Name: Sayali Salunke **Subject:** Data Science Laboratory

Roll No: A-256(A-3 Batch) **Subject-In-Charge:** Prof.Manisha Patil

---

Practical: 01

Aim: Installation and study of any one Data Analytics Tool Framework.

Source Code:

```
import pandas as pd

data = {
    "Name": ["Sayali", "Dhanoo", "Ankita", "Rushii"],
    "Age": [21, 27, 22, 32],
    "Salary": [50000, 40000, 30000, 20000]
}

df = pd.DataFrame(data)

print("Original Data:")
print(df)

avg_age = df["Age"].mean()
avg_salary = df["Salary"].mean()

print("\nAverage Age:", avg_age)
print("Average Salary:", avg_salary)

filtered_data = df[df["Age"] > 30]
print("\nPeople Above 30:")
print(filtered_data)
```

Output:

```
Original Data:
     Name  Age  Salary
0  Sayali   21   50000
1  Dhanoo   27   40000
2  Ankita   22   30000
3  Rushii   32   20000

Average Age: 25.5
Average Salary: 35000.0

People Above 30:
     Name  Age  Salary
3  Rushii   32   20000
```

Name: Sayali Salunke **Subject:** Data Science Laboratory

Roll No: A-256(A-3 Batch) **Subject-In-Charge:** Prof.Manisha Patil

---

Practical: 02

Aim: Installation and study of any one Data Analytics Tool Framework.

Source Code:

```
# 2.Write a python program to demonstrate the use of Numpy.
import numpy as np

arr1 = np.array([1, 2, 3, 4, 5])
arr2 = np.array([5, 4, 3, 2, 1])

sum_arr = arr1 + arr2
dot_product = np.dot(arr1, arr2)
mean_arr1 = np.mean(arr1)
matrix = arr1.reshape(1, 5)

print("Array 1:", arr1)
print("Array 2:", arr2)
print("\nElement-wise Sum:", sum_arr)
print("Dot Product:", dot_product)
print("Mean of Array 1:", mean_arr1)
print("\nReshaped Matrix:\n", matrix)
```

Output:

Array 1: [1 2 3 4 5]

Array 2: [5 4 3 2 1]

Element-wise Sum: [6 6 6 6 6]

Dot Product: 35

Mean of Array 1: 3.0

Reshaped Matrix:

[[1 2 3 4 5]]

Name: Sayali Salunke **Subject:** Data Science Laboratory

Roll No: A-256(A-3 Batch) **Subject-In-Charge:** Prof.Manisha Patil

---

Practical: 03

Aim: Design and develop at least 10 problem statements which demonstrate the use of data structure, functions, Importing / Exporting Data in any data analytics tool.

- **10 Problem Statements with Solutions**

1. Store Employee Data in a Dictionary & Export to JSON

Problem: Create a dictionary of employees (name, age, salary) and save it as JSON.

Solution:

Source Code:

```
#3.1. Store Employee Data in a Dictionary & Export to JSON
import pandas as pd

employees = {
    "Name": ["James", "Cristopher", "Jack"],
    "Age": [21, 22, 23],
    "Salary": [50000, 40000, 30000]
}

df = pd.DataFrame(employees)
df.to_json("employees.json", orient="records", indent=4)

print("Data exported to JSON!")
```

Output:

Data exported to JSON!

2. Read CSV, Filter Rows, and Export to Excel

Problem: Load a CSV file, filter rows where Age > 30, and save as Excel.

Solution:

Source Code:

```python
#3.2 2. Read CSV, Filter Rows, and Export to Excel
import pandas as pd

# Read CSV file
df = pd.read_csv("Student_marks.csv")

# Filter rows where Age > 30
filtered_data = df[df["Age"] > 30]

# Export filtered data to Excel
filtered_data.to_excel("filtered_data.xlsx", index=False)
```

3. Merge Two DataFrames and Save as CSV

Problem: Combine two datasets (e.g., customers.csv and orders.csv) and export the result.

Solution:

Source Code:

```python
#3.3
import pandas as pd

customers = pd.read_csv("/content/customer.csv")
orders = pd.read_csv("/content/order.csv")

merged_data = pd.concat([customers, orders], axis=1)

merged_data.to_csv("merged_data.csv", index=False)
```

4. Convert List of Tuples to DataFrame & Export

Problem: Take a list of tuples (product, price) and save as CSV.

