Laboratory Manual

Data Warehousing and Mining Lab

(CMLR0603)

Semester VI

Bachelor of Technology

(B. Tech.)

in

Computer Engineering Department

Third Year with Effect from AY 2024 -2025

Prepared By:	Signature
Dr. Pinki Vishwakarma **(Associate Professor)**	
Dr. Afreen Banu **(Assistant Professor)**	
Mr. Tejas Hirave **(Assistant Professor)**	

Institute

Vision: To become a globally recognized institution offering quality education and enhancing professional standards	**Mission:** To impart high-quality technical education to the students by providing an excellent academic environment, well-equipped laboratories and training through the motivated teachers.

Department

Vision: ● To develop computer engineering graduates with engineering and managerial skills to acquire high end positions that are globally recognized.	**Mission:** ● To impart computer engineering knowledge. and to provide exposure to the latest technologies. ● To enable students to solve various engineering problems and possesses social, ethical responsibilities ● To foster an attitude of lifelong learning, ensuring they become competent professionals.
Program Educational Objectives (PEO)	**Program Specific Outcomes (PSOs)**
● Graduates will possess the engineering fundamental knowledge and technical skills to build successful career in various domains. ● Graduates will analyze real life problems and build feasible and economically acceptable solutions using latest technologies. ● Graduates will exhibit strong soft skills, teamwork, professional ethics and social responsibilities.	By the end of the educational experience our students will be able to: ● Students will be able to design & develop Computer programs and Automated systems to solve the real world problems for the benefit of society. ● Students will be able to work professionally by applying Software Engineering practices, pursue higher studies and build Entrepreneur skills.

Program Outcomes (POs)

Engineering Graduates will be able to:

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Mapping of PSOs to POs:

PSO Number	PO Number
PSO1	PO1, PO2, PO3, PO4, PO5, PO6, PO7, PO12
PSO2	PO1, PO8, PO9, PO10, PO11, PO12

Sd/-
Program Coordinator
Computer Engineering Program

Program: Third Year B.Tech.	Semester: VI	L	P	C
Data Warehousing & Mining	Course Code: CMCR0603	3	0	3
Data Warehousing & Mining Lab	Lab Code: CMLR0603	0	2	1
		3	2	4

Course Objectives:	
1	To identify the scope and understand the fundamentals of Data Warehousing and Mining.
2	To understand the importance of data warehouses which would assist in providing business insights for data mining applications.
3	To instigate research interest towards advances in data mining.

Course Outcomes:	
After successful completion of this course, the students should be able to	
CO 1:	Articulate and Analyze Data Warehousing fundamentals, Dimensionality modelling principles and the use of ETL techniques and apply OLAP operations.
CO 2:	Perceive the importance of data pre-processing and basics of data mining techniques.
CO 3:	Apply classification algorithms in real world dataset for prediction and classification and concept clustering algorithms and its applications.
CO 4:	Relate to the concepts of market basket analysis in real world applications.

Pre-requisite courses: Database system, Data Warehouse and mining, SQL Databases.

Course Assessment Methods:

DIRECT

1. Continuous Internal Assessment (Theory component)
2. Assignments/Tutorials/Power-point-presentation/Group-discussion/Quiz/seminar/Case studies/Design Thinking/Innovation/Creativity (Blog writing/Vlogging, etc)
3. Pre/Post - Experiment Test/Viva; Experimental Write-Up for each Experiment, Day to Day Experiments /Assignments/Tutorials/Power-point-presentation/Group-discussion/Quiz/seminar/Case studies/Design Thinking/Innovation/Creativity(Blog writing/Vlogging, etc) (Laboratory Component)
4. End Semester Examination (Theory and Laboratory components).

INDIRECT

1. Course-end survey
2. Activity based survey (if any)

DETAILED SYLLABUS

Module 1: Data Warehousing	**11 Hours**

1.1 Introduction to Data Warehouse
1.2 OLTP Systems versus Data Warehouse
1.3 Characteristics of Data Warehouse, Components of Data warehouse Architecture, Data warehouse Architecture
1.4 Data warehouses versus Data Marts
1.5 E-R Modeling versus Dimensional Modeling
1.6 Data Warehouse Schemas; Star Schema, Snowflake Schema, Inside Dimensional Table, Inside Fact Table, Fact Less Fact Table, Fact Constellation Schema
1.7 Update the dimension tables.
1.8 Major steps in ETL process, Data Extraction Methods, Data Transformation; Basic Tasks in Transformation, Major Data Transformation Types
1.9 Data Loading Technique
1.10 What is Multidimensional Data
1.11 OLAP operations: Slice, Dice, Rollup, Drilldown and Pivot

Module 2: Data Mining and Data pre-processing	**08 Hours**

2.1 Introduction to data mining
2.2 Architecture for Data Mining
2.3 KDD process
2.4 Data mining functionalities
2.5 Interestingness measures
2.6 Classification of data mining system
2.7 Major issues in data mining
2.8 Data summarization, data cleaning, data integration and transformation, data reduction, data discretization and concept hierarchy generalization

Module 3: Classification, Prediction and Cluster analysis	**10 Hours**

3.1 Definition
3.2 Decision tree induction
3.3 Bayesian classification
3.4 Introduction to prediction, Linear regression, Multiple linear regression, accuracy, and error measures
3.5 Distance Measures, clustering algorithms: Partitioning- K means and K-medoids,Hierarchical clustering, Agglomerative clustering and Divisive clustering.

Module 4: Mining frequent patterns and associations	**10 Hours**

4.1 Market Basket Analysis, Frequent Item sets, Closed Item sets, and Association Rule,
4.2 Frequent Pattern Mining, Efficient and Scalable Frequent Itemset
4.3 Mining Methods: Apriori Algorithm, Association Rule Generation
4.4 Improving the Efficiency of Apriori, FP growth
Mining various kinds of association rules – multilevel and multidimensional.

Lecture: 3 Hrs./Week		**Total Hours: 39 Hrs.**

LAB COMPONENT CONTENTS

Suggested List of Experiments (Minimum 8 Experiments)

1. Prepare the objective of given Mini- Project with respect to building Data warehouse/Data Mart

    i. Write Detailed Problem statement and design dimensional modelling (creation of star and snowflake schema)
    ii. Implementation of all dimension tables and fact table.

2. Implementation of OLAP operations: Slice, Dice, Rollup, Drilldown and Pivot for the above problem statement (experiment 1).

3. Implementation of Classification algorithm (Decision Tree/Naive Bayes)

4. Implementation of Clustering algorithm(K-means/Agglomerative)

5. Implementation of Association Rule Mining algorithm (Apriori)

6. Implementation of prediction algorithm (Linear regression)

7. Implementation of Data Discretization (any one) & Visualization (any one)

8. Perform data Pre-processing task and Demonstrate Classification algorithm on data sets using data mining tools (WEKA, R tool, XL Miner, Orange etc.

9. Perform data Pre-processing task and Demonstrate Clustering algorithm on data sets using data mining tool (WEKA, R tool, XL Miner, Orange etc.).

10. Perform data Pre-processing task and Demonstrate Association algorithm on data sets using data mining tools (WEKA, R tool, XL Miner, Orange etc.)

One beyond curriculum experiment may be conducted (To be decided by the Subject Teacher)

Practical: 2 Hrs./Week		**Total Hours: 26 Hrs.**

Textbooks:

Textbooks:

1. Paulraj Ponniah, Data Warehousing: Fundamentals for IT Professionals, Wiley India
2. Han, Kamber, Data Mining Concepts and Techniques, Morgan Kaufmann
3. Reema Thareja, Data warehousing, Oxford University Press.
4. M.H. Dunham, Data Mining Introductory and Advanced Topics, Pearson Education

Reference Books:

1. Ian H. Witten, Eibe Frank and Mark A. Hall, Data Mining, 3rd Edition Morgan Kaufmann Publisher.
2. Pang-Ning Tan, Michael Steinbach and Vipin Kumar, Introduction to Data Mining, Pearson Publisher.

Practitioners, Wiley India Private Limited

Course Code	Lab Name	Credits
CMLR0603	Data Warehousing & Mining Lab	1

Continuous Internal Assessment Practical (CIAP): CIAP will be assessed for 50 marks on the following rubrics and scaled down to 10 marks	
1	5 marks – Evaluation of write-up on day-to-day experiment in the laboratory (in terms of aim, components/procedure, expected outcome)
2	The Course In charge will choose any two of the below mentioned components, with each component having weightage of 20 marks each Assignments/ Tutorials/ Powerpoint presentation/ Group discussion/ Quiz/ seminar/ Case studies/ Design Thinking/ Innovation/ Creativity/ Project/ App development
3	Attendance will be having weightage of 5 marks
End Semester Examination (ESEP)	
1	Based on the above contents and entire syllabus of ETDLOCR07031 The End Semester Examination Practical shall be conducted for 100 marks for a duration of three hours and scaled down to 15 marks

Evaluation Method	Passing Requirement
Continuous Internal Assessment (CIAP)+End Semester Examination (ESEP)	Obtained Marks $\geq \geq$ 40 % of maximum marks

Course Outcomes (CO)

CO No.	**CO Statement** (At the end of the course, students will be able to …)	BL
1	Articulate and Analyze Data Warehousing fundamentals, Dimensionality modelling principles and the use of ETL techniques and apply OLAP operations.	3
2	Perceive the importance of data pre-processing and basics of data mining techniques.	3
3	Apply classification algorithms in real world dataset for prediction and classification and concept clustering algorithms and its applications.	3
4	Relate to the concepts of market basket analysis in real world applications.	4

List of Experiments

Expt No.	Title	CO	PO	PSO
1	Build Data warehouse/ Data Mart for the given problem statement (i) Write detailed Problem statement and design dimensional modeling (Creation of star and snowflake schema) (ii) Implementation of all dimension tables and fact table.	1	1,2	1,2
2	Implementation of OLAP operations: Slice, Dice, Rollup, Drilldown and Pivot based on experiment 1	1	1,2	1,2
3	Implementation of Classification Algorithm (Decision Tree / Naïve Bayes)	3	1,2,4	1,2
4	Implementation of Clustering Algorithm K-means / Agglomerative)	3	1,2,4	1,2
5	Implementation of Association Rule Mining Algorithm (Apriori)	4	1,2,4	1,2
6	Implementation of Prediction Algorithm (Linear Regression)	3	1,2,4	1,2
7	Implementation of Data Discretization (any one) & Visualization (any one)	2	1,2,4	1,2
8	Perform data Pre-processing task and demonstrate Classification, Clustering, Association algorithm on data sets using data mining tool WEKA	3,4	1,2,4,5	1,2

Name and Signature:

Date

INDEX

Subject In-charge

Experiment No. 1

1.1 Aim: Prepare the objective of a given Mini Project with respect to building a Data warehouse/ Data Mart.

(i) Write detailed Problem statement and design dimensional modeling(Creation of star and snowflake schema)

(ii) Implementation of all dimension tables and fact table.

1.2 Course Outcome: Articulate and Analyze Data Warehousing fundamentals, Dimensionality
modelling principles and the use of ETL techniques and apply OLAP operations.

1.3 Requirement: Visual Paradigm, Canva, Live SQL

1.4 Related Theory:

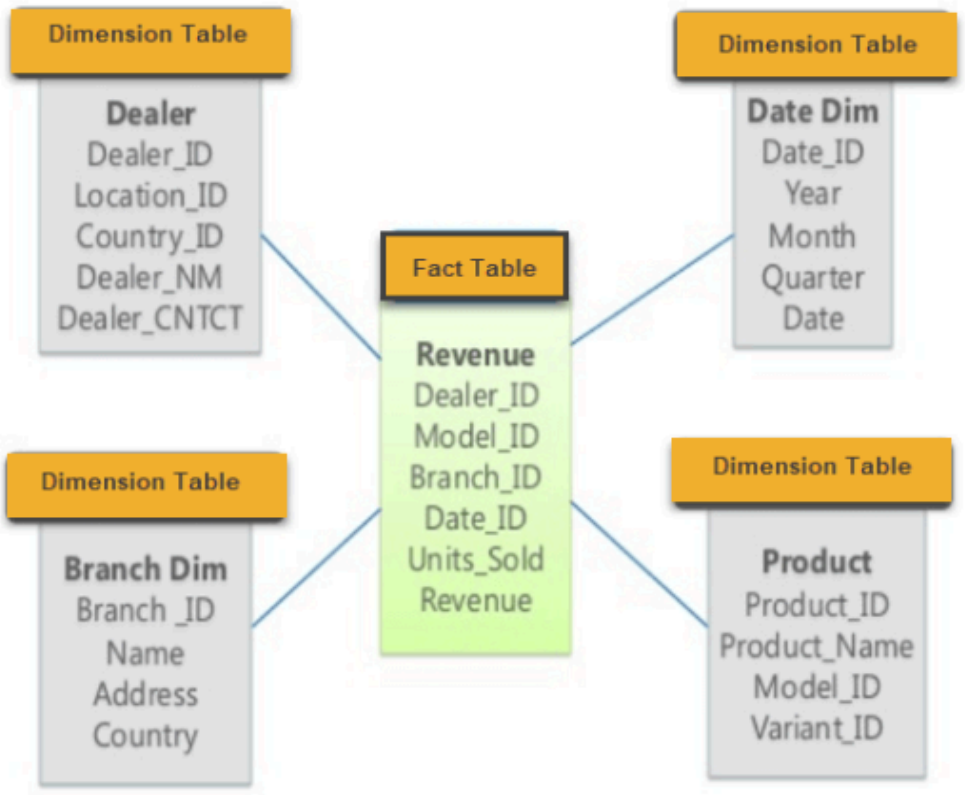
THEORY:

Data warehouse
A data warehouse is a subject oriented, integrated, time varying, non-volatile collection of data that is used primarily in organizational decision making. Dimensional modeling: A star schema is the simplest form of a dimensional model, in which data is organized into facts and dimensions. A fact is an event that is counted or measured, such as a sale or login. A dimension contains reference information about the fact, such as date, product, or customer. A star schema is diagrammed by surrounding each fact with its associated dimensions. The resulting diagram resembles a star. Star schemas are optimized for querying large data sets and are used in data warehouses and data marts to support OLAP cubes, business intelligence and analytic applications, and ad-hoc queries. Within the data warehouse or data mart, a dimension table is associated with a fact table by using a foreign key relationship. The dimension table has a single primary key that uniquely identifies each member record (row).The fact table contains the primary key of each associated dimension table as a foreign key. Combined, these foreign keys form a multi-part composite primary key that uniquely identifies each member record in the fact table. The fact table also contains one or more numeric measures. For example, a simple Sales fact with millions of individual clothing sale records might contain a Product Key, Promotion Key, Customer Key, and Date Key, along with Units Sold and Revenue measures. The Product

dimension would hold reference information such as product name, description, size, and color. The Promotion dimension would hold information such as promotion name and price. The Customer dimension would hold information such as first and last name, birth date, gender, address, etc. The Date dimension would include calendar date, week of year, month, quarter, year, etc. This simple Sales fact will easily support queries such as “total revenue for all clothing products sold during the first quarter of the 2010” or “count of female customers who purchased 5 or more dresses in December 2009”.

STAR SCHEMA

Star Schema in a data warehouse, in which the center of the star can have one fact table and number of associated dimension tables. It is known as star schema as its structure resembles a star. The Star Schema data model is the simplest type of Data Warehouse schema. It is also known as Star Join Schema and is optimized for querying large data sets. In the following Star Schema example, the fact table is at the center which contains keys to every dimension table like Dealer_ID, Model ID, Date_ID, Product_ID, Branch_ID & other attributes like Units sold and revenue.



