Exp 3

Import pandas as pd

# Load Titanic dataset (from seaborn for simplicity)

Import seaborn as sns

Df = sns.load\_dataset(‘titanic’)

Print(“First 5 rows of Titanic dataset:”)

Print(df.head())

# 1. SLICE: Select passengers who are female

Slice\_df = df[df[‘sex’] == ‘female’]

Print(“\nSlice: Only female passengers”)

Print(slice\_df.head())

# 2. DICE: Female passengers in Class 1 or 2

Dice\_df = df[(df[‘sex’] == ‘female’) & (df[‘pclass’].isin([1, 2]))]

Print(“\nDice: Female passengers in Class 1 or 2”)

Print(dice\_df.head())

# 3. ROLL-UP: Total fare collected by passenger class

Grouped = df.groupby(‘pclass’) # Group data by class

Rollup\_df = grouped[‘fare’].sum() # Add up fare for each class

Print(“\nRoll-up: Total fare by Class”)

Print(rollup\_df)

# 4. DRILL-DOWN: Survival count grouped by class and sex

Drilldown\_df = df.groupby([‘pclass’, ‘sex’])[‘survived’].sum()

Print(“\nDrill-down: Survival count by Class and Sex”)

Print(drilldown\_df)

# 5. PIVOT: Survival rate by Class vs Sex

Pivot\_df = df.pivot\_table(values=’survived’,

Index=’pclass’,

Columns=’sex’,

Aggfunc=’mean’)

Print(“\nPivot: Survival rate (mean) by Class vs Sex”)

Print(pivot\_df)

[Dataset link](https://docs.google.com/spreadsheets/d/1qvd-QVL0qgOjgUAPYufSjdcxkC9nvF3n/edit?usp=drivesdk&ouid=109848835302865061001&rtpof=true&sd=true)

Exp 4

import numpy as np  
import matplotlib.pyplot as plt  
  
print("\n\t\tSmoothing by Bin Means\n")  
  
# Input data  
data = list(map(float, input("Enter the data separated by spaces: ").split()))  
num\_bins = int(input("Enter the number of bins: "))  
  
# Sort the data  
data.sort()  
  
# Perform binning and compute bin means  
bin\_means = []  
for i in range(0, len(data), num\_bins):  
    bin\_data = data[i:i+num\_bins]  
    mean\_val = sum(bin\_data) / len(bin\_data)  
    bin\_means.append(mean\_val)  
  
# Print bin means  
print("\nBin means:")  
for i, mean in enumerate(bin\_means):  
    print(f"Bin {i+1}: {mean:.2f}")  
  
# Create histogram  
plt.hist(data, bins=num\_bins, color="pink", edgecolor="black")  
plt.title("Data Discretization Histogram (Bin Means)")  
plt.xlabel("Values")  
plt.ylabel("Frequency")  
plt.show()

Another

from docx import Document  
from docx.shared import Inches, Pt  
from docx.enum.text import WD\_ALIGN\_PARAGRAPH  
  
# Create new document  
doc = Document()  
  
# Set narrow margins  
sections = doc.sections  
for section in sections:  
    section.top\_margin = Inches(0.5)  
    section.bottom\_margin = Inches(0.5)  
    section.left\_margin = Inches(0.5)  
    section.right\_margin = Inches(0.5)  
  
# Title page  
title = doc.add\_paragraph()  
title.alignment = WD\_ALIGN\_PARAGRAPH.CENTER  
run = title.add\_run("DATA WAREHOUSING & DATA MINING VIVA BOOKLET\n")  
run.bold = True  
run.font.size = Pt(20)  
run.font.name = "Times New Roman"  
  
subtitle = doc.add\_paragraph("Prepared for Viva Examination")  
subtitle.alignment = WD\_ALIGN\_PARAGRAPH.CENTER  
subtitle.runs[0].font.name = "Times New Roman"  
subtitle.runs[0].font.size = Pt(14)  
  
doc.add\_page\_break()  
  
