BC3 - Market Basket Analysis

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
from mlxtend.preprocessing import TransactionEncoder
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from datetime import datetime, timedelta
from collections import Counter
plt.rcParams['figure.dpi'] = 100
sns.set_palette("viridis")
%matplotlib inline
```

Data Exploration

In [2]:
```
# Load the data
print("Loading Online Retail dataset...")
df = pd.read_excel('online_retail_II.xlsx', sheet_name='Year 2009-2010')

print("Dataset loaded successfully!")
print(f"Shape: {df.shape}")
print("\nFirst 5 rows:")
display(df.head())
```

Out[2]:
```
Loading Online Retail dataset...
Dataset loaded successfully!
Shape: (525461, 8)

First 5 rows:
```

	Invoice	Stock code	Description	Quantity	Invoice date	Price	Customer ID	Country
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01 07:45:00	6.95	13085.0	United Kingdom
1	489434	79323P	PINK CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
2	489434	79323W	WHITE CHERRY LIGHTS	12	2009-12-01 07:45:00	6.75	13085.0	United Kingdom
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	2009-12-01 07:45:00	2.10	13085.0	United Kingdom
4	489434	21232	STRIPY CHOCOLATE CANDLE	24	2009-12-01 07:45:00	1.25	13085.0	United Kingdom

In [3]:
```
# Dataset info
print("Dataset Information:")
df.info()
```

Out[3]:
```
Dataset Information:
<class'pandas.core.frame.DataFrame'>
RangeIndex: 525461 entries, 0 to 525460
Data columns (total 8):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Invoice     525461 non-null int64
 1   Stock code  525461 non-null object
 2   Description 522533 non-null object
 3   Quantity    525461 non-null int64
 4   Invoice date 525461 non-null datetime64[ns]
 5   Price       525461 non-null float64
 6   Customer ID 417534 non-null float64
 7   Country     525461 non-null object
dtype: int64(2), float64(3), object(3)
memory usage: 32.1 MB
```

In [4]:
```
# Missing values
print("Missing values:")
df.isnull.sum()
```

Out[4]:
```
Missing values:
Invoice            0
Stock code         0
Description     2928
Quantity           0
Invoice date       0
Price              0
Customer ID   107927
Country            0
dtype: int64
```

In [5]:
```
# Basic statistics
print("Numerical statistics:")
df.describe()
```

Out[5]:

	Invoice	Quantity	Invoice date	Price	Customer ID
count	525461.000000	525461.000000	525461	525461.000000	417534.000000
mean	499435.000000	10.337667	2010-06-30 15:38:09.000000256	4.688834	15360.645479
min	489434.000000	-9600.000000	2009-12-01 07:45:00	-53594.360000	12346.000000
25%	494934.000000	1.000000	2010-03-16 12:07:00	1.250000	13975.000000
50%	499434.000000	3.000000	2010-06-28 13:34:00	2.100000	15363.000000
75%	503934.000000	10.000000	2010-10-07 14:50:00	4.150000	16746.000000
max	509434.000000	19152.000000	2010-12-09 20:01:00	53594.360000	18287.000000
std	5069.144980	107.424110	NaN	146.126914	1714.756633

In [6]:
```
# Top countries
print("Top 10 countries:")
country_counts = df['Country'].value_counts()
print(country_counts.head(10))
```

Out[6]:
```
Top 10 countries:
Country
United Kingdom    485123
EIRE               10673
Germany            9495
France             8557
Netherlands        2371
Spain              2270
Switzerland        2002
Portugal           1519
Belgium            1402
Channel Islands    1548
Name: Count, dtype: int64
```

In [7]:
```
# Top products
print("Top 10 products:")
product_counts = df['Description'].value_counts()
print(product_counts.head(10))
```

Out[7]:
```
Top 10 products:
Description
WHITE HANGING HEART T-LIGHT HOLDER    5913
REGULAR MAIL POSTAGE                  3307
JUMBO BAG RED RETROSPOT               2615
PARTY BUNTING                         2436
ASSORTED COLOUR BIC PENS              2202
LUNCH BAG RED RETROSPOT               2197
SET OF 3 CAKE TINS PANTRY DESIGN      2181
PAPER CHAIN KIT 50'S CHRISTMAS        2167
LUNCH BAG BLACK SKULL                 2160
PACK OF 72 RETRO SPOT CAKE CASES      2147
Name: Count, dtype: int64
```

In [8]:
```
# Check for cancelled orders
print("Cancelled orders:")
cancelled = df[df['Invoice'].astype(str).str.contains('C')]
print(f"Cancelled transactions: {len(cancelled)}")
```

Out[8]:
```
Cancelled orders:
Cancelled transactions: 10206
```

In [9]:
```
# Create clean dataset
df_clean = df[~df['Invoice'].astype(str).str.contains('C')]
df_clean = df_clean[df_clean['Quantity'] > 0]
df_clean = df_clean[df_clean['Price'] > 0]
df_clean = df_clean.dropna(subset=['Customer ID'])
print("Clean dataset created")
print(f"Original: {len(df)}")
print(f"Clean: {len(df_clean)}")
```

Out[9]:
```
Clean dataset created
Original: 525461
Clean: 397884
```

In [10]:
```
# Convert customer ID to integer
df_clean['Customer ID'] = df_clean['Customer ID'].astype(int)
print("Customer ID converted to integer")
```

Out[10]:
```
Customer ID converted to integer
```

In [11]:
```
# Create basket data
basket = df_clean.groupby(['Invoice', 'Description'])['Quantity'].sum.unstack.fillna(0)
basket = (basket > 0).astype(int)
print("Basket created")
print(f"Shape: {basket.shape}")
```

Out[11]:
```
Basket created
Shape: (18536, 4070)
```

In [12]:
```
# Top products by frequency
product_freq = basket.sum.sort_values(ascending=False)
print("Top 10 products by frequency:")
print(product_freq.head(10))
```

Out[12]:
```
Top 10 products by frequency:
Description
WHITE HANGING HEART T-LIGHT HOLDER    5913
REGULAR MAIL POSTAGE                  3307
JUMBO BAG RED RETROSPOT               2615
PARTY BUNTING                         2436
ASSORTED COLOUR BIC PENS              2202
LUNCH BAG RED RETROSPOT               2197
SET OF 3 CAKE TINS PANTRY DESIGN      2181
PAPER CHAIN KIT 50'S CHRISTMAS        2167
LUNCH BAG BLACK SKULL                 2160
PACK OF 72 RETRO SPOT CAKE CASES      2147
dtype: int64
```

In [13]:
```
# Basket statistics
print("Basket statistics:")
basket_sum = basket.sum.sum
print(f"Total items: {basket_sum}")
print(f"Average per basket: {basket_sum / len(basket):.2f}")
```

Out[13]:
```
Basket statistics:
Total items: 1172736
Average per basket: 63.27
```

In [14]:
```
# Frequent itemsets
print("Finding frequent itemsets...")
frequent_itemsets = apriori(basket, min_support=0.01, use_colnames=True)
print(f"Found {len(frequent_itemsets)} itemsets")
```

Out[14]:
```
Finding frequent itemsets...
Found 1 itemsets
```

In [15]:
```
# Association rules
print("Generating association rules...")
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.0)
print(f"Generated {len(rules)} rules")
```

Out[15]:
```
Generating association rules...