Example of Star Schema Diagram

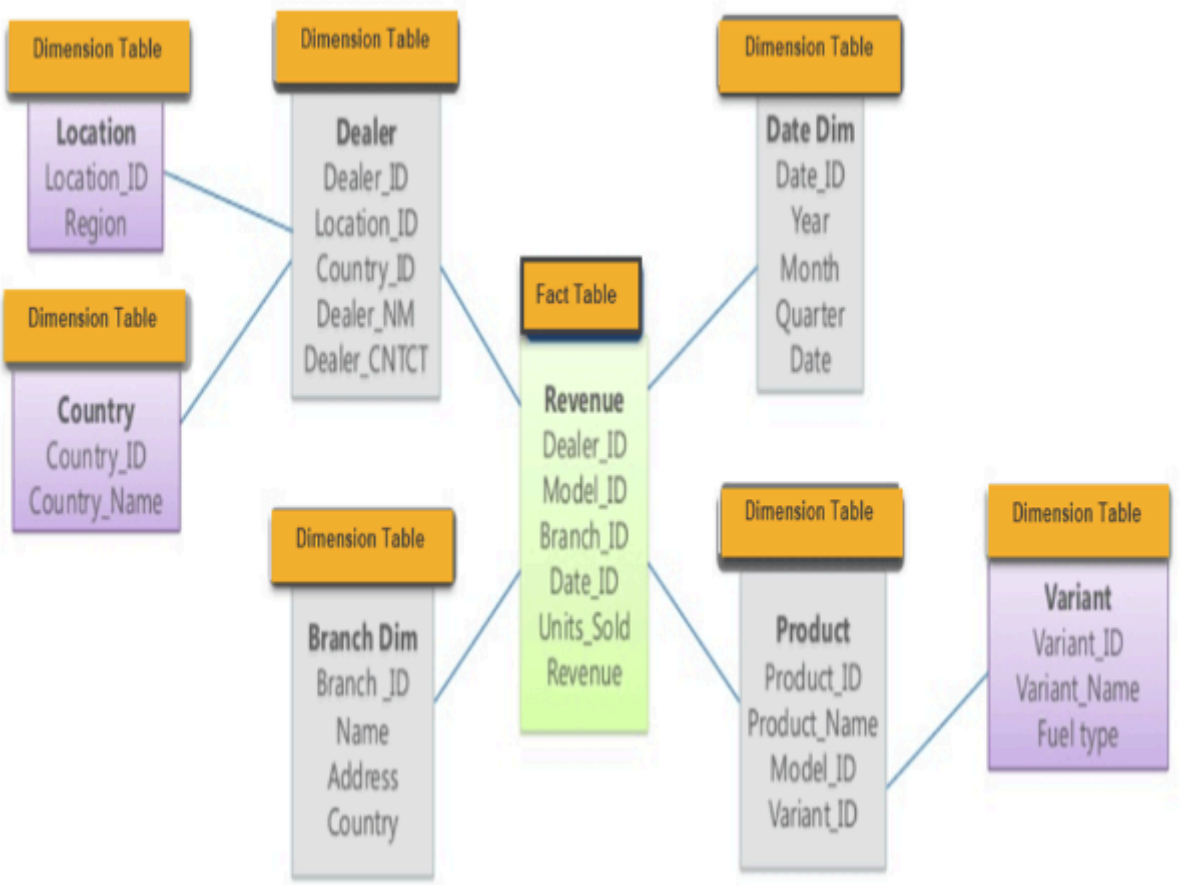
Characteristics of Star Schema:

- Every dimension in a star schema is represented with the only one-dimension table.
- The dimension table should contain the set of attributes.
- The dimension table is joined to the fact table using a foreign key
- The dimension table are not joined to each other

- Fact table would contain key and measure
- The Star schema is easy to understand and provides optimal disk usage.
- The dimension tables are not normalized. For instance, in the above figure, Country_ID does not have a Country lookup table as an OLTP design would have.
- The schema is widely supported by BI Tools

SNOWFLAKE SCHEMA

Snowflake Schema in a data warehouse is a logical arrangement of tables in a multidimensional database such that the ER diagram resembles a snowflake shape. A Snowflake Schema is an extension of a Star Schema, and it adds additional dimensions. The dimension tables are normalized which splits data into additional tables.In the following Snowflake Schema example, Country is further normalized into an individual table.



Example of Snowflake Schema

Characteristics of Snowflake Schema:

- The main benefit of the snowflake schema is that it uses smaller disk space.
- Easier to implement a dimension is added to the Schema
- Due to multiple tables query performance is reduced
- The primary challenge that you will face while using the snowflake Schema is that you

need to perform more maintenance efforts because of the more lookup tables.

1.5 Algorithm:

Steps 1:Create Dimension table
A. Product Dimension table:

```
CREATE table Product_dw
( prod_id int PRIMARY KEY,
Prod_name varchar(60) NOT NULL,
Prod_category varchar(255) NOT NULL,
Brand_name varchar(255) NOT NULL,
Suppl_name varchar(255) NOT NULL,
Prod_price NUMBER );
```

B. Time Dimension table:

```
CREATE table Time_dw
( time_id int PRIMARY KEY,
day DATE NOT NULL,
month varchar(255) NOT NULL,
qt varchar(255) NOT NULL,
yr varchar(255) NOT NULL);
```

C Location Dimension table:

```
CREATE table Location_dw
( loc_id int PRIMARY KEY,
street varchar(60) NOT NULL,
city varchar(255) NOT NULL,
state varchar(255) NOT NULL,
country varchar(255) NOT NULL
);
```

Step 2: Create Sale Fact table

```
CREATE table Fact_sales
( prod_id int REFERENCES Product_dw(prod_id),
time_id int REFERENCES Time_dw(time_id),
loc_id int REFERENCES Location_dw(loc_id),
number_of_unit_sold int NOT NULL,
Total_sales int NOT NULL);
```

Step 3: Insert data into Dimension table and fact table
A. Product Dimension table-

INSERT INTO Product_dw(prod_id,Prod_name,Prod_category,Brand_name, Suppl_name, Prod_price)
VALUES (1, ‘Rice’, ‘Grocery’, 'Dawat',’Ramesh’,140);
INSERT INTO Product_dw (prod_id,Prod_name,Prod_category,Brand_name, Suppl_name, Prod_price)
VALUES (2, ‘Sugar’, ‘Grocery’, 'Dawat',’Ramesh’,50);
INSERT INTO Product_dw (prod_id,Prod_name,Prod_category,Brand_name, Suppl_name, Prod_price)
VALUES (3, ‘Kurta’, ‘Cloth’, 'Max',’Lila’,500);
INSERT INTO Product_dw (prod_id,Prod_name,Prod_category,Brand_name, Suppl_name, Prod_price)

VALUES (4, ‘jacket’, ‘Cloth’, 'Max',’Lila’,700);

B. B.Time Dimension table:
INSERT INTO Time_dw(time_id,day, month,qt,yr)
VALUES (101,DATE '2021-1-17', 'january', 'Q1','2021');
INSERT INTO Time_dw(time_id, day, month,qt,yr)
VALUES (102, DATE '2021-2-14', 'february', 'Q1','2021');
INSERT INTO Time_dw(time_id, day, month,qt,yr)
VALUES (103, DATE '2021-5-21', 'may', 'Q2','2021');
INSERT INTO Time_dw (time_id, day, month,qt,yr)
VALUES (104, DATE '2021-6-26', 'june', 'Q2','2021');
C. Location Dimension table:
INSERT INTO Location_dw(loc_id,street,city,state,country)
VALUES (201,'ML ROAD','MUMBAI','MAHARASHTRA','INDIA');
INSERT INTO Location_dw(loc_id,street,city,state,country)
VALUES (202,'AI ROAD','MUMBAI','MAHARASHTRA','INDIA');
INSERT INTO Location_dw(loc_id,street,city,state,country)
VALUES (203,'BI ROAD','KOLKATA','WEST BENGAL','INDIA');
INSERT INTO Location_dw(loc_id,street,city,state,country)

VALUES (204,'DB ROAD','KOLKATA','WEST BENGAL','INDIA');

D. Sales Fact table:

```
INSERT INTO Fact_sales(prod_id,time_id,loc_id,number_of_unit_sold,Total_sales)
VALUES (1,101,201,400,80000);
INSERT INTO Fact_sales(prod_id,time_id,loc_id,number_of_unit_sold,Total_sales)
VALUES (1,102,201,400,90000);
INSERT INTO Fact_sales(prod_id,time_id,loc_id,number_of_unit_sold,Total_sales)
VALUES (1,103,201,400,70000);
INSERT INTO Fact_sales(prod_id,time_id,loc_id,number_of_unit_sold,Total_sales)
VALUES (1,104,201,400,90000);
.
.
 so on
```

If there are 3-dimension table and each table consists of 4 rows ,in that case fact table consists of 4*4*4=64 rows in fact table

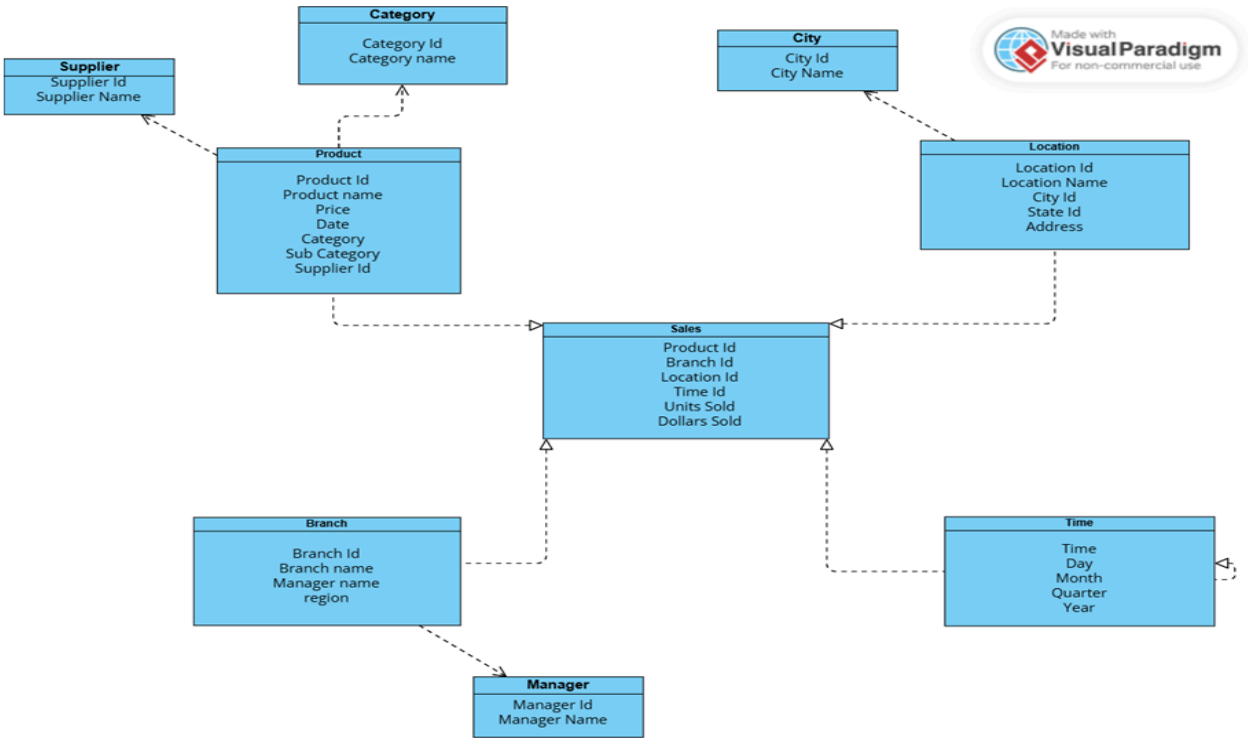
Step 4 : To view any table use select operation

A .Product dimension table

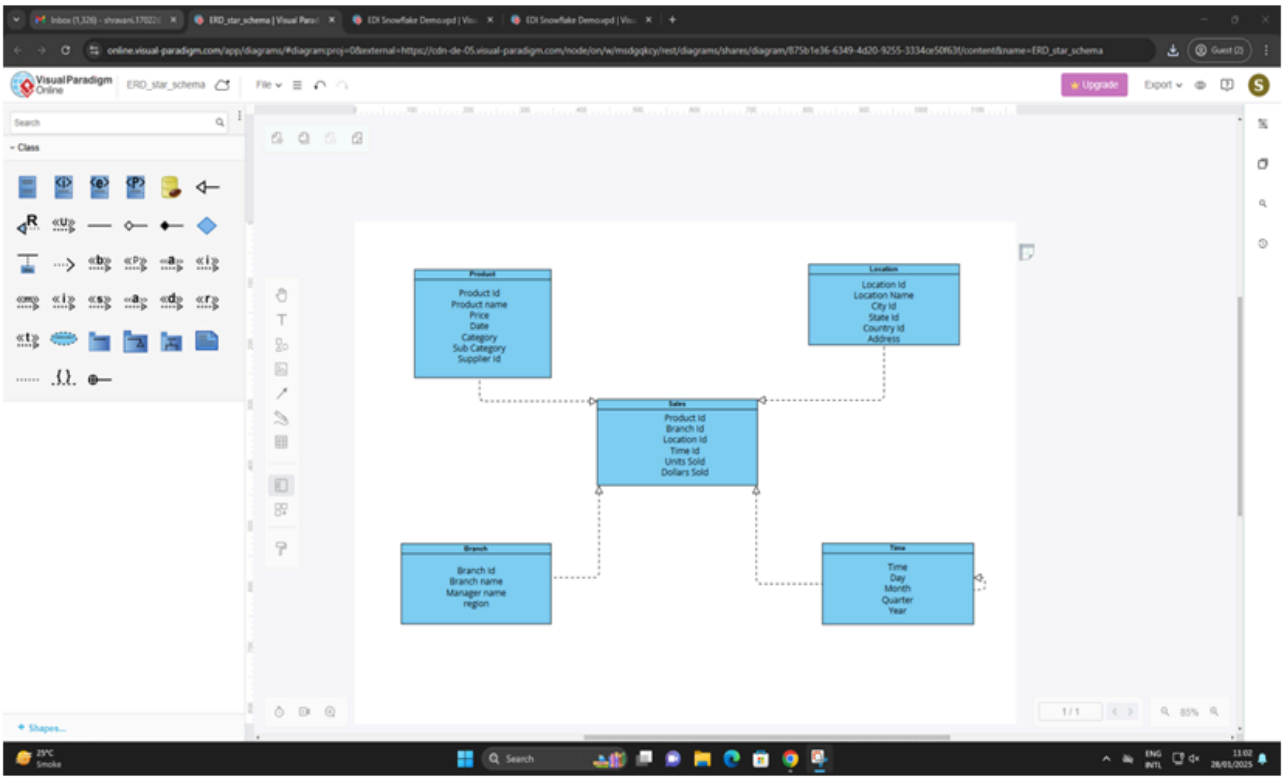
```
Select * from product
```

1.6 Program and Output:

Snowflake Schema:



Star Schema:


Product
Product Id
Product name
Price
Date
Category
Sub Category
Supplier Id
Location
Location Id
Location Name
City Id
State Id
Country Id
Address
Sales
Product Id
Branch Id
Location Id
Time Id
Units Sold
Dollars Sold
Branch
Branch Id
Branch name
Manager name
region
Time
Time
Day
Month
Quarter
Year

1.7 Conclusion:

This experiment successfully demonstrates the creation of a Data Warehouse/Data Mart using dimensional modeling with star and snowflake schemas. All required dimension and fact tables are implemented and deployed.

1.8 Questions:

What are the advantages of using star and snowflake schemas in dimensional modeling for a Data Warehouse?

What are the key differences between fact tables and dimension tables, and how do they work together in this project?

