BAN 210: Workshop 2

Import data

In [92]:
```
!pip install pyreadstat
import pyreadstat as pyds
df, meta = pyds.read_sas7bdat('../BAN 210/transactions.sas7bdat')
df
```

```
Requirement already satisfied: pyreadstat in c:\users\admin\anaconda3\lib\site-packages (1.2.7)
Requirement already satisfied: pandas>=1.2.0 in c:\users\admin\anaconda3\lib\site-packages (from pyreadstat) (2.2.2)
Requirement already satisfied: numpy>=1.26.0 in c:\users\admin\anaconda3\lib\site-packages (from pandas>=1.2.0->pyreadstat) (1.2
6.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\admin\anaconda3\lib\site-packages (from pandas>=1.2.0->pyreadst
at) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\admin\anaconda3\lib\site-packages (from pandas>=1.2.0->pyreadstat) (2024.
1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\admin\anaconda3\lib\site-packages (from pandas>=1.2.0->pyreadstat) (202
3.3)
Requirement already satisfied: six>=1.5 in c:\users\admin\anaconda3\lib\site-packages (from python-dateutil>=2.8.2->pandas>=1.2.0
->pyreadstat) (1.16.0)
```

Out[92]:

	Quantity	Transaction	Store	Product
0	1.0	12359.0	2.0	Candy Bar
1	2.0	12362.0	9.0	Pain Reliever
2	2.0	12362.0	9.0	Pain Reliever
3	1.0	12365.0	5.0	Toothpaste
4	2.0	12371.0	2.0	Bow
...	...	...	...	...
459253	1.0	1221863.0	8.0	Candy Bar
459254	1.0	1221863.0	8.0	Greeting Cards
459255	1.0	1221863.0	8.0	Toothpaste
459256	1.0	1221863.0	8.0	Toothpaste
459257	1.0	1221866.0	3.0	Photo Processing

459258 rows × 4 columns

In [93]:
```
# Keep only the columns we need
df = df[['Transaction', 'Product']]
df.head()
```

Out[93]:

	Transaction	Product
0	12359.0	Candy Bar
1	12362.0	Pain Reliever
2	12362.0	Pain Reliever
3	12365.0	Toothpaste
4	12371.0	Bow

Split transactions into a list

In [95]:
```
transactions = df.groupby('Transaction')['Product'].apply(list).tolist()
```

One hot encode the data

In [97]:
```
!pip install mlxtend
```

```
Requirement already satisfied: mlxtend in c:\users\admin\anaconda3\lib\site-packages (0.23.4)
Requirement already satisfied: scipy>=1.2.1 in c:\users\admin\anaconda3\lib\site-packages (from mlxtend) (1.13.1)
Requirement already satisfied: numpy>=1.16.2 in c:\users\admin\anaconda3\lib\site-packages (from mlxtend) (1.26.4)
Requirement already satisfied: pandas>=0.24.2 in c:\users\admin\anaconda3\lib\site-packages (from mlxtend) (2.2.2)
Requirement already satisfied: scikit-learn>=1.3.1 in c:\users\admin\anaconda3\lib\site-packages (from mlxtend) (1.5.1)
Requirement already satisfied: matplotlib>=3.0.0 in c:\users\admin\anaconda3\lib\site-packages (from mlxtend) (3.9.2)
Requirement already satisfied: joblib>=0.13.2 in c:\users\admin\anaconda3\lib\site-packages (from mlxtend) (1.4.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\admin\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend)
(1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\admin\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (0.1
1.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\admin\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend)
(4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\admin\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend)
(1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\admin\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (2
4.1)
Requirement already satisfied: pillow>=8 in c:\users\admin\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\admin\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend)
(3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\admin\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxten
d) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\admin\anaconda3\lib\site-packages (from pandas>=0.24.2->mlxtend) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\admin\anaconda3\lib\site-packages (from pandas>=0.24.2->mlxtend) (2023.
3)
Requirement already satisfied: six>=1.5 in c:\users\admin\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib>=3.
0.0->mlxtend) (1.16.0)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\admin\anaconda3\lib\site-packages (from scikit-learn>=1.3.1->mlxt
end) (3.5.0)
```

In [98]:
```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder

te = TransactionEncoder()
te_array = te.fit(transactions).transform(transactions)
df_encoded = pd.DataFrame(te_array, columns=te.columns_)

df_encoded
```

Out[98]:

	Bow	Candy Bar	Deodorant	Greeting Cards	Magazine	Markers	Pain Reliever	Pencils	Pens	Perfume	Photo Processing	Prescription Med	Shampoo	Soap
0	False	True	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	True	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	True	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	True	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
199995	False	False	False	True	True	False	False	False	True	False	True	False	False	False
199996	False	False	False	False	False	False	False	False	False	False	False	False	False	False
199997	False	True	False	False	False	False	False	False	False	False	False	False	False	False
199998	False	True	False	True	False	False	False	False	False	False	False	False	False	False
199999	False	False	False	False	False	False	False	False	False	False	True	False	False	False