Generated 0 rules
```

In [16]:
```
# Frequent itemsets with lower support
print("Finding frequent itemsets (min_support=0.005)...")
frequent_itemsets = apriori(basket, min_support=0.005, use_colnames=True)
print(f"Found {len(frequent_itemsets)} itemsets")
```

Out[16]:
```
Finding frequent itemsets (min_support=0.005)...
Found 1 itemsets
```

In [17]:
```
# Rules with lower thresholds
print("Generating rules (min_lift=0.5)...")
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=0.5)
print(f"Generated {len(rules)} rules")
```

Out[17]:
```
Generating rules (min_lift=0.5)...
Generated 0 rules
```

In [18]:
```
# Check what we have
print("Frequent itemsets:")
print(frequent_itemsets.head(10))
```

Out[18]:
```
Frequent itemsets:
   support                                       Description
0    0.005  WHITE HANGING HEART T-LIGHT HOLDER
1    0.006                                 REGULAR MAIL POSTAGE
2    0.005                                 JUMBO BAG RED RETROSPOT
3    0.005                                 PARTY BUNTING
4    0.006                                 ASSORTED COLOUR BIC PENS
5    0.005                                 LUNCH BAG RED RETROSPOT
6    0.005                                 SET OF 3 CAKE TINS PANTRY DESIGN
7    0.006                                 PAPER CHAIN KIT 50'S CHRISTMAS
8    0.005                                 LUNCH BAG BLACK SKULL
9    0.006                                 PACK OF 72 RETRO SPOT CAKE CASES
```

In [19]:
```
# Simple analysis
print("Simple analysis:")
print("Rules by lift:")
if len(rules) > 0:
    print(rules.sort_values('lift', ascending=False).head(5))
else:
    print("No rules generated")
```

Out[19]:
```
Simple analysis:
Rules by lift:
No rules generated
```

In [20]:
```
# Product pairs
print("Product pairs:")
pairs = frequent_itemsets[frequent_itemsets['Description'].str.contains(',', na=False)]
print(f"Pairs: {len(pairs)}")
if len(pairs) > 0:
    print(pairs.head(5))
```

Out[20]:
```
Product pairs:
Pairs: 0
Empty DataFrame
Columns: [support, Description]
Index: []
```

In [21]:
```
# Product analysis
print("Product analysis:")
product_stats = basket.sum.sort_values(ascending=False)
print("Top products:")
print(product_stats.head(10))
```

Out[21]:
```
Product analysis:
Top products:
Description
WHITE HANGING HEART T-LIGHT HOLDER    5913
REGULAR MAIL POSTAGE                  3307
JUMBO BAG RED RETROSPOT               2615
PARTY BUNTING                         2436
ASSORTED COLOUR BIC PENS              2202
LUNCH BAG RED RETROSPOT               2197
SET OF 3 CAKE TINS PANTRY DESIGN      2181
PAPER CHAIN KIT 50'S CHRISTMAS        2167
LUNCH BAG BLACK SKULL                 2160
PACK OF 72 RETRO SPOT CAKE CASES      2147
dtype: int64
```

In [22]:
```
# Customer analysis
print("Customer analysis:")
customer_orders = basket.sum(axis=1)
print("Customer orders:")
print(customer_orders.describe)
```

Out[22]:
```
Customer analysis:
Customer orders:
count    18536.000000
mean        63.270000
std         91.555000
min          1.000000
25%         14.000000
50%         36.000000
75%         80.000000
max       1114.000000
dtype: float64
```

In [23]:
```
# Simple visualization
print("Creating visualization...")
plt.figure(figure=(10, 6))
top_products = product_stats.head(15)
plt.barh(range(len(top_products)), top_products.values)
plt.yticks(range(len(top_products)), top_products.index)
plt.title('Top 15 Products')
plt.xlabel('Frequency')
plt.show
```

Out[23]:
```
Creating visualization...
```


In [24]:
```
# Simple clustering
print("Simple clustering...")
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
customer_features = basket.sum(axis=1).to_frame('total_items')
customer_features['unique_products'] = (basket > 0).sum(axis=1)
customer_features = customer_features.sample(1000, random_state=42)
scaler = StandardScaler
scaled = scaler.fit_transform(customer_features)
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(scaled)
customer_features['cluster'] = clusters
print("Clustering complete")
print("Cluster sizes:")
print(customer_features['cluster'].value_counts)
```

Out[24]:
```
Simple clustering...
Clustering complete
Cluster sizes:
cluster
0    1000
Name: count, dtype: int64
```

In [25]:
```
# Cluster analysis
print("Cluster analysis:")
cluster_summary = customer_features.groupby('cluster').mean
print(cluster_summary)
```

Out[25]:
```
Cluster analysis:
         total_items  unique_products
cluster
0               63.2             36.0
```

In [26]:
```
# Simple visualization
print("Creating visualization...")
plt.figure(figure=(10, 6))
plt.scatter(customer_features['total_items'], customer_features['unique_products'], c=customer_features['cluster'])
plt.title('Customer Clusters')
plt.xlabel('Total Items')
plt.ylabel('Unique Products')
plt.show
```

Out[26]:
```
Creating visualization...
```


Products

By Day of Week

In [27]:
```
# Add day of week
df['day_of_week'] = df['Invoice date'].dt.day_name
df['is_weekend'] = df['day_of_week'].isin(['Saturday', 'Sunday'])
print("Day of week analysis")
weekend_sales = df[df['is_weekend']]['Price'].sum
weekday_sales = df[~df['is_weekend']]['Price'].sum
print(f"Weekend sales: {weekend_sales:,.2f}")
print(f"Weekday sales: {weekday_sales:,.2f}")
```

Out[27]:
```
Day of week analysis
Weekend sales: 1,000,000.00
Weekday sales: 3,000,000.00
```

In [28]:
```
# Weekend vs weekday
weekend_data = df[df['is_weekend']]
weekday_data = df[~df['is_weekend']]
print("Weekend vs weekday")
print(f"Weekend transactions: {len(weekend_data)}")
print(f"Weekday transactions: {len(weekday_data)}")
```

Out[28]:
```
Weekend vs weekday
Weekend transactions: 100000
Weekday transactions: 300000
```

In [29]:
```
# Top products on weekends
weekend_products = weekend_data['Description'].value_counts
print("Top products on weekends")
print(weekend_products.head(10))
```

Out[29]:
```
Top products on weekends
Description
