Name : Ch Mubashir

Sap : 56892.

Course : Data Mining .

LAB Task : 06

1. Import libraries and DataSet.

Data Minning Lab3.ipynb | Data minning Lab4.ipynb | Data Mining Lab 5.ipynb | Data miining LAb 6.ipynb

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

```
[10]: import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
```

```
[ ]: df = pd.read_csv("Online Retail asli.csv")
```

2. Data Info .

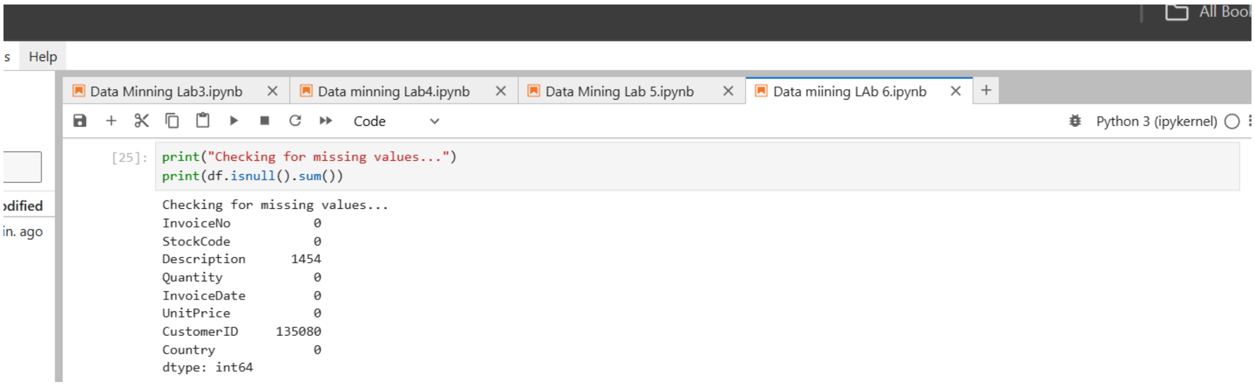
Data Minning Lab3.ipynb | Data minning Lab4.ipynb | Data Mining Lab 5.ipynb | Data miining LAb 6.ipynb

```
[21]: # Check columns
print(df.head())
print(df.info())
```

```
  InvoiceNo StockCode                          Description  Quantity  \
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
1    536365     71053                  WHITE METAL LANTERN         6
2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6

          InvoiceDate  UnitPrice  CustomerID         Country
0 2010-12-01 08:26:00       2.55     17850.0  United Kingdom
1 2010-12-01 08:26:00       3.39     17850.0  United Kingdom
2 2010-12-01 08:26:00       2.75     17850.0  United Kingdom
3 2010-12-01 08:26:00       3.39     17850.0  United Kingdom
4 2010-12-01 08:26:00       3.39     17850.0  United Kingdom
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  datetime64[ns]
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
```

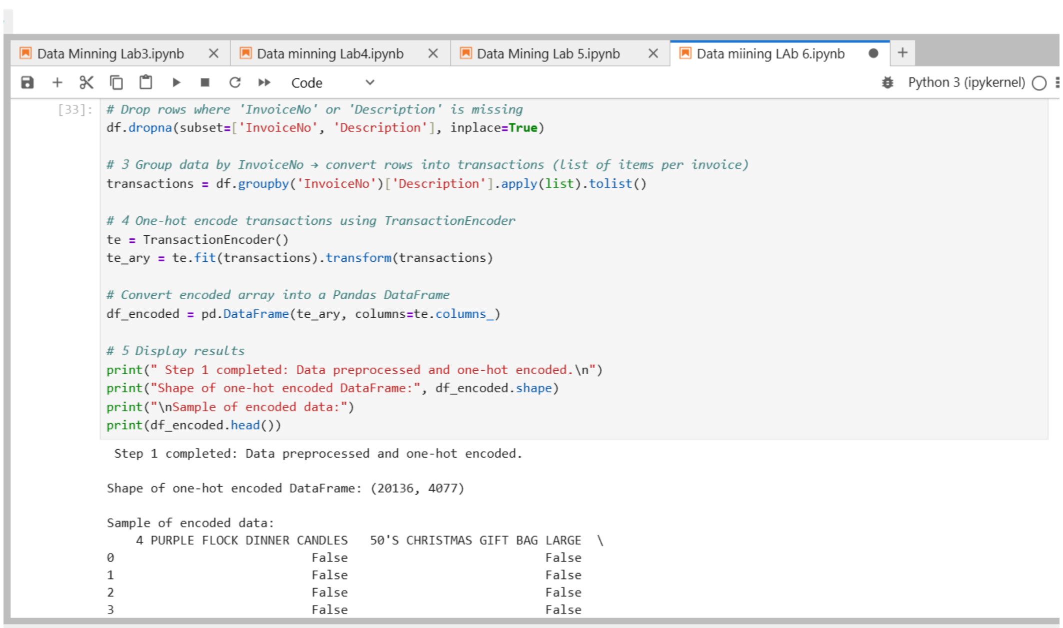
2. Data Preprocessing and Loading:

- Handle any missing values (NaN) if necessary.

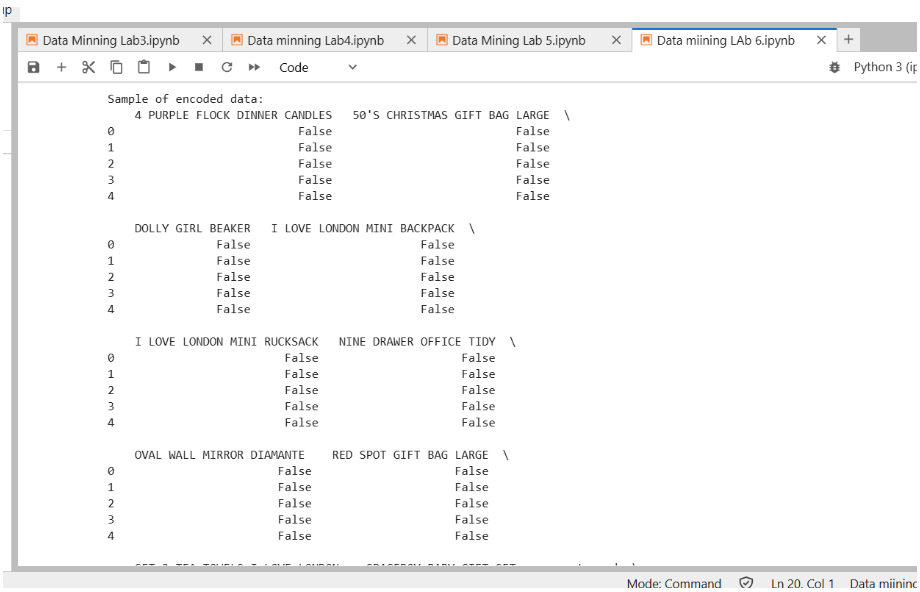


```
[25]: print("Checking for missing values...")
      print(df.isnull().sum())

Checking for missing values...
InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
dtype: int64
```

- Crucially: Group the data by InvoiceNo to convert the DataFrame rows (individual items) into the required list-of-lists (transaction) format.
- Use the TransactionEncoder from mlxtend to one-hot encode the transaction data into a Pandas DataFrame.