# --------------------------  
# Add 180+ Q&A (medium-length answers)  
# --------------------------  
  
# Module-wise Q&A data (sample content, extended to 180+ Q&A)  
# For brevity, using representative samples and patterns for 180 Q&As  
modules = {  
    "Module 1: Data Warehousing Fundamentals": [  
        ("What is a Data Warehouse?", "A Data Warehouse is a centralized repository that stores integrated, subject-oriented, historical data from multiple sources for analysis, reporting, and decision-making."),  
        ("What are the benefits of a Data Warehouse?", "It provides integrated historical data that supports trend analysis, performance evaluation, prediction, and informed decision-making."),  
        ("Differentiate between Data Warehouse and Data Mart.", "A Data Warehouse is enterprise-wide and contains multiple subject areas, whereas a Data Mart focuses on a specific business area or department."),  
        ("What is OLTP?", "Online Transaction Processing (OLTP) manages day-to-day transactions and is optimized for insert and update operations."),  
        ("What is OLAP?", "Online Analytical Processing (OLAP) is used for analyzing large datasets quickly, optimized for complex select queries and multi-dimensional analysis."),  
        ("Differentiate between OLTP and OLAP.", "OLTP handles routine transactional data with high normalization, whereas OLAP supports complex analytical queries on denormalized data for decision-making."),  
        ("What is ETL?", "ETL stands for Extract, Transform, Load. It is the process of extracting data from source systems, transforming it for analysis, and loading it into the data warehouse."),  
        ("Explain Star Schema.", "A Star Schema is a multi-dimensional schema with a central fact table connected directly to dimension tables. It is simple, easy to query, and widely used for OLAP."),  
        ("Explain Snowflake Schema.", "A Snowflake Schema is a normalized version of a star schema where dimension tables are split into multiple related tables to reduce redundancy."),  
        ("What is a Fact Table?", "A Fact Table stores quantitative data (measures) for analysis, such as sales amount or quantity, and is linked to dimension tables."),  
        ("What is a Dimension Table?", "Dimension tables store descriptive attributes related to facts, such as product name, customer, or time, and help qualify measures."),  
        ("Define Factless Fact Table.", "A Factless Fact Table captures events or relationships without storing numeric measures, useful for tracking occurrences or participation."),  
        ("What is a Fact Constellation Schema?", "A Fact Constellation Schema has multiple fact tables sharing dimension tables, supporting complex analysis across multiple subjects."),  
        ("Explain OLAP operations: Slice and Dice.", "Slice selects a single dimension value to view a subset of data, while Dice selects specific values across multiple dimensions to create a subcube."),  
        ("Explain Roll-up and Drill-down.", "Roll-up aggregates data along a hierarchy for a higher-level summary, whereas Drill-down provides more detailed data by moving down the hierarchy."),  
        ("What is Pivot operation?", "Pivot rotates the data axes to view data from different perspectives, facilitating multi-dimensional analysis."),  
        ("What is an Information Package Diagram?", "It represents business processes, their dimensions, and the facts in a diagrammatic form to guide warehouse design."),  
        ("What is metadata?", "Metadata is data about data, describing the structure, types, source, and format of data in the warehouse."),  
        ("What are slowly changing dimensions?", "Dimensions whose attribute values change over time are called slowly changing dimensions, and require strategies to track historical changes."),  
        ("What are ROLAP, MOLAP, and HOLAP?", "ROLAP stores data in relational databases, MOLAP in multi-dimensional cubes, and HOLAP combines both to balance storage and query performance."),  
        ("What are Cubes in OLAP?", "Cubes store summarized multi-dimensional data to enable fast querying across various dimensions, such as sales by month and region."),  
        ("Differentiate E-R and Dimensional Modeling.", "E-R models are used for transactional systems focusing on relationships, while dimensional modeling is designed for analysis with facts and dimensions."),  
        ("What are the major steps in ETL process?", "The steps are: data extraction from sources, cleaning and transforming data, and loading it into the warehouse for analysis."),  
        ("Explain OLAP schema types.", "The main schemas are Star, Snowflake, and Fact Constellation; they organize data for efficient querying and reporting."),  
        ("Why is data warehouse subject-oriented?", "It is designed around major subjects of the organization like sales or finance, enabling focused analysis rather than operational processing."),  
        ("Why is data warehouse time-variant?", "Data is stored with historical timestamps to allow trend analysis and comparison over time."),  
        ("Why is data warehouse non-volatile?", "Once data is entered, it is not updated or deleted, ensuring stable historical data for analysis."),  
        ("Explain dependent and independent data warehouse.", "A dependent warehouse receives data from operational systems; an independent warehouse gets data directly from sources without an operational warehouse."),  
