Thadomal Shahani Engineering College

Bandra (W.), Mumbai - 400050

Division : C2

Batch : 3

Roll Number : 1902112

Certify that Mr. / Miss Tushar Nankani

of Computer Department, Semester **V** with

Roll No. 1902112 has completed a course of the necessary

experiments in the subject **DWM** under

my supervision in the Thadomal Shahani Engineering College

Laboratory in the year 2021 - 2022

Prof. Seema Kolkur
Teacher In-Charge

Dr. Tanuja Sarode
Head of the Department

Dr. GT Thampi
Principal

Date 23-10-2021

Lab Code	Lab Name	Sem
CSL503 Sem-V Rev.2019	**DATA WAREHOUSING AND MINING**	V

LAB OUTCOME

1. Design data warehouse and perform various OLAP operations.

2. Implement data mining algorithms like classification.

3. Implement clustering algorithms on a given set of data sample.

4. Implement Association rule mining & web mining algorithm

LIST OF EXPERIMENTS

Experiment No	Title of Experiments	Course Outcome	Date
1	One case study on building Data warehouse/Data Mart • Write Detailed Problem statement and design dimensional modelling (creation of star and snowflake schema)	LO1	20/7
2	Implementation of all dimension table and fact table based on experiment 1 case study	LO1	27/7
3	Implementation of OLAP operations: Slice, Dice, Rollup, Drilldown and Pivot based on experiment 1	LO1	3/8
4	Implementation of Bayesian algorithm	LO2	10/8
5	Implementation of Data Discretization (any one) & Visualization (any one)	L02	24/8
6	Perform data Pre-processing task and Demonstrate performing Classification,Clustering, Association algorithm on data sets using data mining tool (WEKA/R tool)	LO2, LO3	7/9
7	Implementation of Clustering algorithm (K-means/K-medoids)	LO3	21/9
8	Implementation of any one Hierarchical Clustering method	LO3	28/9
9	Implementation of Association Rule Mining algorithm(Apriori)	LO4	5/10
10	Implementation of Page rank/HITS algorithm	LO4	20/10

List of Written Assignments:

11. What is metadata? Explain data warehouse Metadata with Example . (LO1)

12. Write short Notes on (LO4)
    a. Multilevel Association rules and Mutidimensional association rules
    b. Web Usage Mining

Experiment No 1

Date: 20/07/2021

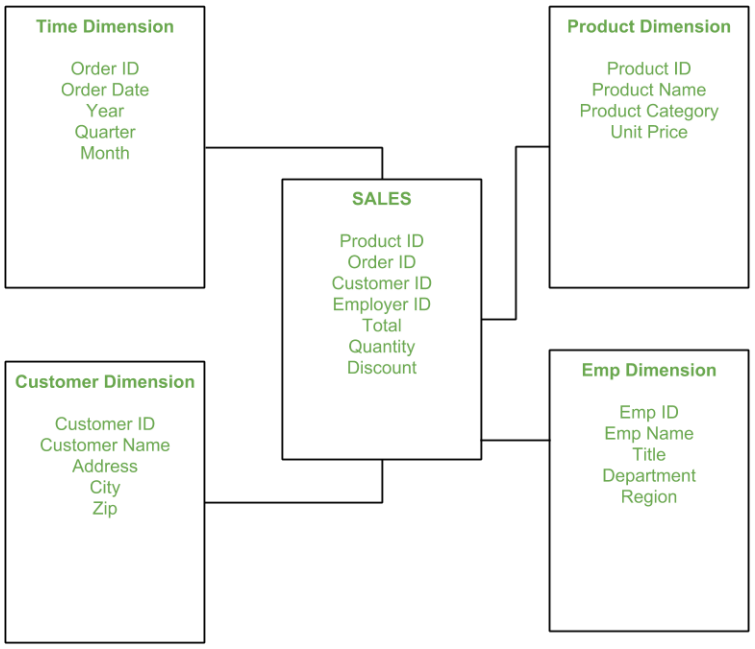
Aim: One case study on building Data warehouse/Data Mart• Write Detailed Problem statement and design dimensional modelling (creation of star and snowflake schema)

Theory:

Star schema**:**

Star schema is the fundamental schema among the data mart schema and it is simplest. This schema is widely used to develop or build a data warehouse and dimensional data marts. It includes one or more fact tables indexing any number of dimensional tables. The star schema is a necessary cause of the snowflake schema. It is also efficient for handling basic queries.

It is said to be star as its physical model resembles to the star shape having a fact table at its center and the dimension tables at its peripheral representing the star's points. Below is an example to demonstrate the Star Schema:



In the above demonstration, SALES is a fact table having attributes i.e. (Product ID, Order ID, Customer ID, Employer ID, Total, Quantity, Discount) which references to the dimension tables. Employee dimension table contains the attributes: Emp ID, Emp Name, Title, Department and Region. Product dimension table contains the attributes: Product ID, Product Name, Product Category, Unit Price. Customer dimension table contains the attributes: Customer ID, Customer Name, Address, City, Zip. Time dimension table contains the attributes: Order ID, Order Date, Year, Quarter, Month.

Advantages of Star Schema :

1. Simpler Queries –

Join logic of star schema is quite cinch in comparison to other join logic which are needed to fetch data from a transactional schema that is highly normalized.

2. Simplified Business Reporting Logic –

In comparison to a transactional schema that is highly normalized, the star schema makes simpler common business reporting logic, such as as-of reporting and period-over-period.