Experiment No. 2

2.1 Aim: Implementation of OLAP operations: Slice, Dice, Rollup, Drilldown and Pivot

2.2 Course Outcome: Articulate and Analyze Data Warehousing fundamentals, Dimensionality modelling principles and the use of ETL techniques and apply OLAP operations.

2.3 Requirement: Visual Paradigm, Canva, Live SQL

2.4 Related Theory:

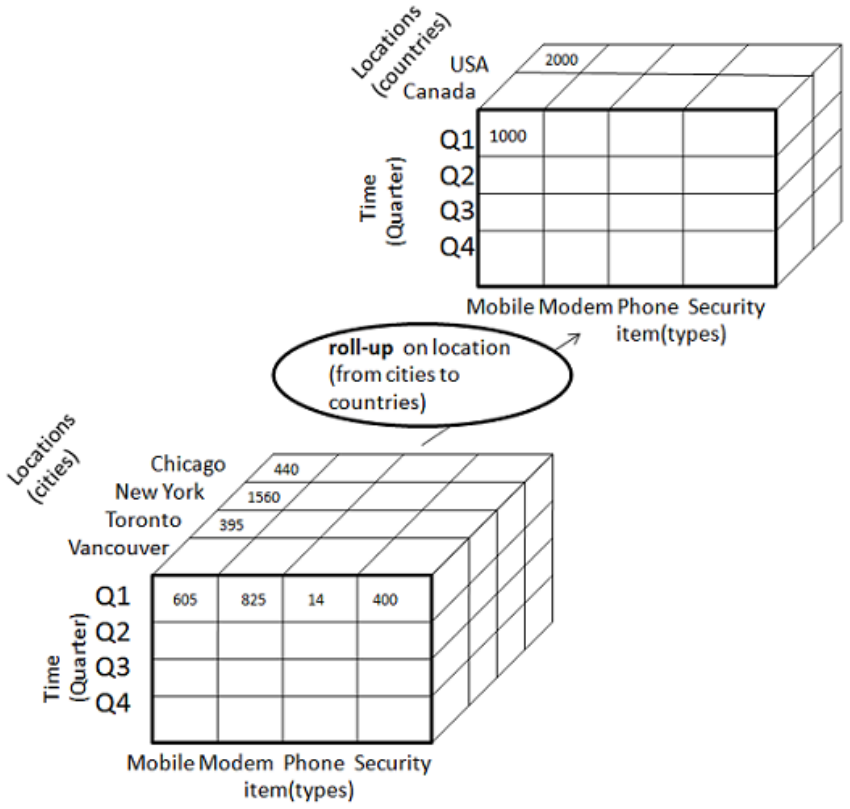
OLAP Operations:

Since OLAP servers are based on a multidimensional view of data. Here is the list of OLAP operations which can be performed on multidimensional data.

Roll Up

Roll-up performs aggregation on a data cube.The following diagram illustrates how roll-up works.

- Roll-up is performed by climbing up a concept hierarchy for the dimension location.
- Initially the concept hierarchy was "street < city < province < country".
- On rolling up, the data is aggregated by ascending the location hierarchy from the level of city to the level of country.
- The data is grouped into cities rather than countries.
- When roll-up is performed, one or more dimensions from the data cube are removed.



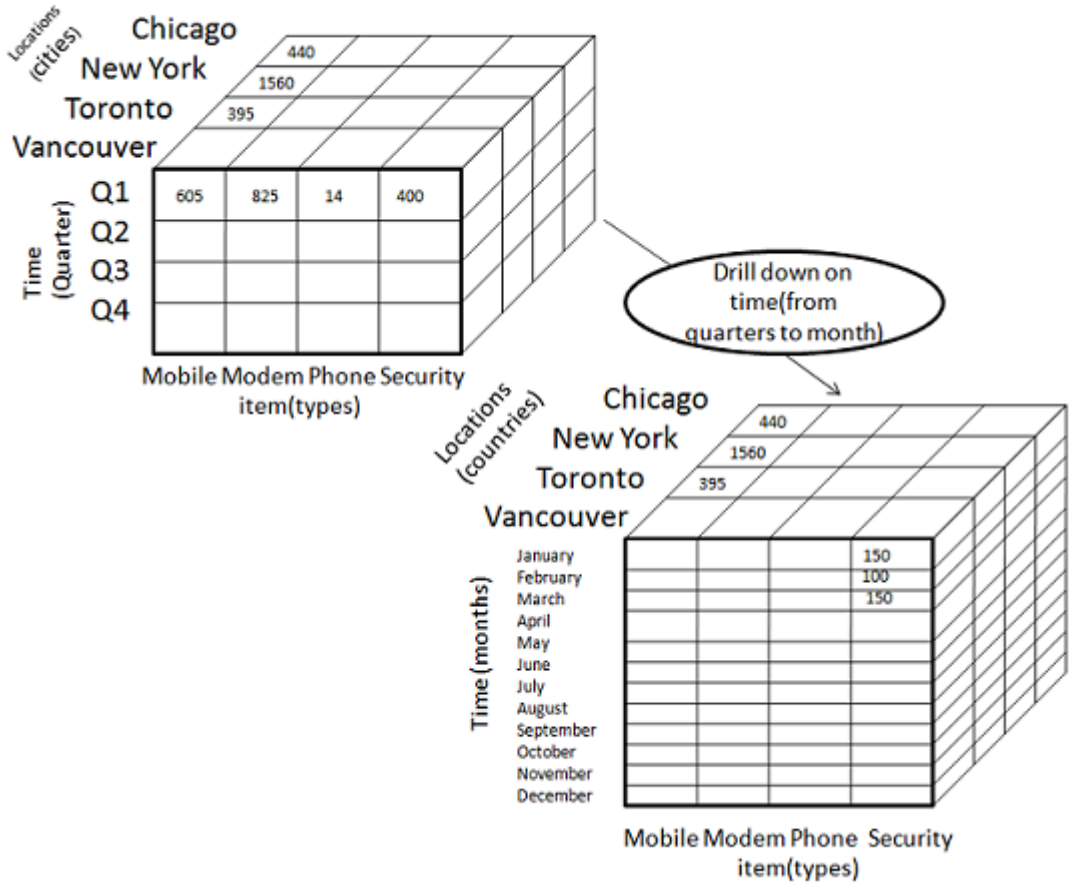
Drill Down

Drill-down is the reverse operation of roll-up. It is performed by either of the following ways:

- By stepping down a concept hierarchy for a dimension
- By introducing a new dimension.

The following diagram illustrates how drill-down works:

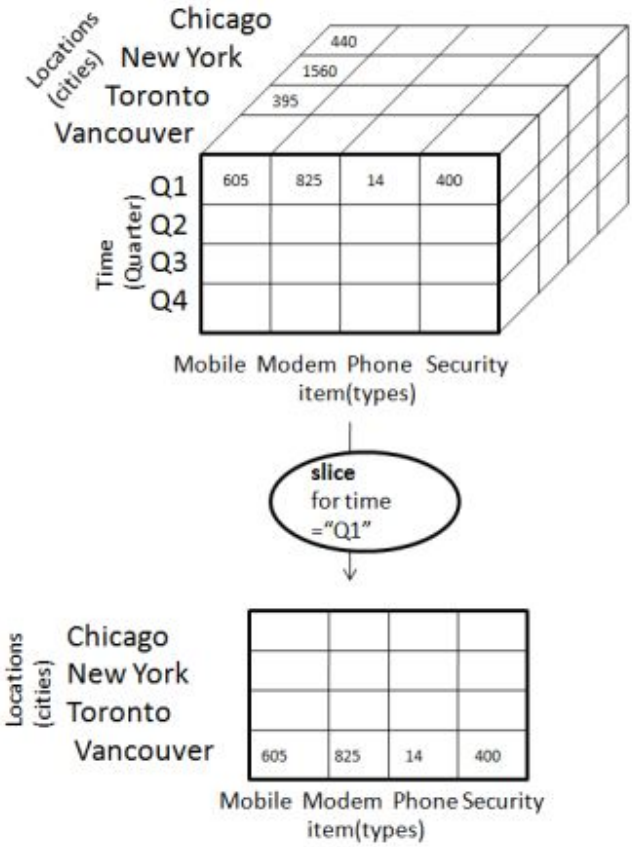
- Drill-down is performed by stepping down a concept hierarchy for the dimension time.
- Initially the concept hierarchy was "day < month < quarter < year."
- On drilling down, the time dimension is descended from the level of quarter to the level of month.
- When drill-down is performed, one or more dimensions from the data cube are added.
- It navigates the data from less detailed data to highly detailed data.



Slice

The slice operation selects one particular dimension from a given cube and provides a new sub-cube. Consider the following diagram that shows how slice works.

- Here Slice is performed for the dimension "time" using the criterion time = "Q1".
- It will form a new sub-cube by selecting one or more dimensions.

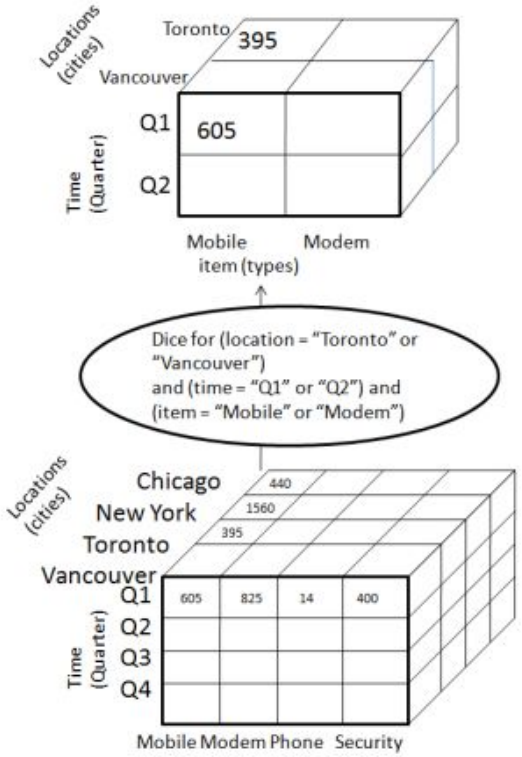


Dice

Dice selects two or more dimensions from a given cube and provides a new sub-cube. Consider the following diagram that shows the dice operation.

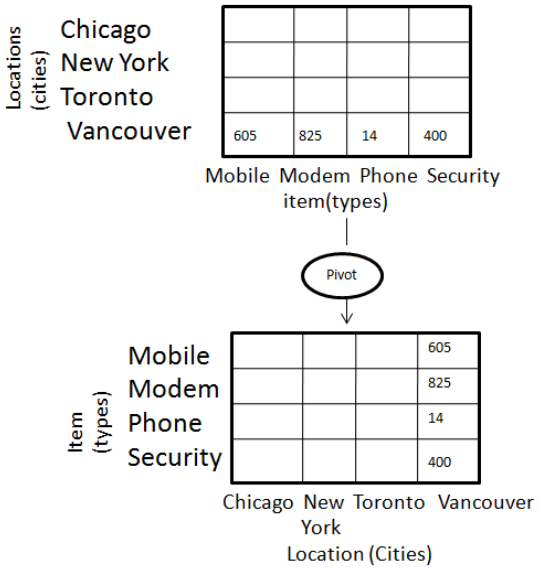
The dice operation on the cube based on the following selection criteria involves three dimensions.

- (location = "Toronto" or "Vancouver")
- (time = "Q1" or "Q2")
- (item =" Mobile" or "Modem")



Pivot

The pivot operation is also known as rotation. It rotates the data axes in view in order to provide an alternative presentation of data. Consider the following diagram that shows the pivot operation. In this the item and location axes in 2-D slice are rotated.



2.5 Algorithm:

OLAP (Online Analytical Processing) operations allow users to interactively analyze multidimensional data from different perspectives. The following steps outline how to perform each key OLAP operation using the sales data warehouse created in Experiment 1.

---

1. Roll-Up Operation

Purpose: To aggregate data by moving up the hierarchy in a dimension.

Steps:

- Identify the dimension with a defined hierarchy (e.g., Location: Street → City → State → Country).

- Use GROUP BY in SQL to aggregate data at a higher level.

Example:

```
SELECT
   l.country,
   SUM(f.Total_sales) AS total_sales
FROM
   Fact_sales f
JOIN
   Location_dw l ON f.loc_id = l.loc_id
GROUP BY
   l.country;
```

---

2. Drill-Down Operation

Purpose: To get more detailed data by descending a hierarchy or adding a new dimension.

Steps:

- Choose a dimension to drill down (e.g., Time: Year → Quarter → Month → Day).
- Modify the query to include a more detailed level.

Example:

```
SELECT
   t.month,
   l.city,
   SUM(f.Total_sales) AS total_sales
FROM
   Fact_sales f
JOIN
   Time_dw t ON f.time_id = t.time_id
JOIN
   Location_dw l ON f.loc_id = l.loc_id
```

```
GROUP BY
    t.month, l.city;
```

---

3. Slice Operation

Purpose: To extract a sub-cube by fixing one dimension to a specific value.

Steps:

- Select a fixed value for one dimension (e.g., time = 'Q1').
- Use WHERE clause to filter that dimension.

Example:

```
SELECT
    p.Prod_name,
    l.city,
    f.Total_sales
FROM
    Fact_sales f
JOIN
    Product_dw p ON f.prod_id = p.prod_id
JOIN
    Time_dw t ON f.time_id = t.time_id
JOIN
    Location_dw l ON f.loc_id = l.loc_id
WHERE
    t.qt = 'Q1';
```

4. Dice Operation

Purpose: To extract a sub-cube based on multiple dimension filters.

Steps:

- Apply multiple filter conditions across different dimensions.

Example:

```
SELECT
   p.Prod_name,
   t.month,
   l.city,
   f.Total_sales
FROM
   Fact_sales f
JOIN
   Product_dw p ON f.prod_id = p.prod_id
JOIN
   Time_dw t ON f.time_id = t.time_id
JOIN
   Location_dw l ON f.loc_id = l.loc_id
WHERE
   (l.city = 'MUMBAI' OR l.city = 'KOLKATA') AND
   (t.qt = 'Q1' OR t.qt = 'Q2') AND
   (p.Prod_name = 'Rice' OR p.Prod_name = 'Kurta');
```

---

5. Pivot Operation

Purpose: To rotate the data axes and change the presentation of the result set.

Steps:

- Use SQL CASE, PIVOT, or presentation tools to rearrange the data axes.

Example (basic SQL-style pivot using CASE):

```
SELECT
   p.Prod_name,
   SUM(CASE WHEN t.month = 'January' THEN f.Total_sales ELSE 0 END) AS Jan_Sales,
   SUM(CASE WHEN t.month = 'February' THEN f.Total_sales ELSE 0 END) AS Feb_Sales
FROM
   Fact_sales f
JOIN
   Product_dw p ON f.prod_id = p.prod_id
JOIN
   Time_dw t ON f.time_id = t.time_idGROUP BY
   p.Prod_name;
```

2.6 Program and Output:

1. SLICE operation:

This query extracts only sales that occurred in the year 2024 from the sales table

SQL Worksheet

```
1  SELECT s.BranchID, s.LocID, s.ProductID, s.TimeID, s.UnitsSold, s.DollarsSold
2  FROM Sale s
3  JOIN TIME T ON s.TimeID = T.TimeID
4  WHERE T.Yr = '2024'
5  ORDER BY s.ProductID;
```

BRANCHID	LOCID	PRODUCTID	TIMEID	UNITSSOLD	DOLLARSSOLD
3	3	1	5	10	100
3	4	1	1	10	100
3	4	1	2	10	100
3	4	1	3	10	100
3	4	1	4	10	100
3	4	1	5	10	100

This query extracts only sales of products that belong to the "Accessories" category.