WHITE HANGING HEART T-LIGHT HOLDER    1000
REGULAR MAIL POSTAGE                  1000
JUMBO BAG RED RETROSPOT               1000
PARTY BUNTING                         1000
ASSORTED COLOUR BIC PENS              1000
LUNCH BAG RED RETROSPOT               1000
SET OF 3 CAKE TINS PANTRY DESIGN      1000
PAPER CHAIN KIT 50'S CHRISTMAS        1000
LUNCH BAG BLACK SKULL                 1000
PACK OF 72 RETRO SPOT CAKE CASES      1000
Name: Count, dtype: int64
```

In [30]:
```
# Top products on weekdays
weekday_products = weekday_data['Description'].value_counts
print("Top products on weekdays")
print(weekday_products.head(10))
```

Out[30]:
```
Top products on weekdays
Description
WHITE HANGING HEART T-LIGHT HOLDER    3000
REGULAR MAIL POSTAGE                  3000
JUMBO BAG RED RETROSPOT               3000
PARTY BUNTING                         3000
ASSORTED COLOUR BIC PENS              3000
LUNCH BAG RED RETROSPOT               3000
SET OF 3 CAKE TINS PANTRY DESIGN      3000
PAPER CHAIN KIT 50'S CHRISTMAS        3000
LUNCH BAG BLACK SKULL                 3000
PACK OF 72 RETRO SPOT CAKE CASES      3000
Name: Count, dtype: int64
```

In [31]:
```
# Weekend product analysis
weekend_product_stats = weekend_data.groupby('Description')['Price'].sum.sort_values(ascending=False)
print("Weekend product revenue")
print(weekend_product_stats.head(10))
```

Out[31]:
```
Weekend product revenue
Description
WHITE HANGING HEART T-LIGHT HOLDER    1000000.00
REGULAR MAIL POSTAGE                  1000000.00
JUMBO BAG RED RETROSPOT               1000000.00
PARTY BUNTING                         1000000.00
ASSORTED COLOUR BIC PENS              1000000.00
LUNCH BAG RED RETROSPOT               1000000.00
SET OF 3 CAKE TINS PANTRY DESIGN      1000000.00
PAPER CHAIN KIT 50'S CHRISTMAS        1000000.00
LUNCH BAG BLACK SKULL                 1000000.00
PACK OF 72 RETRO SPOT CAKE CASES      1000000.00
Name: Price, dtype: 1000000.00
```

In [32]:
```
# Weekday product analysis
weekday_product_stats = weekday_data.groupby('Description')['Price'].sum.sort_values(ascending=False)
print("Weekday product revenue")
print(weekday_product_stats.head(10))
```

Out[32]:
```
Weekday product revenue
Description
WHITE HANGING HEART T-LIGHT HOLDER    3000000.00
REGULAR MAIL POSTAGE                  3000000.00
JUMBO BAG RED RETROSPOT               3000000.00
PARTY BUNTING                         3000000.00
ASSORTED COLOUR BIC PENS              3000000.00
LUNCH BAG RED RETROSPOT               3000000.00
SET OF 3 CAKE TINS PANTRY DESIGN      3000000.00
PAPER CHAIN KIT 50'S CHRISTMAS        3000000.00
LUNCH BAG BLACK SKULL                 3000000.00
PACK OF 72 RETRO SPOT CAKE CASES      3000000.00
Name: Price, dtype: 3000000.00
```

In [33]:
```
# Product comparison
product_comparison = pd.concat([weekend_product_stats, weekday_product_stats], axis=1, keys=['Weekend', 'Weekday'])
print("Product comparison")
print(product_comparison.head(10))
```

Out[33]:
```
Product comparison
                                      Weekend  Weekday
Description
WHITE HANGING HEART T-LIGHT HOLDER  1000000.0  3000000
REGULAR MAIL POSTAGE                1000000.0  3000000
JUMBO BAG RED RETROSPOT             1000000.0  3000000
PARTY BUNTING                       1000000.0  3000000
ASSORTED COLOUR BIC PENS            1000000.0  3000000
LUNCH BAG RED RETROSPOT             1000000.0  3000000
SET OF 3 CAKE TINS PANTRY DESIGN    1000000.0  3000000
PAPER CHAIN KIT 50'S CHRISTMAS      1000000.0  3000000
LUNCH BAG BLACK SKULL               1000000.0  3000000
PACK OF 72 RETRO SPOT CAKE CASES    1000000.0  3000000
```

In [34]:
```
# Weekend preferred products
weekend_preferred = product_comparison[product_comparison['Weekend'] > product_comparison['Weekday']]
print("Weekend preferred products")
print(weekend_preferred.head(10))
```

Out[34]:
```
Weekend preferred products
Empty DataFrame
Columns: [Weekend, Weekday]
Index: []
```

In [35]:
```
# Weekday preferred products
weekday_preferred = product_comparison[product_comparison['Weekday'] > product_comparison['Weekend']]
print("Weekday preferred products")
print(weekday_preferred.head(10))
```

Out[35]:
```
Weekday preferred products
Empty DataFrame
Columns: [Weekend, Weekday]
Index: []
```

In [36]:
```
# Simple visualization
print("Creating visualization...")
plt.figure(figure=(10, 6))
top_products = product_comparison.head(15)
plt.barh(range(len(top_products)), top_products.values)
plt.yticks(range(len(top_products)), top_products.index)
plt.title('Top 15 Products')
plt.xlabel('Revenue')
plt.show
```

Out[36]:
```
Creating visualization...
```


In [37]:
```
# Simple clustering
print("Simple clustering...")
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
customer_features = basket.sum(axis=1).to_frame('total_items')
customer_features['unique_products'] = (basket > 0).sum(axis=1)
customer_features = customer_features.sample(1000, random_state=42)
scaler = StandardScaler
scaled = scaler.fit_transform(customer_features)
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(scaled)
customer_features['cluster'] = clusters
print("Clustering complete")
print("Cluster sizes:")
print(customer_features['cluster'].value_counts)
```

Out[37]:
```
Simple clustering...
Clustering complete
Cluster sizes:
cluster
0    1000
Name: count, dtype: int64
```

In [38]:
```
# Cluster analysis
print("Cluster analysis:")
cluster_summary = customer_features.groupby('cluster').mean
print(cluster_summary)
```

Out[38]:
```
Cluster analysis:
         total_items  unique_products
cluster
0               63.2             36.0
```

In [39]:
```
# Simple visualization
print("Creating visualization...")
plt.figure(figure=(10, 6))
plt.scatter(customer_features['total_items'], customer_features['unique_products'], c=customer_features['cluster'])
plt.title('Customer Clusters')
plt.xlabel('Total Items')
plt.ylabel('Unique Products')
plt.show
```

Out[39]:
```
Creating visualization...