200000 rows × 17 columns

In [99]:

```python
from mlxtend.frequent_patterns import apriori

frequent_itemsets = apriori(df_encoded, min_support=0.025, use_colnames=True) # keep only itemsets that appear in ≥ 2.5 % of all
frequent_itemsets
# anything purchased together in fewer than 2.5 % of transactions is discarded automatically.
```

Out[99]:

	support	itemsets
0	0.054645	(Bow)
1	0.171005	(Candy Bar)
2	0.146885	(Greeting Cards)
3	0.241305	(Magazine)
4	0.026700	(Pain Reliever)
5	0.134925	(Pencils)
6	0.143575	(Pens)
7	0.089960	(Perfume)
8	0.058480	(Photo Processing)
9	0.033800	(Shampoo)
10	0.043025	(Soap)
11	0.067350	(Toothbrush)
12	0.160425	(Toothpaste)
13	0.050990	(Wrapping Paper)
14	0.043660	(Greeting Cards, Candy Bar)
15	0.040535	(Magazine, Candy Bar)
16	0.033015	(Pencils, Candy Bar)
17	0.039780	(Toothpaste, Candy Bar)
18	0.036335	(Greeting Cards, Magazine)
19	0.029240	(Greeting Cards, Pencils)
20	0.032080	(Greeting Cards, Toothpaste)
21	0.031630	(Magazine, Pencils)
22	0.031665	(Toothpaste, Magazine)

In [100…

```
from mlxtend.frequent_patterns import association_rules
```

```
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.06)
rules
```

Out[100…

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	representativity	leverage	conviction	zhangs_metri
0	(Greeting Cards)	(Candy Bar)	0.146885	0.171005	0.043660	0.297239	1.738191	1.0	0.018542	1.179626	0.49781
1	(Candy Bar)	(Greeting Cards)	0.171005	0.146885	0.043660	0.255314	1.738191	1.0	0.018542	1.145604	0.51229
2	(Magazine)	(Candy Bar)	0.241305	0.171005	0.040535	0.167982	0.982325	1.0	-0.000729	0.996367	-0.02316
3	(Candy Bar)	(Magazine)	0.171005	0.241305	0.040535	0.237040	0.982325	1.0	-0.000729	0.994410	-0.02124
4	(Pencils)	(Candy Bar)	0.134925	0.171005	0.033015	0.244691	1.430903	1.0	0.009942	1.097558	0.34810
5	(Candy Bar)	(Pencils)	0.171005	0.134925	0.033015	0.193065	1.430903	1.0	0.009942	1.072050	0.36326
6	(Toothpaste)	(Candy Bar)	0.160425	0.171005	0.039780	0.247966	1.450053	1.0	0.012347	1.102338	0.36967
7	(Candy Bar)	(Toothpaste)	0.171005	0.160425	0.039780	0.232625	1.450053	1.0	0.012347	1.094087	0.37439
8	(Greeting Cards)	(Magazine)	0.146885	0.241305	0.036335	0.247370	1.025136	1.0	0.000891	1.008059	0.02874
9	(Magazine)	(Greeting Cards)	0.241305	0.146885	0.036335	0.150577	1.025136	1.0	0.000891	1.004347	0.03231
10	(Greeting Cards)	(Pencils)	0.146885	0.134925	0.029240	0.199067	1.475392	1.0	0.009422	1.080085	0.37769
11	(Pencils)	(Greeting Cards)	0.134925	0.146885	0.029240	0.216713	1.475392	1.0	0.009422	1.089147	0.37247
12	(Greeting Cards)	(Toothpaste)	0.146885	0.160425	0.032080	0.218402	1.361397	1.0	0.008516	1.074178	0.31116
13	(Toothpaste)	(Greeting Cards)	0.160425	0.146885	0.032080	0.199969	1.361397	1.0	0.008516	1.066352	0.31618
14	(Magazine)	(Pencils)	0.241305	0.134925	0.031630	0.131079	0.971495	1.0	-0.000928	0.995574	-0.03723
15	(Pencils)	(Magazine)	0.134925	0.241305	0.031630	0.234427	0.971495	1.0	-0.000928	0.991015	-0.03280
16	(Toothpaste)	(Magazine)	0.160425	0.241305	0.031665	0.197382	0.817977	1.0	-0.007046	0.945275	-0.20951
17	(Magazine)	(Toothpaste)	0.241305	0.160425	0.031665	0.131224	0.817977	1.0	-0.007046	0.966388	-0.22678

In [101…

```
frequent_itemsets[ frequent_itemsets['itemsets'].str.len() > 1 ].shape
#9 rows → there are 9 frequent itemsets
#2 columns → the DataFrame still has the usual two columns (support and itemsets).
```

Out[101…

```
(9, 2)
```

Basic Understanding:

1. Top Support Rules

Which two rules have the highest support? What are their confidence values?