SQL Worksheet

Clear Find Actions

```
SELECT s.BranchID, s.LocID, s.ProductID, s.TimeID, s.UnitsSold, s.DollarsSold, A.Category, A.Price
FROM Sale s
JOIN Products A ON s.ProductID = A.ProductID
WHERE A.Category = 'Accessories' ;
```

BRANCHID	LOCID	PRODUCTID	TIMEID	UNITSSOLD	DOLLARSSOLD	CATEGORY	PRICE
1	1	2	1	10	100	Accessories	25
1	1	3	1	10	100	Accessories	75
1	1	5	1	10	100	Accessories	50
1	1	2	2	10	100	Accessories	25
1	1	3	2	10	100	Accessories	75
1	1	5	2	10	100	Accessories	50
1	1	2	3	10	100	Accessories	25
1	1	3	3	10	100	Accessories	75

2. DICE Operation:

This query retrieves sales only from the USA and limits it to Region East or West.

SQL Worksheet

Clear Find Actions Save Run

```
SELECT s.BranchID, s.LocID, s.ProductID, s.TimeID, s.UnitsSold, s.DollarsSold, L.coun_try, B.Branchname, B.Region
FROM Sale s
JOIN Loc_dw L ON s.LocID = L.LocId
JOIN Branch B ON s.BranchID = B.BranchID
WHERE L.coun_try = 'USA'
AND B.Region IN ('East', 'West')
ORDER BY B.BranchName, L.coun_try ;
```

BRANCHID	LOCID	PRODUCTID	TIMEID	UNITSSOLD	DOLLARSSOLD	COUN_TRY	BRANCHNAME	REGION
2	1	1	1	10	100	USA	Branch B	East
2	1	2	1	10	100	USA	Branch B	East
2	1	3	1	10	100	USA	Branch B	East
2	1	4	1	10	100	USA	Branch B	East
2	1	5	1	10	100	USA	Branch B	East
2	1	1	2	10	100	USA	Branch B	East

3. Roll Up Operation:

This query calculates the total units sold (SUM(s.UnitsSold)) and the total sales amount (SUM(s.DollarsSold)) for each product (ProductName) and branch (BranchName).

```
SELECT
    p.ProductName, b.BranchName, SUM(s.UnitsSold) AS TotalUnitsSold, SUM(s.DollarsSold) AS TotalDollarsSold
FROM
    Sale s
JOIN
    Products p ON s.ProductID = p.ProductID
JOIN
    BRANCH b ON s.BranchID = b.BranchID
GROUP BY
    ROLLUP (p.ProductName, b.BranchName)
ORDER BY p.ProductName, b.BranchName;
```

PRODUCTNAME	BRANCHNAME	TOTALUNITSSOLD	TOTALDOLLARSSOLD
Keyboard	Branch A	250	2500
Keyboard	Branch B	250	2500
Keyboard	Branch C	250	2500
Keyboard	Branch D	250	2500
Keyboard	Branch E	250	2500
Keyboard	-	1250	12500
Laptop	Branch A	250	2500
Laptop	Branch B	250	2500
Laptop	Branch C	250	2500

4. Pivot Operation:

This query pivots the sales data, showing total sales for each product category across different years.

```
SELECT * FROM (
    SELECT p.product_category, t.years, s.total_amount
    FROM sales s
    JOIN product p ON s.product_id = p.product_id
    JOIN time t ON s.time_id = t.time_id
)
PIVOT (
    SUM(total_amount)
    FOR years IN ('2023' AS "2023", '2024' AS "2024", '2025' AS "2025", '2026' AS "2026")
)
ORDER BY product_category
FETCH FIRST 5 ROWS ONLY
```

PRODUCT_CATEGORY	2023	2024	2025	2026
Accessories	-	-	101378	27006.6
Apparel	-	-	45677.45	11184.2
Electronics	-	-	818417.3	205853.1
Furniture	-	-	68932.7	17616.3

Download CSV

rows selected.

2.7 Conclusion:

OLAP operations like Slice, Dice, Roll-up, Drill-down, and Pivot were successfully applied to the sales data warehouse. These operations allow users to analyze data from multiple angles, making it easier to discover patterns, summarize information, and support better decision-making.

2.8 Questions:

1. What is the difference between the Slice and Dice operations in OLAP?

2. How does the Roll-up operation help in summarizing large volumes of data?

3. Give an example of a scenario where Pivoting the data provides a clearer insight than traditional tabular view?

Experiment No. 3

3.1 Aim: Implementation of Classification Algorithm (Decision Tree / Naïve Bayes)

3.2 Course Outcome: Apply classification algorithms in real world dataset for prediction and classification and concept clustering algorithms and its applications.

3.3 Requirement: Google Colab/ Python IDLE.

3.4 Related Theory:

Introduction to Classification

Classification is a type of supervised machine learning task where the goal is to predict the category or class of an object based on input features. The output variable (also called the target variable) is categorical, and the algorithm learns patterns from a labeled training dataset to classify unseen data.

Decision Tree Algorithm

A Decision Tree is a tree-like structure where:

- **Each internal node** represents a "test" or "decision" on an attribute (e.g., whether an age is greater than 30).
- **Each branch** represents the outcome of the test (e.g., True/False).
- **Each leaf node** represents a class label or the target variable.

How It Works:

- The Decision Tree splits the data based on features that maximize information gain (entropy reduction) or minimize impurity (Gini index).
- This process continues recursively to create a tree where the most significant features are placed closer to the root.

Advantages:

- Easy to interpret and visualize.

- Can handle both numerical and categorical data.
- Non-linear relationships between features do not affect performance.

Disadvantages:

- Prone to overfitting, especially with deep trees.
- Sensitive to small variations in data, leading to unstable models.

Naïve Bayes Algorithm

Naïve Bayes is a probabilistic classification technique based on **Bayes' Theorem**, which assumes that the features are independent of each other given the class label.

How It Works:

- **Bayes' Theorem** calculates the probability of a class given a set of features.
- It uses the formula:
  P(C∣X)=P(X∣C)P(C)P(X)P(C|X) = \frac{P(X|C)P(C)}{P(X)}P(C∣X)=P(X)P(X∣C)P(C)
  Where:
  - P(C∣X)P(C|X)P(C∣X) is the posterior probability (probability of class CCC given the features XXX).
  - P(X∣C)P(X|C)P(X∣C) is the likelihood (probability of observing features XXX given class CCC).
  - P(C)P(C)P(C) is the prior probability of class CCC.
  - P(X)P(X)P(X) is the evidence (probability of features XXX occurring).

Advantages:

- Simple and fast, especially for large datasets.

- Performs well with high-dimensional data.
- Works well with categorical and numerical data, especially when features are independent.

Disadvantages:

- The "naïve" assumption (features are independent) may not hold in many real-world scenarios, potentially reducing accuracy.

3.6 Procedure:

Data Collection

- Collect or import a dataset suitable for classification (e.g., Iris, Titanic, Student Performance, etc.).
- Ensure the dataset contains both independent (features) and dependent (target) variables.

Data Preprocessing

- Handle missing values and remove duplicates.
- Convert categorical data into numerical format (if needed).
- Split the dataset into training and testing sets (e.g., 80% train, 20% test).

Algorithm Selection

- Choose a classification algorithm (e.g., Decision Tree or Naïve Bayes).
- Import the required libraries (e.g., sklearn.tree, sklearn.naive_bayes).

Model Training

- Fit the chosen model to the training dataset using the fit() function.

Prediction

- Use the trained model to predict the outcome on the test dataset using the predict() function.

Model Evaluation

- Evaluate the performance of the model using metrics such as:
    - Accuracy
    - Confusion Matrix
    - Precision, Recall, and F1-Score

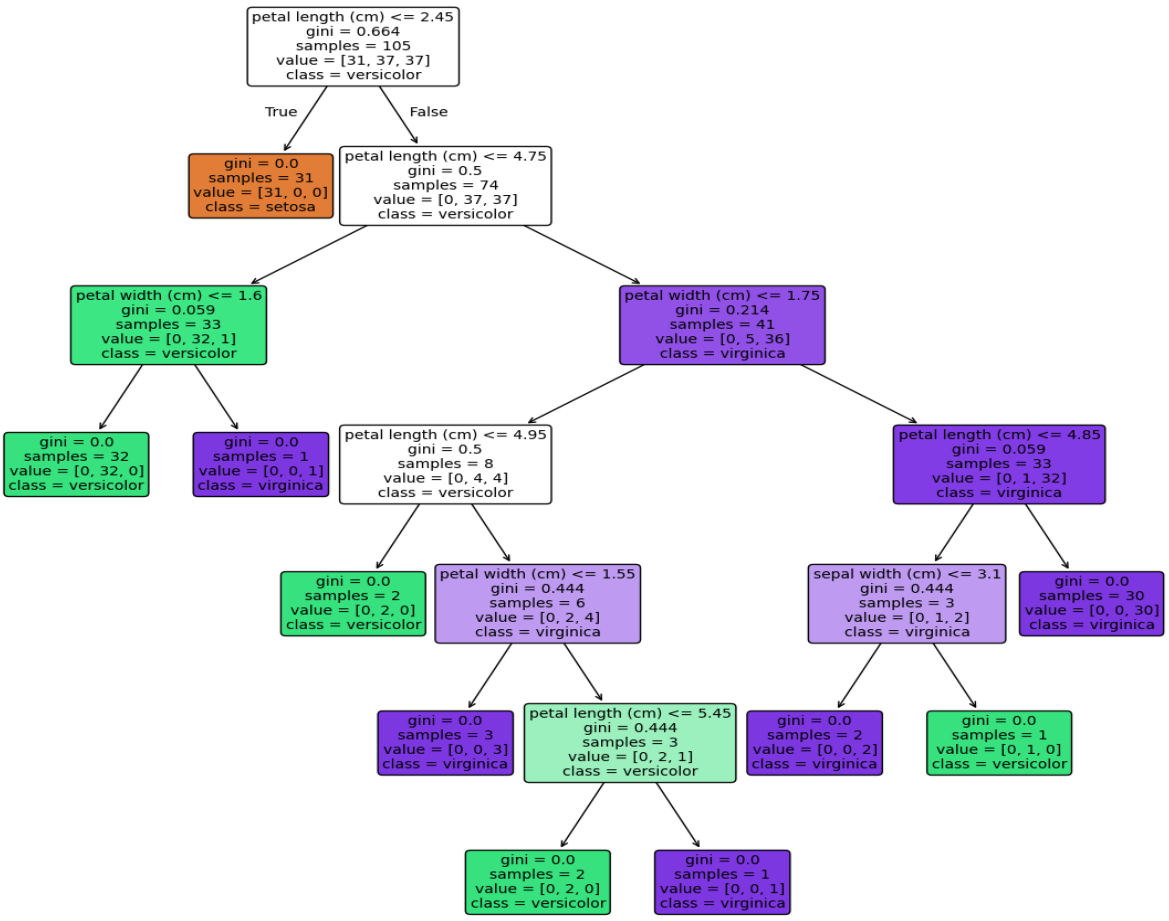
Visualization (optional)

- For Decision Tree: Use plot_tree() to visualize the tree structure.
- For Naïve Bayes: Display class probabilities or likelihoods.

3.7 Program and Output:

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree
# Step 1: Load the dataset (Iris dataset in this case)
iris = load_iris()
X = iris.data  # Features (independent variables)
y = iris.target  # Target labels (dependent variable)

# Step 2: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Step 3: Initialize the Decision Tree Classifier
dt_classifier = DecisionTreeClassifier(random_state=42)

# Step 4: Train the classifier with training data
dt_classifier.fit(X_train, y_train)
```

```
# Step 5: Predict the target labels on the test data
y_pred = dt_classifier.predict(X_test)

# Step 6: Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

# Additional evaluation: Classification Report
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=iris.target_names))

# Plot the decision tree
plt.figure(figsize=(14, 14))
plot_tree(dt_classifier, filled=True, feature_names=iris.feature_names, class_names=iris.target_names,
rounded=True, fontsize=10)
plt.show()
```



Naive Bayes:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
import numpy as np
import matplotlib.pyplot as plt
```

In [21]:

```
# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target
```

In [22]:

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42) # 70% training and 30% test
```

In [23]:

```
# Create a Gaussian Naive Bayes classifier
gnb = GaussianNB()
```

In [24]:

```
# Train the classifier
gnb.fit(X_train, y_train)
```

Out[24]:

GaussianNB()

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [25]:

```
# Make predictions on the testing set
y_pred = gnb.predict(X_test)
```

In [26]:

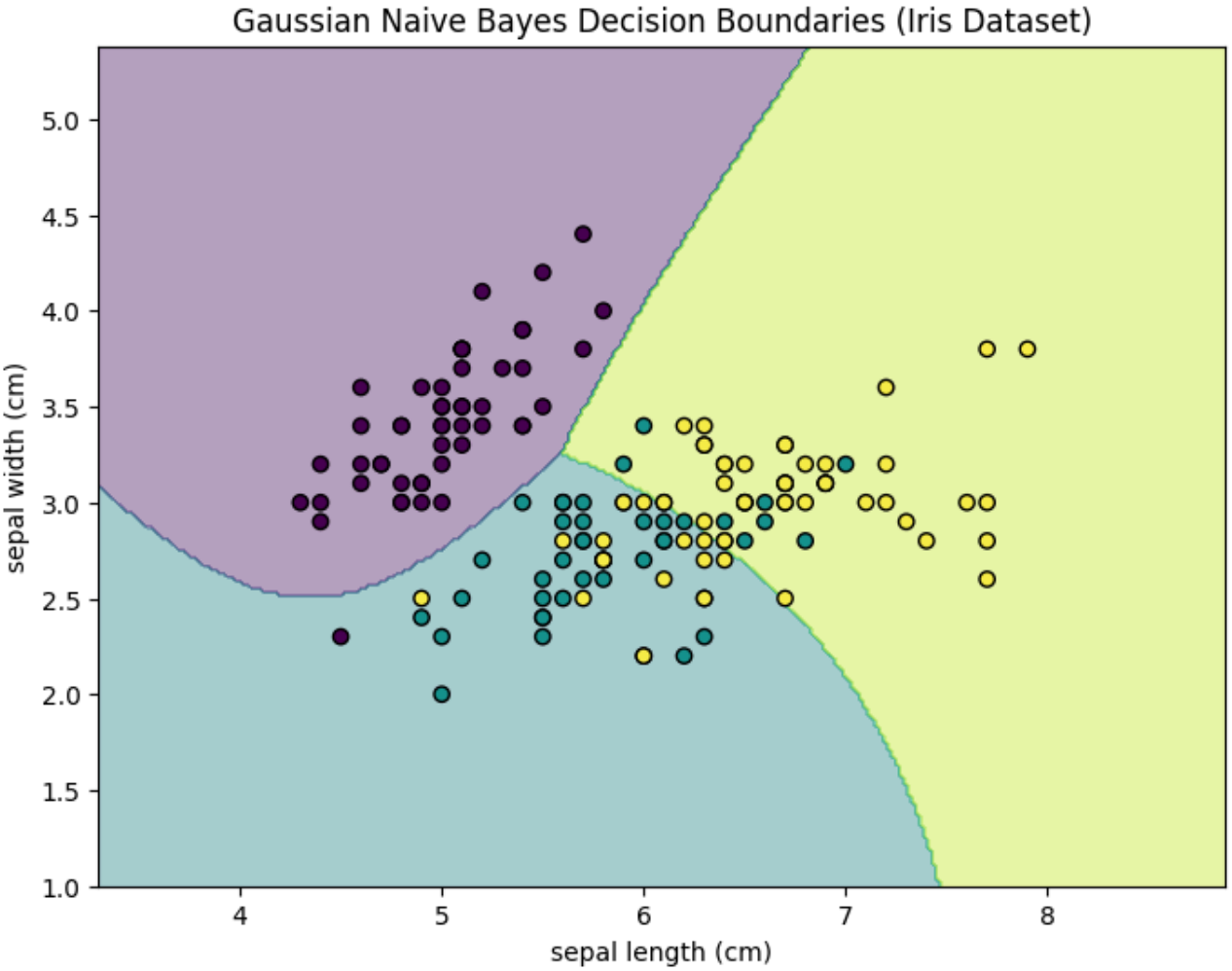
```
# Evaluate the accuracy of the classifier
```

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9777777777777777
```

In [27]:

```
# Load the Iris dataset
iris = load_iris()
X = iris.data[:, :2]  # Use only the first two features for visualization
y = iris.target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
# Create a Gaussian Naive Bayes classifier
gnb = GaussianNB()
gnb.fit(X_train, y_train)
# Create a meshgrid for plotting decision boundaries
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),
                     np.arange(y_min, y_max, 0.02))
# Predict the class for each point in the meshgrid
Z = gnb.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
# Plot the decision boundaries and data points
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, alpha=0.4)
plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k')
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
plt.title('Gaussian Naive Bayes Decision Boundaries (Iris Dataset)')
plt.show()
```


Gaussian Naive Bayes Decision Boundaries (Iris Dataset)
sepal width (cm)
sepal length (cm)
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
4
5
6
7
8

3.8 Conclusion:

The classification algorithm was successfully implemented using the Decision Tree and Naïve Bayes model. The model effectively learned patterns from the training data and predicted outcomes on the test data.