Solution:

Source Code:

```python
#3.4
products = [("Laptop", 1000), ("Phone", 800), ("Tablet", 500)]

df = pd.DataFrame(products, columns=["Product", "Price"])

df.to_csv("products.csv", index=False)
```

5. Read JSON Data, Compute Average, and Export

Problem: Load JSON data, compute the average salary, and save results.

Solution:

Source Code:

```
#3.5
import pandas as pd

data = pd.read_json("employees.json")

avg_salary = data["Salary"].mean()

result = {"Average Salary": avg_salary}

pd.DataFrame([result]).to_csv("avg_salary.csv", index=False)
```

6. Generate Random Data, Store in DataFrame, Export to Excel

Problem: Create random sales data and export it.

Solution:

Source Code:

```
#3.6
import pandas as pd
import numpy as np

sales_data = {
    "Product": ["A", "B", "C"],
    "Sales": np.random.randint(100, 1000, 3)
}

df = pd.DataFrame(sales_data)

df.to_excel("sales.xlsx", index=False)
```

7. Read Excel, Apply Discount, and Save

Problem: Load an Excel file, apply a 10% discount on prices, and save.

Solution:

Source Code:

```python
#3.7
import pandas as pd

df = pd.read_excel("products.xlsx")

df["Discounted_Price"] = df["Price"] * 0.9

df.to_csv("discounted_products.csv", index=False)
```

8. Group Data by Category & Export Summary

Problem: Group sales data by category and compute total sales.

Solution:

Source Code:

```python
#3.8
import pandas as pd

sales = pd.read_csv("sales.csv")

grouped = sales.groupby("Product")["Sales"].sum()

grouped.to_csv("sales_summary.csv")
```

9. Convert Dictionary to DataFrame & Save as CSV

Problem: Store student records (name, marks) in a dictionary and export.

Solution:

Source Code:

```python
#3.9
import pandas as pd

students = {
    "Name": ["Dhanashri", "Neha", "Soham"],
    "Marks": [85, 80, 75]
}
df = pd.DataFrame(students)
df.to_csv("students.csv", index=False)
```

10. Read Web Data (API) & Store in CSV

Problem: Fetch data from a JSON API and save it.

Solution:

Source Code:

```python
#3.10
import pandas as pd
import requests

response = requests.get(
    "https://data.gov.sg/api/action/datastore_search?resource_id=f1765b54-a209-4718-8d38-a39237f502b3"
)

data = pd.DataFrame(response.json())

data.to_csv("api_data.csv", index=False)
```

Name: Sayali Salunke **Subject:** Data Science Laboratory

Roll No: A-256(A-3 Batch) **Subject-In-Charge:** Prof.Manisha Patil

---

Practical: 04

Aim: Design and develop at least 5 problem statements which demonstrate the use of Control Structures of any data analytics tool.

Source Code:

1. **If-Else Control Structure**

```
#4.1
# Program to check whether a number is even or odd

# Input from user
num = int(input("Enter a number: "))

# if-else control structure
if num % 2 == 0:
    print(f"{num} is Even")
else:
    print(f"{num} is Odd")
```

Output:

Enter a number: 5

5 is Odd

2. **For Loop(Sum of Numbers from 1 to 100)**

```
#4.2
# Program to calculate the sum of numbers from 1 to 100 using for loop

total = 0

for i in range(1, 101):
    total += i

print("The sum of numbers from 1 to 100 is:", total)
```

Output:

The sum of numbers from 1 to 100 is: 5050

3. **While Loop(Reverse a Number)**

```
#4.3
# Program to reverse digits of a number using while loop

num = int(input("Enter a number: "))
rev = 0

while num > 0:
    digit = num % 10
    rev = rev * 10 + digit
    num //= 10

print("Reversed number is:", rev)
```

Output:

Enter a number: 567

Reversed number is: 765

4. **If–Elif–Else Control Structure (Check Positive, Negative, or Zero)**

```
#4.4
# Program to check whether a number is positive, negative, or zero

num = int(input("Enter a number: "))

if num > 0:
    print("The number is Positive")
elif num < 0:
    print("The number is Negative")
else:
    print("The number is Zero")
```

Output:

Enter a number: 7

The number is Positive

5. Do While Simulation (Print Numbers 1 to 5)

```
#4.5
# Program to simulate do...while loop in Python

num = 1

while True:
    print(num)
    num += 1
    if num > 5:   # exit condition
        break
```

Output:

1

2

3

4

5

Name: Sayali Salunke **Subject:** Data Science Laboratory

Roll No: A-256(A-3 Batch) **Subject-In-Charge:** Prof.Manisha Patil

---

Practical: 05

Aim: Implement KNN Classification techniques using any data analytics tool.

