DSA 2040 US 2025 End Semester Exam

Practical Exam: Data Warehousing and Data Mining
Exam Type: Practical
Total Marks: 100
Instructions:

- This exam is designed to test your practical skills in data warehousing and data mining. You will need access to a computer with Python (version 3.x), libraries such as pandas, numpy, scikit-learn, sqlite3, and any SQL client (e.g., DB Browser for SQLite).
- For datasets: You have the option to either use publicly available datasets where specified (e.g., from UCI ML Repository or scikit-learn built-in) or generate synthetic data yourself. If generating synthetic data, ensure it mimics the structure and scale of the described dataset (e.g., similar columns, row count around 500-1000 for practicality). Do not use any data from Kaggle; if choosing public datasets, stick to UCI or built-in libraries. Include the generation code in your submission if you opt for synthetic data.
- All code must be well-commented, modular, and error-free. Include explanations in markdown cells or comments.
- Submission: Create a GitHub repository named "DSA 2040_Practical_Exam_[YourName and last three digits of ID]". Upload everything attempted, including all code files (.py or .ipynb), generated datasets (as CSV or included in code), SQL scripts, reports (PDF or Markdown), screenshots of outputs, visualizations, and any partial attempts or debugging notes. Include a README.md with an overview of your submission, which datasets you used/generated, how to run your code, and a self-assessment of what was completed.
- Plagiarism: Zero tolerance; use your own work.
- Marks Breakdown: Data Warehousing (50 marks), Data Mining (50 marks).
- Resources: You may refer to official documentation (e.g., pandas docs, scikit-learn docs) but not external code snippets or forums during the exam.

Section 1: Data Warehousing (50 Marks)

Task 1: Data Warehouse Design (15 Marks)

Scenario: You are designing a data warehouse for a retail company that sells products across categories (e.g., electronics, clothing). The company tracks sales, customers, products, and time. Key requirements:

- Support queries like total sales by product category per quarter, customer demographics analysis, and inventory trends.

Instructions:

1. Design a **star schema** for this data warehouse. Include at least one fact table and 3-4 dimension tables. Specify:
    - Fact table: Measures (e.g., sales amount, quantity) and foreign keys.
    - Dimension tables: Attributes (e.g., customer ID, name, location; product ID, name, category; time ID, date, quarter, year).
    - Use a diagram (hand-drawn or using tools like Draw.io; export as image and include in submission).
2. Explain why you chose star schema over snowflake (2-3 sentences).
3. Write SQL CREATE TABLE statements for the fact and dimension tables (assume SQLite syntax).

Marks Allocation:

- Schema design and diagram: 8 marks
- Explanation: 3 marks
- SQL statements: 4 marks

Submission: Include schema diagram (image file), explanation in README.md, and SQL script (.sql file).

Task 2: ETL Process Implementation (20 Marks)

Dataset Option: Either use the "Online Retail" dataset from UCI ML Repository (download as CSV: https://archive.ics.uci.edu/dataset/352/online+retail) with columns like InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country, OR generate synthetic data in Python (e.g., using pandas and faker library if available, or manually with random values). For synthetic: Create ~1000 rows with similar columns (e.g., random invoices, products, quantities 1-50, prices 1-100, dates over 2 years, 100 unique customers, 5-10 countries). Include generation code if chosen.

Instructions:

1. **Extract:** Write Python code to read the CSV file (or generate and use the DataFrame) into a pandas DataFrame. Handle any missing values or data types (e.g., convert InvoiceDate to datetime).
2. **Transform:**
    - Calculate a new column: TotalSales = Quantity * UnitPrice.
    - Create dimension-like extracts: Group by CustomerID to create a customer summary (e.g., total purchases, country).
    - Filter data for sales in the last year (assume current date as August 12, 2025).
    - Handle outliers: Remove rows where Quantity < 0 or UnitPrice <= 0.
3. **Load:** Use sqlite3 in Python to create a database file (retail_dw.db). Load the transformed data into a fact table (SalesFact) and at least two dimension tables (e.g., CustomerDim, TimeDim).
4. Write a function to perform the full ETL and log the number of rows processed at each stage.

Marks Allocation:

- Extraction and transformation code (including generation if chosen): 8 marks
- Loading to database: 6 marks
- Functionality and error handling: 4 marks
- Comments and logging: 2 marks

Submission: Python script (.py or .ipynb) named etl_retail.py. Include the .db file if small, generated CSV if applicable, or screenshots of table contents post-load.