3.9 Questions:

1.What is a Decision Tree classifier, and how does it work?

2.Explain the Naive Bayes algorithm and its underlying assumptions.

3.Compare the working principles of Decision Tree and Naive Bayes classifiers.

4.What are the different types of Decision Tree splitting criteria?

Experiment No. 4

4.1 Aim: Implementation of Clustering Algorithm (K-means / Agglomerative)

4.2 Course Outcome: Apply classification algorithms in real world dataset for prediction and classification and concept clustering algorithms and its applications.

4.3 Requirement: Google Colab/ IDLE.

4.4 Related Theory:

Introduction to Clustering

Clustering is a type of **unsupervised learning** where the goal is to group similar data points into clusters, without predefined labels. Each group, or cluster, contains data points that are similar to each other, and dissimilar to data points in other clusters. Clustering is widely used in tasks like customer segmentation, anomaly detection, and image compression.

K-means Clustering Algorithm

K-means is a centroid-based algorithm, which partitions the data into **K** clusters. The algorithm works by iterating through the following steps:

1. **Initialization**: Randomly select K points as cluster centers (centroids).
2. **Assignment Step**: Assign each data point to the nearest centroid.
3. **Update Step**: Calculate the new centroids by taking the mean of the points in each cluster.
4. **Repeat**: Repeat steps 2 and 3 until the centroids do not change or the algorithm converges.

Advantages of K-means:

- Simple and fast, especially for large datasets.
- Works well when the clusters are spherical and equally sized.

Disadvantages of K-means:

- Sensitive to the initial placement of centroids.

- Not suitable for clusters of varying shapes and sizes.

Agglomerative Hierarchical Clustering

Agglomerative clustering is a **bottom-up** approach where each data point starts in its own cluster, and pairs of clusters are merged as the algorithm progresses.

Steps involved:

1. **Initialization**: Start with each data point as a single cluster.
2. **Merging**: Find the two clusters that are the most similar (based on a distance measure, such as Euclidean distance) and merge them.
3. **Repeat**: Repeat step 2 until all data points are merged into a single cluster.

Advantages of Agglomerative Clustering:

- Does not require the number of clusters to be specified in advance.
- Produces a dendrogram that shows how clusters are merged, which can be useful for understanding the structure of the data.

Disadvantages of Agglomerative Clustering:

- Computationally expensive for large datasets.
- Sensitive to the choice of distance metric.

Evaluation of Clustering

Since clustering is unsupervised, it is challenging to evaluate the performance without ground truth labels. However, some techniques used include:

- **Silhouette Score**: Measures how similar an object is to its own cluster compared to other clusters.
- **Inertia**: Used in K-means to measure the sum of squared distances from each point to its assigned centroid.

4.6 Algorithm:

1. Data Collection

- Choose a dataset with multiple features suitable for clustering (e.g., customer or product data).

2. Data Preprocessing

- Handle missing values and scale the data (if necessary) to ensure fair contribution from each feature.

3. Algorithm Selection

- **K-means**: Requires specifying the number of clusters (K).
- **Agglomerative**: Does not require predefining the number of clusters.

4. Implement Clustering

- **K-means**: Fit the model and assign each point to a cluster, based on K.
- **Agglomerative**: Fit the model and let the algorithm merge clusters iteratively.

5. Evaluate the Clusters

- Use metrics like **Silhouette Score** to evaluate clustering quality.
- **K-means**: Check **inertia** to measure within-cluster distances.

6. Visualize the Results

- Use **PCA** for dimensionality reduction and create scatter plots to visualize cluster separation.

7. Interpret and Analyze

- Review cluster characteristics and assess how well-separated the clusters are.

8. Fine-Tuning (Optional)

- **K-means**: Adjust K or reinitialize centroids if necessary.
- **Agglomerative**: Try different linkage criteria or adjust the distance threshold.

4.7 Program and Output:

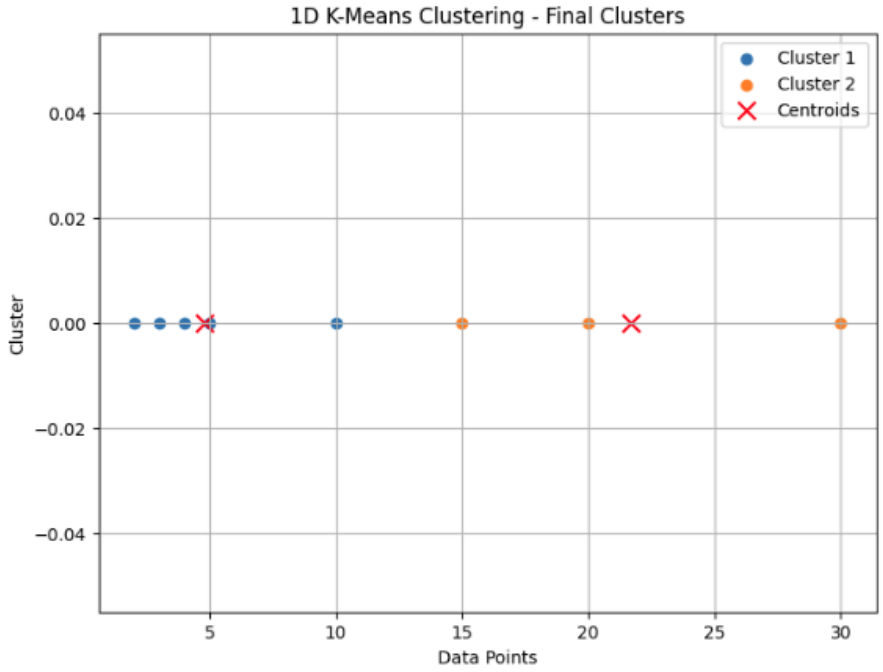
```
import numpy as np
import matplotlib.pyplot as plt
def mean(l):
    if not l:  # Check if the list is empty
        return 0  # Return 0 for an empty list
    n = len(l)
```

```
    sum_val = 0  # Avoid using 'sum' as a variable name
    for i in l:
        sum_val += i
    mean_val = sum_val / n
    return mean_val  # Return the calculated mean


def cluster(k1, k2, data):
    if not data:
        return  # Return instead of using exit()


    pc1 = []
    while True:
        c1 = []
        c2 = []
        for i in range(len(data)):
            if abs(data[i] - k1) <= abs(data[i] - k2):
                c1.append(data[i])
            else:
                c2.append(data[i])


        print("Cluster 1: ", c1)
        print("Cluster 2: ", c2)


        new_k1 = mean(c1)
        new_k2 = mean(c2)


        if sorted(pc1) == sorted(c1):
            break
        else:
            pc1 = c1[:]
            k1 = new_k1
            k2 = new_k2
    plt.figure(figsize=(8, 6))
    plt.scatter(c1, [0] * len(c1), label="Cluster 1")  # Plot Cluster 1
    plt.scatter(c2, [0] * len(c2), label="Cluster 2")  # Plot Cluster 2
    plt.scatter([k1, k2], [0, 0], marker='x', s=100, c='red', label='Centroids')  # Plot final centroids
    plt.xlabel("Data Points")
    plt.ylabel("Cluster")
    plt.title("1D K-Means Clustering - Final Clusters")
    plt.legend()
    plt.grid(True)
    plt.show()
```

return k1,k2

```
data = [2, 3, 4, 10, 20, 15, 5, 30]
k1 = data[0]
k2 = data[1]
cluster(k1, k2, data)
```

```
Cluster 1:  [2]
Cluster 2:  [3, 4, 10, 20, 15, 5, 30]
Cluster 1:  [2, 3, 4, 5]
Cluster 2:  [10, 20, 15, 30]
Cluster 1:  [2, 3, 4, 10, 5]
Cluster 2:  [20, 15, 30]
Cluster 1:  [2, 3, 4, 10, 5]
Cluster 2:  [20, 15, 30]
```




```
(4.8, 21.666666666666668)
```

4.7 Conclusion:

The k-means algorithm is a powerful and widely-used tool for clustering data into distinct groups based on their similarities. By iteratively assigning data points to clusters and updating the centroids, it efficiently identifies patterns and structures within the data. Its simplicity and scalability make it suitable for a variety of applications, from customer segmentation to image compression.

4.8 Questions:

1. **What is the K-means clustering algorithm, and how does it work?**
2. **How do you determine the optimal number of clusters in K-means?**
3. **What are the common distance metrics used in Agglomerative Clustering?**

Experiment No. 5

5.1 Aim: Implementation of Association Rule Mining Algorithm (Apriori)

5.2 Course Outcome: Relate to the concepts of market basket analysis in real world applications.

.

5.3 Requirement: Google Colab/ Python IDLE.

5.4 Related Theory:

Association Rule Mining

The Apriori algorithm is a data mining technique used for association rule mining, a process that identifies frequent patterns and dependencies among items in a dataset. It works by iteratively finding frequent itemsets (combinations of items) and generating association rules based on them.

1. Finding Frequent Itemsets:

The algorithm starts by identifying frequent single items (1-itemsets) based on a minimum support threshold.

It then iteratively generates candidate itemsets (combinations of items) of increasing length, for example, 2-itemsets, 3-itemsets, and so on.

For each iteration, it prunes (removes) candidates that don't meet the minimum support threshold, effectively reducing the number of candidates to be considered.

This process continues until no more frequent itemsets can be found.

2. Generating Association Rules:

Once frequent itemsets are identified, association rules are generated from them.

An association rule typically takes the form of "IF X THEN Y", where X and Y are sets of items.

The algorithm calculates metrics like support, confidence, and lift to evaluate the strength and relevance of the generated rules.

3. Example:

In a retail setting, the Apriori algorithm could identify that customers who frequently buy milk and bread also tend to buy eggs.

This would generate an association rule like "IF milk and bread THEN eggs".

The support would indicate how frequently this pattern occurs in the dataset, the confidence would tell how likely it is that eggs are purchased when milk and bread are, and the lift would show how much more likely it is that eggs are purchased with milk and bread compared to being purchased independently.

4. Benefits:

Discovering Hidden Patterns:

The algorithm helps uncover relationships and patterns that might not be immediately obvious.

Actionable Insights:

The generated association rules can be used for various applications, such as market basket analysis, recommendation systems, and customer behavior analysis.

Efficiency:

The Apriori algorithm uses an efficient "downward closure" property, which states that any subset of a frequent itemset is also frequent, to prune the search space and reduce computation.

5.5 Algorithm:

1. Install and import libraries.
2. Prepare or load the dataset.
3. Transform data using TransactionEncoder.
4. Apply apriori to get frequent itemsets.
5. Use association_rules to derive rules.
6. Analyze, filter, or save the results.

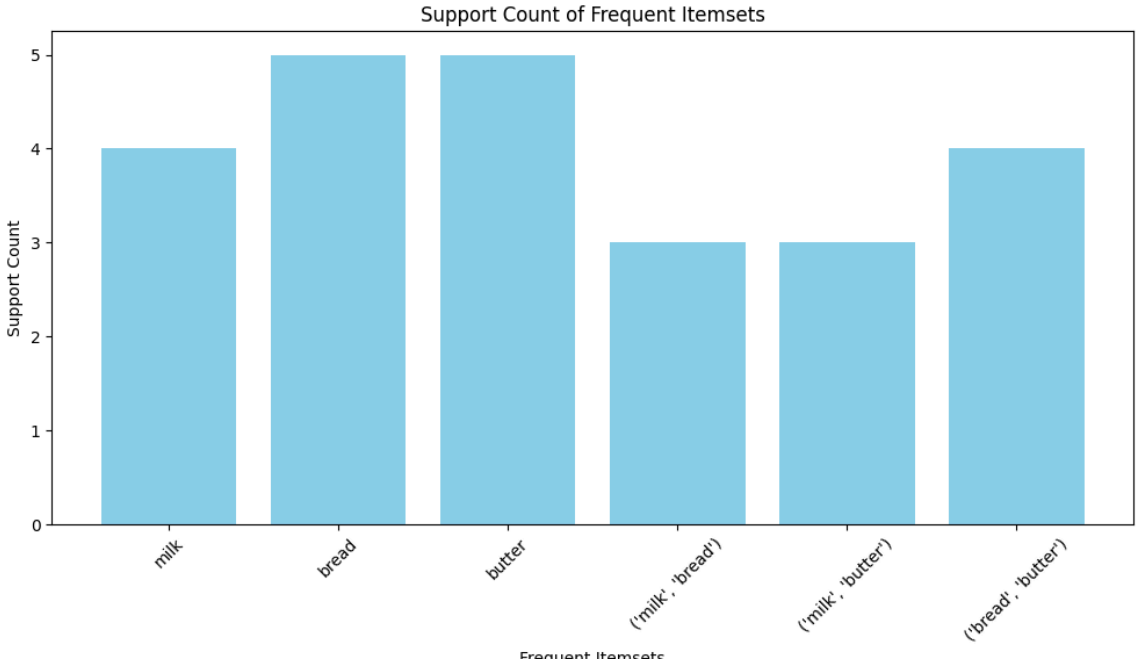
5.6 Program and Output:

```
from itertools import combinations
import matplotlib.pyplot as plt

def get_frequent_itemsets(transactions, min_support):
    itemsets = {}
    for transaction in transactions:
        for item in transaction:
            if item in itemsets:
                itemsets[item] += 1
            else:
                itemsets[item] = 1
    frequent_itemsets = {item: support for item, support in itemsets.items() if support >=
min_support}
    return frequent_itemsets

def get_candidate_itemsets(frequent_itemsets, k):
    candidates = []
    frequent_items = list(frequent_itemsets.keys())
    for combination in combinations(frequent_items, k):
        candidates.append(combination)
    return candidates

def apriori(transactions, min_support):
    k = 1
    frequent_itemsets = get_frequent_itemsets(transactions, min_support)
    all_frequent_itemsets = [frequent_itemsets]
    while frequent_itemsets:
        k += 1
        candidates = get_candidate_itemsets(frequent_itemsets, k)
        candidate_supports = {candidate: 0 for candidate in candidates}
        for transaction in transactions:
            for candidate in candidates:
                if set(candidate).issubset(set(transaction)):
                    candidate_supports[candidate] += 1
        frequent_itemsets = {itemset: support for itemset, support in candidate_supports.items() if
support >= min_support}
        if frequent_itemsets:
            all_frequent_itemsets.append(frequent_itemsets)
    return all_frequent_itemsets
```

```
transactions = [
    ['milk', 'bread', 'butter'],
    ['bread', 'butter'],
    ['milk', 'bread'],
    ['milk', 'butter'],
    ['bread', 'butter'],
    ['milk', 'bread', 'butter']
]

min_support = 2
frequent_itemsets = apriori(transactions, min_support)

for level, itemsets in enumerate(frequent_itemsets, start=1):
    print(f"\nFrequent {level}-itemsets:")
    for itemset, support in itemsets.items():
        print(f"{itemset}: {support}")

support_data = []
labels = []

for level, itemsets in enumerate(frequent_itemsets, start=1):
    for itemset, support in itemsets.items():
        support_data.append(support)
        labels.append(str(itemset))

plt.figure(figsize=(10, 6))
plt.bar(labels, support_data, color='skyblue')
plt.xlabel('Frequent Itemsets')
plt.ylabel('Support Count')
plt.title('Support Count of Frequent Itemsets')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Output:

```
Frequent 1-itemsets:
milk: 4
bread: 5
butter: 5

Frequent 2-itemsets:
('milk', 'bread'): 3
('milk', 'butter'): 3
('bread', 'butter'): 4
```



5.7 Conclusion:

The Apriori algorithm efficiently identifies frequent itemsets by iteratively generating and filtering candidates based on a minimum support threshold. It is widely used in market basket analysis and data mining to find associations between items. While effective, it can become computationally expensive for large datasets, requiring optimizations for better performance. Overall, this experiment demonstrates how Apriori helps uncover useful patterns in transactional data.

5.8 Questions:

Q1. What is the Apriori algorithm in Association Rule Mining?

Q2. What is the significance of support, confidence, and lift in Apriori?

Q3. How does changing the minimum support and confidence thresholds affect the number and quality of association rules generated using the Apriori algorithm?

Experiment No. 6

6.1 Aim: Implementation of Prediction Algorithm (Linear Regression)

6.2 Course Outcome: Apply classification algorithms in real world dataset for prediction and classification and concept clustering algorithms and its applications.

6.3 Requirement: Google Colab/ Python IDLE

6.4 Related Theory:

Linear Regression

Linear regression analysis is used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable.

1. Data Preparation:

The first step involves preparing the data by cleaning it, handling missing values, and transforming variables as needed.

2. Model Formulation:

A linear equation is formulated to represent the relationship between the dependent variable (Y) and the independent variable(s) (X): $Y = a + bX + e$, where 'a' is the intercept, 'b' is the slope, and 'e' is the error. If there are multiple independent variables, this becomes multiple linear regression.

3. Parameter Estimation:

The parameters (a and b, and the coefficients for multiple regression) are estimated using the least squares method, which minimizes the sum of squared errors (deviations from the regression line).

4. Model Evaluation:

The model's performance is assessed using metrics like R-squared (which measures the proportion of variance explained by the model) and residual analysis (which examines the errors to check for assumptions like normality and independence).

5. Interpretation and Prediction:

Once the model is deemed valid, the coefficients can be interpreted to understand the relationship between the variables, and the model can be used to make predictions.

Types of Linear Regression:

Simple Linear Regression: Uses one independent variable to predict the dependent variable.

Multiple Linear Regression: Uses two or more independent variables to predict the dependent variable.

Polynomial Regression: Extends linear regression by allowing for non-linear relationships by including polynomial terms of the independent variables.

Other Variations: Includes techniques like ridge regression, Lasso regression, and elastic net regression, which are used to handle multicollinearity and improve model stability.

6.5 Algorithm:

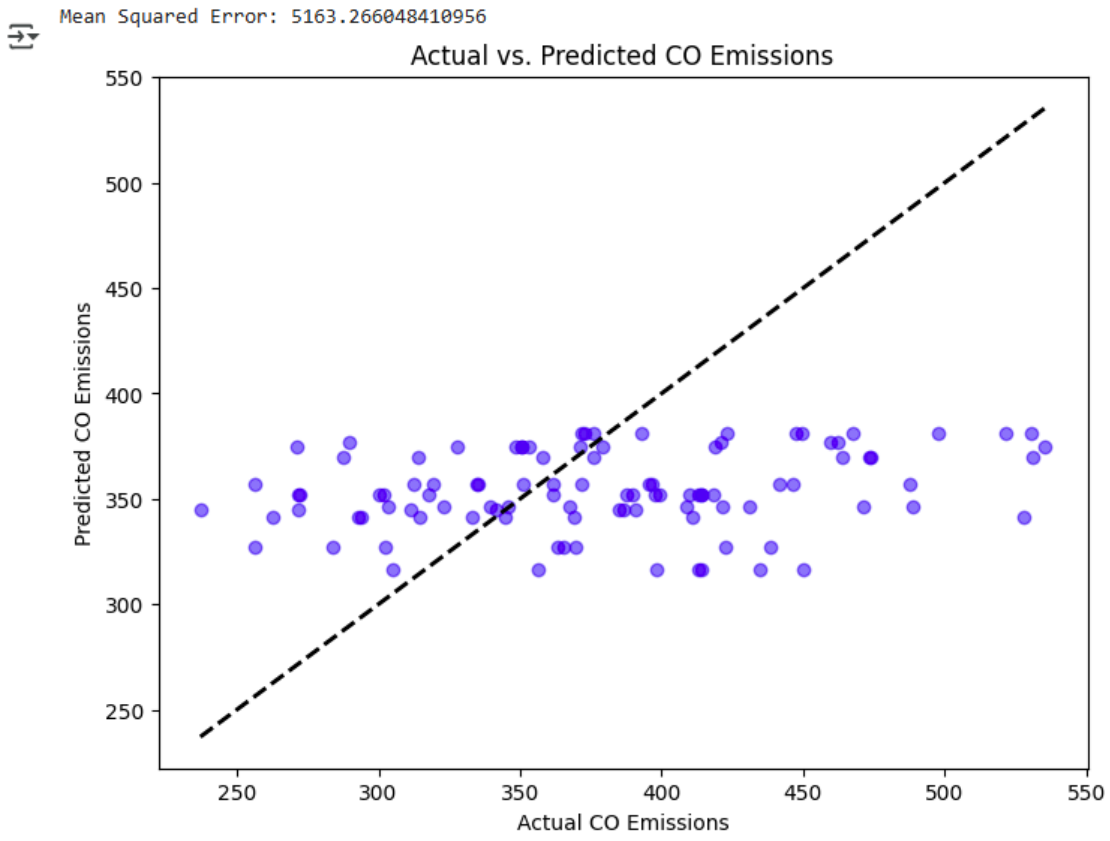
1. Import Libraries: Import necessary Python libraries for data handling, linear regression, and evaluation.
2. Prepare Data: Organize your data into independent (X) and dependent (Y) variables.
3. Split Data: Divide the data into training and testing sets.
4. Create Model: Initialize and train the linear regression model with the training data.
5. Make Predictions: Use the model to predict values from the test data.
6. Evaluate Model: Calculate the model's accuracy using metrics like MSE and R-squared.
7. Visualize Results: Plot the original data points and the regression line.
8. Interpret Results: Examine the coefficients and evaluate the goodness of fit.

6.6 Program and Output:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error


data = pd.read_csv('data.csv')
```

```
X = pd.get_dummies(data[['manufacturer', 'fuel_type']], drop_first=True)
y = data['co_emissions']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, color='blue', alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2)
plt.xlabel("Actual CO Emissions")
plt.ylabel("Predicted CO Emissions")
plt.title("Actual vs. Predicted CO Emissions")
plt.show()
```

Output:



6.7 Conclusion:

In this experiment, we implemented Simple Linear Regression to model the relationship between a dependent and an independent variable. The process involved preparing the data, training the model, and evaluating its performance using metrics like Mean Squared Error (MSE) and R-squared. The model was visualized by plotting the regression line against the data points, showing how well it fit the data.

6.8 Questions:

What are the key steps involved in implementing a simple linear regression model using Python and scikit-learn?

How can you evaluate the performance of a linear regression model in Python? List and explain at least two metrics.

What is the role of the train_test_split() function in building a linear regression model, and why is it important?

Experiment No. 7

7.1 Aim: Implementation of Data Discretization(any one) and Data Visualization (any one)

7.2 Course Outcome: Perceive the importance of data pre-processing and basics of data mining techniques.

7.4 Requirement: Google Colab/ IDLE

7.5 Related Theory:

1. Data Discretization (Equal Width Discretization)

Data Discretization is the process of transforming continuous data into a discrete form by dividing the range of continuous values into distinct intervals or categories. It is commonly used in data mining and machine learning, especially when algorithms require categorical data or when simplifying complex data for better interpretability.

There are various methods of data discretization, such as:

Equal Width Discretization (EWD): In this method, the range of data is divided into equal-width intervals or bins. Each bin has the same width, but the number of data points within each bin can vary. This is the most straightforward discretization method.

Equal Frequency Discretization (EFD): In this method, the range of data is divided such that each bin contains the same number of data points, but the width of each bin may vary.

K-Means Discretization: This method uses the K-means clustering algorithm to group data into a predefined number of clusters and then assigns the clusters as discrete labels.

Equal Width Discretization (EWD) divides the continuous range into a set of intervals where the width of each interval is constant. It's useful when there is a need for uniform bin sizes and a straightforward, simple method. The key concept is to map continuous values into these intervals, often represented as categorical labels.

Example: If a dataset contains values ranging from 10 to 50, and we choose 4 bins (num_bins = 4), the width of each bin is calculated as:

\text{Bin Width} = \frac{\text{max value} - \text{min value}}{\text{num_bins}} = \frac{50 - 10}{4} = 10

So, the bins would be [10-20], [20-30], [30-40], and [40-50].

Advantages of Discretization:

Reduces the complexity of continuous data, making it easier to analyze and visualize.

Can improve the performance of algorithms that work better with categorical data (e.g., decision trees).

Helps in dealing with noise and outliers in the data.

Disadvantages of Discretization:

Can lose information due to data simplification.

Binning methods like equal width may not represent the data distribution well, especially if the data is skewed.

2. Data Visualization (Histogram)

Data Visualization refers to the graphical representation of data and information. It involves the use of charts, graphs, and plots to display patterns, trends, and insights in data, making it easier to understand, interpret, and communicate findings.

Histogram is one of the most commonly used visualization techniques to display the distribution of continuous data. It consists of bars where the x-axis represents the range of data (divided into intervals or bins), and the y-axis represents the frequency or count of data points that fall within each bin.

Theory Behind Histograms:

Bins (or intervals): The continuous data is divided into intervals (or bins), and each bin has a range of values.

Frequency/Count: The number of data points that fall within each bin is counted and represented as the height of the bar for that bin.

Visualization of Distribution: Histograms help in visualizing the distribution, central tendency, variability, skewness, and any outliers in the data.

Key Features:

Shape of Distribution: The histogram allows you to understand the shape of the data distribution (normal, skewed, bimodal, etc.).

Central Tendency: Helps visualize the mean, median, and mode of the dataset.

Spread of Data: Indicates how widely the data is spread out (variance and standard deviation).

Example: A histogram of a dataset of student exam scores can show if most students scored in the middle range, whether scores are skewed to the high or low end, or if there are any unusual patterns like bimodal distributions.

Advantages of Histograms:

Provides an intuitive and quick visual representation of data distribution.

Easily identifies the presence of skewness, central tendencies, and outliers.

Helps in comparing multiple distributions.

Disadvantages:

Histograms are sensitive to bin sizes. Choosing too few bins can oversimplify data, while too many bins can lead to overfitting.

The choice of bin width is crucial in interpreting the data correctly. Too wide a bin can obscure important patterns, while too narrow a bin may highlight irrelevant details.

7.5 Algorithm:

1. Import Libraries: Import necessary libraries (pandas, numpy, matplotlib).
2. Prepare Data: Create a sample dataset containing continuous values.
3. Define Bins: Set the number of bins for discretization.

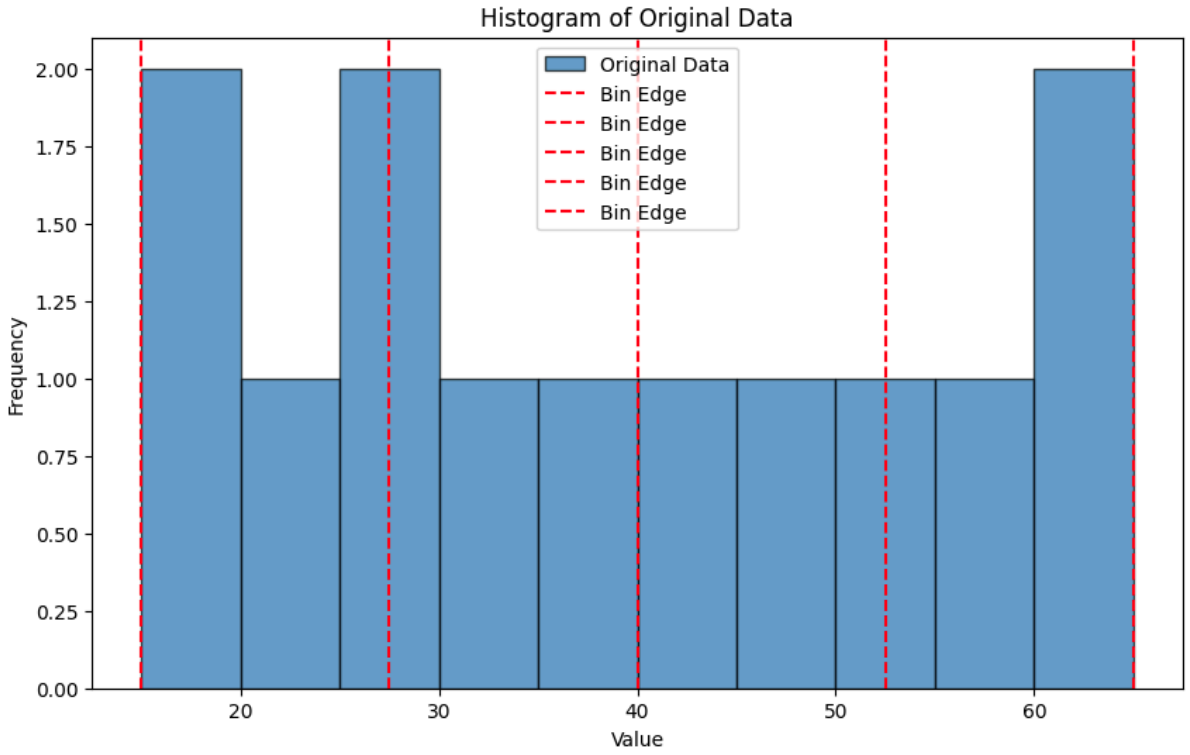
4. Apply Equal Width Discretization: Divide the data into equal-width intervals.
5. Display Results: Print the original and discretized data.
6. Visualize Data: Create a histogram to visualize the data distribution and mark the discretization bins.
7. Interpret Results: Analyze the discretized data and its visualization.