Source Code:

```
# Import libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
# Step 1: Load dataset (Iris dataset for example)
from sklearn.datasets import load_iris
iris = load_iris()
# Convert to DataFrame for easy viewing
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df["target"] = iris.target
print("First 5 rows of dataset:")
print(df.head())
# Step 2: Split features and target
X = df.iloc[:, :-1]   # features
y = df.iloc[:, -1]    # target
# Step 3: Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)
# Step 4: Feature scaling (important for KNN)
scaler = StandardScaler()
```

```
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# Step 5: Train KNN model
knn = KNeighborsClassifier(n_neighbors=5)  # k = 5
knn.fit(X_train, y_train)
# Step 6: Make predictions
y_pred = knn.predict(X_test)
# Step 7: Evaluate model
print("\nAccuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

Output:

```
First 5 rows of dataset:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   target
0       0
1       0
2       0
3       0
4       0

Accuracy: 1.0

Confusion Matrix:
 [[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

Name: Sayali Salunke **Subject:** Data Science Laboratory

Roll No: A-256(A-3 Batch) **Subject-In-Charge:** Prof.Manisha Patil

---

Practical: 06

Aim: Implement Naive Bayes Classification techniques using any data analytics tool.

Source Code:

```
# Import libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Step 1: Load dataset (Iris dataset for example)
from sklearn.datasets import load_iris
iris = load_iris()
# Convert to DataFrame for easy understanding
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df["target"] = iris.target
print("First 5 rows of dataset:")
print(df.head())
# Step 2: Split features and target
X = df.iloc[:, :-1]   # features
y = df.iloc[:, -1]    # target
# Step 3: Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)
# Step 4: Train Naive Bayes model
```

```
nb = GaussianNB()
nb.fit(X_train, y_train)
# Step 5: Make predictions
y_pred = nb.predict(X_test)
# Step 6: Evaluate model
print("\nAccuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

Output:

```
First 5 rows of dataset:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   target
0       0
1       0
2       0
3       0
4       0

Accuracy: 0.9777777777777777

Confusion Matrix:
 [[19  0  0]
 [ 0 12  1]
 [ 0  0 13]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      0.92      0.96        13
           2       0.93      1.00      0.96        13

    accuracy                           0.98        45
   macro avg       0.98      0.97      0.97        45
weighted avg       0.98      0.98      0.98        45
```

Name: Sayali Salunke **Subject:** Data Science Laboratory

Roll No: A-256(A-3 Batch) **Subject-In-Charge:** Prof.Manisha Patil

---

Practical: 07

Aim: Implement K means Clustering techniques using any data analytics tool.

Source Code:

```
# Import libraries
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Step 1: Load dataset (Iris dataset for example)
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)

print("First 5 rows of dataset:")
print(df.head())

# Step 2: Feature scaling (optional but recommended)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df)

# Step 3: Apply K-Means clustering
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
df["Cluster"] = kmeans.fit_predict(X_scaled)