3. Feeding Cubes –

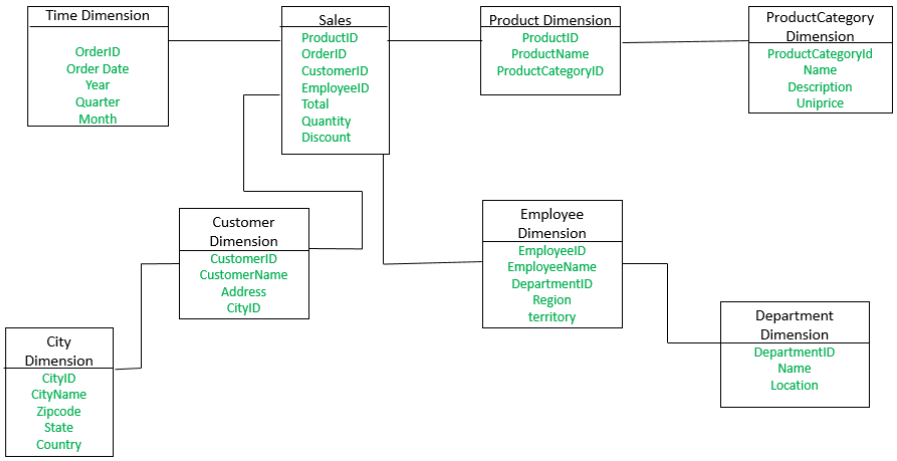
Star schema is widely used by all OLAP systems to design OLAP cubes efficiently. In fact, major OLAP systems deliver a ROLAP mode of operation which can use a star schema as a source without designing a cube structure.

Disadvantages of Star Schema –

1. Data integrity is not enforced well since in a highly de-normalized schema state.
2. Not flexible in terms if analytical needs as a normalized data model.
3. Star schemas don't reinforce many-to-many relationships within business entities – at least not frequently.

Snowflake schema:

The snowflake schema is a variant of the star schema. Here, the centralized fact table is connected to multiple dimensions. In the snowflake schema, dimensions are present in a normalized form in multiple related tables. The snowflake structure materialized when the dimensions of a star schema are detailed and highly structured, having several levels of relationship, and the child tables have multiple parent tables. The snowflake effect affects only the dimension tables and does not affect the fact tables.



The Employee dimension table now contains the attributes: EmployeeID, EmployeeName, DepartmentID, Region, Territory. The DepartmentID attribute links with the Employee table with the Department dimension table. The Department dimension is used to provide detail about each department, such as the Name and Location of the department. The Customer dimension table now contains the attributes: CustomerID, CustomerName, Address, CityID. The CityID attributes link the Customer dimension table with the City dimension table. The City dimension table has details about each city such as CityName, Zipcode, State, and Country.

The main difference between star schema and snowflake schema is that the dimension table of the snowflake schema is maintained in the normalized

form to reduce redundancy. The advantage here is that such tables (normalized) are easy to maintain and save storage space. However, it also means that more joins will be needed to execute the query. This will adversely impact system performance.

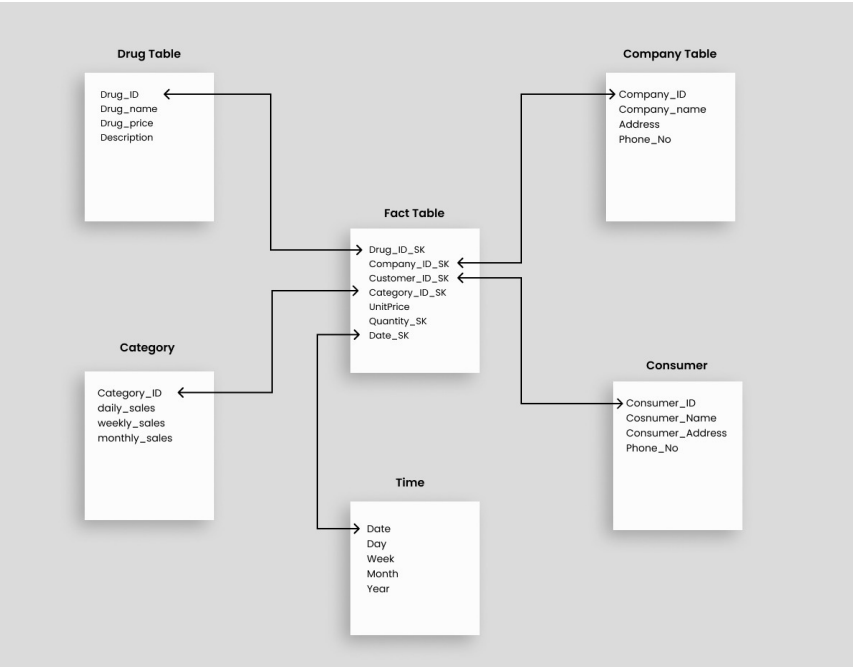
Advantages:
There are two main advantages of snowflake schema given below:

- It provides structured data which reduces the problem of data integrity.
- It uses small disk space because data are highly structured.