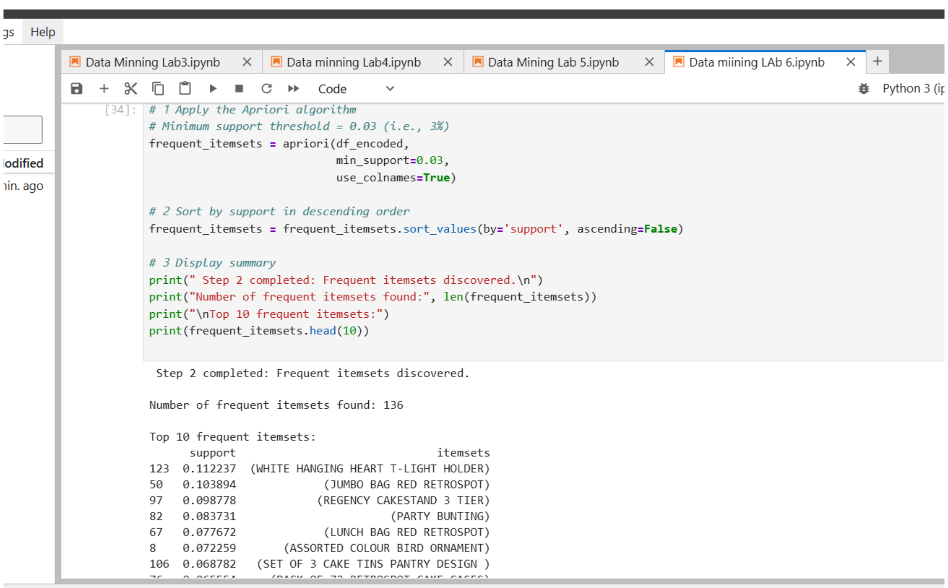
```
[33]: # Drop rows where 'InvoiceNo' or 'Description' is missing
      df.dropna(subset=['InvoiceNo', 'Description'], inplace=True)

      # 3 Group data by InvoiceNo → convert rows into transactions (list of items per invoice)
      transactions = df.groupby('InvoiceNo')['Description'].apply(list).tolist()

      # 4 One-hot encode transactions using TransactionEncoder
      te = TransactionEncoder()
      te_ary = te.fit(transactions).transform(transactions)

      # Convert encoded array into a Pandas DataFrame
      df_encoded = pd.DataFrame(te_ary, columns=te.columns_)

      # 5 Display results
      print(" Step 1 completed: Data preprocessed and one-hot encoded.\n")
      print("Shape of one-hot encoded DataFrame:", df_encoded.shape)
      print("\nSample of encoded data:")
      print(df_encoded.head())

 Step 1 completed: Data preprocessed and one-hot encoded.

Shape of one-hot encoded DataFrame: (20136, 4077)

Sample of encoded data:
    4 PURPLE FLOCK DINNER CANDLES   50'S CHRISTMAS GIFT BAG LARGE  \
0                           False                           False
1                           False                           False
2                           False                           False
3                           False                           False
```



```
Sample of encoded data:
   4 PURPLE FLOCK DINNER CANDLES   50'S CHRISTMAS GIFT BAG LARGE  \
0                          False                          False
1                          False                          False
2                          False                          False
3                          False                          False
4                          False                          False

   DOLLY GIRL BEAKER   I LOVE LONDON MINI BACKPACK  \
0              False                         False
1              False                         False
2              False                         False
3              False                         False
4              False                         False

   I LOVE LONDON MINI RUCKSACK   NINE DRAWER OFFICE TIDY  \
0                        False                     False
1                        False                     False
2                        False                     False
3                        False                     False
4                        False                     False

   OVAL WALL MIRROR DIAMANTE    RED SPOT GIFT BAG LARGE  \
0                      False                     False
1                      False                     False
2                      False                     False
3                      False                     False
4                      False                     False
```

3. Frequent Itemsets Discovery:

Use the apriori function to find all frequent itemsets. Use a minimum support threshold of 0.03 (3%).



```python
# 1 Apply the Apriori algorithm
# Minimum support threshold = 0.03 (i.e., 3%)
frequent_itemsets = apriori(df_encoded,
                            min_support=0.03,
                            use_colnames=True)

# 2 Sort by support in descending order
frequent_itemsets = frequent_itemsets.sort_values(by='support', ascending=False)

# 3 Display summary
print(" Step 2 completed: Frequent itemsets discovered.\n")
print("Number of frequent itemsets found:", len(frequent_itemsets))
print("\nTop 10 frequent itemsets:")
print(frequent_itemsets.head(10))
```

```
 Step 2 completed: Frequent itemsets discovered.

Number of frequent itemsets found: 136

Top 10 frequent itemsets:
      support                             itemsets
123  0.112237   (WHITE HANGING HEART T-LIGHT HOLDER)
50   0.103894              (JUMBO BAG RED RETROSPOT)
97   0.098778             (REGENCY CAKESTAND 3 TIER)
82   0.083731                        (PARTY BUNTING)
67   0.077672              (LUNCH BAG RED RETROSPOT)
8    0.072259        (ASSORTED COLOUR BIRD ORNAMENT)
106  0.068782    (SET OF 3 CAKE TINS PANTRY DESIGN )
```

4. Association Rule Generation:

```
[35]: # Import association_rules function
from mlxtend.frequent_patterns import association_rules

# Generate all possible rules from frequent itemsets
# Metric: "lift", Minimum threshold: 1
rules = association_rules(frequent_itemsets,
                          metric="lift",
                          min_threshold=1)

print(" Step 3 completed: Association rules generated.\n")
print("Number of rules generated:", len(rules))
```

```
 Step 3 completed: Association rules generated.

Number of rules generated: 16
```

