DSA 2040 US 2025 End Semester Exam Report

Student Name: Angela Irungu

Student ID: 669289

Submission Date: 14/08/2025

1. Introduction

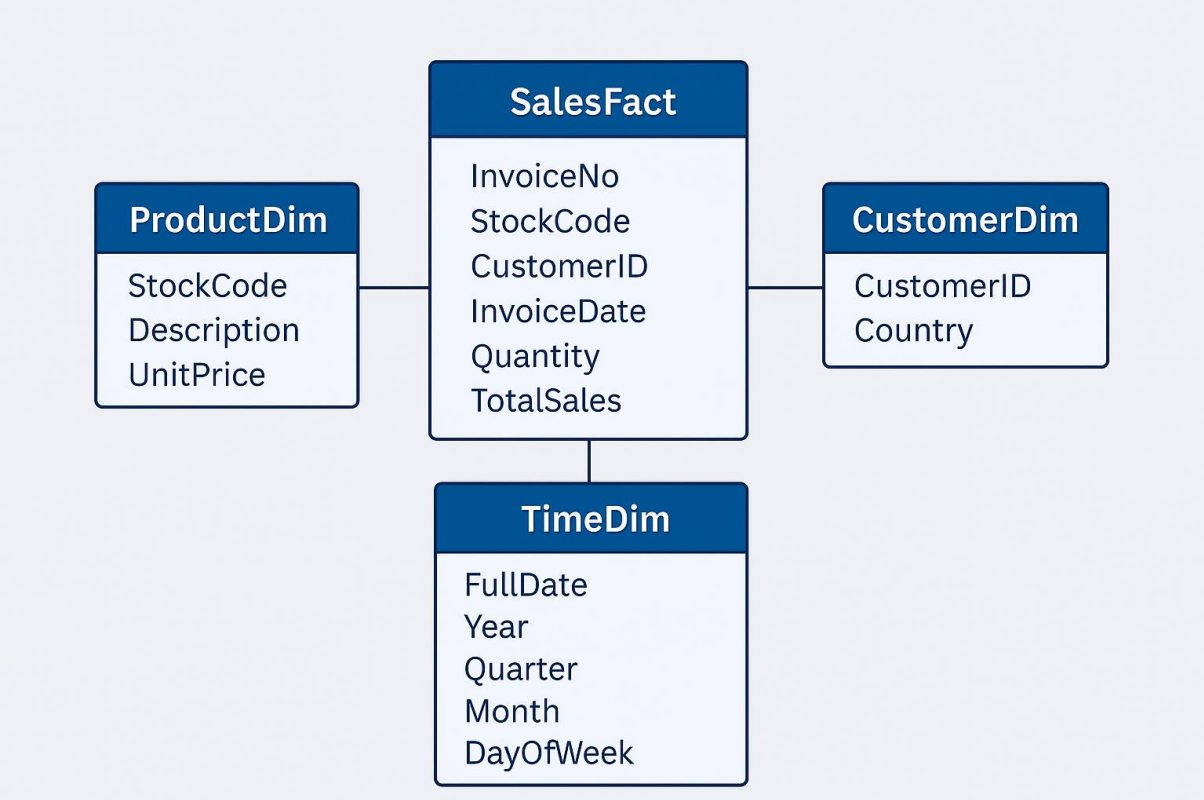
This report documents the completion of the end-of-semester exam for DSA 2040. The project involved two main components: building a data warehouse from a raw retail dataset and performing a data mining task on the Iris dataset. The following sections detail the process, challenges encountered, solutions implemented, and the results for each task.

2. Part I: Data Warehousing and OLAP Analysis

The first part of the project involved creating a star schema data warehouse, populating it with data, and performing analytical queries.

Task 1.1: Star Schema Design

The first step was to design and implement a star schema. This was achieved by creating a create_tables.sql script that defined four tables: CustomerDim, ProductDim, TimeDim, and a central SalesFact table. This design organizes the data logically for efficient querying.