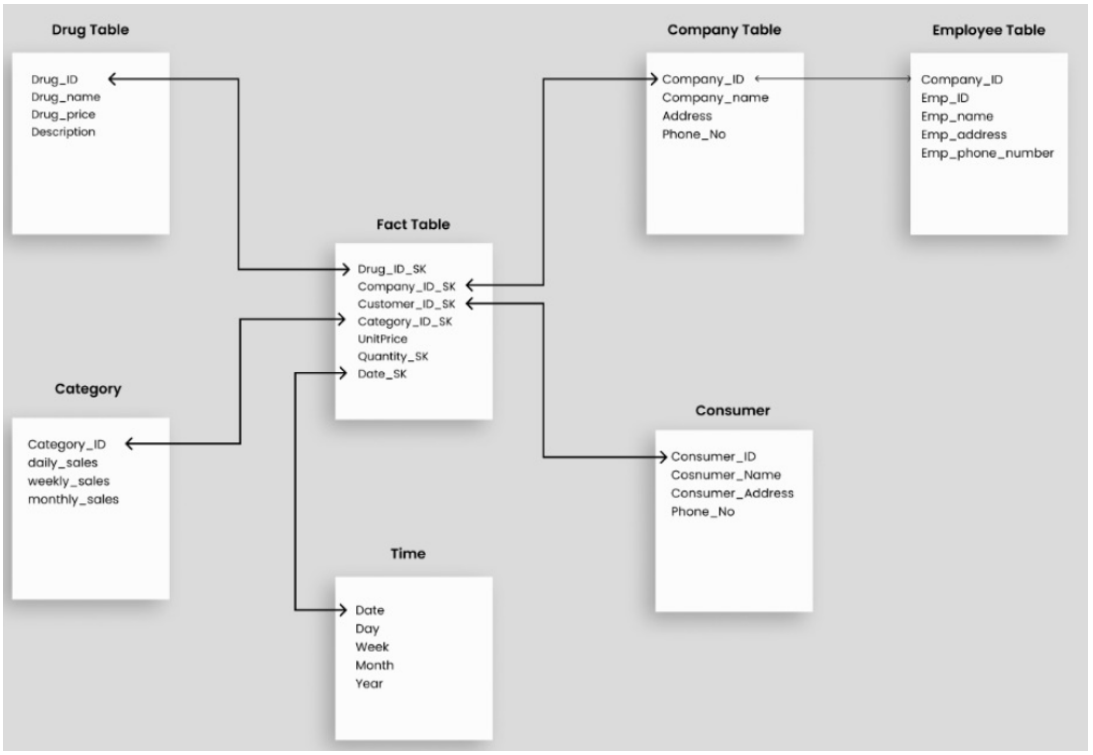
Disadvantages:

- Snowflaking reduces space consumed by dimension tables but compared with the entire data warehouse the saving is usually insignificant.
- Avoid snowflaking or normalization of a dimension table, unless required and appropriate.
- Do not snowflake hierarchies of one dimension table into separate tables. Hierarchies should belong to the dimension table only and should never be snowflakes.
- Multiple hierarchies that can belong to the same dimension have been designed at the lowest possible detail.

Output:



Star schema for a pharmaceutical company



Snowflake schema for a pharmaceutical company

Experiment No 2

Date: 27/07/2021

Aim: Implementation of all dimension table and fact table based one experiment 1 case study

Theory:

Fact Table:

A fact table is a primary table in a dimensional model.

A Fact Table contains:

- Measurements/facts
- Foreign key to dimension table

Dimension table:

- A dimension table contains dimensions of a fact.
- They are joined to the fact table via a foreign key.
- Dimension tables are de-normalized tables.
- The Dimension Attributes are the various columns in a dimension table
- Dimensions offers descriptive characteristics of the facts with the help of their attributes
- No set limit set for given for number of dimensions
- The dimension can also contain one or more hierarchical relationships

KEY DIFFERENCE:

- Fact table contains measurements, metrics, and facts about a business process while the Dimension table is a companion to the fact table which contains descriptive attributes to be used as query constraining.
- Fact table is located at the center of a star or snowflake schema, whereas the Dimension table is located at the edges of the star or snowflake schema.
- Fact table is defined by their grain or its most atomic level whereas Dimension table should be wordy, descriptive, complete, and quality assured.

- Fact table helps to store report labels whereas the Dimension table contains detailed data.
- Fact table does not contain a hierarchy whereas the Dimension table contains hierarchies.