7.6 Program and Output:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data = pd.Series([15, 22, 37, 45, 18, 30, 60, 25, 40, 28, 50, 55, 65])
num_bins = 4
min_value = data.min()
max_value = data.max()
bin_width = (max_value - min_value) / num_bins
bins = [min_value + i * bin_width for i in range(num_bins + 1)]
labels = [f"Bin {i+1}" for i in range(num_bins)]
discretized_data = pd.cut(data, bins=bins, labels=labels, include_lowest=True)
print("Original Data:")
print(data)
print("\nDiscretized Data:")
print(discretized_data)
plt.figure(figsize=(10, 6))
plt.hist(data, bins=10, edgecolor='black', alpha=0.7, label='Original Data')
plt.title('Histogram of Original Data')
plt.xlabel('Value')
plt.ylabel('Frequency')
for bin_edge in bins:
    plt.axvline(x=bin_edge, color='red', linestyle='--', label="Bin Edge")
plt.legend()
plt.show()
```

Output:

```
Original Data:
0     15
1     22
2     37
3     45
4     18
5     30
6     60
7     25
8     40
9     28
10    50
11    55
12    65
dtype: int64

Discretized Data:
0     Bin 1
1     Bin 1
2     Bin 2
3     Bin 3
4     Bin 1
5     Bin 2
6     Bin 4
7     Bin 1
8     Bin 2
9     Bin 2
10    Bin 3
11    Bin 4
12    Bin 4
dtype: category
Categories (4, object): ['Bin 1' < 'Bin 2' < 'Bin 3' < 'Bin 4']
```


Histogram of Original Data
Original Data
Bin Edge
Bin Edge
Bin Edge
Bin Edge
Bin Edge
Frequency
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
20
30
40
50
60
Value

7.8 Conclusion:

Thus running the discretization and visualization, the continuous values were transformed into discrete bins and how the histogram displays the data's frequency distribution.
You can analyze how well the discretization reflects the data distribution and how the histogram gives insights into the central tendency, spread, and any skewness in the data.

7.9 Questions:

Q1 How does the process of Equal Width Discretization transform continuous data into discrete categories, and what role does the choice of the number of bins play in the outcome?

Q2 What insights can be derived from the histogram of the original continuous data, and how does it help in understanding the distribution of the data after discretization?

Q3 What are the potential advantages and limitations of using Equal Width Discretization for continuous data, and how can the choice of bin width affect the analysis of the data?

Experiment No. 8

8.1 Aim:

Perform data Pre-processing task and demonstrate Classification, Clustering, Association algorithm on data sets using data mining tool WEKA

8.2 Course Outcome: Relate to the concepts of market basket analysis in real world applications.

8.4 Requirement: Weka tool 3.9.6

8.5 Related Theory:

About Weka:

Weka (Waikato Environment for Knowledge Analysis) is a popular, open-source suite of machine learning software developed at the University of Waikato in New Zealand. It provides a collection of algorithms for data mining tasks such as classification, regression, clustering, association rule mining, and data pre-processing. Weka is written in Java and comes with a graphical user interface (GUI) as well as a command-line interface for users to interact with.

Key Features of Weka:

Wide Range of Algorithms:

Weka includes numerous algorithms for classification (e.g., decision trees, Naive Bayes, SVM), regression (e.g., linear regression, logistic regression), clustering (e.g., K-means, DBSCAN), association rule mining (e.g., Apriori), and attribute selection.

It also provides advanced techniques such as ensemble learning (bagging, boosting), dimensionality reduction (PCA), and evaluation methods (cross-validation, train-test split).

Preprocessing Tools:

Weka has many built-in tools for data preprocessing, such as filtering, discretization, normalization, and attribute selection. These tools help prepare the data for modeling.

Graphical User Interface (GUI):

The GUI allows users to interact with Weka without needing to write any code. Key components of the Weka GUI include:

Explorer: For data loading, preprocessing, model building, evaluation, and visualization.

Experimenter: For running experiments and comparing different machine learning algorithms.

Knowledge Flow: A flow-based environment for building machine learning workflows.

Simple CLI: A simple command-line interface for running algorithms on datasets.

Integration with Java:

Since Weka is written in Java, it can easily be integrated into Java-based applications. You can use the Weka library in Java code to apply machine learning algorithms programmatically.

Support for Various File Formats:

Weka supports several data file formats, including ARFF (Attribute-Relation File Format), CSV, and C4.5. The ARFF format is the native format for Weka datasets.

Visualization:

Weka provides various built-in visualization tools, including:

Scatter plot for visualizing relationships between attributes.

ROC (Receiver Operating Characteristic) and Confusion Matrix for evaluating classification models.

Cluster visualization for unsupervised learning models.

Extensibility:

Weka is highly extensible, allowing users to add new machine learning algorithms or preprocessing filters via Java programming. Weka also integrates with other libraries, like MOA (Massive Online Analysis) for stream mining and WEKA-Loader for connecting to databases.

Cross-validation and Evaluation:

Weka offers tools for model evaluation using cross-validation, train/test splits, and custom evaluation procedures. It also provides detailed performance metrics such as accuracy, precision, recall, F1 score, and confusion matrices for classification tasks.

Weka Components and Modules:

Weka Explorer:

A comprehensive GUI to load datasets, preprocess data, build models, evaluate models, and visualize results. It provides easy access to Weka's algorithms for both supervised and unsupervised learning.

Weka Experimenter:

A module for running and comparing multiple experiments to evaluate the performance of different algorithms on the same dataset. This tool helps in systematically testing machine learning models.

Weka Knowledge Flow:

This graphical interface allows users to create machine learning workflows by dragging and dropping various components, including data sources, classifiers, filters, and evaluators. It is a powerful tool for automating data processing and analysis.

Weka Command-Line Interface (CLI):

A lightweight interface for executing machine learning tasks via the command line. It allows automation and scripting of data analysis tasks.

Use Cases:

Educational Tool: Weka is commonly used in academia for teaching machine learning concepts due to its user-friendly GUI and variety of algorithms.

Data Mining: Data scientists and researchers use Weka for exploring and analyzing datasets to extract meaningful patterns and insights.

Rapid Prototyping: Developers and data scientists can quickly prototype and test machine learning models on small to medium-sized datasets using Weka.

Strengths:

Ease of Use: Weka is known for its user-friendly interface, making it accessible to beginners.

Wide Variety of Algorithms: Weka offers a large selection of machine learning algorithms for classification, regression, clustering, and association rule mining.

Visualization Tools: The visualization tools provided in Weka help users interpret and understand their models better.

Limitations:

Scalability: Weka is primarily designed for smaller datasets, and may not scale well for very large datasets (millions of instances).

Limited Support for Deep Learning: While Weka provides a rich set of algorithms, it is limited in terms of deep learning models when compared to specialized frameworks like TensorFlow or PyTorch.

Memory Usage: Weka can consume a lot of memory when working with large datasets, which may limit its use in resource-constrained environments.

KMeans

The K-Means algorithm is one of the most widely used unsupervised learning algorithms for clustering. It aims to partition a set of data points into K distinct clusters based on their features. The goal is to minimize the variance within each cluster while ensuring the points in different clusters are as dissimilar as possible.

K-Means Algorithm Overview:

Input: A dataset of n points, and a number K specifying how many clusters you want to form.

Output: A set of K clusters with their centroids and the assignment of each data point to one of the clusters.

8.6 Algorithm:

1. Initialize the centroids:
    - Randomly choose K data points from the dataset to serve as the initial centroids of the clusters.
2. Assign each data point to the nearest centroid:
    - For each data point, calculate the distance (usually Euclidean distance) to each centroid, and assign the data point to the cluster with the nearest centroid.
3. Update the centroids:
    - After all points have been assigned to clusters, recalculate the centroids by taking the mean (average) of all points in each cluster.
4. Repeat steps 2 and 3 until convergence:
    - Continue reassigning data points to the nearest centroid and updating centroids. The algorithm stops when the centroids no longer change, or the changes are minimal (i.e., convergence is reached), or after a fixed number of iterations.

Pseudocode for K-Means:

1. Initialize K centroids randomly from the data points.

2. Repeat until convergence:

   a. Assign each data point to the nearest centroid.

   b. Update each centroid to the mean of the points assigned to it.

3. Output the K clusters and their centroids.

K Means in Weka using Iris Dataset:

Process:

1.Go to Knowledge flow and select ARFLoader as the DataSource and configure iris dataset.

2. From Evaluation, select TrainingSet Maker and from Clusterers select Simple KMeans

3. For Evaluation select ClusterersPerformanceEvaluator

4. For the final output from Visualisation select TextViewer

5. Connect all of them with ARFLoader to TrainingSetMaker passing Dataset, Training SetMaker to Simple KMeans passing trainingSet, Simple KMeans to Clustered PerformanceEvaluator passing batchClustered and passing text to Textviewer.

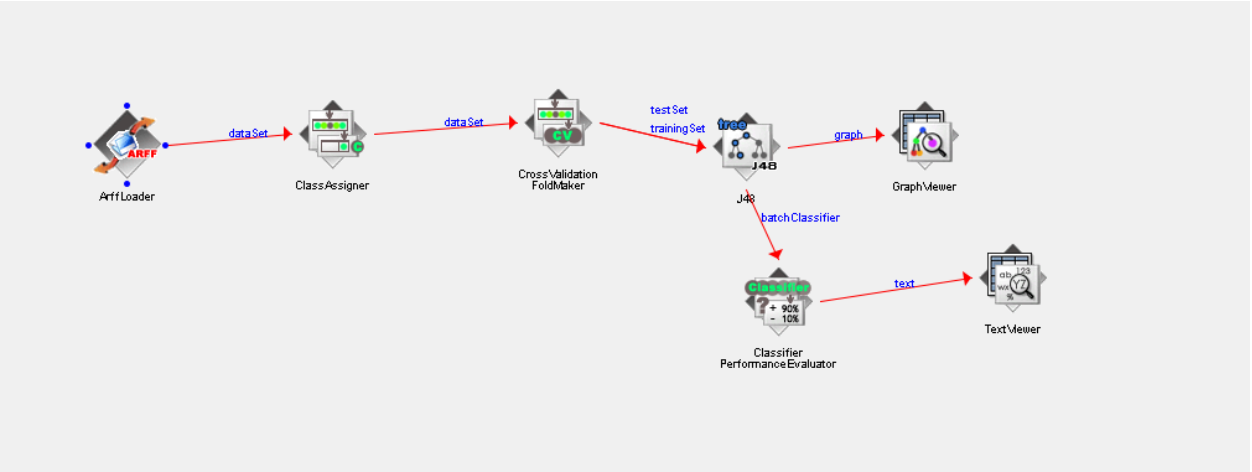
6. Check all the connections and execute.

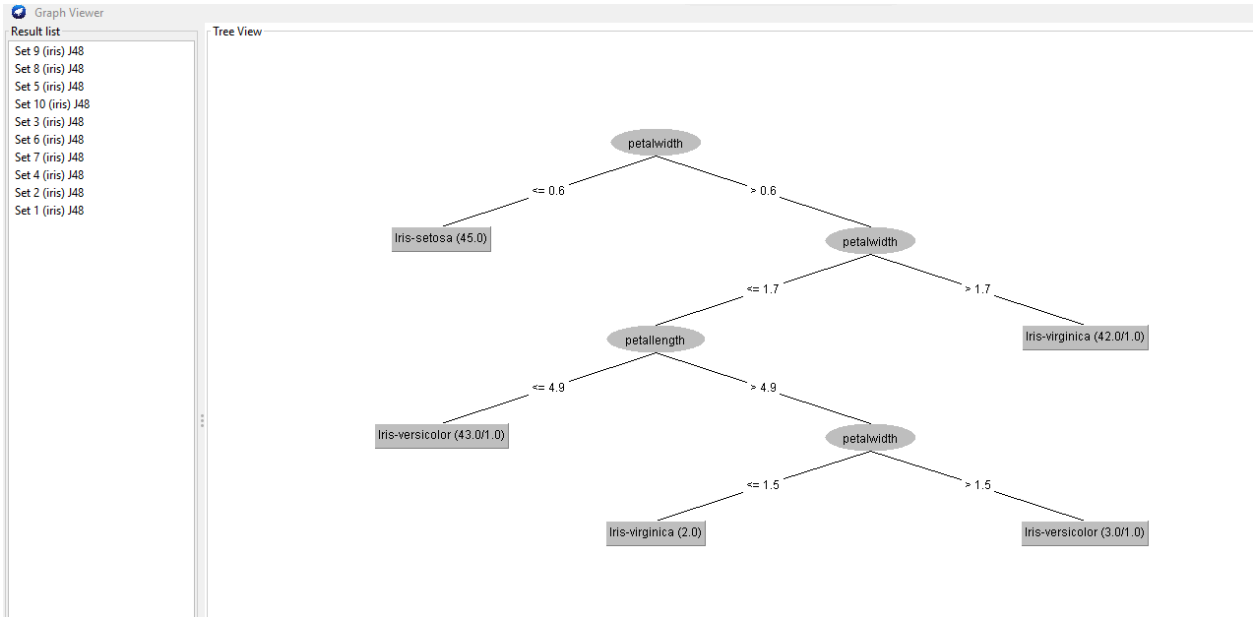
7. In Explorer, open the file for Iris dataset and view Pre-Processing and Clustering Model as the output.

8.7 Program and Output:

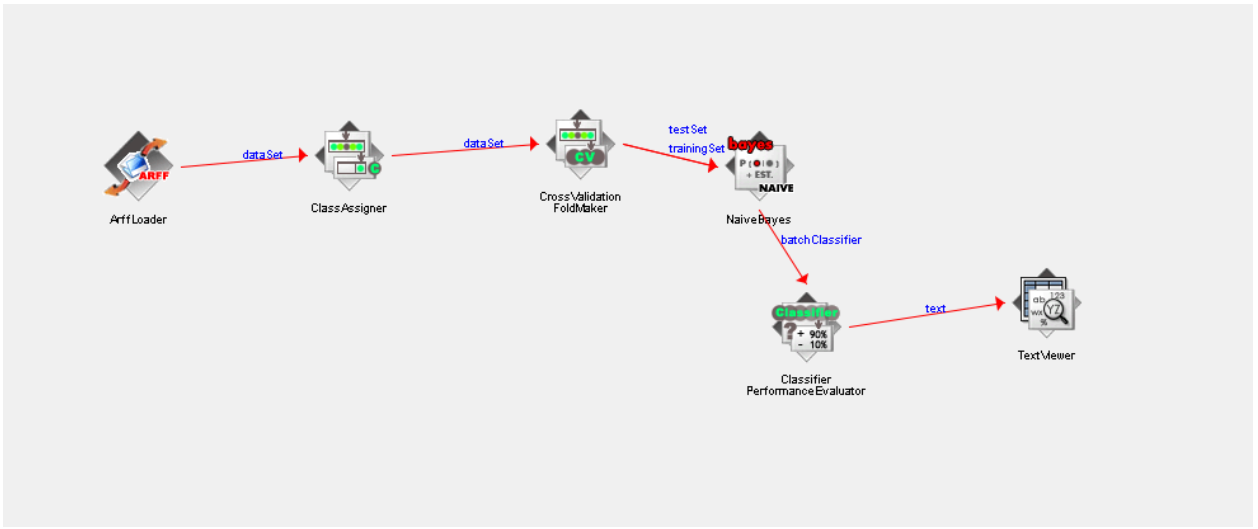
CLASSIFICATION:

DECISION TREE:





NAÏVE BAYESIAN:



Text Viewer

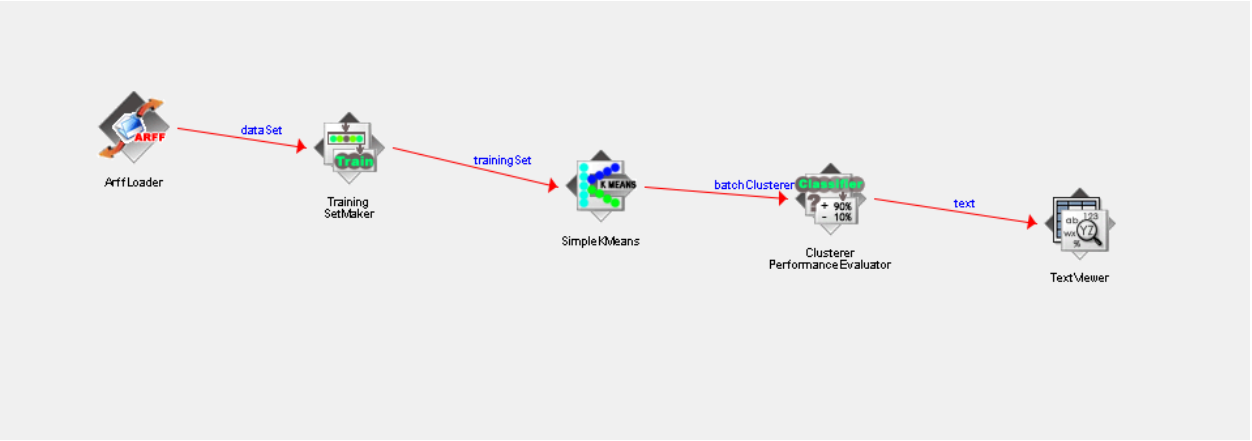
Result list

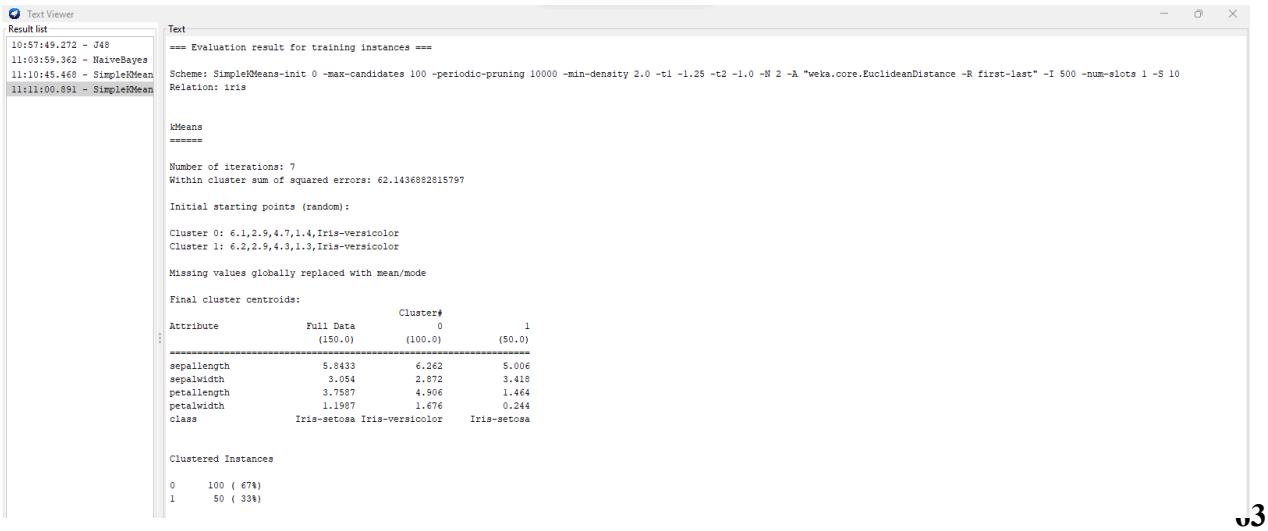
10:57:49.272 - J48
11:03:59.362 - NaiveBayes

Text