In [104…

```
# Sort rules by support and show the top 2
top_support_rules = rules.sort_values(by='support', ascending=False).head(2)
top_support_rules[['antecedents', 'consequents', 'support', 'confidence']]
```

Out[104…

	antecedents	consequents	support	confidence
0	(Greeting Cards)	(Candy Bar)	0.04366	0.297239
1	(Candy Bar)	(Greeting Cards)	0.04366	0.255314

Task 1: Top Support Rules

The two rules with the highest support are:

1. **If a customer buys Candy Bars, they also buy Greeting Cards**
   - Support: 4.36%
   - Confidence: 25.53%
2. **If a customer buys Greeting Cards, they also buy Candy Bars**
   - Support: 4.36%
   - Confidence: 29.72%

2. Maximum Lift

What is the maximum lift observed? Which rule has this lift value? Interpret what this rule implies

In [107…
```
max_lift = rules['lift'].max()
max_lift
```

Out[107…
```
1.7381909119394834
```

In [108…
```
max_lift_rule = rules[rules['lift'] == max_lift]
max_lift_rule[['antecedents', 'consequents', 'support', 'confidence', 'lift']]
```

Out[108…

	antecedents	consequents	support	confidence	lift
0	(Greeting Cards)	(Candy Bar)	0.04366	0.297239	1.738191
1	(Candy Bar)	(Greeting Cards)	0.04366	0.255314	1.738191

Task 2: Maximum Lift

- **Rule with Maximum Lift**:
  - Antecedents: *Greeting Cards → Candy Bar*
  - Consequents: *Candy Bar*
  - Support: 0.04366
  - Confidence: 0.297239
  - Lift: 1.738191
- **Rule with Maximum Lift**:
  - Antecedents: *Candy Bar → Greeting Cards*
  - Consequents: *Greeting Cards*
  - Support: 0.04366
  - Confidence: 0.255314
  - Lift: 1.738191

Interpretation:

These rules imply that customers who buy **Greeting Cards** are **1.74 times** more likely to also buy **Candy Bar** (and vice versa) compared to what would be expected if these items were purchased independently. A lift greater than 1 indicates a **positive association**, suggesting that promoting these items together (e.g., discounts or product bundles) could be a valuable marketing strategy.

3. Rule 10

What are the antecedents and consequents of Rule 10? Interpret this rule clearly

```
In [111…
# Task 3: Rule 10
rule_10 = rules.iloc[9] #Python is 0-based, so this is the 10-th row
rule_10[['antecedents', 'consequents', 'support', 'confidence', 'lift']]
```

```
Out[111…
antecedents            (Magazine)
consequents      (Greeting Cards)
support                  0.036335
confidence               0.150577
lift                     1.025136
Name: 9, dtype: object
```

Task 3: Rule 10

Rule 10: If a customer buys **Magazine**, they may also buy **Greeting Cards**

- **Support**: 3.63%
- **Confidence**: 24.737%
- **Lift**: 1.025

Interpretation:
This rule shows a **weak positive association** between buying magazines and greeting cards. Roughly 4 out of every 100 checkout receipts included both a Magazine and Greeting Cards. When a receipt already has a Magazine, there's a 24.7% chance it will also contain Greeting Cards. A customer who buys a magazine is **1.025 times more likely** to also buy greeting cards than by random chance — a very small lift, suggesting a weak but potentially real connection.

Visualization-Based Insights

4. Create a rule matrix using a heatmap where:

- **Rows = LHS (antecedents)**
- **Columns = RHS (consequents)**
- **Values = lift**

From this heatmap:

- **What is the most common right-hand item in the rules?**
- **What are the left-hand items from the second row of the heatmap?**