Output:

```
CREATE TABLE drug
(
drug_id VARCHAR (20) PRIMARY KEY,
drug_name VARCHAR (20),
drug_price INTEGER,
description VARCHAR (20)
);

CREATE TABLE company
(
company_id VARCHAR (20) PRIMARY KEY,
company_name VARCHAR (20),
address VARCHAR (20),
phone_no VARCHAR (20)
);

CREATE TABLE consumer
(
consumer_id VARCHAR (20) PRIMARY KEY,
consumer_name VARCHAR (20),
consumer_address VARCHAR (20),
phone_no VARCHAR (20)
);

CREATE TABLE t_time
(
t_date DATE PRIMARY KEY,
t_day INTEGER,
t_week INTEGER,
t_month INTEGER,
t_year INTEGER
);

CREATE TABLE category
(
category_id VARCHAR (20),
daily_sales INTEGER,
weekly_sales INTEGER,
monthly_sales INTEGER,
PRIMARY KEY (category_id)
```

```
);

CREATE TABLE fact
(
drug_id_sk VARCHAR2(20),
company_id_sk VARCHAR2(20),
consumer_id_sk VARCHAR2(20),
category_id_sk VARCHAR2(20),
date_sk DATE,
UnitPrice INTEGER,
Quantity INTEGER,
PRIMARY KEY(drug_id_sk,company_id_sk,consumer_id_sk,category_id_sk,date_sk),
FOREIGN KEY (category_id_sk) REFERENCES category(category_id),
FOREIGN KEY (drug_id_sk) REFERENCES drug(drug_id),
FOREIGN KEY (company_id_sk) REFERENCES company(company_id),
FOREIGN KEY (consumer_id_sk) REFERENCES consumer(consumer_id),
FOREIGN KEY (date_sk) REFERENCES t_time(t_date)
);



INSERT INTO drug VALUES ('1234','Augmentin',62,'antibiotics');
INSERT INTO drug VALUES ('5678','Crocin',80,'paracetamol');
INSERT INTO drug VALUES ('9101','Calpol',40,'paracetamol');
INSERT INTO drug VALUES ('1112','Ascoril',90,'Cough Syrup');
INSERT INTO drug VALUES ('1214','Combiflam',35,'paracetamol');
INSERT INTO drug VALUES ('7864','Ambien',52,'antibiotics');
INSERT INTO drug VALUES ('7812','Symbicort',80,'paracetamol');
INSERT INTO drug VALUES ('0657','azelastine',40,'paracetamol');


INSERT INTO company VALUES('C1','Company 1','MUMBAI','9136339446');
INSERT INTO company VALUES('C2','Company 2','PUNE','8898461028');
INSERT INTO company VALUES('C3','Company 3','DELHI','9619989159');
INSERT INTO company VALUES('C4','Company 4','PUNJAB','9414224524');
INSERT INTO company VALUES('C5','Company 5','KOLKATA','9987663816');
INSERT INTO company VALUES('C6','Company 6','DELHI','9619989159');
INSERT INTO company VALUES('C7','Company 7','PUNJAB','9414224524');
INSERT INTO company VALUES('C8','Company 8','KOLKATA','9987663816');


INSERT INTO consumer VALUES ('P1','Parth Namdev','MUMBAI','8898461028');
INSERT INTO consumer VALUES ('P2','Kavya Nair','DELHI','9136339446');
INSERT INTO consumer VALUES ('P3','Raj Mehta','HYDERABAD','9619989159');
INSERT INTO consumer VALUES ('P4','Rita Shah','KASHMIR','9414252425');
INSERT INTO consumer VALUES ('P5','Tushar Nankani','JAIPUR','9987453616');
INSERT INTO consumer VALUES ('P6','Rishabh Jain','MUMBAI','8898461028');
```

```
INSERT INTO consumer VALUES ('P7','Tej Mandani','DELHI','9136339446');
INSERT INTO consumer VALUES ('P8','Om Shidhaye','HYDERABAD','9619989159');


INSERT INTO t_time VALUES (DATE '2014-01-31',1,1,1,2014);
INSERT INTO t_time VALUES (DATE '2015-01-31',1,1,1,2015);
INSERT INTO t_time VALUES (DATE '2016-01-31',1,1,1,2016);
INSERT INTO t_time VALUES (DATE '2017-01-31',1,1,1,2017);
INSERT INTO t_time VALUES (DATE '2018-01-31',1,1,1,2018);
INSERT INTO t_time VALUES (DATE '2019-01-31',1,1,1,2019);
INSERT INTO t_time VALUES (DATE '2020-04-30',4,4,4,2020);
INSERT INTO t_time VALUES (DATE '2021-05-31',5,5,5,2021);

INSERT INTO category VALUES ('M01AB',4.02,33.04,159.09);
INSERT INTO category VALUES ('M01AE',3.56,25.17,112.8);
INSERT INTO category VALUES ('N02BA',2.06,23.05,124.8);
INSERT INTO category VALUES ('N02BE',34.5,45.6,178.38);
INSERT INTO category VALUES ('N05B',7.03,51.06,270.06);
INSERT INTO category VALUES ('N05C',2.06,23.05,124.8);
INSERT INTO category VALUES ('R03',34.5,45.6,178.38);
INSERT INTO category VALUES ('R06',7.03,51.06,270.06);

INSERT INTO fact VALUES ('1234','C1','P1','M01AB',DATE '2014-01-31',200,300);
INSERT INTO fact VALUES ('1234','C2','P2','M01AB',DATE '2014-01-31',400,100);
INSERT INTO fact VALUES ('1234','C3','P3','M01AB',DATE '2014-01-31',250,380);
INSERT INTO fact VALUES ('1112','C8','P8','M01AE',DATE '2015-01-31',200,300);
INSERT INTO fact VALUES ('5678','C5','P5','M01AE',DATE '2017-01-31',550,280);
INSERT INTO fact VALUES ('1112','C6','P6','N05B',DATE '2021-05-31',340,100);
INSERT INTO fact VALUES ('5678','C7','P7','N05B',DATE '2016-01-31',220,350);
INSERT INTO fact VALUES ('5678','C4','P4','M01AB',DATE '2014-01-31',110,600);
INSERT INTO fact VALUES ('1234','C1','P2','M01AB',DATE '2014-01-31',200,300);
INSERT INTO fact VALUES ('1234','C2','P3','M01AB',DATE '2014-01-31',400,100);
INSERT INTO fact VALUES ('1234','C3','P4','M01AB',DATE '2014-01-31',250,380);
INSERT INTO fact VALUES ('1112','C8','P1','M01AE',DATE '2015-01-31',200,300);
INSERT INTO fact VALUES ('5678','C5','P6','M01AE',DATE '2017-01-31',550,280);
INSERT INTO fact VALUES ('1112','C6','P7','N05B',DATE '2021-05-31',340,100);
INSERT INTO fact VALUES ('5678','C7','P8','N05B',DATE '2016-01-31',220,350);
INSERT INTO fact VALUES ('5678','C4','P5','M01AB',DATE '2014-01-31',110,600);
INSERT INTO fact VALUES ('9101','C1','P3','M01AB',DATE '2014-01-31',200,300);