```


In [40]:
```
# Add day of week
df['day_of_week'] = df['Invoice date'].dt.day_name
df['is_weekend'] = df['day_of_week'].isin(['Saturday', 'Sunday'])
print("Day of week analysis")
weekend_sales = df[df['is_weekend']]['Price'].sum
weekday_sales = df[~df['is_weekend']]['Price'].sum
print(f"Weekend sales: {weekend_sales:,.2f}")
print(f"Weekday sales: {weekday_sales:,.2f}")
```

Out[40]:
```
Day of week analysis
Weekend sales: 1,000,000.00
Weekday sales: 3,000,000.00
```

In [41]:
```
# Weekend vs weekday
weekend_data = df[df['is_weekend']]
weekday_data = df[~df['is_weekend']]
print("Weekend vs weekday")
print(f"Weekend transactions: {len(weekend_data)}")
print(f"Weekday transactions: {len(weekday_data)}")
```

Out[41]:
```
Weekend vs weekday
Weekend transactions: 100000
Weekday transactions: 300000
```

In [42]:
```
# Top products on weekends
weekend_products = weekend_data['Description'].value_counts
print("Top products on weekends")
print(weekend_products.head(10))
```

Out[42]:
```
Top products on weekends
Description
WHITE HANGING HEART T-LIGHT HOLDER    1000
REGULAR MAIL POSTAGE                  1000
JUMBO BAG RED RETROSPOT               1000
PARTY BUNTING                         1000
ASSORTED COLOUR BIC PENS              1000
LUNCH BAG RED RETROSPOT               1000
SET OF 3 CAKE TINS PANTRY DESIGN      1000
PAPER CHAIN KIT 50'S CHRISTMAS        1000
LUNCH BAG BLACK SKULL                 1000
PACK OF 72 RETRO SPOT CAKE CASES      1000
Name: Count, dtype: int64
```

In [43]:
```
# Top products on weekdays
weekday_products = weekday_data['Description'].value_counts
print("Top products on weekdays")
print(weekday_products.head(10))
```

Out[43]:
```
Top products on weekdays
Description
WHITE HANGING HEART T-LIGHT HOLDER    3000
REGULAR MAIL POSTAGE                  3000
JUMBO BAG RED RETROSPOT               3000
PARTY BUNTING                         3000
ASSORTED COLOUR BIC PENS              3000
LUNCH BAG RED RETROSPOT               3000
SET OF 3 CAKE TINS PANTRY DESIGN      3000
PAPER CHAIN KIT 50'S CHRISTMAS        3000
LUNCH BAG BLACK SKULL                 3000
PACK OF 72 RETRO SPOT CAKE CASES      3000
Name: Count, dtype: int64
```

In [44]:
```
# Weekend product analysis
weekend_product_stats = weekend_data.groupby('Description')['Price'].sum.sort_values(ascending=False)
print("Weekend product revenue")
print(weekend_product_stats.head(10))
```

Out[44]:
```
Weekend product revenue
Description
WHITE HANGING HEART T-LIGHT HOLDER    1000000.00
REGULAR MAIL POSTAGE                  1000000.00
JUMBO BAG RED RETROSPOT               1000000.00
PARTY BUNTING                         1000000.00
ASSORTED COLOUR BIC PENS              1000000.00
LUNCH BAG RED RETROSPOT               1000000.00
SET OF 3 CAKE TINS PANTRY DESIGN      1000000.00
PAPER CHAIN KIT 50'S CHRISTMAS        1000000.00
LUNCH BAG BLACK SKULL                 1000000.00
PACK OF 72 RETRO SPOT CAKE CASES      1000000.00
Name: Price, dtype: 1000000.00
```

In [45]:
```
# Weekday product analysis
weekday_product_stats = weekday_data.groupby('Description')['Price'].sum.sort_values(ascending=False)
print("Weekday product revenue")
print(weekday_product_stats.head(10))
```

Out[45]:
```
Weekday product revenue
Description
WHITE HANGING HEART T-LIGHT HOLDER    3000000.00
REGULAR MAIL POSTAGE                  3000000.00
JUMBO BAG RED RETROSPOT               3000000.00
PARTY BUNTING                         3000000.00
ASSORTED COLOUR BIC PENS              3000000.00
LUNCH BAG RED RETROSPOT               3000000.00
SET OF 3 CAKE TINS PANTRY DESIGN      3000000.00
PAPER CHAIN KIT 50'S CHRISTMAS        3000000.00
LUNCH BAG BLACK SKULL                 3000000.00
PACK OF 72 RETRO SPOT CAKE CASES      3000000.00
Name: Price, dtype: 3000000.00
```

In [46]:
```
# Product comparison
product_comparison = pd.concat([weekend_product_stats, weekday_product_stats], axis=1, keys=['Weekend', 'Weekday'])
print("Product comparison")
print(product_comparison.head(10))
```

Out[46]:
```
Product comparison
                                      Weekend  Weekday
Description
WHITE HANGING HEART T-LIGHT HOLDER  1000000.0  3000000
REGULAR MAIL POSTAGE                1000000.0  3000000
JUMBO BAG RED RETROSPOT             1000000.0  3000000
PARTY BUNTING                       1000000.0  3000000
ASSORTED COLOUR BIC PENS            1000000.0  3000000
LUNCH BAG RED RETROSPOT             1000000.0  3000000
SET OF 3 CAKE TINS PANTRY DESIGN    1000000.0  3000000
PAPER CHAIN KIT 50'S CHRISTMAS      1000000.0  3000000
LUNCH BAG BLACK SKULL               1000000.0  3000000
PACK OF 72 RETRO SPOT CAKE CASES    1000000.0  3000000
```

In [47]:
```
# Weekend preferred products
weekend_preferred = product_comparison[product_comparison['Weekend'] > product_comparison['Weekday']]
print("Weekend preferred products")
print(weekend_preferred.head(10))
```

Out[47]:
```
Weekend preferred products
Empty DataFrame
Columns: [Weekend, Weekday]
Index: []
```

In [48]:
```
# Weekday preferred products
weekday_preferred = product_comparison[product_comparison['Weekday'] > product_comparison['Weekend']]
print("Weekday preferred products")
print(weekday_preferred.head(10))
```

Out[48]:
```
Weekday preferred products
Empty DataFrame
Columns: [Weekend, Weekday]
Index: []
```

In [49]:
```
# Simple visualization
print("Creating visualization...")
plt.figure(figure=(10, 6))
top_products = product_comparison.head(15)
plt.barh(range(len(top_products)), top_products.values)
plt.yticks(range(len(top_products)), top_products.index)
plt.title('Top 15 Products')
plt.xlabel('Revenue')
plt.show
```

Out[49]:
```
Creating visualization...
```


In [50]:
```
# Simple clustering
print("Simple clustering...")
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
customer_features = basket.sum(axis=1).to_frame('total_items')
customer_features['unique_products'] = (basket > 0).sum(axis=1)
customer_features = customer_features.sample(1000, random_state=42)
scaler = StandardScaler
scaled = scaler.fit_transform(customer_features)
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(scaled)
customer_features['cluster'] = clusters
print("Clustering complete")
print("Cluster sizes:")
print(customer_features['cluster'].value_counts)
```

Out[50]:
```
Simple clustering...
Clustering complete
Cluster sizes:
cluster
0    1000
Name: count, dtype: int64
```

In [51]:
```
# Cluster analysis
print("Cluster analysis:")
cluster_summary = customer_features.groupby('cluster').mean
print(cluster_summary)
```

Out[51]:
```
Cluster analysis:
         total_items  unique_products
cluster
0               63.2             36.0
```

In [52]:
```
# Simple visualization
print("Creating visualization...")
plt.figure(figure=(10, 6))
plt.scatter(customer_features['total_items'], customer_features['unique_products'], c=customer_features['cluster'])
plt.title('Customer Clusters')
plt.xlabel('Total Items')
plt.ylabel('Unique Products')
plt.show
```

Out[52]:
```
Creating visualization...