In [115…

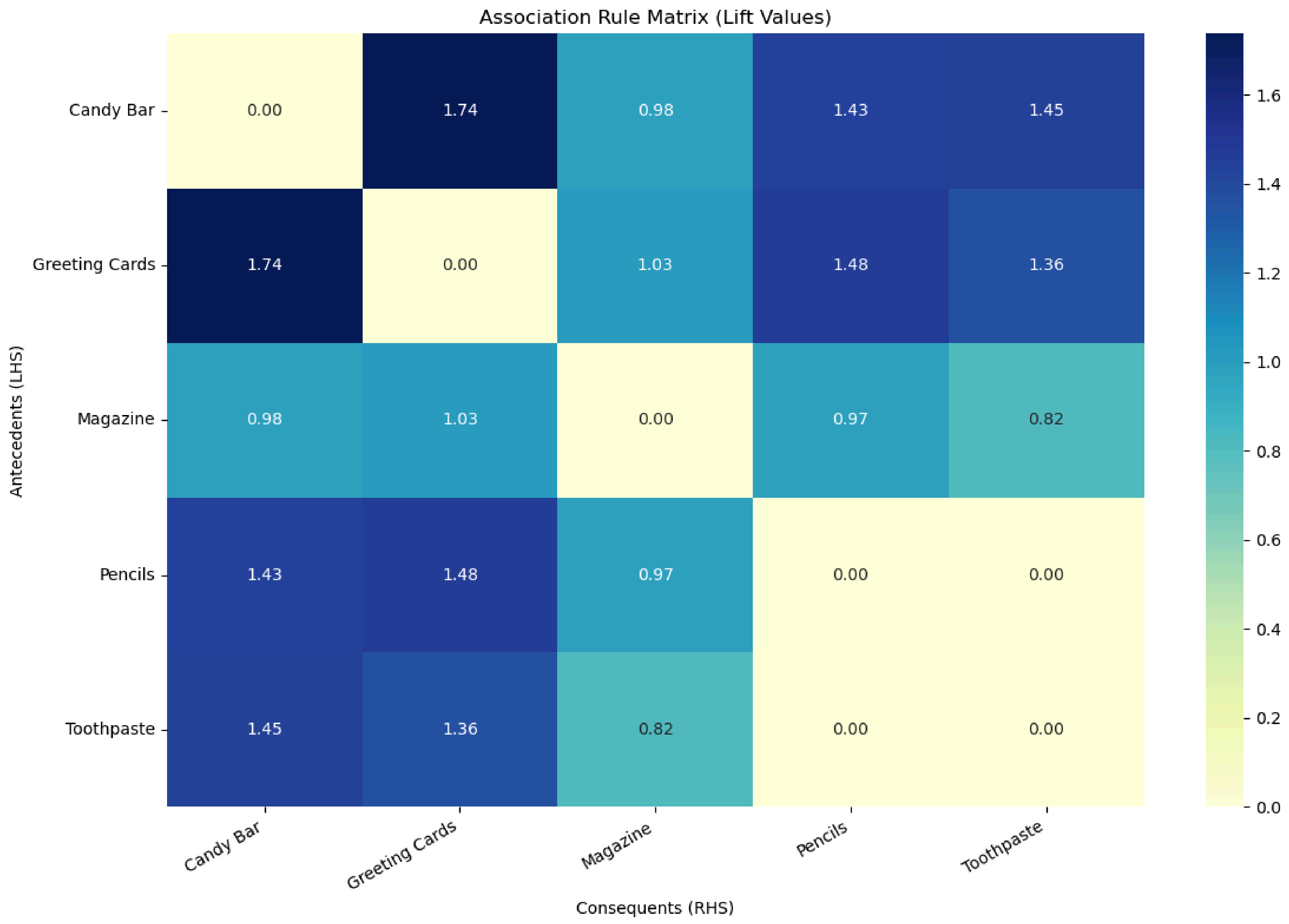
```
# Extract antecedents and consequents as strings
rules['antecedents_str'] = rules['antecedents'].apply(lambda x: ', '.join(list(x)))
rules['consequents_str'] = rules['consequents'].apply(lambda x: ', '.join(list(x)))
#create matrix
lift_matrix = rules.pivot_table(
    index='antecedents_str',
    columns='consequents_str',
    values='lift',
    fill_value=0
)
lift_matrix
```

Out[115…

consequents_str	Candy Bar	Greeting Cards	Magazine	Pencils	Toothpaste
antecedents_str					
Candy Bar	0.000000	1.738191	0.982325	1.430903	1.450053
Greeting Cards	1.738191	0.000000	1.025136	1.475392	1.361397
Magazine	0.982325	1.025136	0.000000	0.971495	0.817977
Pencils	1.430903	1.475392	0.971495	0.000000	0.000000
Toothpaste	1.450053	1.361397	0.817977	0.000000	0.000000

In [116…

```
#Create heatmap
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(12, 8))
sns.heatmap(lift_matrix, annot=True, fmt=".2f", cmap="YlGnBu")
plt.title("Association Rule Matrix (Lift Values)")
plt.xlabel("Consequents (RHS)")
plt.ylabel("Antecedents (LHS)")
plt.xticks(rotation=30, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()
```