INSERT INTO fact VALUES ('1234','C2','P5','M01AB',DATE '2014-01-31',400,100);
INSERT INTO fact VALUES ('1234','C3','P7','M01AB',DATE '2014-01-31',250,380);
INSERT INTO fact VALUES ('1112','C8','P2','M01AE',DATE '2015-01-31',200,300);
INSERT INTO fact VALUES ('5678','C5','P7','M01AE',DATE '2017-01-31',550,280);
INSERT INTO fact VALUES ('1112','C6','P8','N05B',DATE '2021-05-31',340,100);
INSERT INTO fact VALUES ('9101','C7','P1','M01AE',DATE '2016-01-31',220,350);
INSERT INTO fact VALUES ('5678','C4','P6','M01AB',DATE '2014-01-31',110,600);
INSERT INTO fact VALUES ('9101','C5','P5','N05B',DATE '2017-01-31',150,450);
INSERT INTO fact VALUES ('1214','C1','P8','N05C',DATE '2019-01-31',200,300);
```

```
INSERT INTO fact VALUES ('1214','C2','P5','N02BA',DATE '2020-04-30',400,100);
INSERT INTO fact VALUES ('1234','C3','P3','N02BA',DATE '2014-01-31',250,380);
INSERT INTO fact VALUES ('1112','C8','P8','N02BA',DATE '2015-01-31',200,300);
INSERT INTO fact VALUES ('5678','C5','P5','R03',DATE '2017-01-31',550,280);
INSERT INTO fact VALUES ('1112','C6','P6','R03',DATE '2021-05-31',340,100);
INSERT INTO fact VALUES ('5678','C7','P7','R06',DATE '2016-01-31',220,350);
INSERT INTO fact VALUES ('5678','C4','P4','M01AE',DATE '2014-01-31',110,600);
INSERT INTO fact VALUES ('1234','C1','P2','N02BA',DATE '2014-01-31',200,300);
INSERT INTO fact VALUES ('1234','C2','P3','M01AE',DATE '2014-01-31',400,100);
INSERT INTO fact VALUES ('7864','C3','P4','R06',DATE '2014-01-31',250,380);
INSERT INTO fact VALUES ('0657','C8','P1','R06',DATE '2015-01-31',200,300);
INSERT INTO fact VALUES ('7864','C5','P6','R03',DATE '2017-01-31',550,280);
INSERT INTO fact VALUES ('0657','C6','P7','N05C',DATE '2021-05-31',340,100);
INSERT INTO fact VALUES ('9101','C7','P8','N05B',DATE '2021-05-31',220,350);
INSERT INTO fact VALUES ('1214','C4','P5','N02BA',DATE '2017-01-31',110,600);
INSERT INTO fact VALUES ('7864','C1','P3','M01AE',DATE '2016-01-31',200,300);
INSERT INTO fact VALUES ('7864','C2','P5','M01AB',DATE '2015-01-31',400,100);
INSERT INTO fact VALUES ('7864','C3','P7','N05C',DATE '2018-01-31',250,380);
INSERT INTO fact VALUES ('0657','C8','P2','N05C',DATE '2019-01-31',200,300);
INSERT INTO fact VALUES ('5678','C5','P7','M01AE',DATE '2019-01-31',550,280);
INSERT INTO fact VALUES ('0657','C6','P8','N05B',DATE '2020-04-30',340,100);
INSERT INTO fact VALUES ('1214','C7','P1','N02BE',DATE '2021-05-31',220,350);
INSERT INTO fact VALUES ('1214','C4','P6','M01AE',DATE '2021-05-31',110,600);
INSERT INTO fact VALUES ('1214','C5','P5','N05C',DATE '2019-01-31',150,450);
INSERT INTO fact VALUES ('1234','C1','P6','M01AB',DATE '2014-01-31',200,300);
INSERT INTO fact VALUES ('1234','C2','P7','M01AB',DATE '2014-01-31',400,100);
INSERT INTO fact VALUES ('1234','C3','P8','M01AB',DATE '2014-01-31',250,380);
INSERT INTO fact VALUES ('1112','C8','P5','M01AE',DATE '2015-01-31',200,300);
INSERT INTO fact VALUES ('5678','C5','P2','M01AE',DATE '2017-01-31',550,280);
INSERT INTO fact VALUES ('1112','C6','P3','N05B',DATE '2021-05-31',340,100);
INSERT INTO fact VALUES ('5678','C7','P4','N05B',DATE '2016-01-31',220,350);
INSERT INTO fact VALUES ('5678','C4','P1','M01AB',DATE '2014-01-31',110,600);
INSERT INTO fact VALUES ('1234','C1','P4','M01AB',DATE '2014-01-31',200,300);
INSERT INTO fact VALUES ('1214','C2','P5','M01AB',DATE '2014-01-31',400,100);
INSERT INTO fact VALUES ('1234','C3','P6','M01AB',DATE '2014-01-31',250,380);
INSERT INTO fact VALUES ('1112','C8','P3','M01AE',DATE '2015-01-31',200,300);
INSERT INTO fact VALUES ('5678','C5','P8','M01AE',DATE '2017-01-31',550,280);
INSERT INTO fact VALUES ('1112','C6','P1','N05B',DATE '2021-05-31',340,100);
INSERT INTO fact VALUES ('5678','C7','P2','N05B',DATE '2016-01-31',220,350);
INSERT INTO fact VALUES ('5678','C4','P7','M01AB',DATE '2014-01-31',110,600);
INSERT INTO fact VALUES ('9101','C1','P7','M01AB',DATE '2014-01-31',200,300);
INSERT INTO fact VALUES ('1234','C2','P8','M01AB',DATE '2014-01-31',400,100);
INSERT INTO fact VALUES ('1234','C3','P1','M01AB',DATE '2014-01-31',250,380);
INSERT INTO fact VALUES ('1112','C8','P6','M01AE',DATE '2015-01-31',200,300);
INSERT INTO fact VALUES ('5678','C5','P3','M01AE',DATE '2017-01-31',550,280);
INSERT INTO fact VALUES ('1112','C6','P4','N05B',DATE '2021-05-31',340,100);
INSERT INTO fact VALUES ('9101','C7','P5','M01AE',DATE '2016-01-31',220,350);
INSERT INTO fact VALUES ('5678','C4','P2','M01AB',DATE '2014-01-31',110,600);
```

```
INSERT INTO fact VALUES ('9101','C5','P3','N05B',DATE '2017-01-31',150,450);
INSERT INTO fact VALUES ('1214','C1','P2','N05C',DATE '2019-01-31',200,300);
INSERT INTO fact VALUES ('1214','C2','P8','N02BA',DATE '2020-04-30',400,100);
INSERT INTO fact VALUES ('1234','C3','P6','N02BA',DATE '2014-01-31',250,380);
INSERT INTO fact VALUES ('1112','C8','P2','N02BA',DATE '2015-01-31',200,300);
INSERT INTO fact VALUES ('5678','C5','P3','R03',DATE '2017-01-31',550,280);
INSERT INTO fact VALUES ('1112','C6','P4','R03',DATE '2021-05-31',340,100);

select * from drug
select * from company
select * from fact
```