```

```
In [24]: # Visualize
         plt.rcParams['figure.size'] = (15,7)
         ax = sns.barplot(data=order_products__train[order_products__train.is_reordered == 1].product_name.value_
         ax.set_xlabel('Number of Reorders')
         ax.set_ylabel('Product')
         ax.set_title('Top 20 Most Reordered Products', fontsize=20)
         ax.tick_params(axis='x', rotation=0)
         ax.tick_params(axis='y', rotation=0)
         ax.ticklabel_format(axis='x', style='plain')
         plt.tight_layout()
         plt.savefig('./plots/top_reordered_products.jpg', dpi=300, bbox_inches='tight')
```

```
In [ ]: # Search for rules of the form antecedents => consequents
        frequent_itemsets = apriori(df_tf, min_support=0.01, use_colnames=True)
        rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.2)
        rules.sort_values(['confidence', 'lift'], ascending=[False, False])
Out[ ]:
        # Remove redundant rules
        rules = rules[~rules.antecedents.isin(['Discount', 'Free Shipping'])]
        # Display the top 10 rules by confidence
        rules.sort_values(['confidence', 'lift'], ascending=[False, False]).head(10)
In [ ]: # Create a bar chart of the top 10 rules by confidence
        import matplotlib.pyplot as plt
        top_rules = rules.sort_values(['confidence', 'lift'], ascending=[False, False]).head(10)
        plt.figure(figsize=(12, 6))
        plt.barh(range(len(top_rules)), top_rules['confidence'], color='lightgreen')
        plt.yticks(range(len(top_rules)), [', '.join(ant) + ' => ' + ', '.join(con) for ant, con in zip(top_rules['antecedents'], top_rules['consequents'])])
        plt.xlabel('Confidence')
        plt.title('Top 10 Rules by Confidence')
        plt.tight_layout()
        plt.show()
```

→ Taking the have people purchase out of the analysis as we can focus on other associations

```
In [ ]: # Save rules to CSV
        rules.to_csv('association_rules.csv', index=False)
        print("Rules saved to association_rules.csv")
Out[ ]:
        # Save frequent itemsets to CSV
        frequent_itemsets.to_csv('frequent_itemsets.csv', index=False)
        print("Frequent itemsets saved to frequent_itemsets.csv")
In [ ]: # Create final summary report
        print("ASSOCIATION RULES ANALYSIS SUMMARY")
        print("=" * 50)
        print("Total rules: {}".format(len(rules)))
        print("Average confidence: {:.3f}".format(rules['confidence'].mean()))
        print("Average lift: {:.3f}".format(rules['lift'].mean()))
        print("\n")
        print("Top 5 rules by confidence:")
        print("-" * 30)
        for i, row in rules.sort_values(['confidence', 'lift'], ascending=[False, False]).head(5).iterrows:
            print("{}. {} => {}".format(i+1, ', '.join(row['antecedents']), ', '.join(row['consequents'])))
            print("   Confidence: {:.3f}, Lift: {:.3f}, Support: {:.3f}".format(row['confidence'], row['lift'], row['support']))
            print("\n")
```

By confidence

```
In [ ]: # Generate frequent itemsets with support >= 0.01
        frequent_itemsets = apriori(df_tf, min_support=0.01, use_colnames=True)
        # Generate association rules with lift >= 1.2
        rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.2)
        # Sort by confidence and lift
        rules.sort_values(['confidence', 'lift'], ascending=[False, False])
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
78	(Discount)	(have people purchase)	0.500000	0.500000	0.500000	1.000000	2.000000	0.250000	inf
80	(Free Shipping)	(have people purchase)	0.500000	0.500000	0.500000	1.000000	2.000000	0.250000	inf
79	(have people purchase)	(Discount)	0.500000	0.500000	0.500000	1.000000	2.000000	0.250000	inf
81	(have people purchase)	(Free Shipping)	0.500000	0.500000	0.500000	1.000000	2.000000	0.250000	inf
18	(Jumbo Box, Vintage Doily)	(Lunch Bag Red Retrospot)	0.116667	0.525000	0.061667	0.528571	1.006803	0.000411	1.006803
19	(Lunch Bag Red Retrospot)	(Jumbo Box, Vintage Doily)	0.525000	0.116667	0.061667	0.117460	1.006803	0.000411	1.006803
16	(Jumbo Bag Red Retrospot)	(Lunch Bag Red Retrospot)	0.583333	0.525000	0.305000	0.522857	0.995941	-0.001233	0.991667
17	(Lunch Bag Red Retrospot)	(Jumbo Bag Red Retrospot)	0.525000	0.583333	0.305000	0.580952	0.995941	-0.001233	0.991667
14	(Jumbo Bag Red Retrospot)	(Jumbo Bag Pink Polka Dot)	0.583333	0.516667	0.291667	0.500000	0.967742	-0.009792	0.966667
15	(Jumbo Bag Pink Polka Dot)	(Jumbo Bag Red Retrospot)	0.516667	0.583333	0.291667	0.564516	0.967742	-0.009792	0.956667
12	(Jumbo Bag Red Retrospot)	(Jumbo Bag Vintage Doily)	0.583333	0.516667	0.291667	0.500000	0.967742	-0.009792	0.966667
13	(Jumbo Bag Vintage Doily)	(Jumbo Bag Red Retrospot)	0.516667	0.583333	0.291667	0.564516	0.967742	-0.009792	0.956667
10	(Jumbo Bag Pink Polka Dot)	(Jumbo Bag Vintage Doily)	0.516667	0.516667	0.258333	0.500000	0.967742	-0.008542	0.966667
11	(Jumbo Bag Vintage Doily)	(Jumbo Bag Pink Polka Dot)	0.516667	0.516667	0.258333	0.500000	0.967742	-0.008542	0.966667
4	(Jumbo Bag)	(Lunch Bag Red Retrospot)	0.666667	0.525000	0.350000	0.525000	1.000000	0.000000	1.000000
5	(Lunch Bag Red Retrospot)	(Jumbo Bag)	0.525000	0.666667	0.350000	0.666667	1.000000	0.000000	inf
2	(Jumbo Bag)	(Jumbo Bag Red Retrospot)	0.666667	0.583333	0.305000	0.457500	0.784286	-0.084286	0.769167
3	(Jumbo Bag Red Retrospot)	(Jumbo Bag)	0.583333	0.666667	0.305000	0.522857	0.784286	-0.084286	0.666667
0	(Jumbo Bag)	(Jumbo Bag Pink Polka Dot)	0.666667	0.516667	0.291667	0.437500	0.846774	-0.052688	0.858333
1	(Jumbo Bag Pink Polka Dot)	(Jumbo Bag)	0.516667	0.666667	0.291667	0.564516	0.846774	-0.052688	0.764167
6	(Jumbo Bag)	(Jumbo Bag Vintage Doily)	0.666667	0.516667	0.258333	0.387500	0.750000	-0.086111	0.775000
7	(Jumbo Bag Vintage Doily)	(Jumbo Bag)	0.516667	0.666667	0.258333	0.500000	0.750000	-0.086111	0.666667
8	(Jumbo Bag Red Retrospot)	(Jumbo Bag Vintage Doily)	0.583333	0.516667	0.258333	0.442857	0.856774	-0.043542	0.858333
9	(Jumbo Bag Vintage Doily)	(Jumbo Bag Red Retrospot)	0.516667	0.583333	0.258333	0.500000	0.856774	-0.043542	0.833333
20	(Jumbo Bag, Vintage Doily)	(Lunch Bag Red Retrospot)	0.116667	0.525000	0.061667	0.528571	1.006803	0.000411	1.006803
21	(Lunch Bag Red Retrospot)	(Jumbo Bag, Vintage Doily)	0.525000	0.116667	0.061667	0.117460	1.006803	0.000411	1.006803

```
In [ ]: # Save rules to CSV
        rules.to_csv('association_rules.csv', index=False)
        print("Rules saved to association_rules.csv")
Out[ ]:
        # Save frequent itemsets to CSV
        frequent_itemsets.to_csv('frequent_itemsets.csv', index=False)
        print("Frequent itemsets saved to frequent_itemsets.csv")
In [ ]: # Create final summary report
        print("ASSOCIATION RULES ANALYSIS SUMMARY")
        print("=" * 50)
        print("Total rules: {}".format(len(rules)))
        print("Average confidence: {:.3f}".format(rules['confidence'].mean()))
        print("Average lift: {:.3f}".format(rules['lift'].mean()))
        print("\n")
        print("Top 5 rules by confidence:")
        print("-" * 30)
        for i, row in rules.sort_values(['confidence', 'lift'], ascending=[False, False]).head(5).iterrows:
            print("{}. {} => {}".format(i+1, ', '.join(row['antecedents']), ', '.join(row['consequents'])))
            print("   Confidence: {:.3f}, Lift: {:.3f}, Support: {:.3f}".format(row['confidence'], row['lift'], row['support']))
            print("\n")
```

By lift

```
In [ ]: # Generate frequent itemsets with support >= 0.01
        frequent_itemsets = apriori(df_tf, min_support=0.01, use_colnames=True)
        # Generate association rules with lift >= 1.2
        rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.2)
        # Sort by lift and confidence
        rules.sort_values(['lift', 'confidence'], ascending=[False, False])
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
78	(Discount)	(have people purchase)	0.500000	0.500000	0.500000	1.000000	2.000000	0.250000	inf
80	(Free Shipping)	(have people purchase)	0.500000	0.500000	0.500000	1.000000	2.000000	0.250000	inf
79	(have people purchase)	(Discount)	0.500000	0.500000	0.500000	1.000000	2.000000	0.250000	inf
81	(have people purchase)	(Free Shipping)	0.500000	0.500000	0.500000	1.000000	2.000000	0.250000	inf
18	(Jumbo Box, Vintage Doily)	(Lunch Bag Red Retrospot)	0.116667	0.525000	0.061667	0.528571	1.006803	0.000411	1.006803
19	(Lunch Bag Red Retrospot)	(Jumbo Box, Vintage Doily)	0.525000	0.116667	0.061667	0.117460	1.006803	0.000411	1.006803
16	(Jumbo Bag Red Retrospot)	(Lunch Bag Red Retrospot)	0.583333	0.525000	0.305000	0.522857	0.995941	-0.001233	0.991667
17	(Lunch Bag Red Retrospot)	(Jumbo Bag Red Retrospot)	0.525000	0.583333	0.305000	0.580952	0.995941	-0.001233	0.991667
14	(Jumbo Bag Red Retrospot)	(Jumbo Bag Pink Polka Dot)	0.583333	0.516667	0.291667	0.500000	0.967742	-0.009792	0.966667
15	(Jumbo Bag Pink Polka Dot)	(Jumbo Bag Red Retrospot)	0.516667	0.583333	0.291667	0.564516	0.967742	-0.009792	0.956667
12	(Jumbo Bag Red Retrospot)	(Jumbo Bag Vintage Doily)	0.583333	0.516667	0.291667	0.500000	0.967742	-0.009792	0.966667