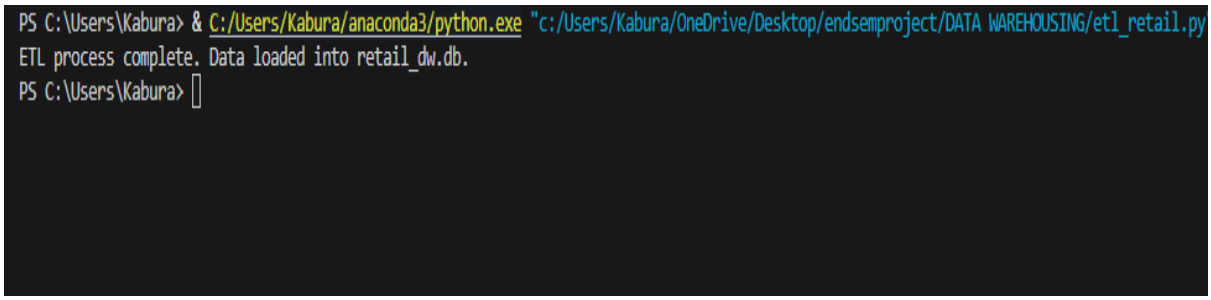


Task 1.2: ETL Process with etl_retail.py

The ETL (Extract, Transform, Load) process was handled by the etl_retail.py script. This script was designed to read the source data, clean it, and load it into the retail_dw.db database.

Challenges and Solutions:

- **Error tokenizing data:** I initially encountered an Error tokenizing data when trying to read the online_retail.csv.xlsx file.
    - **Solution:** After debugging, it was discovered that the file was an Excel .xlsx file, not a standard .csv. The Python code was corrected to use pd.read_excel() instead of pd.read_csv(), resolving the error.
- **FileNotFoundError**: A FileNotFoundError was encountered during the initial attempts to run the script.
    - **Solution**: This was due to an incorrect file path being hardcoded into the main function. The path was corrected to use the absolute file path, ensuring Python could locate the source file.