print("\nCluster centers (scaled):")
```

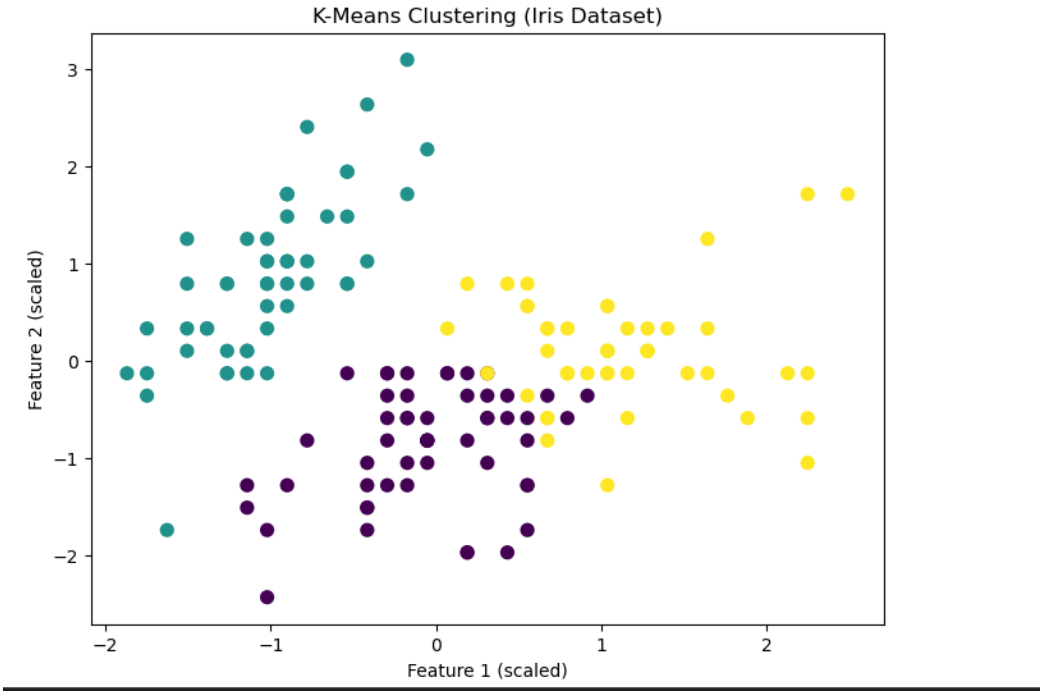
```
print(kmeans.cluster_centers_)


# Step 4: Visualization (using first two features)
plt.figure(figsize=(8, 6))
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=df["Cluster"], cmap="viridis", s=50)
plt.title("K-Means Clustering (Iris Dataset)")
plt.xlabel("Feature 1 (scaled)")
plt.ylabel("Feature 2 (scaled)")
plt.show()
```

Output:

```
First 5 rows of dataset:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: UserWarning: KMeans is known to have a
are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(

Cluster centers (scaled):
[[-0.05021989 -0.88337647  0.34773781  0.2815273 ]
 [-1.01457897  0.85326268 -1.30498732 -1.25489349]
 [ 1.13597027  0.08842168  0.99615451  1.01752612]]
```



Name: Sayali Salunke **Subject:** Data Science Laboratory

Roll No: A-256(A-3 Batch) **Subject-In-Charge:** Prof.Manisha Patil

---

Practical: 08

Aim: Implement DBScan Clustering techniques using any data analytics tool.

Source Code:

#8. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) using Python with scikit-learn.

```
# Import libraries

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.datasets import load_iris

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import DBSCAN


# Step 1: Load dataset (Iris dataset for example)

iris = load_iris()

df = pd.DataFrame(iris.data, columns=iris.feature_names)


print("First 5 rows of dataset:")

print(df.head())


# Step 2: Feature scaling (important for DBSCAN)

scaler = StandardScaler()

X_scaled = scaler.fit_transform(df)


# Step 3: Apply DBSCAN

dbscan = DBSCAN(eps=0.8, min_samples=5)  # eps = neighborhood radius, min_samples = minimum points