13	(Jumbo Bag Vintage Doily)	(Jumbo Bag Red Retrospot)	0.516667	0.583333	0.291667	0.564516	0.967742	-0.009792	0.956667
10	(Jumbo Bag Pink Polka Dot)	(Jumbo Bag Vintage Doily)	0.516667	0.516667	0.258333	0.500000	0.967742	-0.008542	0.966667
11	(Jumbo Bag Vintage Doily)	(Jumbo Bag Pink Polka Dot)	0.516667	0.516667	0.258333	0.500000	0.967742	-0.008542	0.966667
4	(Jumbo Bag)	(Lunch Bag Red Retrospot)	0.666667	0.525000	0.350000	0.525000	1.000000	0.000000	1.000000
5	(Lunch Bag Red Retrospot)	(Jumbo Bag)	0.525000	0.666667	0.350000	0.666667	1.000000	0.000000	inf
2	(Jumbo Bag)	(Jumbo Bag Red Retrospot)	0.666667	0.583333	0.305000	0.457500	0.784286	-0.084286	0.769167
3	(Jumbo Bag Red Retrospot)	(Jumbo Bag)	0.583333	0.666667	0.305000	0.522857	0.784286	-0.084286	0.666667
0	(Jumbo Bag)	(Jumbo Bag Pink Polka Dot)	0.666667	0.516667	0.291667	0.437500	0.846774	-0.052688	0.858333
1	(Jumbo Bag Pink Polka Dot)	(Jumbo Bag)	0.516667	0.666667	0.291667	0.564516	0.846774	-0.052688	0.764167
6	(Jumbo Bag)	(Jumbo Bag Vintage Doily)	0.666667	0.516667	0.258333	0.387500	0.750000	-0.086111	0.775000
7	(Jumbo Bag Vintage Doily)	(Jumbo Bag)	0.516667	0.666667	0.258333	0.500000	0.750000	-0.086111	0.666667
8	(Jumbo Bag Red Retrospot)	(Jumbo Bag Vintage Doily)	0.583333	0.516667	0.258333	0.442857	0.856774	-0.043542	0.858333
9	(Jumbo Bag Vintage Doily)	(Jumbo Bag Red Retrospot)	0.516667	0.583333	0.258333	0.500000	0.856774	-0.043542	0.833333
20	(Jumbo Bag, Vintage Doily)	(Lunch Bag Red Retrospot)	0.116667	0.525000	0.061667	0.528571	1.006803	0.000411	1.006803
21	(Lunch Bag Red Retrospot)	(Jumbo Bag, Vintage Doily)	0.525000	0.116667	0.061667	0.117460	1.006803	0.000411	1.006803

```
In [ ]: # Save rules to CSV
        rules.to_csv('association_rules.csv', index=False)
        print("Rules saved to association_rules.csv")
Out[ ]:
        # Save frequent itemsets to CSV
        frequent_itemsets.to_csv('frequent_itemsets.csv', index=False)
        print("Frequent itemsets saved to frequent_itemsets.csv")
In [ ]: # Create final summary report
        print("ASSOCIATION RULES ANALYSIS SUMMARY")
        print("=" * 50)
        print("Total rules: {}".format(len(rules)))
        print("Average confidence: {:.3f}".format(rules['confidence'].mean()))
        print("Average lift: {:.3f}".format(rules['lift'].mean()))
        print("\n")
        print("Top 5 rules by confidence:")
        print("-" * 30)
        for i, row in rules.sort_values(['confidence', 'lift'], ascending=[False, False]).head(5).iterrows:
            print("{}. {} => {}".format(i+1, ', '.join(row['antecedents']), ', '.join(row['consequents'])))
            print("   Confidence: {:.3f}, Lift: {:.3f}, Support: {:.3f}".format(row['confidence'], row['lift'], row['support']))
            print("\n")
```

Exploration

```
In [ ]: # Generate frequent itemsets with support >= 0.01
        frequent_itemsets = apriori(df_tf, min_support=0.01, use_colnames=True)
        # Generate association rules with lift >= 1.2
        rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.2)
        # Sort by support, confidence, and lift
        rules.sort_values(['support', 'confidence', 'lift'], ascending=[False, False, False])
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
78	(Discount)	(have people purchase)	0.500000	0.500000	0.500000	1.000000	2.000000	0.250000	inf
80	(Free Shipping)	(have people purchase)	0.500000	0.500000	0.500000	1.000000	2.000000	0.250000	inf
79	(have people purchase)	(Discount)	0.500000	0.500000	0.500000	1.000000	2.000000	0.250000	inf
81	(have people purchase)	(Free Shipping)	0.500000	0.500000	0.500000	1.000000	2.000000	0.250000	inf
18	(Jumbo Box, Vintage Doily)	(Lunch Bag Red Retrospot)	0.116667	0.525000	0.061667	0.528571	1.006803	0.000411	1.006803
19	(Lunch Bag Red Retrospot)	(Jumbo Box, Vintage Doily)	0.525000	0.116667	0.061667	0.117460	1.006803	0.000411	1.006803
16	(Jumbo Bag Red Retrospot)	(Lunch Bag Red Retrospot)	0.583333	0.525000	0.305000	0.522857	0.995941	-0.001233	0.991667
17	(Lunch Bag Red Retrospot)	(Jumbo Bag Red Retrospot)	0.525000	0.583333	0.305000	0.580952	0.995941	-0.001233	0.991667
14	(Jumbo Bag Red Retrospot)	(Jumbo Bag Pink Polka Dot)	0.583333	0.516667	0.291667	0.500000	0.967742	-0.009792	0.966667
15	(Jumbo Bag Pink Polka Dot)	(Jumbo Bag Red Retrospot)	0.516667	0.583333	0.291667	0.564516	0.967742	-0.009792	0.956667
12	(Jumbo Bag Red Retrospot)	(Jumbo Bag Vintage Doily)	0.583333	0.516667	0.291667	0.500000	0.967742	-0.009792	0.966667
13	(Jumbo Bag Vintage Doily)	(Jumbo Bag Red Retrospot)	0.516667	0.583333	0.291667	0.564516	0.967742	-0.009792	0.956667
10	(Jumbo Bag Pink Polka Dot)	(Jumbo Bag Vintage Doily)	0.516667	0.516667	0.258333	0.500000	0.967742	-0.008542	0.966667
11	(Jumbo Bag Vintage Doily)	(Jumbo Bag Pink Polka Dot)	0.516667	0.516667	0.258333	0.500000	0.967742	-0.008542	0.966667
4	(Jumbo Bag)	(Lunch Bag Red Retrospot)	0.666667	0.525000	0.350000	0.525000	1.000000	0.000000	1.000000
5	(Lunch Bag Red Retrospot)	(Jumbo Bag)	0.525000	0.666667	0.350000	0.666667	1.000000	0.000000	inf
2	(Jumbo Bag)	(Jumbo Bag Red Retrospot)	0.666667	0.583333	0.305000	0.457500	0.784286	-0.084286	0.769167
3	(Jumbo Bag Red Retrospot)	(Jumbo Bag)	0.583333	0.666667	0.305000	0.522857	0.784286	-0.084286	0.666667
0	(Jumbo Bag)	(Jumbo Bag Pink Polka Dot)	0.666667	0.516667	0.291667	0.437500	0.846774	-0.052688	0.858333
1	(Jumbo Bag Pink Polka Dot)	(Jumbo Bag)	0.516667	0.666667	0.291667	0.564516	0.846774	-0.052688	0.764167
6	(Jumbo Bag)	(Jumbo Bag Vintage Doily)	0.666667	0.516667	0.258333	0.387500	0.750000	-0.086111	0.775000
7	(Jumbo Bag Vintage Doily)	(Jumbo Bag)	0.516667	0.666667	0.258333	0.500000	0.750000	-0.086111	0.666667
8	(Jumbo Bag Red Retrospot)	(Jumbo Bag Vintage Doily)	0.583333	0.516667	0.258333	0.442857	0.856774	-0.043542	0.858333
9	(Jumbo Bag Vintage Doily)	(Jumbo Bag Red Retrospot)	0.516667	0.583333	0.258333	0.500000	0.856774	-0.043542	0.833333
20	(Jumbo Bag, Vintage Doily)	(Lunch Bag Red Retrospot)	0.116667	0.525000	0.061667	0.528571	1.006803	0.000411	1.006803
21	(Lunch Bag Red Retrospot)	(Jumbo Bag, Vintage Doily)	0.525000	0.116667	0.061667	0.117460	1.006803	0.000411	1.006803

```
In [ ]: # Save rules to CSV
        rules.to_csv('association_rules.csv', index=False)
        print("Rules saved to association_rules.csv")
Out[ ]:
        # Save frequent itemsets to CSV
        frequent_itemsets.to_csv('frequent_itemsets.csv', index=False)
        print("Frequent itemsets saved to frequent_itemsets.csv")
In [ ]: # Create final summary report
        print("ASSOCIATION RULES ANALYSIS SUMMARY")
        print("=" * 50)
        print("Total rules: {}".format(len(rules)))
        print("Average confidence: {:.3f}".format(rules['confidence'].mean()))
        print("Average lift: {:.3f}".format(rules['lift'].mean()))
        print("\n")
        print("Top 5 rules by confidence:")
        print("-" * 30)
        for i, row in rules.sort_values(['confidence', 'lift'], ascending=[False, False]).head(5).iterrows:
            print("{}. {} => {}".format(i+1, ', '.join(row['antecedents']), ', '.join(row['consequents'])))
            print("   Confidence: {:.3f}, Lift: {:.3f}, Support: {:.3f}".format(row['confidence'], row['lift'], row['support']))
            print("\n")
```

Analysis by time (week, weekend and hours of day):

→ By week

```
In [ ]: df_week = df_final[df_final['week_end'] == df_final['week_end'].iloc[0]]
        df_week.shape
Out[ ]: (14,8)
In [ ]: df_week['today_id'].nunique()
Out[ ]: 1
In [ ]: df_week.groupby('week_id')['order_id'].nunique()
Out[ ]:
        week_id
        2009-12    14
        Name: order_id, dtype: int64
In [ ]: df_week.groupby('week_id')['today_id'].nunique()
Out[ ]:
        week_id
        2009-12    1
        Name: today_id, dtype: int64
In [ ]: # Create frequent itemsets and association rules
        frequent_itemsets = apriori(df_tf, min_support=0.01, use_colnames=True)
        rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.2)
        # Sort by confidence and lift
        rules.sort_values(['confidence', 'lift'], ascending=[False, False])
```

By confidence

```
In [ ]: # Generate frequent itemsets with support >= 0.01
        frequent_itemsets = apriori(df_tf, min_support=0.01, use_colnames=True)
        # Generate association rules with lift >= 1.2
        rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.2)
        # Sort by confidence and lift
        rules.sort_values(['confidence', 'lift'], ascending=[False, False])
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
47	(Dust Cloth, Tidy Home)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	0.500000	0.417857	0.250000	0.500000	1.196581	0.041964	1.165179
46	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home)	0.417857	0.500000	0.250000	0.598291	1.196581	0.041964	1.248214
45	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	0.417857	0.250000	0.250000	0.598291	2.392162	0.146429	1.870536
44	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	0.250000	0.250000	0.250000	1.000000	4.000000	0.187500	inf
43	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home)	0.250000	0.500000	0.250000	1.000000	2.000000	0.125000	inf