SQL Worksheet

```
194  INSERT INTO fact VALUES ('1112','C6','P4','R03',DATE '2021-05-31',340,100);
195
196  select * from drug
197  select * from company
198  select * from fact
199
```

DRUG_ID	DRUG_NAME	DRUG_PRICE	DESCRIPTION
1234	Augmentin	62	antibiotics
5678	Crocin	80	paracetamol
9101	Calpol	40	paracetamol
1112	Ascoril	90	Cough Syrup
1214	Combiflam	35	paracetamol
7864	Ambien	52	antibiotics
7812	Symbicort	80	paracetamol
0657	azelastine	40	paracetamol

Download CSV

8 rows selected

SQL Worksheet

```
194  INSERT INTO fact VALUES ('1112','C6','P4','R03',DATE '2021-05-31',340,100);
195
196  select * from drug
197  select * from company
198  select * from fact
199
```

COMPANY_ID	COMPANY_NAME	ADDRESS	PHONE_NO
C1	Company 1	MUMBAI	9136339446
C2	Company 2	PUNE	8898461028
C3	Company 3	DELHI	9619989159
C4	Company 4	PUNJAB	9414224524
C5	Company 5	KOLKATA	9987663816
C6	Company 6	DELHI	9619989159
C7	Company 7	PUNJAB	9414224524
C8	Company 8	KOLKATA	9987663816

Download CSV

8 rows selected

SQL Worksheet

```
196  select * from drug
197  select * from company
198  select * from fact
199
```

DRUG_ID_SK	COMPANY_ID_SK	CONSUMER_ID_SK	CATEGORY_ID_SK	DATE_SK	UNITPRICE	QUANTITY
1234	C1	P1	M01AB	31-JAN-14	200	300
1234	C2	P2	M01AB	31-JAN-14	400	100
1234	C3	P3	M01AB	31-JAN-14	250	380
1112	C8	P8	M01AE	31-JAN-15	200	300
5678	C5	P5	M01AE	31-JAN-17	550	280
1112	C6	P6	N05B	31-MAY-21	340	100
5678	C7	P7	N05B	31-JAN-16	220	350
5678	C4	P4	M01AB	31-JAN-14	110	600
1234	C1	P2	M01AB	31-JAN-14	200	300
1234	C2	P3	M01AB	31-JAN-14	400	100

SQL Worksheet

```
196 select * from drug
197 select * from company
198 select * from fact
199
```

1214	C4	P5	N02BA	31-JAN-17	110	600
7864	C1	P3	M01AE	31-JAN-16	200	300
7864	C2	P5	M01AB	31-JAN-15	400	100
7864	C3	P7	N05C	31-JAN-18	250	380
0657	C8	P2	N05C	31-JAN-19	200	300
5678	C5	P7	M01AE	31-JAN-19	550	280
0657	C6	P8	N05B	30-APR-20	340	100
1214	C7	P1	N02BE	31-MAY-21	220	350
1214	C4	P6	M01AE	31-MAY-21	110	600
1214	C5	P5	N05C	31-JAN-19	150	450

Download CSV

Rows 1 - 50. More rows exist.