Task 3: OLAP Queries and Analysis (15 Marks)

Using the Data Warehouse from Task 2:

Instructions:

1. Write and execute 3 OLAP-style SQL queries:
    - Roll-up: Total sales by country and quarter (group by country and quarter from TimeDim).
    - Drill-down: Sales details for a specific country (e.g., UK or a generated one) by month.
    - Slice: Total sales for electronics category (assume you categorize products; add a category column during transform or generation if needed).
2. Use Python (pandas or SQL) to visualize one query result (e.g., bar chart of sales by country using matplotlib). Save as image.
3. Analyze results: In a short report (200-300 words), discuss insights (e.g., top-selling countries, trends) and how the warehouse supports decision-making. Note if using synthetic data affected realism.

Marks Allocation:

- SQL queries: 6 marks (2 each)
- Visualization: 4 marks
- Analysis report: 5 marks

Submission: SQL queries in a .sql file, visualization image, report in PDF/Markdown.

Section 2: Data Mining (50 Marks)

Task 1: Data Preprocessing and Exploration (15 Marks)

Dataset Option: Either use the "Iris" dataset from scikit-learn (built-in: from sklearn.datasets import load_iris) with features: sepal length, sepal width, petal length, petal width, and class (species), OR generate synthetic data (e.g., using numpy.random for 150 samples, 3 clusters mimicking species with Gaussian distributions for features). Include generation code if chosen.

Instructions:

1. Load the dataset in Python using pandas or scikit-learn (or generate).
2. Preprocess:
    - Handle any missing values (demonstrate checks).
    - Normalize features using Min-Max scaling.
    - Encode the class label if needed (e.g., one-hot for some models).
3. Explore:
    - Compute summary statistics (mean, std, etc.) using pandas.describe().
    - Visualize: Pairplot (using seaborn) and correlation heatmap.
    - Identify any potential outliers using boxplots.
4. Write a function to split data into train/test (80/20).

Marks Allocation:

- Loading and preprocessing (including generation if chosen): 6 marks
- Exploration and visualizations: 6 marks
- Split function: 3 marks

Submission: Python script named preprocessing_iris.py. Include visualizations as images and generated data code/CSV if applicable.

Task 2: Clustering (15 Marks)

Using Preprocessed Data from Task 1:

Instructions:

1. Apply K-Means clustering (from scikit-learn) with k=3 (since 3 species or clusters).
    - Fit the model on features (exclude class).
    - Predict clusters and compare with actual classes using Adjusted Rand Index (ARI).
2. Experiment: Try k=2 and k=4; plot elbow curve to justify optimal k.
3. Visualize clusters (e.g., scatter plot of petal length vs width, colored by cluster).
4. Analyze: In 150-200 words, discuss cluster quality, misclassifications, and real-world applications (e.g., customer segmentation). Note if synthetic data impacted results.

Marks Allocation:

- Implementation and metrics: 7 marks
- Experimentation and visualization: 4 marks
- Analysis: 4 marks

Submission: Python script named clustering_iris.py. Images and analysis in report.

Task 3: Classification and Association Rule Mining (20 Marks)

Part A: Classification (10 Marks)
Using Preprocessed Data from Task 1:

1. Train a Decision Tree classifier (scikit-learn) on train set.
    - Predict on test set; compute accuracy, precision, recall, F1-score.
    - Visualize the tree (using plot_tree).
2. Compare with another classifier (e.g., KNN with k=5); report which is better and why.

Part B: Association Rule Mining (10 Marks)
Dataset Option: Generate synthetic transactional data (e.g., 20-50 transactions, each a list of 3-8 items from a pool of 20 items like ['milk', 'bread', 'beer', 'diapers', 'eggs', etc.]; use random.choices to create baskets with patterns, like frequent co-occurrences). Use Python lists or pandas. Include generation code.

1. Apply Apriori algorithm (use mlxtend library; assume installed or provide alternative implementation if needed).
    - Find rules with min_support=0.2, min_confidence=0.5.
    - Sort by lift and display top 5 rules.
2. Analyze: Discuss one rule's implications (e.g., for retail recommendations).

Marks Allocation:

- Classification implementation and metrics: 5 marks
- Comparison and visualization: 5 marks
- Apriori implementation (including generation): 5 marks
- Rule analysis: 5 marks

Submission: Python script named mining_iris_basket.py. /ipynb.Include outputs, generated data, and analysis.

Total Marks: 100
Ensure all code runs without errors. Test your submissions locally. Include all attempted work in the repo, even if incomplete, for partial credit. If generating data, ensure it's reproducible (use seeds). Good luck!