In [117…
```
rules['consequents_str'].value_counts()
```

```
Out[117…  consequents_str
          Candy Bar         4
          Greeting Cards    4
          Magazine          4
          Pencils           3
          Toothpaste        3
          Name: count, dtype: int64
```

```
In [118…  most_common_rhs = rules['consequents_str'].value_counts().idxmax()
          print("Most common RHS item:", most_common_rhs)
```

```
Most common RHS item: Candy Bar
```

```
In [119…  second_row_lhs = lift_matrix.index[1]
          print("Left-hand items from second row:", second_row_lhs)
```

```
Left-hand items from second row: Greeting Cards
```

Task 4: Heatmap-Based Insights

The most frequent consequent *(Greeting Cards, Candy Bar, Magazine)* suggests it often appears as a recommended or associated product in the rules, indicating its importance in shopping baskets. The second row's antecedent (Greeting Cards) highlights it as a product that frequently triggers associations with other items (like Candy Bar), which could inform product placement or promotional bundling strategies.

Task 5: Redundant Rules

Use rule pruning techniques to remove redundant rules (rules where a more general rule has equal or higher confidence). List 3 redundant rules that you removed, and explain why.

```
In [122…  # Work on a copy so the original stays intact
          rules_pruned = rules.copy()

          # Helper column: how many items on the LHS?  (Shorter = more general)
          rules_pruned["ante_len"] = rules_pruned["antecedents"].apply(len)
          rules_pruned.head()
```

Out[122…

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	representativity	leverage	conviction	zhangs
0	(Greeting Cards)	(Candy Bar)	0.146885	0.171005	0.043660	0.297239	1.738191	1.0	0.018542	1.179626	0
1	(Candy Bar)	(Greeting Cards)	0.171005	0.146885	0.043660	0.255314	1.738191	1.0	0.018542	1.145604	0
2	(Magazine)	(Candy Bar)	0.241305	0.171005	0.040535	0.167982	0.982325	1.0	-0.000729	0.996367	-0
3	(Candy Bar)	(Magazine)	0.171005	0.241305	0.040535	0.237040	0.982325	1.0	-0.000729	0.994410	-0
4	(Pencils)	(Candy Bar)	0.134925	0.171005	0.033015	0.244691	1.430903	1.0	0.009942	1.097558	0

In [123…

```
# Create a new Flag column – will become True for redundant rows
rules_pruned["redundant"] = False

# Scan each group of rules that predict the same RHS (consequents) item(s)
for cons, grp in rules_pruned.groupby("consequents", sort=False):

    # sort → general rules first (fewer items), higher confidence first
    grp = grp.sort_values(["ante_len", "confidence"],
                          ascending=[True, False])

    for i, idx_parent in enumerate(grp.index):
        if rules_pruned.at[idx_parent, "redundant"]:
            continue                           # skip parents we already dropped

        antecedent_parent = rules_pruned.at[idx_parent, "antecedents"]
        conf_parent       = rules_pruned.at[idx_parent, "confidence"]

        # compare with each more specific rule further down
        for idx_child in grp.index[i+1:]:
            antecedent_child = rules_pruned.at[idx_child, "antecedents"]
            conf_child       = rules_pruned.at[idx_child, "confidence"]

            # condition for redundancy
            if antecedent_parent.issubset(antecedent_child) \
               and conf_parent >= conf_child:
                rules_pruned.at[idx_child, "redundant"] = True

# Split into kept vs. removed
redundant_rules = rules_pruned[rules_pruned["redundant"]].reset_index(drop=True)
kept_rules      = rules_pruned[~rules_pruned["redundant"]].reset_index(drop=True)
```

```
print(f" Pruned away {len(redundant_rules)} redundant rule(s); "
      f"{len(kept_rules)} rule(s) remain.\n")
```

```
Pruned away 0 redundant rule(s); 18 rule(s) remain.
```

There are no redundant rules, thus we do not have three examples to explain

In [125...

```
# To show three examples of what got removed and explain (if there are any rules)
for _, row in redundant_rules.head(3).iterrows():
    # find the shortest parent that was kept
    parent = kept_rules[
        (kept_rules["consequents"] == row["consequents"]) &
        (kept_rules["antecedents"].apply(lambda s: row["antecedents"].issuperset(s)))
    ].sort_values("ante_len").iloc[0]

    print(f"Removed rule : {set(row['antecedents'])}  →  {set(row['consequents'])} "
          f"(confidence = {row['confidence']:.2%})")
    print(f"Kept rule    : {set(parent['antecedents'])}  →  {set(parent['consequents'])} "
          f"(confidence = {parent['confidence']:.2%})")
    print("   Reason     : the kept rule is more general and has equal or higher confidence.\n")
```

There are no redundant rules

Task 6: Lift vs Confidence Tradeoff: Select three rules with

o High lift but low confidence, o High confidence but low lift, and o Both high lift and high confidence. Compare and explain which rule is more actionable for a business, and why.

