Design and Implementation of Apriori Algorithm.

1. Introduction to Apriori Algorithm

The **Apriori Algorithm** is a fundamental technique in **frequent itemset mining** and **association rule learning**. It is primarily used in **market basket analysis** to find items that frequently appear together.

2. Key Concepts

- **Frequent Itemset**: Item combinations that appear together frequently in transactions.
- **Support**: The proportion of transactions that contain an itemset.
- **Confidence**: The likelihood of an item appearing given another item.
- **Lift**: The strength of an association between two items.

3. Design of Apriori Algorithm

1. **Set a minimum support threshold**.
2. **Generate candidate itemsets** (C1) and count their occurrences.
3. **Filter out infrequent itemsets** to form L1.
4. **Generate larger itemsets (L2, L3, ...)** using previous frequent itemsets.
5. **Repeat until no more frequent itemsets can be found**.
6. **Generate association rules** based on confidence and lift.

---

4. Implementation of Apriori Algorithm

We will use Python and the **mlxtend** library for implementation.

Step 1: Install Required Libraries

bash

CopyEdit

```
pip install mlxtend pandas
```

Step 2: Import Required Modules

python

CopyEdit

```
import pandas as pd
```

```
from mlxtend.frequent_patterns import apriori, association_rules
```

Step 3: Load Sample Transaction Data

```
# Sample dataset (Market Basket Transactions)
dataset = [
   ['Milk', 'Bread', 'Eggs'],
   ['Milk', 'Diaper', 'Beer', 'Bread'],
   ['Milk', 'Diaper', 'Beer', 'Cola'],
   ['Bread', 'Butter'],
   ['Milk', 'Diaper', 'Beer', 'Bread', 'Butter'],
]

# Convert to a pandas DataFrame
df = pd.DataFrame(dataset)
print(df.head())
```

Step 4: Convert Transactions to One-Hot Encoded Format

```
from mlxtend.preprocessing import TransactionEncoder
# Convert dataset into a one-hot encoded format
te = TransactionEncoder()
te_array = te.fit(dataset).transform(dataset)
df_encoded = pd.DataFrame(te_array, columns=te.columns_)

# Display transformed dataset
print(df_encoded)
```

Step 5: Apply the Apriori Algorithm

```
# Find frequent itemsets with a minimum support of 0.4
frequent_itemsets = apriori(df_encoded, min_support=0.4, use_colnames=True)
print(frequent_itemsets)
```

Step 6: Generate Association Rules

```
# Generate association rules with a minimum confidence of 0.6
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.6)


# Display rules
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```

---

5. Conclusion

- **Apriori is efficient for small to medium-sized datasets.**
- **For large-scale data, optimizations like FP-Growth or parallel computing are preferred.**
- **This implementation does not use Spark but effectively finds frequent patterns and rules.**