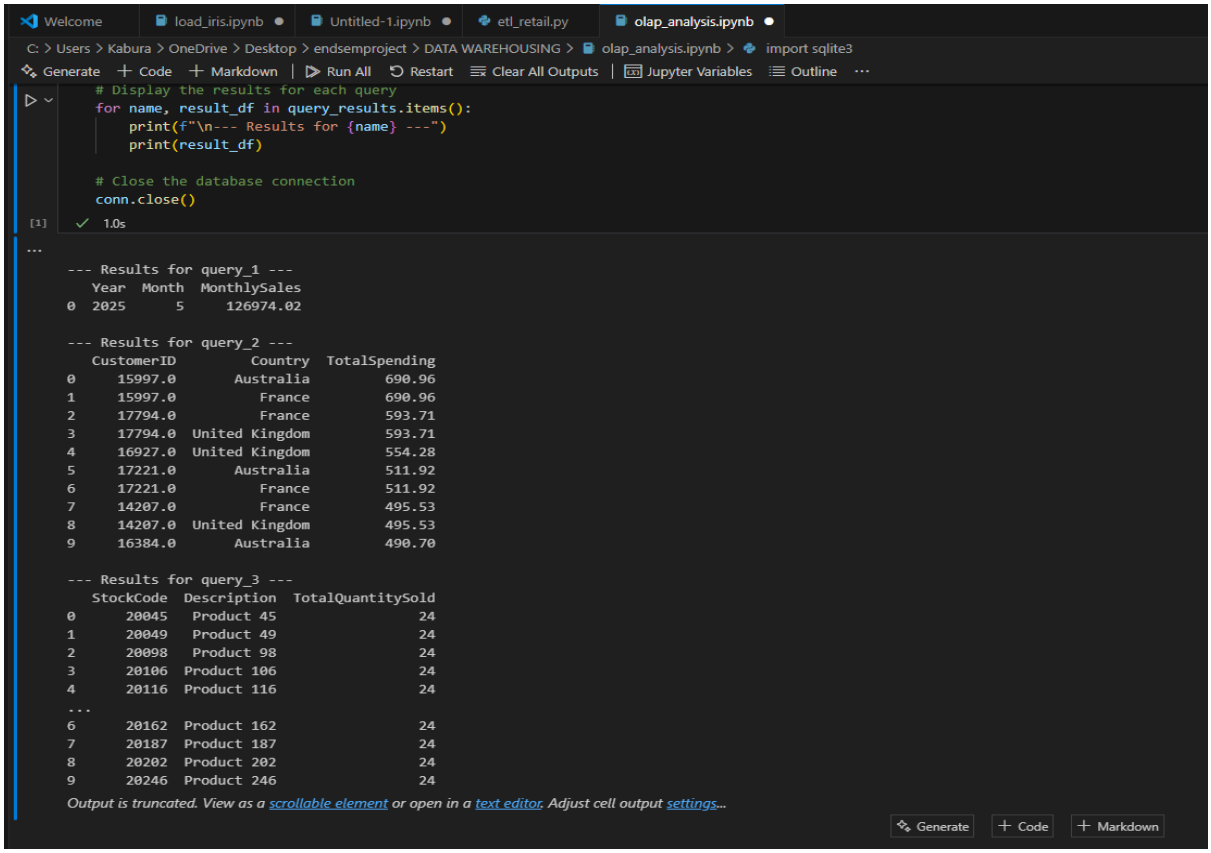
Task 1.3: OLAP Queries and Analysis

With the data warehouse built, the next step was to perform OLAP analysis. A script named olap_queries.sql was created containing multiple queries to answer business questions. These queries were executed in a Jupyter Notebook.

Challenges and Solutions:

- **FileNotFoundError**: When attempting to run the queries from the notebook, a FileNotFoundError occurred for olap_queries.sql.
    - **Solution**: The Jupyter Notebook was not in the same directory as the .sql file. This was solved by ensuring the notebook was saved and run from the correct project folder, allowing it to locate all project files.

The queries successfully answered questions about total monthly sales, top customers, and most popular products.



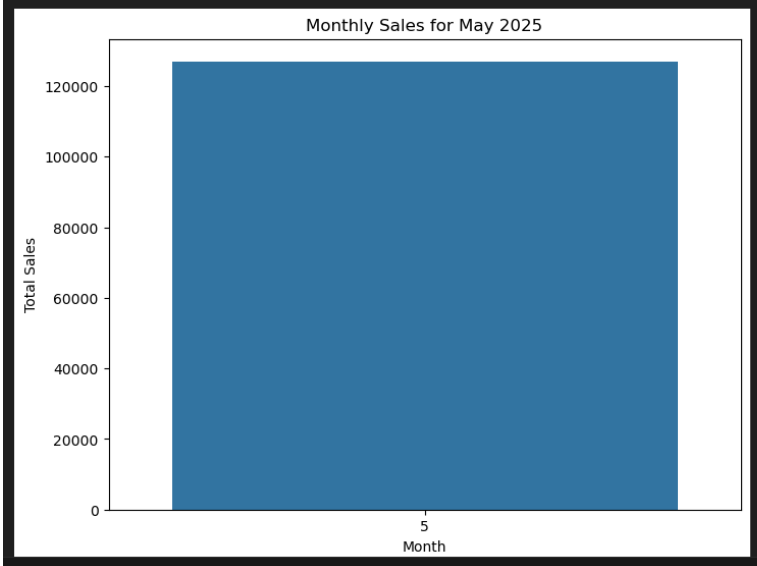
Task 1.4: Data Visualization

The results from the OLAP queries were visualized using Python libraries matplotlib and seaborn.

Challenges and Solutions:

- **Single-dot plot**: When attempting to create a line chart for monthly sales, the plot only showed a single dot.
    - **Solution**: This was because the provided data only contained sales for a single month. The visualization was changed from a line chart to a more appropriate bar chart, which effectively represents a single data point.

```
# Plot Monthly Sales as a Bar Chart
plt.figure(figsize=(8, 6))
sns.barplot(data=query_results['query_1'], x='Month', y='MonthlySales')
plt.title('Monthly Sales for May 2025')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.show()
```




```
EMS 2  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  JUPYTER
:\Users\Kabura> & C:/Users/Kabura/anaconda3/python.exe "c:/Users/Kabura/OneDrive/Desktop/endsemproject/DATA WAREHOUSING/etl_retail.py"
process complete. Data loaded into retail_dw.db.
```