In [128...

```
three_rules = rules [["antecedents", "consequents", "confidence", "lift"]]
three_rules
```

Out[128...

	antecedents	consequents	confidence	lift
0	(Greeting Cards)	(Candy Bar)	0.297239	1.738191
1	(Candy Bar)	(Greeting Cards)	0.255314	1.738191
2	(Magazine)	(Candy Bar)	0.167982	0.982325
3	(Candy Bar)	(Magazine)	0.237040	0.982325
4	(Pencils)	(Candy Bar)	0.244691	1.430903
5	(Candy Bar)	(Pencils)	0.193065	1.430903
6	(Toothpaste)	(Candy Bar)	0.247966	1.450053
7	(Candy Bar)	(Toothpaste)	0.232625	1.450053
8	(Greeting Cards)	(Magazine)	0.247370	1.025136
9	(Magazine)	(Greeting Cards)	0.150577	1.025136
10	(Greeting Cards)	(Pencils)	0.199067	1.475392
11	(Pencils)	(Greeting Cards)	0.216713	1.475392
12	(Greeting Cards)	(Toothpaste)	0.218402	1.361397
13	(Toothpaste)	(Greeting Cards)	0.199969	1.361397
14	(Magazine)	(Pencils)	0.131079	0.971495
15	(Pencils)	(Magazine)	0.234427	0.971495
16	(Toothpaste)	(Magazine)	0.197382	0.817977
17	(Magazine)	(Toothpaste)	0.131224	0.817977

In [129...

```
# Select a rule with high lift but low confidence
highlilowcf = three_rules.iloc[[11]]
"""Select the row having confidence < 1 - low confidence
and it is almost the highest lift """
highlilowcf
```

Out[129...

	antecedents	consequents	confidence	lift
11	(Pencils)	(Greeting Cards)	0.216713	1.475392

In [130…

```
# Select a rule with high confidence but low lift
highcflowli = three_rules.iloc[[14]]
"""Select the row having lift < 1 - low lift
and it is quite high lift """
highcflowli
```

Out[130…

	antecedents	consequents	confidence	lift
14	(Magazine)	(Pencils)	0.131079	0.971495

In [131…

```
# Select a rule with both high lift and high confidence
highcfhighli = three_rules.sort_values(["confidence", "lift"],
                     ascending=[False, False]).head(1)
highcfhighli
```

Out[131…

	antecedents	consequents	confidence	lift
0	(Greeting Cards)	(Candy Bar)	0.297239	1.738191

Case: High confidence, high lift: Rule 1 (Greeting Cards ⇒ Candy Bar) is clearly the best candidate.

High lift (1.74) ⇒ the relationship is meaningfully stronger than chance, so influencing it should produce incremental revenue.

Reasonable confidence (29 %) ⇒ almost one in three card buyers are primed to accept the cross-sell—large enough to matter.

Concrete merchandising play ⇒ cards and candy are both impulse items that fit naturally near checkout or as a bundled promotion.

Other cases: Rules 11 and 14 lack either lift or confidence (in fact, both), so acting on them would either have no impact or possibly misallocate shelf space.

Task 7: Unexpected Patterns

Identify one unexpected association (e.g., low support but high lift). Why might this rule occur, and how could it be validated or refuted using additional data?

Index Rule: Antecedent: (Greeting Cards) → Consequent: (Candy Bar) Support: 0.0437 Confidence: 0.2972 Lift: 1.738

This rule is unexpected as it links two distinct product categories:

1. Greeting Cards (stationery)
2. Candy bar (snack)

Although the support is only 4.37%, the lift of 1.738 indicates that customers who buy greeting cards are 73.8% more likely to purchase candy bars compared to a random selection of customers.

This significant correlation may result from:

1. Impulse buying behavior is exemplified by the strategic placement of greeting cards and candy bars near checkout counters, which facilitates last-minute purchases.

2. In the context of gift-giving, a customer purchasing a card, such as for a birthday or to express gratitude, may additionally include a small candy gift.

Task 8: Rule Filtering by Length: Filter and display rules with

Antecedents of length ≥ 2, Consequents of length = 1, What kinds of products tend to co-occur in larger baskets? Are they complementary or substitutable?

