I Edgar Setyan 217240789 # 217240789 acknowledge that I have contributed at least 100% time and effort to the preparation of this report and work discussed herein.
Edgar Setyan

This code snippet imports the pandas and mlxtend libraries. It then reads the pre-processed CSV file and sets the index to the Date column. The remaining columns are converted to boolean values.

The apriori function from the mlxtend library is used to find frequent itemsets with a minimum support of 0.01 and a minimum length of 2. The itemsets column of the resulting DataFrame is then converted to a list of sets.

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

# Read the pre-processed CSV file
data = pd.read_csv('C:\\Data Mining Project\\Data Mining Task
3\\217240789-t0.csv', parse_dates=['Date'])

# Set the index to the 'Date' column and convert remaining columns to
boolean values
data = data.set_index('Date').astype(bool)
```

This code snippet imports the pandas and mlxtend libraries. It then reads the pre-processed CSV file and sets the index to the Date column. The remaining columns are converted to boolean values.

The apriori function from the mlxtend library is used to find frequent itemsets with a minimum support of 0.01 and a minimum length of 2. The itemsets column of the resulting DataFrame is then converted to a list of sets.

```
# Find frequent itemsets with minimum support of 0.01 and minimum length of
2
frequent_itemsets = apriori(data, min_support=0.01, use_colnames=True)
frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x:
len(x))
frequent_itemsets = frequent_itemsets[frequent_itemsets['length'] >= 2]
```

This code snippet uses the apriori function to find frequent itemsets. The min_support parameter specifies the minimum support threshold, and the use_colnames parameter specifies whether to use the column names in the resulting DataFrame. The length column of the resulting DataFrame is then computed, which is the number of items in each itemset. The length column is then used to filter the DataFrame to only include itemsets with a length of at least 2.

```
# Sort the frequent item sets by support in descending order
frequent_itemsets = frequent_itemsets.sort_values(by='support',
ascending=False)

# Get the top 10 frequent item sets and compute their support
top_10_frequent_items = frequent_itemsets.head(10)
# Print the top 10 frequent item sets and their support
for i, row in top_10_frequent_items.iterrows():
    print(f"{i+1}. Itemset: {row['itemsets']}, Support:
{round(row['support'],4)}")
```

This code snippet sorts the frequent itemsets by support in descending order and prints the top 10 frequent itemsets. The support of an itemset is the proportion of transactions that contain the itemset. The iterrows method is used to iterate over the DataFrame and print the itemsets and their support.

```
# Create association rules using metric 'confidence' and min_threshold=0.01
(adjust as desired)
rules = association_rules(frequent_itemsets,support_only=True,
metric='confidence', min_threshold=0.01)

# Sort the rules by confidence in descending order
rules_sorted_by_confidence = rules.sort_values(by='confidence',
ascending=False)

# Get the top 5 rules with highest confidence and compute their measures of
interest
top_5_high_confidence_rules = rules_sorted_by_confidence.head(5)

# Print the top 5 rules with highest confidence and their measures of
interest
for i, row in top_5_high_confidence_rules.iterrows():
    rule = row[0]

    support = round(row["support"], 4)
```

```
    confidence = round(row["confidence"], 4)
    lift = round(row["lift"], 4)


print("Rule:", set(rule))
print("Support:", support)
print("Confidence:", confidence)
print("Lift:", lift)
```

```
# Get the bottom 5 rules with lowest confidence
bottom_5_lowest_confidence_rules = rules_sorted_by_confidence.tail(5)

# Print the bottom 5 rules with lowest confidence and their measures of
interest
for i, row in bottom_5_lowest_confidence_rules.iterrows():
    rule = row["antecedents"], "=>", row["consequents"]

    support = round(row["support"], 4)
    confidence = round(row["confidence"], 4)
    lift = round(row["lift"], 4)
    leverage = round(row["leverage"], 4)
    conviction = round(row["conviction"], 4)

    print("Rule:", rule)
    print("Support:", support)
    print("Confidence:", confidence)
    print("Lift:", lift)
    print("Leverage:", leverage)
    print("Conviction:", conviction)
    print("---")
```

This code snippet first gets the bottom 5 rules with the lowest confidence from the rules_sorted_by_confidence DataFrame. The tail method is used to get the bottom 5 rows from the DataFrame, and the confidence column is used to sort the rows in descending order.

The iterrows method is then used to iterate over the DataFrame and print the rules and their measures of interest. The i variable is used to keep track of the row number, and the row variable contains the current row from the DataFrame.

The print statement prints the rule, the support, the confidence, the lift, the leverage, and the conviction of the rule. The round function is used to round the measures of interest to four decimal places.

This code snippet can be used to print the bottom 5 rules with the lowest confidence from a DataFrame of association rules. The confidence of a rule is a measure of how often the consequent of the rule occurs given that the antecedent of the rule occurs. The higher the confidence, the more likely the consequent of the rule occurs given that the antecedent of the rule occurs.