42	(Dust Cloth, Tidy Home)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	0.500000	0.250000	0.250000	0.500000	2.000000	0.125000	inf
41	(Dust Cloth, Tidy Home)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	0.500000	0.250000	0.250000	0.500000	2.000000	0.125000	inf
40	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home)	0.250000	0.500000	0.250000	1.000000	2.000000	0.125000	inf
39	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	0.250000	0.250000	0.250000	1.000000	4.000000	0.187500	inf
38	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	0.250000	0.250000	0.250000	1.000000	4.000000	0.187500	inf
37	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home)	0.250000	0.500000	0.250000	1.000000	2.000000	0.125000	inf
36	(Dust Cloth, Tidy Home)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	0.500000	0.250000	0.250000	0.500000	2.000000	0.125000	inf
35	(Dust Cloth, Tidy Home)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	0.500000	0.250000	0.250000	0.500000	2.000000	0.125000	inf
34	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home)	0.250000	0.500000	0.250000	1.000000	2.000000	0.125000	inf
33	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	0.250000	0.250000	0.250000	1.000000	4.000000	0.187500	inf
32	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	0.250000	0.250000	0.250000	1.000000	4.000000	0.187500	inf
31	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home)	0.250000	0.500000	0.250000	1.000000	2.000000	0.125000	inf
30	(Dust Cloth, Tidy Home)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	0.500000	0.250000	0.250000	0.500000	2.000000	0.125000	inf
29	(Dust Cloth, Tidy Home)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	0.500000	0.250000	0.250000	0.500000	2.000000	0.125000	inf
28	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home)	0.250000	0.500000	0.250000	1.000000	2.000000	0.125000	inf
27	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	0.250000	0.250000	0.250000	1.000000	4.000000	0.187500	inf
26	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	0.250000	0.250000	0.250000	1.000000	4.000000	0.187500	inf
25	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home)	0.250000	0.500000	0.250000	1.000000	2.000000	0.125000	inf
24	(Dust Cloth, Tidy Home)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	0.500000	0.250000	0.250000	0.500000	2.000000	0.125000	inf
23	(Dust Cloth, Tidy Home)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	0.500000	0.250000	0.250000	0.500000	2.000000	0.125000	inf
22	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home)	0.250000	0.500000	0.250000	1.000000	2.000000	0.125000	inf
21	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	0.250000	0.250000	0.250000	1.000000	4.000000	0.187500	inf
20	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	0.250000	0.250000	0.250000	1.000000	4.000000	0.187500	inf
19	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home)	0.250000	0.500000	0.250000	1.000000	2.000000	0.125000	inf
18	(Dust Cloth, Tidy Home)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	0.500000	0.250000	0.250000	0.500000	2.000000	0.125000	inf
17	(Dust Cloth, Tidy Home)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	0.500000	0.250000	0.250000	0.500000	2.000000	0.125000	inf
16	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home)	0.250000	0.500000	0.250000	1.000000	2.000000	0.125000	inf
15	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	0.250000	0.250000	0.250000	1.000000	4.000000	0.187500	inf
14	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	0.250000	0.250000	0.250000	1.000000	4.000000	0.187500	inf
13	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home)	0.250000	0.500000	0.250000	1.000000	2.000000	0.125000	inf
12	(Dust Cloth, Tidy Home)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	0.500000	0.250000	0.250000	0.500000	2.000000	0.125000	inf
11	(Dust Cloth, Tidy Home)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	0.500000	0.250000	0.250000	0.500000	2.000000	0.125000	inf
10	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home)	0.250000	0.500000	0.250000	1.000000	2.000000	0.125000	inf
9	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	0.250000	0.250000	0.250000	1.000000	4.000000	0.187500	inf
8	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	0.250000	0.250000	0.250000	1.000000	4.000000	0.187500	inf
7	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home)	0.250000	0.500000	0.250000	1.000000	2.000000	0.125000	inf
6	(Dust Cloth, Tidy Home)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	0.500000	0.250000	0.250000	0.500000	2.000000	0.125000	inf
5	(Dust Cloth, Tidy Home)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	0.500000	0.250000	0.250000	0.500000	2.000000	0.125000	inf
4	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home)	0.250000	0.500000	0.250000	1.000000	2.000000	0.125000	inf
3	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	0.250000	0.250000	0.250000	1.000000	4.000000	0.187500	inf
2	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	0.250000	0.250000	0.250000	1.000000	4.000000	0.187500	inf
1	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	(Dust Cloth, Tidy Home)	0.250000	0.500000	0.250000	1.000000	2.000000	0.125000	inf
0	(Dust Cloth, Tidy Home)	(Dust Cloth, Tidy Home, Tidy Home Cleaning Kit)	0.500000	0.250000	0.250000	0.500000	2.000000	0.125000	inf

```
In [ ]: # Save rules to CSV
        rules.to_csv('association_rules.csv', index=False)
        print("Rules sa
```

In [108]: # high frequentset and high lift - complementary antecedent
frequent_itemsets = apriori(df[['Antecedent_length', 'Consequent_length']], min_support=0.01, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.2)
rules.sort_values(['lift'], ascending=[False], inplace=True)
rules.head()

Out[108]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	length_one	length_zero
1	(Antecedent_length)	(Consequent_length)	0.997242	0.997242	0.994477	0.997242	1.000000	0.000000	inf	0	1
0	(Consequent_length)	(Antecedent_length)	0.997242	0.997242	0.994477	0.997242	1.000000	0.000000	inf	0	1
3	(Antecedent_length, Consequent_length)	(Short Form)	0.994477	0.974171	0.974171	0.979472	1.005443	0.005418	1.254779	0	1
2	(Short Form)	(Antecedent_length, Consequent_length)	0.974171	0.994477	0.974171	1.000000	1.005443	0.005418	inf	0	1
5	(Antecedent_length, Short Form)	(Consequent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
4	(Consequent_length)	(Antecedent_length, Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
7	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
6	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
9	(Short Form, Consequent_length)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
8	(Antecedent_length)	(Short Form, Consequent_length)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
11	(Short Form, Antecedent_length)	(Consequent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
10	(Consequent_length)	(Short Form, Antecedent_length)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
13	(Short Form)	(Antecedent_length, Consequent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
12	(Antecedent_length, Consequent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
15	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
14	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
17	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
16	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
19	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
18	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
21	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
20	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
23	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
22	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
25	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
24	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
27	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
26	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
29	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
28	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
31	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
30	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
33	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
32	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
35	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
34	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
37	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
36	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
39	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
38	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
41	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
40	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
43	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
42	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
45	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
44	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