```
=== Evaluation result ===

Scheme: NaiveBayes
Relation: iris

=== Summary ===

Correctly Classified Instances         144               96      %
Incorrectly Classified Instances         6                4      %
Kappa statistic                          0.94
Mean absolute error                      0.0342
Root mean squared error                  0.155
Relative absolute error                  7.6997 %
Root relative squared error             32.8794 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-setosa
                 0.960    0.040    0.923      0.960    0.941      0.911    0.992     0.983     Iris-versicolor
                 0.920    0.020    0.958      0.920    0.939      0.910    0.992     0.986     Iris-virginica
Weighted Avg.    0.960    0.020    0.960      0.960    0.960      0.940    0.994     0.989

=== Confusion Matrix ===

  a  b  c   <-- classified as
 50  0  0 |  a = Iris-setosa
  0 48  2 |  b = Iris-versicolor
  0  4 46 |  c = Iris-virginica
```

K MEANS:





Text Viewer

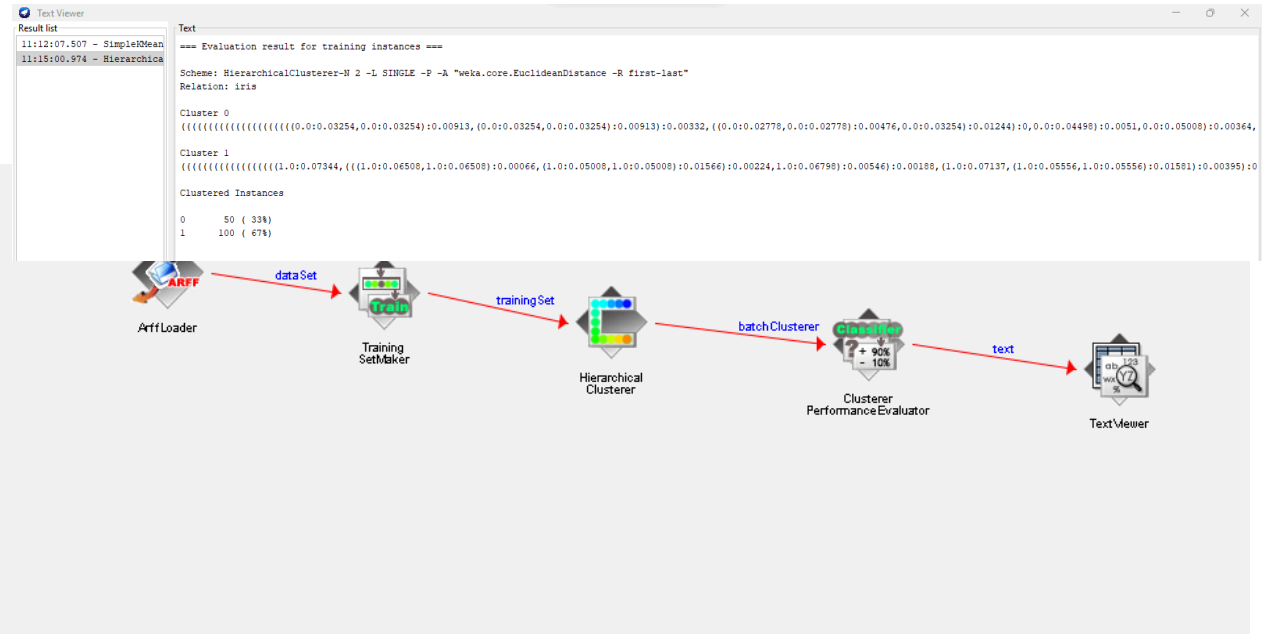
Result list

10:57:49.272 - J48
11:03:59.362 - NaiveBayes
11:10:45.468 - SimpleKMean
11:11:00.891 - SimpleKMean

Text

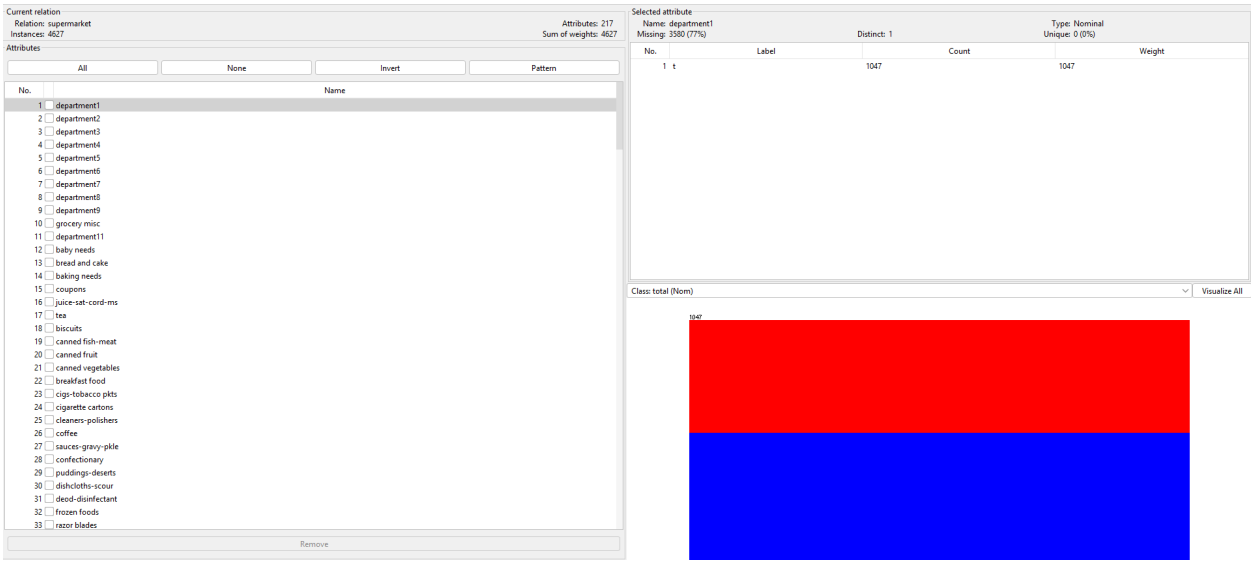
```
=== Evaluation result for training instances ===

Scheme: SimpleKMeans-init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10
Relation: iris


kMeans
======

Number of iterations: 7
Within cluster sum of squared errors: 62.1436882815797

Initial starting points (random):

Cluster 0: 6.1,2.9,4.7,1.4,Iris-versicolor
Cluster 1: 6.2,2.9,4.3,1.3,Iris-versicolor

Missing values globally replaced with mean/mode

Final cluster centroids:
                                Cluster#
Attribute               Full Data           0           1
                          (150.0)     (100.0)      (50.0)
=============================================================
sepallength                5.8433      6.262       5.006
sepalwidth                 3.054       2.872       3.418
petallength                3.7587      4.906       1.464
petalwidth                 1.1987      1.676       0.244
class                 Iris-setosa Iris-versicolor  Iris-setosa


Clustered Instances

0      100 ( 67%)
1       50 ( 33%)
```

AGGLOMERATIVE:

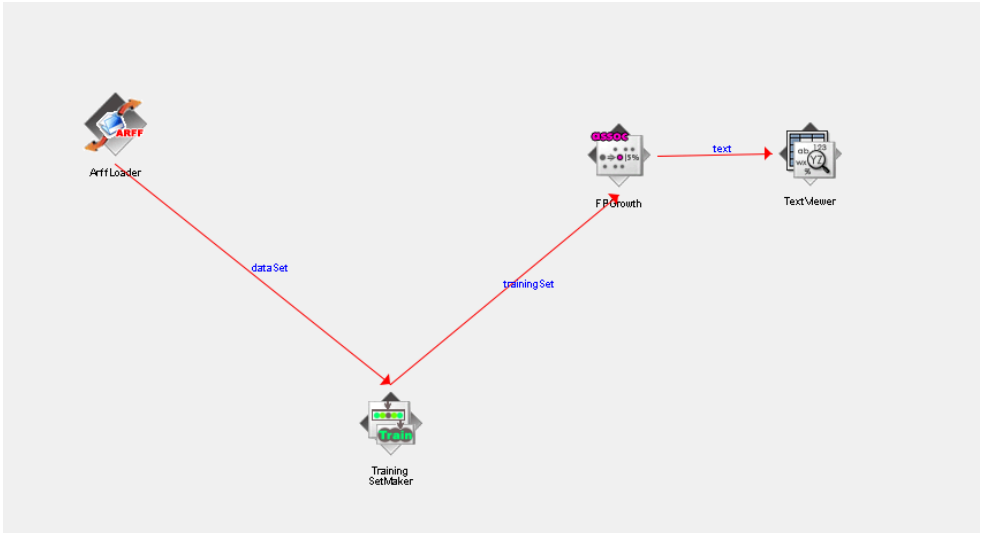


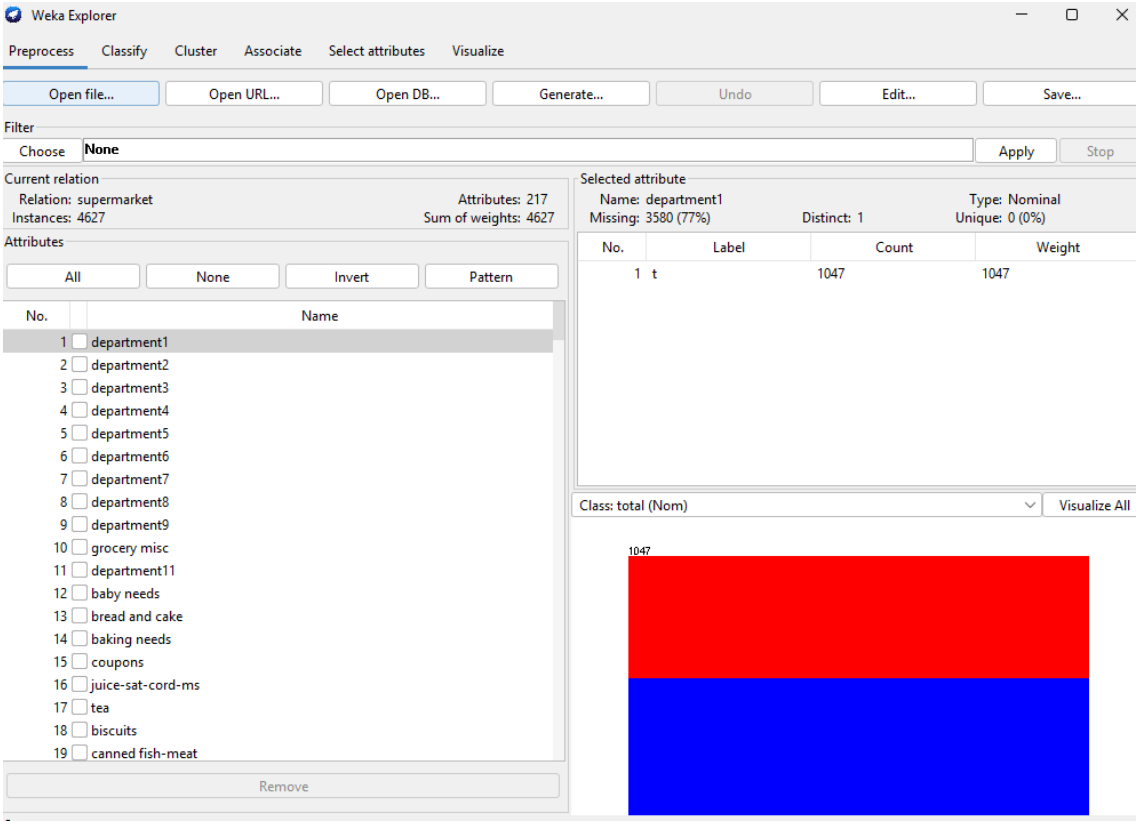
ASSOCIATION:

APRIORI ALOGORITHM:



FP Growth:





8.8 Conclusion:

Thus data Pre-processing task to demonstrate Classification, Clustering and Association algorithm on data sets using data mining tool WEKA is implemented successfully.

8.9 Questions:

1. **Which WEKA panel allows you to load datasets and apply filters before modeling?**
2. **What kind of learning does clustering fall under?Which algorithm in WEKA is used to generate association rules**
3. **What evaluation metric in WEKA tells you how well your classification model is performing?**
4. **True or False: In WEKA, normalization is a pre-processing step used to scale data values between 0 and 1.**

Manavir Education Trust's
Shah & Anchor Kutchhi Engineering College
An Autonomous Institute Affiliated to University of Mumbai

UG Program in Computer Engineering
(Accredited by NBA for 3 years from AY 2022-23)
Academic Year: 2024-2025

Laboratory Manual

Data Warehousing and Mining Lab

(CMLR0603)

Semester VI

Bachelor of Technology

(B. Tech.)

in

Computer Engineering Department

Third Year with Effect from AY 2024 -2025