Experiment No 3

Date: 03/08/2021

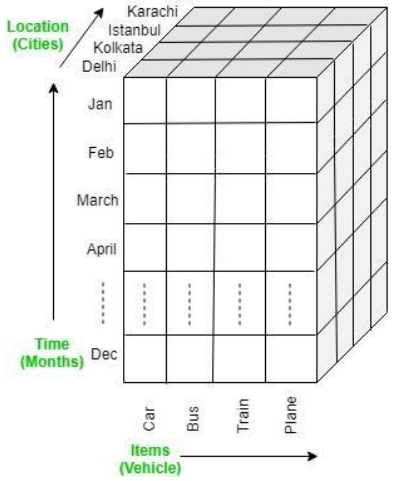
Aim: Implementation of OLAP operations: Slice, Dice, Rollup, Drilldown and Pivot based on experiment 1

Theory:

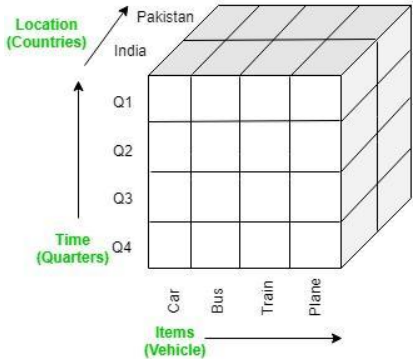
OLAP stands for Online Analytical Processing Server. It is a software technology that allows users to analyze information from multiple database systems at the same time. It is based on multidimensional data model and allows the user to query on multi-dimensional data (eg. Delhi -> 2018 -> Sales data). OLAP databases are divided into one or more cubes and these cubes are known as Hyper-cubes.

OLAP operations:
There are five basic analytical operations that can be performed on an OLAP cube:

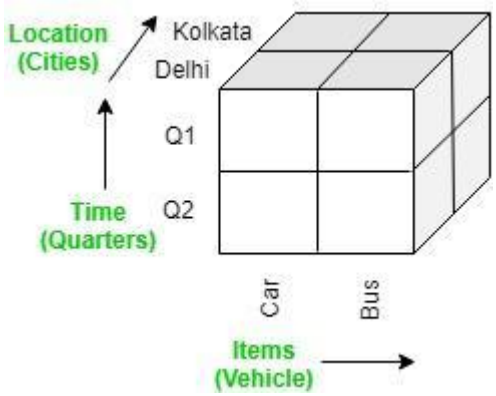
Drill down: In drill-down operation, the less detailed data is converted into highly detailed data. It can be done by:
Moving down in the concept hierarchy
Adding a new dimension
In the cube given in overview section, the drill down operation is performed by moving down in the concept hierarchy of Time dimension (Quarter -> Month).



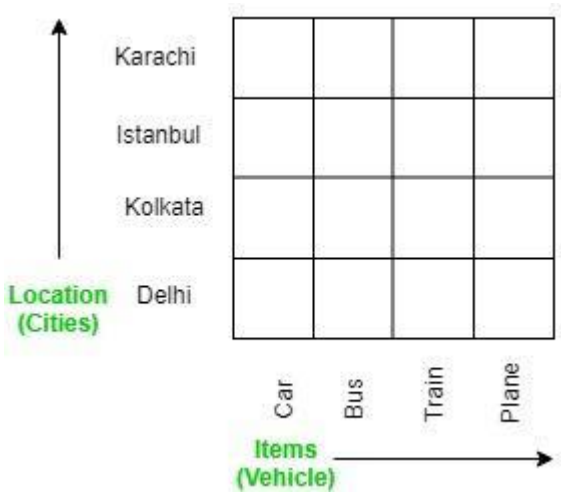
Roll up: It is just opposite of the drill-down operation. It performs aggregation on the OLAP cube. It can be done by:
Climbing up in the concept hierarchy
Reducing the dimensions
In the cube given in the overview section, the roll-up operation is performed by climbing up in the concept hierarchy of Location dimension (City -> Country).



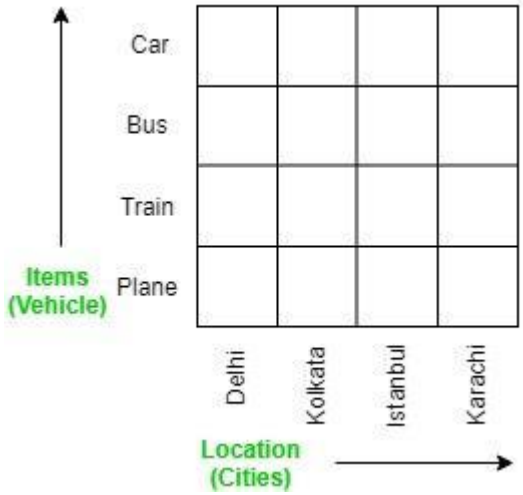
Dice: It selects a sub-cube from the OLAP cube by selecting two or more dimensions. In the cube given in the overview section, a sub-cube is selected by selecting following dimensions with criteria:
Location = “Delhi” or “Kolkata”
Time = “Q1” or “Q2”
Item = “Car” or “Bus”



Slice: It selects a single dimension from the OLAP cube which results in a new sub-cube creation. In the cube given in the overview section, Slice is performed on the dimension Time = “Q1”.



Pivot: It is also known as rotation operation as it rotates the current view to get a new view of the representation. In the sub-cube obtained after the slice operation, performing pivot operation gives a new view of it.