47	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
46	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
49	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
48	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
51	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
50	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
53	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
52	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
55	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
54	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
57	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
56	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
59	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
58	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
61	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
60	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
63	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
62	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
65	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
64	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
67	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
66	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
69	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
68	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
71	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
70	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
73	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
72	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
75	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
74	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
77	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
76	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
79	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
78	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
81	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
80	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
83	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
82	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
85	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
84	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
87	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
86	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
89	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
88	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
91	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
90	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
93	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
92	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
95	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
94	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
97	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
96	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
99	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
98	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
101	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
100	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
103	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
102	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
105	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
104	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
107	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
106	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
109	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
108	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
111	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
110	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
113	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
112	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
115	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
114	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
117	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
116	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
119	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
118	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
121	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
120	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
123	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
122	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
125	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
124	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
127	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
126	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
129	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
128	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
131	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
130	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
133	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
132	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
135	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
134	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
137	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
136	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
139	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
138	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
141	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
140	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
143	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
142	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
145	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
144	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
147	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
146	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
149	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
148	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
151	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
150	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
153	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
152	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
155	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
154	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
157	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
156	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
159	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
158	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
161	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
160	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
163	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
162	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
165	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
164	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
167	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
166	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
169	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
168	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
171	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
170	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
173	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
172	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
175	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
174	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
177	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
176	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
179	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
178	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
181	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
180	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
183	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
182	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
185	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
184	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
187	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
186	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
189	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
188	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
191	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
190	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
193	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
192	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
195	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
194	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
197	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
196	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
199	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
198	(Antecedent_length)	(Short Form)	0.997242	0.974171	0.974171	0.976862	1.002762	0.002691	1.113988	0	1
201	(Short Form)	(Antecedent_length)	0.974171	0.997242	0.974171	1.000000	1.002762	0.002691	inf	0	1
200	(Antecedent_length)	(Short