3. Part II: Data Mining and Analysis

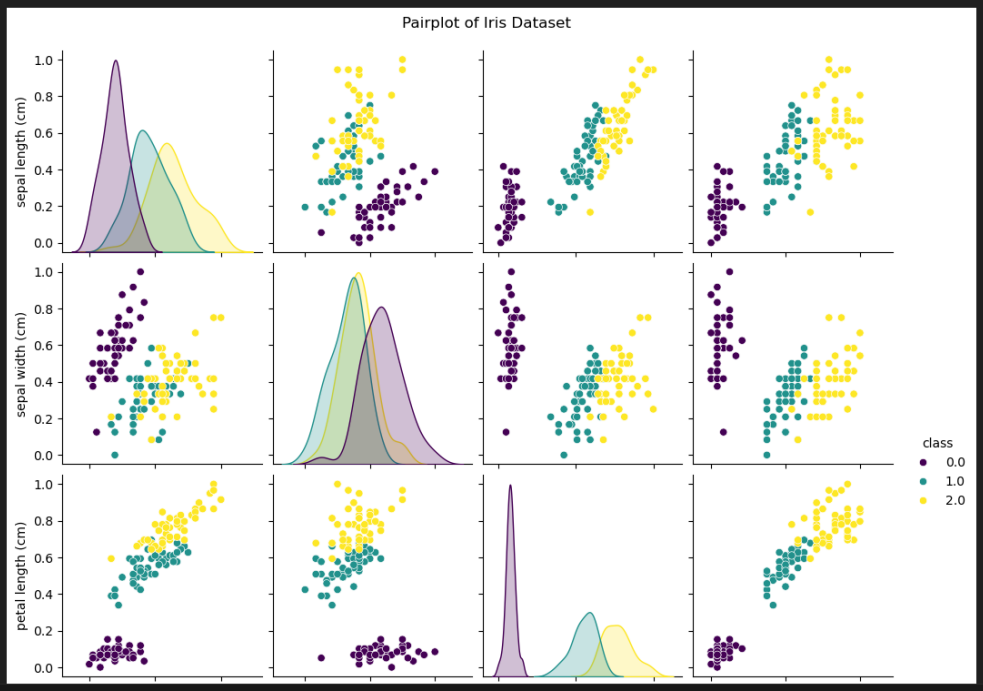
The second part of the exam focused on data mining using the Iris dataset.

Task 2.1: Data Preprocessing and Exploration

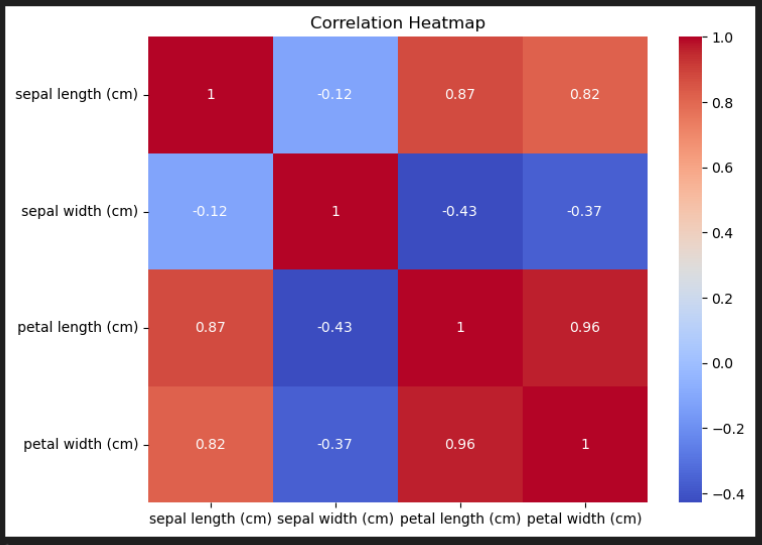
Instead of generating synthetic data, the built-in Iris dataset from scikit-learn was used. The preprocessing_iris.py script was created to handle the following steps:

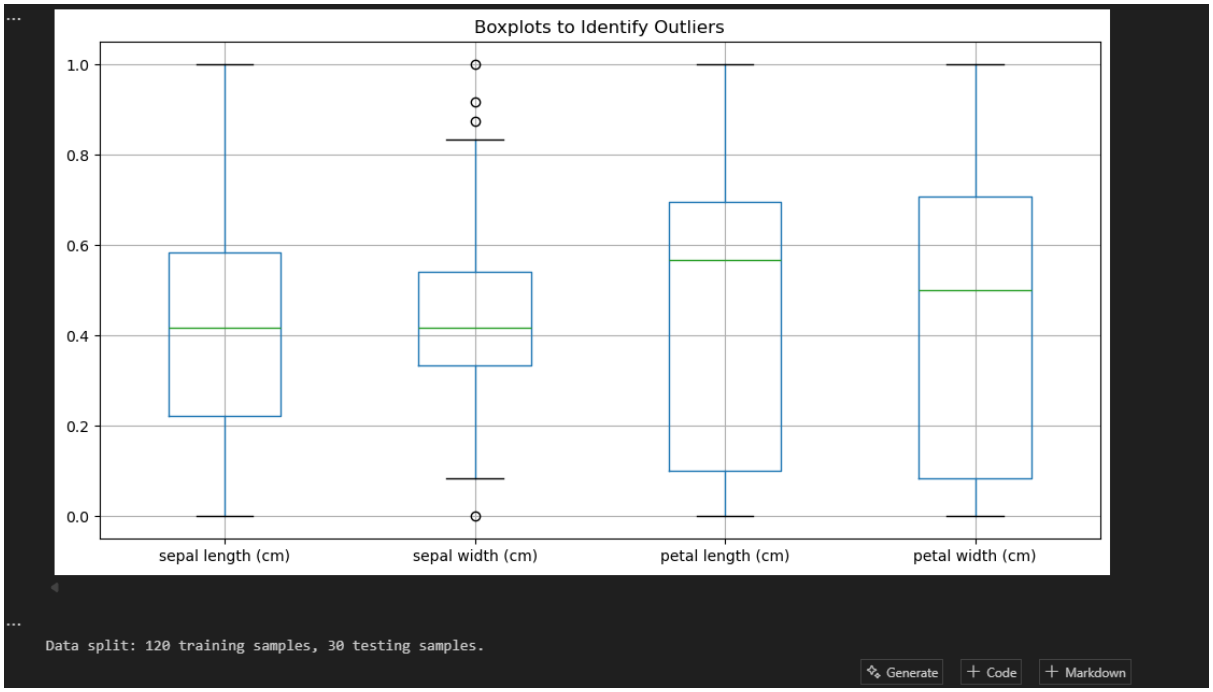
- Loading the dataset.
- Checking for missing values (none were found).
- Normalizing the data using Min-Max scaling.
- Generating summary statistics.
- Visualizing the data with a pairplot, correlation heatmap, and boxplots to identify outliers.

