0.3	(Café, Manteiga, Pão)	3

Regras

```
mlx_rules = association_rules(mlx_itemsets, metric="confidence", min_threshold=0.8)
```

mlx\_rules

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	representativity	leverage	conviction	zhangs
0	(Café)	(Manteiga)	0.3	0.5	0.3	1.0	2.0	1.0	0.15	inf	(
1	(Café)	(Pão)	0.3	0.5	0.3	1.0	2.0	1.0	0.15	inf	(
2	(Manteiga)	(Pão)	0.5	0.5	0.4	0.8	1.6	1.0	0.15	2.5	(
3	(Pão)	(Manteiga)	0.5	0.5	0.4	0.8	1.6	1.0	0.15	2.5	(
4	(Café, Manteiga)	(Pão)	0.3	0.5	0.3	1.0	2.0	1.0	0.15	inf	(
5	(Café, Pão)	(Manteiga)	0.3	0.5	0.3	1.0	2.0	1.0	0.15	inf	(

```
item_mapping = {index: product for index, product in enumerate(items)}
print("Regras geradas:")
for index, row in mlx_rules.iterrows():
    antecedent_items = [item_mapping.get(item, item) for item in row['antecedents']]
    consequent_items = [item_mapping.get(item, item) for item in row['consequents']]
    antecedent_items = [str(item) for item in antecedent_items]
    consequent_items = [str(item) for item in consequent_items]
    print(f"Quem leva {' , '.join(antecedent_items)} leva {' , '.join(consequent_items)}")
```

Regras geradas:  
Quem leva Café leva Manteiga  
Quem leva Café leva Pão  
Quem leva Manteiga leva Pão  
Quem leva Pão leva Manteiga  
Quem leva Café, Manteiga leva Pão  
Quem leva Café, Pão leva Manteiga  
Quem leva Café leva Manteiga, Pão

## ❯ Questão 06

O artigo em questão oferece uma revisão abrangente sobre os métodos de visualização utilizados no processo de Association Rule Mining (ARM), destacando sua importância na etapa de pós-processamento e na interpretação dos resultados, especialmente em contextos que envolvem Inteligência Artificial Explicável (XAI). O trabalho evidencia como técnicas de visualização facilitam a compreensão de regras de associação permitindo que usuários explorem dados complexos de forma mais acessível.

Os autores analisam tanto abordagens tradicionais — como scatter plots, grafos e mosaicos — quanto métodos mais recentes, como diagramas de Ishikawa, mapas de metrô e representações moleculares. A categorização das visualizações com base em critérios como foco, quantidade de regras e medidas de interesse contribui para um panorama claro das aplicações e limitações de cada técnica, além de revelar tendências emergentes no campo.

Por fim, o artigo propõe uma taxonomia detalhada das técnicas de visualização em ARM e discute desafios ainda presentes na área, como a complexidade dos conjuntos de regras e a falta de ferramentas padronizadas. A pesquisa também aponta direções futuras, incentivando o desenvolvimento de métodos interativos e adaptativos, fundamentais para integrar cada vez mais a visualização no processo de tomada de decisão baseado em mineração de dados.