        ("What is a hybrid data warehouse?", "A hybrid warehouse integrates data from both central warehouses and operational systems for flexible analysis."),  
        ("Explain update to dimension tables.", "Dimension tables can be updated using slowly changing dimension techniques to maintain historical accuracy while reflecting changes.")  
    ],  
    # Module 2 sample (30 questions)  
    "Module 2: Data Mining, Exploration and Preprocessing": [  
        ("What is Data Mining?", "Data Mining is the process of discovering hidden patterns, relationships, or knowledge from large datasets using statistical and computational techniques."),  
        ("Define KDD process.", "Knowledge Discovery in Databases (KDD) is the overall process including data selection, preprocessing, transformation, mining, and interpretation of results."),  
        ("What are descriptive and predictive tasks in data mining?", "Descriptive tasks summarize data characteristics, while predictive tasks use existing data to predict future trends or classify new data."),  
        ("What are task primitives in data mining?", "Task primitives define the type of data mining task, the kind of knowledge to be discovered, and constraints on the data or patterns."),  
        ("What are applications of data mining?", "Applications include market analysis, fraud detection, customer relationship management, healthcare analysis, and scientific research."),  
        ("What is data exploration?", "Data exploration involves examining datasets to understand distributions, relationships, and quality before analysis."),  
        ("Types of data attributes?", "Nominal, ordinal, interval, and ratio — each represents different measurement levels and require different handling."),  
        ("What is data visualization?", "Visualization techniques like charts, histograms, and scatter plots help understand patterns and relationships in data."),  
        ("What is data preprocessing?", "Data preprocessing cleans, integrates, transforms, reduces, and discretizes data to prepare it for mining algorithms."),  
        ("Explain data cleaning.", "Data cleaning removes inconsistencies, duplicates, missing values, and noise to improve quality of analysis."),  
        ("Explain data integration.", "Data integration combines data from multiple sources into a unified dataset for coherent analysis."),  
        ("Explain data transformation.", "Data transformation converts data into appropriate formats, scales, or encoding to facilitate analysis."),  
        ("What is data reduction?", "Data reduction reduces data volume while preserving analytical integrity, using techniques like sampling, aggregation, or dimensionality reduction."),  
        ("Explain discretization.", "Discretization converts continuous attributes into categorical bins to simplify analysis."),  
        ("What is concept hierarchy generation?", "It maps low-level data values to higher-level concepts to support roll-up and aggregation in mining."),  
        ("Why is preprocessing important?", "Preprocessing improves data quality, reduces computational costs, and increases the accuracy of mining results."),  
        ("Explain data summarization.", "Data summarization provides aggregate descriptions, such as mean, standard deviation, or total counts for attributes."),  
        ("Define outliers.", "Outliers are data points that deviate significantly from normal patterns, potentially affecting analysis."),  
        ("Define missing values handling.", "Missing values can be handled by imputation, deletion, or estimation to maintain dataset completeness."),  
        ("Explain normalization.", "Normalization scales numeric values to a standard range, improving algorithm performance and comparability."),  
        ("What is attribute selection?", "Selecting the most relevant attributes reduces dimensionality and improves classifier performance."),  
        ("Explain data discretization techniques.", "Techniques include binning, clustering-based discretization, and decision tree-based discretization."),  
        ("What are the main issues in data mining?", "Issues include data quality, scalability, heterogeneous data sources, privacy concerns, and algorithm efficiency."),  
        ("What is data characterization?", "Data characterization summarizes general features of a target dataset, such as average sales of top products."),  
        ("What is data discrimination?", "Data discrimination compares features of a target class against contrasting classes to highlight differences."),  
        ("Explain statistical description of data.", "Statistical description uses mean, median, variance, and correlation to summarize datasets."),  
        ("What is concept hierarchy in OLAP?", "A hierarchy organizes dimension levels from fine-grained to aggregated, supporting drill-down and roll-up."),  
        ("Explain distributed data warehouse.", "Distributed warehouses store data across multiple sites and integrate them for analysis while maintaining consistency."),  
        ("Define virtual data warehouse.", "A virtual warehouse provides a logical view of integrated data without physically storing it, often using middleware."),  
        ("List roles of data warehouse manager.", "They manage data marts, handle user queries, control concurrency, maintain metadata, and ensure data quality."),  
        ("What are different types of cuboids?", "Cuboids are aggregates in different dimensional combinations: 0-D apex cuboid, n-D base cuboid, and intermediate-level cuboids.")  
    ]  
}  
  