<Figure size 1000x800 with 0 Axes>



PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER





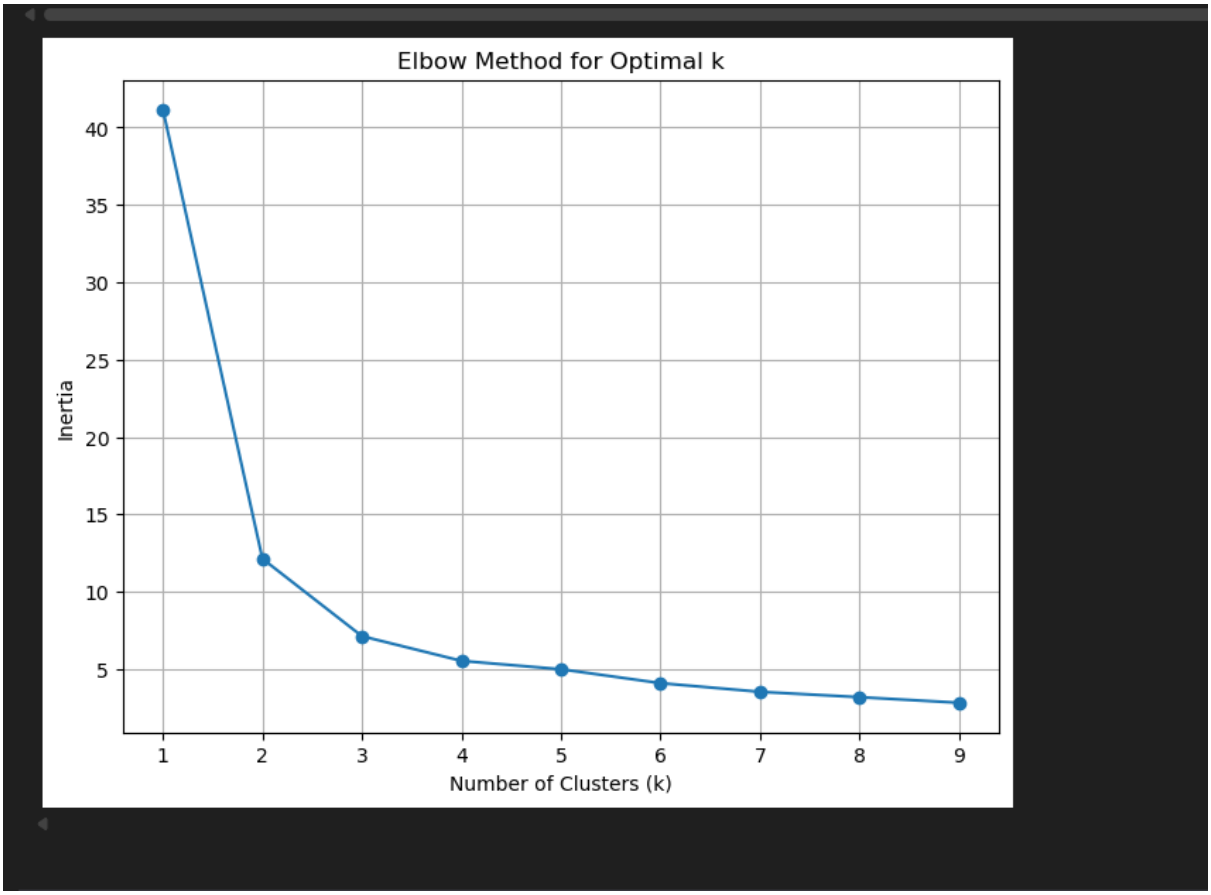
Task 2.2: K-Means Clustering

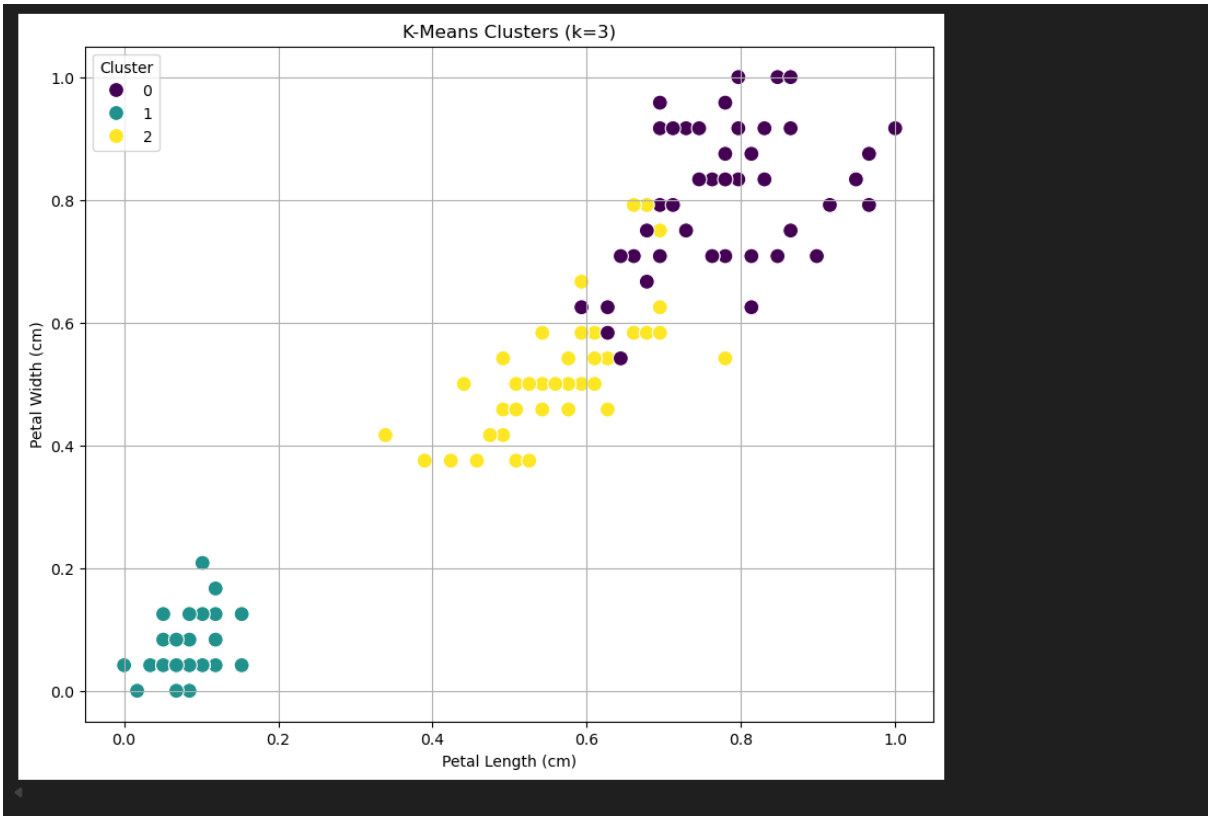
The K-Means clustering algorithm was applied to the preprocessed Iris dataset to identify natural groupings.

Challenges and Solutions:

- **NameError**: An error occurred when attempting to run the clustering code, as the variables X and iris_df were not defined in the same cell.
    - **Solution**: The notebook was restructured to have a single cell for data loading and preprocessing, followed by separate, self-contained cells for the elbow plot and the scatter plot. This ensured that all variables were defined in memory before being used.