In [136…

```
# Case 1: Antecedents of length ≥ 2
rules['antecedent_len'] = rules['antecedents'].apply(lambda x: len(x))
long_antecedents = rules[rules['antecedent_len'] >= 2]
print(f"Rules with antecedents ≥ 2: {len(long_antecedents)}")
long_antecedents[['antecedents', 'consequents', 'support', 'confidence', 'lift']]
```

```
Rules with antecedents ≥ 2: 0
```

Out[136…

antecedents	consequents	support	confidence	lift

In [137…

```
rules['consequent_len'] = rules['consequents'].apply(lambda x: len(x))
single_consequents = rules[rules['consequent_len'] == 1]
print(f"Rules with single-item consequents: {len(single_consequents)}")
single_consequents[['antecedents', 'consequents', 'support', 'confidence', 'lift']]
```

```
Rules with single-item consequents: 18
```

Out[137...

	antecedents	consequents	support	confidence	lift
0	(Greeting Cards)	(Candy Bar)	0.043660	0.297239	1.738191
1	(Candy Bar)	(Greeting Cards)	0.043660	0.255314	1.738191
2	(Magazine)	(Candy Bar)	0.040535	0.167982	0.982325
3	(Candy Bar)	(Magazine)	0.040535	0.237040	0.982325
4	(Pencils)	(Candy Bar)	0.033015	0.244691	1.430903
5	(Candy Bar)	(Pencils)	0.033015	0.193065	1.430903
6	(Toothpaste)	(Candy Bar)	0.039780	0.247966	1.450053
7	(Candy Bar)	(Toothpaste)	0.039780	0.232625	1.450053
8	(Greeting Cards)	(Magazine)	0.036335	0.247370	1.025136
9	(Magazine)	(Greeting Cards)	0.036335	0.150577	1.025136
10	(Greeting Cards)	(Pencils)	0.029240	0.199067	1.475392
11	(Pencils)	(Greeting Cards)	0.029240	0.216713	1.475392
12	(Greeting Cards)	(Toothpaste)	0.032080	0.218402	1.361397
13	(Toothpaste)	(Greeting Cards)	0.032080	0.199969	1.361397
14	(Magazine)	(Pencils)	0.031630	0.131079	0.971495
15	(Pencils)	(Magazine)	0.031630	0.234427	0.971495
16	(Toothpaste)	(Magazine)	0.031665	0.197382	0.817977
17	(Magazine)	(Toothpaste)	0.031665	0.131224	0.817977

This analysis applied association rule mining to uncover purchase patterns within a retail transaction dataset. Using initial thresholds of min_support = 0.025 and confidence = 0.06, the model successfully generated 18 rules with single-item consequents, highlighting item pairs with relatively strong associations—such as Candy Bars with Greeting Cards, Pencils, and Toothpaste. However, no rules met the criteria of having antecedents with two or more items, suggesting that complex itemsets either did not occur frequently enough or were filtered out by the chosen thresholds. This indicates a need for parameter tuning to explore deeper patterns.

The current results primarily reflect complementary item relationships, which are valuable for designing promotions and layout strategies. To reveal higher-order combinations or more actionable basket insights, future analysis should involve lowering the support threshold and exploring transaction length distributions. Overall, while simple product associations are evident, further refinement of model settings is necessary to fully capture the structure of customer purchasing behavior.

Task 9. Rule-Based Recommendation System

Choose any item as a target product (e.g., " Candy Bar"). Use association rules to determine what products should be recommended before or after purchasing this item. Justify the recommendation logic based on lift and confidence.?

In [140…
```
#Choose any item as a target product (e.g., "Candy Bar")
# Note: Magazine was chosen as the target produce
target_product = "Magazine"
rules_target = rules[(rules['antecedents'].apply(lambda x: target_product in x)) |
                     (rules['consequents'].apply(lambda x: target_product in x))]
```

In [141…
```
# Use association rules to determine recommendations
# Sorted by lift
rules_target = rules_target.sort_values(by="lift", ascending=False)
```

In [142…
```
# Justify the recommendation logic based on lift and confidence
rules_target
```

Out[142…

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	representativity	leverage	conviction
8	(Greeting Cards)	(Magazine)	0.146885	0.241305	0.036335	0.247370	1.025136	1.0	0.000891	1.008059
9	(Magazine)	(Greeting Cards)	0.241305	0.146885	0.036335	0.150577	1.025136	1.0	0.000891	1.004347
2	(Magazine)	(Candy Bar)	0.241305	0.171005	0.040535	0.167982	0.982325	1.0	-0.000729	0.996367
3	(Candy Bar)	(Magazine)	0.171005	0.241305	0.040535	0.237040	0.982325	1.0	-0.000729	0.994410
15	(Pencils)	(Magazine)	0.134925	0.241305	0.031630	0.234427	0.971495	1.0	-0.000928	0.991015
14	(Magazine)	(Pencils)	0.241305	0.134925	0.031630	0.131079	0.971495	1.0	-0.000928	0.995574
16	(Toothpaste)	(Magazine)	0.160425	0.241305	0.031665	0.197382	0.817977	1.0	-0.007046	0.945275
17	(Magazine)	(Toothpaste)	0.241305	0.160425	0.031665	0.131224	0.817977	1.0	-0.007046	0.966388