# For brevity, duplicate pattern for Modules 3-6  
modules["Module 3: Classification"] = [("Q"+str(i)+"?","Answer for Q"+str(i)+".") for i in range(1,26)]  
modules["Module 4: Clustering"] = [("Q"+str(i)+"?","Answer for Q"+str(i)+".") for i in range(1,26)]  
modules["Module 5: Frequent Pattern & Association Mining"] = [("Q"+str(i)+"?","Answer for Q"+str(i)+".") for i in range(1,31)]  
modules["Module 6: Web Mining"] = [("Q"+str(i)+"?","Answer for Q"+str(i)+".") for i in range(1,31)]

Exp 5

# Naive Bayes Classifier Example in R  
  
# Attributes  
colour <- c('Red', 'Red', 'Red', 'Yellow', 'Yellow', 'Yellow', 'Yellow', 'Yellow', 'Red', 'Red')  
type\_car <- c('sports', 'sports', 'sports', 'sports', 'sports', 'SUV', 'SUV', 'SUV', 'SUV', 'sports')  
origin <- c('domestic', 'domestic', 'domestic', 'domestic', 'imported', 'imported', 'imported', 'domestic', 'imported', 'imported')  
stolen <- c('yes', 'no', 'yes', 'no', 'yes', 'no', 'yes', 'no', 'no', 'yes')  
  
# Priors  
p\_y <- sum(stolen == 'yes') / length(stolen)  
p\_n <- sum(stolen == 'no') / length(stolen)  
p\_yes <- sum(stolen == 'yes')  
p\_no <- sum(stolen == 'no')  
  
# Likelihood counters  
p\_r\_yes <- p\_r\_no <- 0  
p\_s\_yes <- p\_s\_no <- 0  
p\_d\_yes <- p\_d\_no <- 0  
  
# Calculate frequency counts  
for (i in 1:length(stolen)) {  
  if (colour[i] == 'Red' && stolen[i] == 'yes') p\_r\_yes <- p\_r\_yes + 1  
  if (colour[i] == 'Red' && stolen[i] == 'no') p\_r\_no <- p\_r\_no + 1  
  
  if (type\_car[i] == 'SUV' && stolen[i] == 'yes') p\_s\_yes <- p\_s\_yes + 1  
  if (type\_car[i] == 'SUV' && stolen[i] == 'no') p\_s\_no <- p\_s\_no + 1  
  
  if (origin[i] == 'domestic' && stolen[i] == 'yes') p\_d\_yes <- p\_d\_yes + 1  
  if (origin[i] == 'domestic' && stolen[i] == 'no') p\_d\_no <- p\_d\_no + 1  
}  
  
# Probabilities  
prob\_yes <- (p\_r\_yes/p\_yes) \* (p\_d\_yes/p\_yes) \* (p\_s\_yes/p\_yes) \* p\_y  
prob\_no <- (p\_r\_no/p\_no) \* (p\_d\_no/p\_no) \* (p\_s\_no/p\_no) \* p\_n  
  
cat("P(Yes) =", prob\_yes, "   P(No) =", prob\_no, "\n")  
  
if (prob\_yes > prob\_no) {  
  print("Hence it is classified as: YES (Stolen)")  
} else {  
  print("Hence it is classified as: NO (Not Stolen)")  
}

Another

import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.naive\_bayes import CategoricalNB  
from sklearn.preprocessing import LabelEncoder  
from sklearn.metrics import accuracy\_score, classification\_report  
  
# Load dataset (Car Evaluation Dataset from Kaggle)  
# Make sure 'car\_evaluation.csv' is downloaded from Kaggle and in the same folder  
df = pd.read\_csv("car\_evaluation.csv")  
  
print("First 5 rows of dataset:")  
print(df.head())  
  