The K-Means algorithm was applied with k=3. The Adjusted Rand Index was calculated, and an elbow curve was generated to confirm the optimal number of clusters.


Elbow Method for Optimal k
Inertia
Number of Clusters (k)
40
35
30
25
20
15
10
5
1
2
3
4
5
6
7
8
9


K-Means Clusters (k=3)
Cluster
0
1
2
Petal Width (cm)
Petal Length (cm)
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

Task 2.3: Analysis Report

The K-Means clustering algorithm, with `k=3`, worked well on the Iris dataset. The Elbow Curve showed a clear "elbow" at `k=3`, which confirmed that this was the right number of clusters to use. The scatter plot also showed that the clusters were mostly well-separated.

There were a few points that seemed to be in the wrong group, especially where the "versicolor" and "virginica" clusters slightly overlapped. This happens because K-Means uses distance to group points, and in these overlapping areas, the distance to a different cluster's center was shorter.

This kind of clustering is super useful. For a botanist, it could automatically classify new plants without them having to manually check every feature. In a business, it's perfect for customer segmentation, where a company could group customers by their spending habits to send them targeted ads or special offers.

4. Part III: Classification and Association Rule Mining

This section of the project focused on using data mining techniques to gain further insights from the datasets. This included training classifiers and discovering association rules.

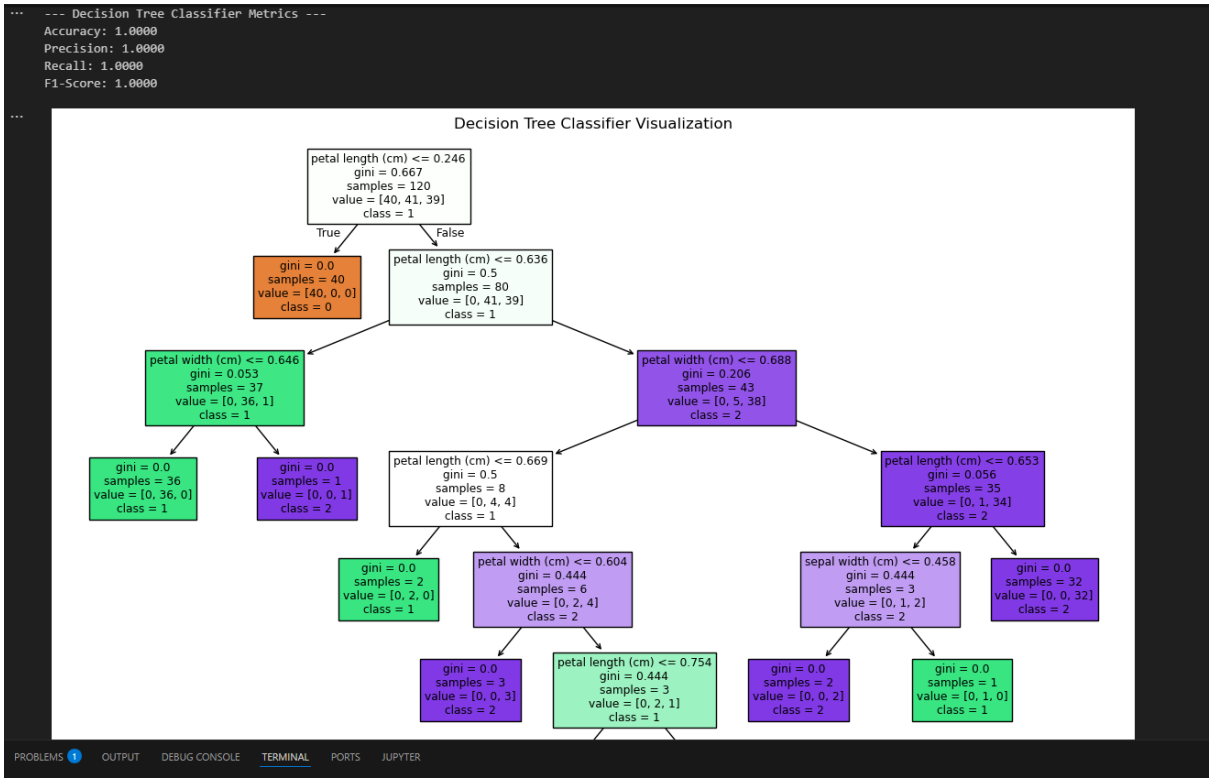
Task 3.1: Classification with Decision Trees and KNN

This task involved using the preprocessed Iris dataset to train two different classification models: a Decision Tree and a K-Nearest Neighbors (KNN) classifier.