clusters = dbscan.fit_predict(X_scaled)
```

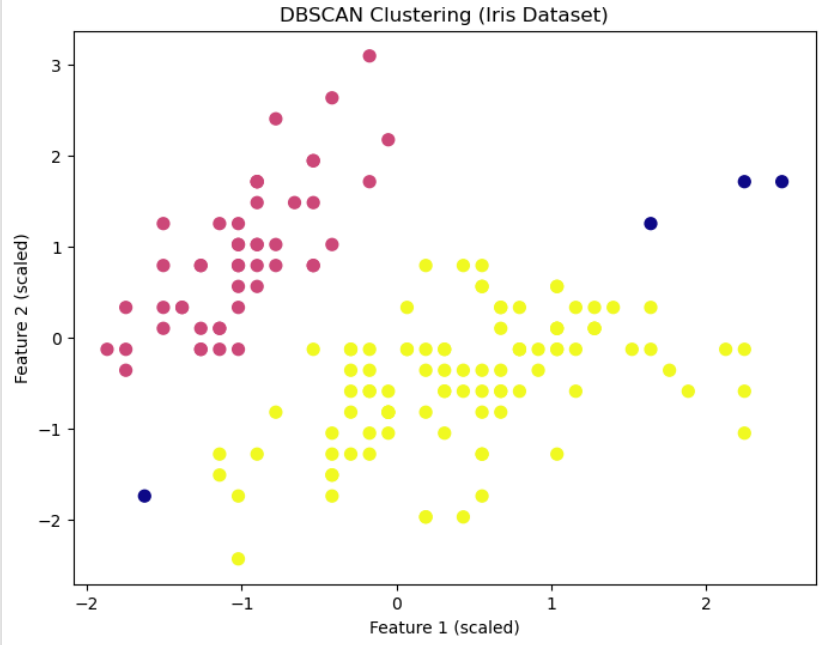
```
# Step 4: Add cluster labels to DataFrame
df["Cluster"] = clusters
print("\nCluster labels assigned by DBSCAN:")
print(df["Cluster"].value_counts())

# Step 5: Visualization (using first two features)
plt.figure(figsize=(8, 6))
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=clusters, cmap="plasma", s=50)
plt.title("DBSCAN Clustering (Iris Dataset)")
plt.xlabel("Feature 1 (scaled)")
plt.ylabel("Feature 2 (scaled)")
plt.show()
```

Output:

```
First 5 rows of dataset:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

Cluster labels assigned by DBSCAN:
Cluster
 1    97
 0    49
-1     4
Name: count, dtype: int64
```



Name: Sayali Salunke **Subject:** Data Science Laboratory

Roll No: A-256(A-3 Batch) **Subject-In-Charge:** Prof.Manisha Patil

---

Practical: 09

Aim: Implement Eclat Association Rule Mining techniques using any data analytics tool.

Source Code:

```
from itertools import combinations


# Sample dataset
dataset = [
    ["Milk", "Bread", "Butter"],
    ["Bread", "Eggs"],
    ["Milk", "Bread", "Eggs", "Butter"],
    ["Bread", "Butter"],
    ["Milk", "Bread", "Eggs"]
]
# Support function
def support(itemset):
    return sum(1 for t in dataset if set(itemset).issubset(t)) / len(dataset)


# Frequent itemsets
items = sorted(set(i for t in dataset for i in t))
freq = [(s, support(s)) for r in range(1, len(items)+1) for s in combinations(items, r) if
support(s) >= 0.4]


print("Frequent Itemsets:")
for s, sup in freq: print(s, "=> Support:", round(sup, 2))


# Association Rules
print("\nAssociation Rules:")
```

```
for s, sup in freq:
    if len(s) > 1:
        for i in range(1, len(s)):
            for a in combinations(s, i):
                b = tuple(set(s)-set(a))
                conf = support(s) / support(a)
                if conf >= 0.6:
                    print(a, "->", b, "| Support:", round(sup, 2), "Confidence:", round(conf, 2))
```

Output:

```
Frequent Itemsets:
('Bread',) => Support: 1.0
('Butter',) => Support: 0.6
('Eggs',) => Support: 0.6
('Milk',) => Support: 0.6
('Bread', 'Butter') => Support: 0.6
('Bread', 'Eggs') => Support: 0.6
('Bread', 'Milk') => Support: 0.6
('Butter', 'Milk') => Support: 0.4
('Eggs', 'Milk') => Support: 0.4
('Bread', 'Butter', 'Milk') => Support: 0.4
('Bread', 'Eggs', 'Milk') => Support: 0.4