# Encode categorical values into numbers  
encoders = {}  
for column in df.columns:  
    le = LabelEncoder()  
    df[column] = le.fit\_transform(df[column])  
    encoders[column] = le  
  
# Features (X) and Target (y)  
X = df.drop("class", axis=1)   # features  
y = df["class"]                # target (car acceptability)  
  
# Split into train and test  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)  
  
# Train Naive Bayes classifier  
model = CategoricalNB()  
model.fit(X\_train, y\_train)  
  
# Predictions  
y\_pred = model.predict(X\_test)  
  
# Results  
print("\nAccuracy:", accuracy\_score(y\_test, y\_pred))  
print("\nClassification Report:\n", classification\_report(y\_test, y\_pred, target\_names=encoders["class"].classes\_))

Exp 6

Data(iris)

# Normalization

Normalize <- function(x) { (x – min(x)) / (max(x) – min(x)) }

Iris\_norm <- as.data.frame(lapply(iris[1:4], normalize))

# Histogram before & after normalization

Par(mfrow=c(1,2))

Hist(iris$Sepal.Length, main=”Original”, col=”skyblue”)

Hist(iris\_norm$Sepal.Length, main=”Normalized”, col=”lightgreen”)

Another

# Install required packages (run only once if not installed)

Install.packages(“rpart”)

Install.packages(“rpart.plot”)

# Load libraries

Library(rpart)

Library(rpart.plot)

# Train Decision Tree

Model <- rpart(Species ~ ., data=iris, method=”class”)

# Plot Tree

Rpart.plot(model, main=”Decision Tree”)

Another

Set.seed(123)

Iris\_km <- kmeans(iris[,1:4], centers=3)

# Plot clustering

Plot(iris[,1:2], col=iris\_km$cluster, pch=19,

Main=”K-Means Clustering”, xlab=”Sepal.Length”, ylab=”Sepal.Width”)

Points(iris\_km$centers[,1:2], col=1:3, pch=8, cex=2)

Another

# Install if not already installed

Install.packages(“arules”)

Install.packages(“arulesViz”)

Library(arules)

Library(arulesViz)

# Load dataset

Data(“Groceries”)

# Generate rules

Rules <- apriori(Groceries, parameter=list(supp=0.01, conf=0.5))

# Show top 5 rules

Inspect(head(rules, 5))

# Plot top rules (graph)

Plot(head(rules, 10), method=”graph”, engine=”igraph”)

Exp 7

Print(“K-means Clustering”)

Dataset = [2, 3, 4, 10, 11, 12, 20, 25, 30]

M1 = int(input(“Enter m1: “))

M2 = int(input(“Enter m2: “))

P1 = p2 = 0

While m1 != p1 or m2 != p2:

K1, k2 = [], []

For I in dataset:

If abs(I – m1) < abs(I – m2):

K1.append(i)

Else:

K2.append(i)

Avg1 = sum(k1) / len(k1)

Avg2 = sum(k2) / len(k2)

Print(“Clusters:”, k1, k2)

P1, p2 = m1, m2

M1, m2 = avg1, avg2

Print(“m1 = {}, m2 = {}”.format(m1, m2))

Exp 8

Import numpy as np

Import pandas as pd

Import matplotlib.pyplot as plt

From scipy.cluster import hierarchy

From scipy.spatial.distance import pdist, squareform

# --- Data ---

A = np.array([2, 8, 9, 1, 8.5])

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

Labels = [‘P1’, ‘P2’, ‘P3’, ‘P4’, ‘P5’]

# Build DataFrame

Data = pd.DataFrame({‘a’: a, ‘b’: b}, index=labels)

Print(“\nInput points:\n”, data)

# --- Scatter plot ---

Plt.figure(figsize=(8, 5))

Plt.scatter(data[‘a’], data[‘b’])

For idx, row in data.iterrows():

Plt.annotate(idx, (row[‘a’], row[‘b’]), fontsize=12)

Plt.title(‘Scatter Plot of points’)

Plt.xlabel(‘a’)

Plt.ylabel(‘b’)

Plt.show()

# --- Distance matrix ---

Dist\_vector = pdist(data[[‘a’, ‘b’]], metric=’euclidean’)

Dist\_matrix = squareform(dist\_vector)

Dist\_df = pd.DataFrame(dist\_matrix, index=labels, columns=labels)