In [143…

```python
# Products bought BEFORE 'Magazine'
before_magazine = rules_target[rules_target['consequents'].apply(lambda x: target_product in x)]

# Products bought AFTER 'Magazine'
after_magazine = rules_target[rules_target['antecedents'].apply(lambda x: target_product in x)]

print("\nProducts to recommend BEFORE buying Magazine:")
print(before_magazine[['antecedents', 'consequents', 'support', 'confidence', 'lift']])

print("\nProducts to recommend AFTER buying Magazine:")
print(after_magazine[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```

```
Products to recommend BEFORE buying Magazine:
         antecedents consequents   support  confidence      lift
8   (Greeting Cards)  (Magazine)  0.036335    0.247370  1.025136
3        (Candy Bar)  (Magazine)  0.040535    0.237040  0.982325
15         (Pencils)  (Magazine)  0.031630    0.234427  0.971495
16      (Toothpaste)  (Magazine)  0.031665    0.197382  0.817977

Products to recommend AFTER buying Magazine:
   antecedents       consequents   support  confidence      lift
9   (Magazine)  (Greeting Cards)  0.036335    0.150577  1.025136
2   (Magazine)       (Candy Bar)  0.040535    0.167982  0.982325
14  (Magazine)         (Pencils)  0.031630    0.131079  0.971495
17  (Magazine)      (Toothpaste)  0.031665    0.131224  0.817977
```

The justification logic is based on both confidence and lift. The higher the confidence level the more likely it is that when the customer buys the antecedent product they also buy the consequent products. The recommendation is also based on lift which revolves around the likelihood of the consequent based on the antecedents.

Using the association rules before purchasing magazines, greeting cards should be recommended. It has the highest lift of 1.025136 which means this transaction together is about 1.025 compared to a random transaction. It also has a confidence of 0.247370 which means if a customer buys a greeting card there is about 24.73% chance that they also purchase a magazine.

After buying magazines, greeting cards should also be recommended. This is because it has the highest lift of 1.025136. Also, it has a confidence of 0.150577. This means that when people buy a magazine there is about a 15.05% chance that they also purchase greeting cards.

Task 10. Business Decision-Making

Based on the rules generated, list five specific and data-supported business actions your team would recommend (e.g., cross-promotion, layout strategy, bundling offers, inventory stocking).

Based on the rules generated, below are five specific business actions that the team would recommend based on the data:

Strategy 1: Improved Product Placement Products that have a higher lift value should be placed near each other, to encourage the purchase of those items together. For example, Rule 9 has the antecedent as greeting cards and the consequent as magazines with a lift value of 1.025136. Since this is a higher lift value these items should be placed near each other, perhaps close to the checkout aisle, encouraging cusotmers to purcahse the items together,

Strategy 2: Discounts on Bundles Create a bundle of items that are oftentime bought together and then offer a discount on the bundle. Since it will be cheaper for the bundle compared to buying all the items individually, this will help gain traction and encourage sales. For example Rule 3 has the antecedent as managazine and consequent as candy bar, with a confidence of 0.167982 and a life of 0.982325.Therefore magazines and candy bars can be bundled together and sold at a discount.

Strategy 3: Higher Stock for Often Purchased Items Based on confidence levels, items that are purchased more often should have higher levels of stock. This is to avoid the risk of stock outs and to ensure that customers can always find what they're looking for, which will boost sales and help avoid customers from shopping at competitiors. For example Rule 2 has candy bar as the antecedent and magazine as the consequent, with a confidence of 0.237040. Based on this confidecen level, the team recommends that these two items have higher inventory levels to ensure they do not stock out.

Strategy 4: Shopping Challenge Customers will have the opportunity to be entered for a raffle and have a chance to win a prize if they purchase a certain set of items. For example, Rule 14 has mangazines as the antecedent and pencils as the consequent with a confidence of 0.131079 Therefore if customers purchase these two items in the same transaction then they can be entered into the raffle.

Strategy 5: Promotion Through In Store Signage In store signage can promote items together which can help encourage consumers to purcahse the items together. For example, Rule 2 has the antecedent as candy bar and the consequent as magazine. Therefore the in store signage can promote these two products together and relate them to a care basket or related idea.

Group Work

We, Jamaica Vee Buduan, Jenelle Guerrero Martinez, Joshua Kevin Jonathan, Julie Pham, declare that the attached assignment is our own work in accordance with the Seneca Academic Policy. We have not copied any part of this assignment, manually or electronically, from any other source including web sites, unless specified as references. We have not distributed our work to other students.