Base de Dados

Nº; Leite; Café; Cerveja; Pão; Manteiga; Arroz; Feijão

Show command palette (Ctrl+Shift+P)

1; Não ; Sim ; Não ; Sim; Sim ; Não ; Não

2; Sim ; Não ; Sim ; Sim; Sim ; Não ; Não

3; Não ; Sim ; Não ; Sim; Sim ; Não ; Não

4; Sim ; Sim ; Não ; Sim; Sim ; Não ; Não

5; Não ; Não ; Sim ; Não; Não ; Não ; Não

6; Não ; Não ; Não ; Não; Sim ; Não ; Não

7; Não ; Não ; Não ; Sim; Não ; Não ; Não

8; Não ; Não ; Não ; Não; Não ; Não ; Sim

9; Não ; Não ; Não ; Não; Não ; Sim ; Sim

10; Não ; Não ; Não ; Não; Não ; Sim ; Não

Questao 2

```
# Instalar dependência
# $ pip install apyori

# Importar dependências
import pandas as pd
from apyori import apriori

# Ler .cvs
df = pd.read_csv('supermercado.csv', sep=';', encoding='utf-8', header=None)
print( "DataFrame:" )
print( df )
print( "DataFrame (linhas, colunas):" )
print( df.shape )
print( "// ------------------------------------------------- //")

# Separando o cabeçalho dos dados
items = df.iloc[0, 1:].tolist()
transactions = df.iloc[1:].reset_index(drop=True)
print( "Itens:" )
print( items )
print( "Transações:" )
print( transactions )
print( "// ------------------------------------------------- //")

# Transformando o dataframe em uma lista de transações
transactions_list = []
for index, row in transactions.iterrows():
    transaction = []
    for i in range(len(items)):
        if row[i + 1] == 'Sim':
            transaction.append(items[i])
    transactions_list.append(transaction)
transactions_list = sorted(transactions_list, key=lambda x: len(x))
print( "Lista de Transações:" )
for i in range(len(transactions_list)):
    print( transactions_list[i] )
print( "// ------------------------------------------------- //")

# 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( "Quantidade de Regras Obtidas:" )
print( len(saida) )
print( "Regras Obtidas:" )
for i in range(len(saida)):
    print( saida[i] )
print( "// ------------------------------------------------- //")

# Transformando o resultado em um dataframe para facilitar a vizualização
antecedente = []
```

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

Show command palette (Ctrl+Shift+P)

```
        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': confianca,
                'lift': lift
            })
    print( "DataFrame das Regras: " )
    print( regrasFinais )
    print( "// ------------------------------------------------ //")

    # Ordenando resultados pela métrica lift
    print( "DataFrame das Regras Ordenada por 'lift': " )
    regrasFinais.sort_values(by='lift', ascending =False)
    print( regrasFinais )
```

Questao 3

```
    # Instalar dependência
    # $ pip install apyori

    # Importar dependências
    import pandas as pd
    from itertools import combinations
    from collections import Counter

    # Ler .cvs
    df = pd.read_csv('supermercado.csv', sep=';', encoding='utf-8', header=None)
    print( "DataFrame:" )
    print( df )
    print( "DataFrame (linhas, colunas):" )
    print( df.shape )
    print( "// ------------------------------------------------ //")

    # Separando o cabeçalho dos dados
    items = df.iloc[0, 1:].tolist()
    transactions = df.iloc[1:].reset_index(drop=True)
    print( "Itens:" )
    print( items )
    print( "Transações:" )
    print( transactions )
    print( "// ------------------------------------------------ //")

    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})")
```



Questao 4

```
# Instalar dependência
# $ pip install apyori

# Importar dependências
import pandas as pd
from apyori import apriori

# Ler .cvs
df = pd.read_csv('supermercado.csv', sep=';', encoding='utf-8', header=None)
print( "DataFrame:" )
print( df )
print( "DataFrame (linhas, colunas):" )
print( df.shape )
print( "// ------------------------------------------------ //")

# Separando o cabeçalho dos dados
items = df.iloc[0, 1:].tolist()
transactions = df.iloc[1:].reset_index(drop=True)
print( "Itens:" )
print( items )
print( "Transações:" )
print( transactions )
print( "// ------------------------------------------------ //")

# Transformando o dataframe em uma lista de transações
transactions_list = []
for index, row in transactions.iterrows():
    transaction = []
    for i in range(len(items)):
        if row[i + 1] == 'Não':
            transaction.append(items[i])
    transactions_list.append(transaction)
transactions_list = sorted(transactions_list, key=lambda x: len(x))
print( "Lista de Transações:" )
for i in range(len(transactions_list)):
    print( transactions_list[i] )
print( "// ------------------------------------------------ //")

# 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( "Quantidade de Regras Obtidas:" )
print( len(saida) )
print( "Regras Obtidas:" )
for i in range(len(saida)):
    print( saida[i] )
print( "// ------------------------------------------------ //")

# 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': confianca,
                'lift': lift
            })
    print( "DataFrame das Regras: " )
    print( regrasFinais )
    print( "// ------------------------------------------------ //")

    # Ordenando resultados pela métrica lift
    print( "DataFrame das Regras Ordenada por 'lift': " )
    regrasFinais.sort_values(by='lift', ascending =False)
    print( regrasFinais )
```

Show command palette (Ctrl+Shift+P)

Questao 6

```
    # Instalar dependência
    # $ pip install mlxtend
    # https://rasbt.github.io/mlxtend/user_guide/frequent_patterns/apriori/
    # https://rasbt.github.io/mlxtend/user_guide/frequent_patterns/association_rules/

    # Importar dependências
    import pandas as pd
    from mlxtend.preprocessing import TransactionEncoder
    from mlxtend.frequent_patterns import apriori, association_rules

    # Ler .cvs
    df = pd.read_csv( 'supermercado.csv', sep=';', encoding='utf-8')
    print( "DataFrame:" )
    print( df )
    print( "DataFrame (linhas, colunas):" )
    print( df.shape )
    print( "// ------------------------------------------------ //")

    # Transformando o DataFrame em uma lista de transações
    items = list(df.columns.values)
    transactions_list = []
    for index, row in df.iterrows():
        transaction = []
        for i in range(len(items)):
            if row.iloc[i] == 'Sim':
                transaction.append(items[i])
        transactions_list.append(transaction)
    print( "Lista de Transações:" )
    print( transactions_list )
    print( "// ------------------------------------------------ //")

    # Codificando para o formato esperado
    te = TransactionEncoder()
    te_ary = te.fit(transactions_list).transform(transactions_list)
    df = pd.DataFrame(te_ary, columns=te.columns_)
    print( "DataFrame Codificado:" )
    print( df )
    print( "// ------------------------------------------------ //")

    ## Selecionando e Filtrando Resultados

    # ItemSets
    frequent_itemsets = apriori(df, min_support=0.3, use_colnames=True)
    frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x))
    print( "ItemSets:" )
```

```
print( frequent_itemsets )
print( "// ------------------------------------------------ //")

# Regras
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.8)
```

Show command palette (Ctrl+Shift+P)

```
print( rules )
print( "// ------------------------------------------------ //")

item_mapping = {index: product for index, product in enumerate(items)}
print("Regras geradas:")
for index, row in 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)}")
```