Np.fill\_diagonal(dist\_df.values, np.nan) # hide 0 self-distances

Print(“\nPairwise distance matrix:\n”, dist\_df)

# --- Closest pair ---

Mask = np.triu(np.ones(dist\_matrix.shape), k=1).astype(bool)

Masked = np.where(mask, dist\_matrix, np.nan)

Min\_val = np.nanmin(masked)

Min\_pos = np.where(masked == min\_val)

I, j = min\_pos[0][0], min\_pos[1][0]

Print(f”\nClosest pair: {labels[i]} and {labels[j]} with distance = {min\_val:.4f}”)

# --- Dendrogram ---

Plt.figure(figsize=(6, 5))

Plt.title(“Dendrogram (single linkage)”)

Z = hierarchy.linkage(data[[‘a’, ‘b’]], method=’single’, metric=’euclidean’)

Hierarchy.dendrogram(Z, labels=labels)

Plt.xlabel(‘Points’)

Plt.ylabel(‘Distance’)

Plt.show()

Exp 9

import numpy as np  
import pandas as pd  
import plotly.express as px  
from mlxtend.frequent\_patterns import apriori, association\_rules  
from mlxtend.preprocessing import TransactionEncoder  
import warnings  
  
# -------------------- Hide Warnings --------------------  
warnings.filterwarnings("ignore", category=DeprecationWarning)  
warnings.filterwarnings("ignore", category=FutureWarning)  
  
# -------------------- Load Dataset --------------------  
data = pd.read\_csv("spotify-2023.csv", encoding='ISO-8859-1')  
  
# -------------------- Top 10 Frequently Sold Products --------------------  
print("Top 10 frequently sold products (Tabular Representation)\n")  
top\_10\_tracks = data['track\_name'].value\_counts().head(10)  
print(top\_10\_tracks)  
  
# Plot top 10 songs  
fig = px.bar(  
    x=top\_10\_tracks.index,  
    y=top\_10\_tracks.values,  
    labels={"x": "Songs", "y": "Count"},  
    title="Top 10 Frequently Sold Products (Graphical Representation)"  
)  
fig.show()  
  
# -------------------- Release Year Analysis --------------------  
if "released\_year" in data.columns:  
    year\_counts = data['released\_year'].value\_counts().sort\_index()  
  
    fig1 = px.bar(  
        x=year\_counts.index,  
        y=year\_counts.values,  
        labels={"x": "Release Year", "y": "Count"},  
        title="Number of Tracks Released Per Year"  
    )  
    fig1.show()  
else:  
    print("\n⚠️ 'released\_year' column not found. Skipping year analysis.")  
  
# -------------------- Apriori Algorithm with mlxtend --------------------  
if "released\_year" in data.columns:  
    # Prepare transactions (each year = basket of songs)  
    transactions = data.groupby("released\_year")['track\_name'].apply(list)  
  
    # Convert transactions into one-hot encoded DataFrame  
    te = TransactionEncoder()  
    te\_array = te.fit(transactions).transform(transactions)  
    df\_transactions = pd.DataFrame(te\_array, columns=te.columns\_)  
  
    # Apply Apriori  
    freq\_items = apriori(df\_transactions, min\_support=0.003, use\_colnames=True)  
  
    # Generate Association Rules  
    rules = association\_rules(freq\_items, metric="lift", min\_threshold=3)  
  
    # Display rules  
    print("\nAssociation Rule Mining Results:\n")  
    for idx, row in rules.iterrows():  
        if len(row['antecedents']) == 1 and len(row['consequents']) == 1:  # Keep only 1->1 rules  
            print("Rule: ", list(row['antecedents'])[0], " -> ", list(row['consequents'])[0])  
            print("Support: ", round(row['support'], 4))  
            print("Confidence: ", round(row['confidence'], 4))  
            print("Lift: ", round(row['lift'], 4))  
            print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")  
else:  
    print("\n⚠️ 'released\_year' column not found. Skipping Apriori analysis.")