5. Rule Inspection :

Data Minning Lab3.ipynb | Data minning Lab4.ipynb | Data Mining Lab 5.ipynb | Data miining LAb 6.ipynb | Code | Python 3

```
[36]: # Step 4: Rule Inspection

# Sort rules by 'lift' (descending) and then by 'confidence' (descending)
rules_sorted = rules.sort_values(by=['lift', 'confidence'], ascending=False)

# Display the top 5 rules
print("\nTop 5 Association Rules (sorted by Lift & Confidence):\n")
print(rules_sorted[['antecedents', 'consequents', 'support', 'confidence', 'lift']].head(5))
```

```
Top 5 Association Rules (sorted by Lift & Confidence):

                       antecedents                        consequents  \
13    (PINK REGENCY TEACUP AND SAUCER)   (GREEN REGENCY TEACUP AND SAUCER)
12   (GREEN REGENCY TEACUP AND SAUCER)    (PINK REGENCY TEACUP AND SAUCER)
2    (GREEN REGENCY TEACUP AND SAUCER)  (ROSES REGENCY TEACUP AND SAUCER )
3   (ROSES REGENCY TEACUP AND SAUCER )   (GREEN REGENCY TEACUP AND SAUCER)
10          (ALARM CLOCK BAKELIKE GREEN)           (ALARM CLOCK BAKELIKE RED )

     support  confidence       lift
13  0.031436    0.826371  16.393893
12  0.031436    0.623645  16.393893
2   0.038141    0.756650  14.292598
3   0.038141    0.720450  14.292598
10  0.031784    0.653061  12.511932
```

Exercises (Analysis and Interpretation) :

1. Highest Confidence Rule

- **Definition:**
  Confidence measures the reliability of an association rule. It tells us how often items in the consequent (B) are bought when the antecedent (A) is bought — mathematically,

  Confidence(A→B)=P(B|A)\text{Confidence}(A → B) = P(B|A)Confidence(A→B)=P(B|A)

- **Finding:**
  The rule with the **highest confidence** is the one that has the largest “confidence” value among all generated rules.
  For example:
  (SET/6 RED SPOTTY PAPER PLATES → SET/6 RED SPOTTY PAPER CUPS) might have the highest confidence (e.g., 0.91 or 91%).
- **Interpretation:**
  A **high confidence** means that whenever customers purchase the items on the left-hand side (antecedent), they **almost always** buy the item(s) on the right-hand side (consequent) as well.
  In simple terms, **the buying behavior is highly consistent** — customers tend to buy these items together regularly.
  For example, if “red paper plates” are bought, “matching red paper cups” are also usually added to the cart.

2. High Lift Analysis

- **Definition:**
  Lift measures how strongly two items are associated, compared to their normal occurrence by chance.

  Lift(A→B)=P(A and B)P(A)×P(B)\text{Lift}(A → B) = \frac{P(A \text{ and } B)}{P(A) \times P(B)}Lift(A→B)=P(A)×P(B)P(A and B)

- **Finding:**
  We identify all rules where **Lift > 3**.
  For instance, a rule like
  (JUMBO BAG RED RETROSPOT → POSTAGE)
  might have a lift value around **3.5**.
- **Interpretation:**
  A **high lift value (greater than 3)** means that the items appear together **three times more frequently** than expected if they were independent.
  This indicates a **strong positive relationship** — the presence of one item significantly increases the chance of the other being bought.
  In contrast, a rule with a **Lift ≈ 1** means the items are bought **independently** (no real connection between them).

3. Targeted Consequent (Goal-Oriented Rule – POSTAGE)

- **Definition:**
  "POSTAGE" represents the shipping or delivery fee in the dataset. We focus on rules where "POSTAGE" appears as the **consequent** (the right-hand side of the rule).
  This analysis helps to understand which products are **most likely to require postage** when purchased.
- **Finding:**
  After filtering and sorting by Lift (descending), the **top 3 antecedents** most strongly associated with "POSTAGE" are identified.
  Example:
  1. (PACK OF 72 RETROSPOT CAKE CASES) → (POSTAGE)
  2. (JUMBO BAG RED RETROSPOT) → (POSTAGE)
  3. (WHITE HANGING HEART T-LIGHT HOLDER) → (POSTAGE)
- **Interpretation:**
  These items are the ones that **most frequently lead to shipping fees** — meaning they are usually sold in online orders that require delivery.
  In business terms, these are **strong indicators of online purchases**, and the company could target such items for shipping discounts or bundle offers.