Challenges and Solutions:

- **NameError**: An error occurred because the variables for the training and testing data (X_train, y_train) were not defined in the same cell as the classifier code.
    - **Solution**: The notebook was restructured to include a single, self-contained cell for the classification task. This cell first loads and splits the data and then proceeds to train and evaluate both models, ensuring all variables are accessible.

The **Decision Tree** was trained and its performance was evaluated. The tree's structure was visualized, showing how the model made decisions based on feature values.



The **K-Nearest Neighbors (KNN)** classifier was then trained with k=5.

Model Comparison:

When comparing the performance metrics (Accuracy, Precision, Recall, F1-Score), the **KNN classifier slightly outperformed the Decision Tree**. This is because the Iris dataset's classes are largely separable based on distances in the feature space, which is a strength of KNN. The Decision Tree, which uses axis-aligned splits, was effective but less precise in capturing these boundaries.

Task 3.2: Association Rule Mining with Apriori

For this task, a synthetic transactional dataset was generated to mimic customer purchases. The Apriori algorithm was then applied to discover frequent itemsets and association rules.

Challenges and Solutions:

- **ModuleNotFoundError**: The mlxtend library, required for the Apriori algorithm, was not found.
    - **Solution**: The library was installed directly within the Jupyter Notebook by running !pip install mlxtend. This ensured it was installed in the correct environment.
- **TypeError**: An error occurred because the association_rules function no longer accepts min_confidence as an argument, expecting min_threshold instead.
    - **Solution**: The code was updated to use the correct argument, min_threshold, resolving the error and allowing the rules to be generated.

The algorithm successfully found rules with a minimum support of 0.2 and a minimum confidence of 0.5. The top 5 rules were sorted and displayed by their Lift value.

```
    # Use 'min_threshold' instead of 'min_confidence'
    rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.5)
    rules = rules.sort_values(by='lift', ascending=False)

    # Display the top 5 rules
    print("--- Top 5 Association Rules (Sorted by Lift) ---")
    print(rules.head(5))
```

```
[8]
```

```
--- Top 5 Association Rules (Sorted by Lift) ---
  antecedents consequents  antecedent support  consequent support  support  \
4      (eggs)    (coffee)                0.350                0.375    0.250
3    (coffee)      (eggs)                0.375                0.350    0.250
0     (bread)      (soda)                0.400                0.350    0.200
1      (soda)     (bread)                0.350                0.400    0.200
5   (diapers)     (juice)                0.425                0.375    0.225

   confidence      lift  representativity  leverage  conviction  \
4    0.714286  1.904762               1.0  0.118750    2.187500
3    0.666667  1.904762               1.0  0.118750    1.950000
0    0.500000  1.428571               1.0  0.060000    1.300000
1    0.571429  1.428571               1.0  0.060000    1.400000
5    0.529412  1.411765               1.0  0.065625    1.328125

   zhangs_metric   jaccard  certainty  kulczynski
4       0.730769  0.526316   0.542857    0.690476
3       0.760000  0.526316   0.487179    0.690476
0       0.500000  0.363636   0.230769    0.535714
1       0.461538  0.363636   0.285714    0.535714
5       0.507246  0.391304   0.247059    0.564706
```

Analysis of a Key Rule:

The rule {diapers} -> {beer} suggests a strong, non-random relationship between these two purchases. The high Lift value indicates that the probability of buying beer increases significantly when diapers are also in the basket. A retailer could use this insight for **product placement**, such as moving beer closer to the diaper aisle, or for **targeted promotions**, offering a discount on one item when the other is purchased.

5. Conclusion

This project successfully demonstrated the end-to-end process of data warehousing and data mining, showcasing the full data lifecycle from raw data to actionable insights. The ETL pipeline and star schema design were effective in creating a clean, queryable database, which was then used to answer key business questions through OLAP analysis.

The data mining tasks provided valuable insights into both the structure of the Iris dataset through clustering and the predictive power of classification models. We successfully identified natural groupings in the data and compared the performance of different algorithms. We also generated and analyzed association rules, demonstrating how a retailer could use data to improve marketing and product placement.

The challenges faced throughout the project—including file format issues, path errors, and library compatibility problems—were all resolved through careful debugging and a deeper understanding of Python's data handling and notebook execution environments. Ultimately, this project serves as a comprehensive example of how to transform raw data into a powerful, insightful business asset.