Association Rules:
('Bread',) -> ('Butter',) | Support: 0.6 Confidence: 0.6
('Butter',) -> ('Bread',) | Support: 0.6 Confidence: 1.0
('Bread',) -> ('Eggs',) | Support: 0.6 Confidence: 0.6
('Eggs',) -> ('Bread',) | Support: 0.6 Confidence: 1.0
('Bread',) -> ('Milk',) | Support: 0.6 Confidence: 0.6
('Milk',) -> ('Bread',) | Support: 0.6 Confidence: 1.0
('Butter',) -> ('Milk',) | Support: 0.4 Confidence: 0.67
('Milk',) -> ('Butter',) | Support: 0.4 Confidence: 0.67
('Eggs',) -> ('Milk',) | Support: 0.4 Confidence: 0.67
('Milk',) -> ('Eggs',) | Support: 0.4 Confidence: 0.67
('Butter',) -> ('Milk', 'Bread') | Support: 0.4 Confidence: 0.67
('Milk',) -> ('Butter', 'Bread') | Support: 0.4 Confidence: 0.67
('Bread', 'Butter') -> ('Milk',) | Support: 0.4 Confidence: 0.67
('Bread', 'Milk') -> ('Butter',) | Support: 0.4 Confidence: 0.67
('Butter', 'Milk') -> ('Bread',) | Support: 0.4 Confidence: 1.0
('Eggs',) -> ('Milk', 'Bread') | Support: 0.4 Confidence: 0.67
('Milk',) -> ('Eggs', 'Bread') | Support: 0.4 Confidence: 0.67
('Bread', 'Eggs') -> ('Milk',) | Support: 0.4 Confidence: 0.67
('Bread', 'Milk') -> ('Eggs',) | Support: 0.4 Confidence: 0.67
('Eggs', 'Milk') -> ('Bread',) | Support: 0.4 Confidence: 1.0
```

Name: Sayali Salunke **Subject:** Data Science Laboratory

Roll No: A-256(A-3 Batch) **Subject-In-Charge:** Prof.Manisha Patil

---

Practical: 10

Aim: Implement Apriori Association Rule Mining techniques using any data analytics tool.

Source Code:

```
#10 : Implement Apriori Association Rule Mining techniques using any data analytics tool.
from itertools import combinations

# Sample dataset
dataset = [
    ["Milk", "Bread", "Butter"],
    ["Bread", "Eggs"],
    ["Milk", "Bread", "Eggs", "Butter"],
    ["Bread", "Butter"],
    ["Milk", "Bread", "Eggs"]
]

# Support function
def support(itemset, transactions):
    return sum(1 for t in transactions if set(itemset).issubset(t)) / len(transactions)

# Generate frequent itemsets
def apriori(transactions, min_support=0.4):
    items = sorted(set(i for t in transactions for i in t))
    freq = []
    for r in range(1, len(items)+1):
        for s in combinations(items, r):
            sup = support(s, transactions)
```

```
            if sup >= min_support:
                freq.append((s, sup))
    return freq

# Generate association rules
def generate_rules(freq, transactions, min_conf=0.6):
    rules = []
    for s, sup in freq:
        if len(s) > 1:
            for i in range(1, len(s)):
                for a in combinations(s, i):
                    b = tuple(set(s) - set(a))
                    conf = support(s, transactions) / support(a, transactions)
                    if conf >= min_conf:
                        rules.append((a, b, round(sup,2), round(conf,2)))
    return rules

# Run Apriori
frequent_itemsets = apriori(dataset, min_support=0.4)
print("Frequent Itemsets:")
for s, sup in frequent_itemsets:
    print(s, "=> Support:", round(sup, 2))

print("\nAssociation Rules:")
rules = generate_rules(frequent_itemsets, dataset, min_conf=0.6)
for a, b, sup, conf in rules:
    print(a, "->", b, "| Support:", sup, "Confidence:", conf)
```

Output:

```
Frequent Itemsets:
('Bread',) => Support: 1.0
('Butter',) => Support: 0.6
('Eggs',) => Support: 0.6
('Milk',) => Support: 0.6
('Bread', 'Butter') => Support: 0.6
('Bread', 'Eggs') => Support: 0.6
('Bread', 'Milk') => Support: 0.6
('Butter', 'Milk') => Support: 0.4
('Eggs', 'Milk') => Support: 0.4
('Bread', 'Butter', 'Milk') => Support: 0.4
('Bread', 'Eggs', 'Milk') => Support: 0.4