Exp 10

import pandas as pd  
  
# Points  
points = ['P1', 'P2', 'P3', 'P4']  
  
# Adjacency matrix for the graph  
g = pd.DataFrame({  
    'Point': points,  
    'P1': [0, 1, 1, 0],  
    'P2': [0, 0, 0, 1],  
    'P3': [1, 1, 0, 1],  
    'P4': [0, 0, 1, 0]  
})  
  
# Set index  
g = g.set\_index('Point')  
print("Adjacency Matrix:")  
print(g, "\n")  
  
# Out-degree sums for each column  
A = g['P1'].sum()  
B = g['P2'].sum()  
C = g['P3'].sum()  
D = g['P4'].sum()  
print(f"A = {A}, B = {B}, C = {C}, D = {D}\n")  
  
# Convert adjacency to records  
rec = g.to\_records(index=False)  
row = list(rec)  
  
Ain, Bin, Cin, Din = row[0], row[1], row[2], row[3]  
  
# Iteration 1  
I1 = [0.25, 0.25, 0.25, 0.25]  
print("Iteration 1")  
print("P1, P2, P3, P4")  
print(I1, "\n")  
  
# Iteration 2  
I2 = [0, 0, 0, 0]  
  
# P1  
for i in range(4):  
    if Ain[i] == 1:  
        if i == 1:  
            I2[0] += round(I1[0] / B, 3)  
        elif i == 2:  
            I2[0] += round(I1[0] / C, 3)  
        elif i == 3:  
            I2[0] += round(I1[0] / D, 3)  
I2[0] = round(I2[0], 3)  
  
# P2  
for i in range(4):  
    if Bin[i] == 1:  
        if i == 0:  
            I2[1] += round(I1[1] / A, 3)  
        elif i == 2:  
            I2[1] += round(I1[1] / C, 3)  
        elif i == 3:  
            I2[1] += round(I1[1] / D, 3)  
I2[1] = round(I2[1], 3)  
  
# P3  
for i in range(4):  
    if Cin[i] == 1:  
        if i == 0:  
            I2[2] += round(I1[2] / A, 3)  
        elif i == 1:  
            I2[2] += round(I1[2] / B, 3)  
        elif i == 3:  
            I2[2] += round(I1[2] / D, 3)  
I2[2] = round(I2[2], 3)  
  
# P4  
for i in range(4):  
    if Din[i] == 1:  
        if i == 0:  
            I2[3] += round(I1[3] / A, 3)  
        elif i == 1:  
            I2[3] += round(I1[3] / B, 3)  
        elif i == 2:  
            I2[3] += round(I1[3] / C, 3)  
I2[3] = round(I2[3], 3)  
  
print("Iteration 2")  
print("P1, P2, P3, P4")  
print(I2, "\n")  
  
# Iteration 3  
I3 = [0, 0, 0, 0]  
  
# P1  
for i in range(4):  
    if Ain[i] == 1:  
        if i == 1:  
            I3[0] += round(I2[0] / B, 3)  
        elif i == 2:  
            I3[0] += round(I2[0] / C, 3)  
        elif i == 3:  
            I3[0] += round(I2[0] / D, 3)  
I3[0] = round(I3[0], 3)  
  
# P2  
for i in range(4):  
    if Bin[i] == 1:  
        if i == 0:  
            I3[1] += round(I2[1] / A, 3)  
        elif i == 2:  
            I3[1] += round(I2[1] / C, 3)  
        elif i == 3:  
            I3[1] += round(I2[1] / D, 3)  
I3[1] = round(I3[1], 3)  
  
# P3  
for i in range(4):  
    if Cin[i] == 1:  
        if i == 0:  
            I3[2] += round(I2[2] / A, 3)  
        elif i == 1:  
            I3[2] += round(I2[2] / B, 3)  
        elif i == 3:  
            I3[2] += round(I2[2] / D, 3)  
I3[2] = round(I3[2], 3)  
  
# P4  
for i in range(4):  
    if Din[i] == 1:  
        if i == 0:  
            I3[3] += round(I2[3] / A, 3)  
        elif i == 1:  
            I3[3] += round(I2[3] / B, 3)  
        elif i == 2:  
            I3[3] += round(I2[3] / C, 3)  
I3[3] = round(I3[3], 3)  
  
print("Iteration 3")  
print("P1, P2, P3, P4")  
print(I3, "\n")  
  
print("The page with the minimum value has PageRank 1, and the page with the maximum value has PageRank 4")