Association Rules:
('Bread',) -> ('Butter',) | Support: 0.6 Confidence: 0.6
('Butter',) -> ('Bread',) | Support: 0.6 Confidence: 1.0
('Bread',) -> ('Eggs',) | Support: 0.6 Confidence: 0.6
('Eggs',) -> ('Bread',) | Support: 0.6 Confidence: 1.0
('Bread',) -> ('Milk',) | Support: 0.6 Confidence: 0.6
('Milk',) -> ('Bread',) | Support: 0.6 Confidence: 1.0
('Butter',) -> ('Milk',) | Support: 0.4 Confidence: 0.67
('Milk',) -> ('Butter',) | Support: 0.4 Confidence: 0.67
('Eggs',) -> ('Milk',) | Support: 0.4 Confidence: 0.67
('Milk',) -> ('Eggs',) | Support: 0.4 Confidence: 0.67
('Butter',) -> ('Milk', 'Bread') | Support: 0.4 Confidence: 0.67
('Milk',) -> ('Bread', 'Butter') | Support: 0.4 Confidence: 0.67
('Bread', 'Butter') -> ('Milk',) | Support: 0.4 Confidence: 0.67
('Bread', 'Milk') -> ('Butter',) | Support: 0.4 Confidence: 0.67
('Butter', 'Milk') -> ('Bread',) | Support: 0.4 Confidence: 1.0
('Eggs',) -> ('Milk', 'Bread') | Support: 0.4 Confidence: 0.67
('Milk',) -> ('Eggs', 'Bread') | Support: 0.4 Confidence: 0.67
('Bread', 'Eggs') -> ('Milk',) | Support: 0.4 Confidence: 0.67
('Bread', 'Milk') -> ('Eggs',) | Support: 0.4 Confidence: 0.67
('Eggs', 'Milk') -> ('Bread',) | Support: 0.4 Confidence: 1.0
```

Name: Sayali Salunke **Subject:** Data Science Laboratory

Roll No: A-256(A-3 Batch) **Subject-In-Charge:** Prof.Manisha Patil

---

Practical: 11

Aim: Visualize all the statistical measures (mean, mode, median, range, inter quartile range, etc.) using Histograms, Boxplots, scatter plots, etc.

Source Code:

```
# Statistical Measures and Visualization

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from scipy import stats


# Step 1: Sample data

data = [12, 15, 14, 10, 18, 20, 18, 22, 24, 20, 25, 30, 28, 18, 15, 14]


# Step 2: Convert to pandas DataFrame

df = pd.DataFrame(data, columns=["Values"])


# Step 3: Calculate Statistical Measures

mean_val = np.mean(data)

median_val = np.median(data)

mode_val = stats.mode(data, keepdims=True).mode[0]   # ✅ FIXED

range_val = np.max(data) - np.min(data)

q1, q3 = np.percentile(data, [25, 75])

iqr_val = q3 - q1

std_val = np.std(data)
```

```
print("📊 Statistical Measures:")
print(f"Mean: {mean_val}")
print(f"Median: {median_val}")
print(f"Mode: {mode_val}")
print(f"Range: {range_val}")
print(f"Q1 (25%): {q1}")
print(f"Q3 (75%): {q3}")
print(f"IQR: {iqr_val}")
print(f"Standard Deviation: {std_val}")

# Step 4: Visualizations

# Histogram with Mean, Median, Mode
plt.figure(figsize=(8,5))
sns.histplot(df["Values"], bins=8, kde=True, color="skyblue")
plt.axvline(mean_val, color='red', linestyle='--', label=f"Mean={mean_val}")
plt.axvline(median_val, color='green', linestyle='-', label=f"Median={median_val}")
plt.axvline(mode_val, color='blue', linestyle=':', label=f"Mode={mode_val}")
plt.legend()
plt.title("Histogram with Mean, Median, Mode")
plt.show()

# Boxplot
plt.figure(figsize=(6,4))
sns.boxplot(y=df["Values"], color="lightcoral")
plt.title("Boxplot (Median, IQR, Outliers)")
plt.show()

# Scatter plot
plt.figure(figsize=(7,5))
```

```
plt.scatter(range(len(data)), data, color="purple", s=70)
plt.axhline(mean_val, color="red", linestyle="--", label="Mean")
plt.axhline(median_val, color="green", linestyle="-", label="Median")
plt.legend()
plt.title("Scatter Plot with Mean & Median")
plt.show()
```

Output:

```
Statistical Measures:
Mean: 18.9375
Median: 18.0
Mode: 18
Range: 20
Q1 (25%): 14.75
Q3 (75%): 22.5
IQR: 7.75
Standard Deviation: 5.527982792122277
```