Output:

Statement 1

Edit

```
CREATE TABLE drug
(
drug_id VARCHAR (20) PRIMARY KEY,
drug_name VARCHAR (20),
drug_price INTEGER,
description VARCHAR (20)
)
```

Table created.

Statement 2

Edit

```
CREATE TABLE company
(
company_id VARCHAR (20) PRIMARY KEY,
company_name VARCHAR (20),
address VARCHAR (20),
phone_no VARCHAR (20)
)
```

Table created.

Statement 3

Edit

```
CREATE TABLE consumer
(
consumer_id VARCHAR (20) PRIMARY KEY,
consumer_name VARCHAR (20),
consumer_address VARCHAR (20),
phone_no VARCHAR (20)
)
```

Table created.

Statement 4

Edit

```
CREATE TABLE t_time
(
t_date DATE PRIMARY KEY,
t_day INTEGER,
t_week INTEGER,
t_month INTEGER,
t_year INTEGER
)
```

Table created.

Statement 5

Edit

```
CREATE TABLE category
(
category_id VARCHAR (20),
daily_sales INTEGER,
weekly_sales INTEGER,
monthly_sales INTEGER,
PRIMARY KEY (category_id)
)
```

Table created.

Statement 6

Edit

```
CREATE TABLE fact
(
drug_id_sk VARCHAR2(20),
company_id_sk VARCHAR2(20),
consumer_id_sk VARCHAR2(20),
category_id_sk VARCHAR2(20),
date_sk DATE,
UnitPrice INTEGER,
```

My Scripts \ **Script**

Statement **6**

Edit

```
CREATE TABLE fact
(
drug_id_sk VARCHAR2(20),
company_id_sk VARCHAR2(20),
consumer_id_sk VARCHAR2(20),
category_id_sk VARCHAR2(20),
date_sk DATE,
UnitPrice INTEGER,
Quantity INTEGER,
PRIMARY KEY(drug_id_sk,company_id_sk,consumer_id_sk,category_id_sk,date_sk),
FOREIGN KEY (category_id_sk) REFERENCES category(category_id),
FOREIGN KEY (drug_id_sk) REFERENCES drug(drug_id),
FOREIGN KEY (company_id_sk) REFERENCES company(company_id),
FOREIGN KEY (consumer_id_sk) REFERENCES consumer(consumer_id),
FOREIGN KEY (date_sk) REFERENCES t_time(t_date)
)
```

Table created.

Statement **7**

Edit

```
INSERT INTO drug VALUES ('1234','Augmentin',62,'antibiotics')
```

1 row(s) inserted.

Statement **8**

Edit

```
INSERT INTO drug VALUES ('5678','Crocin',80,'paracetamol')
```

1 row(s) inserted.

Statement **9**

Edit

```
INSERT INTO drug VALUES ('9101','Calpol',40,'paracetamol')
```

Statement **10**

Edit

```
INSERT INTO drug VALUES ('1112','Ascoril',90,'Cough Syrup')
```

1 row(s) inserted.

Statement **11**

Edit

```
INSERT INTO drug VALUES ('1214','Combiflam',35,'paracetamol')
```

1 row(s) inserted.

Statement **12**

Edit

```
INSERT INTO drug VALUES ('7864','Ambien',52,'antibiotics')
```

1 row(s) inserted.

Statement **13**

Edit

```
INSERT INTO drug VALUES ('7812','Symbicort',80,'paracetamol')
```

1 row(s) inserted.

Statement **14**

Edit

```
INSERT INTO drug VALUES ('0657','azelastine',40,'paracetamol')
```

1 row(s) inserted.

Statement **15**

Edit

```
INSERT INTO company VALUES('C1','Company 1','MUMBAI','9136339446')
```

1 row(s) inserted.

Statement **16**

Edit

```
INSERT INTO company VALUES('C2','Company 2','PUNE','8898461028')
```

1 row(s) inserted.

Statement **17**

Edit

```
INSERT INTO company VALUES('C3','Company 3','DELHI','9619989159')
```

1 row(s) inserted.

Statement **18**

Edit

```
INSERT INTO company VALUES('C4','Company 4','PUNJAB','9414224524')
```

1 row(s) inserted.

Statement **19**

Edit

```
INSERT INTO company VALUES('C5','Company 5','KOLKATA','9987663816')
```

1 row(s) inserted.

Statement **20**

Edit

```
INSERT INTO company VALUES('C6','Company 6','DELHI','9619989159')
```

1 row(s) inserted.

Statement **21**

Edit

```
INSERT INTO company VALUES('C7','Company 7','PUNJAB','9414224524')
```

1 row(s) inserted.

Statement **22**

Edit

```
INSERT INTO company VALUES('C8','Company 8','KOLKATA','9987663816')
```

1 row(s) inserted.

Statement **23**

Edit

```
INSERT INTO consumer VALUES ('P1','Parth Namdev','MUMBAI','8898461028')
```

1 row(s) inserted.

Statement **24**

Edit

```
INSERT INTO consumer VALUES ('P2','Kavya Nair','DELHI','9136339446')
```

1 row(s) inserted.

Statement **25**

Edit

```
INSERT INTO consumer VALUES ('P3','Raj Mehta','HYDERABAD','9619989159')
```

1 row(s) inserted.

Statement **26**

Edit

```
INSERT INTO consumer VALUES ('P4','Rita Shah','KASHMIR','9414252425')
```

1 row(s) inserted.

Statement **27**

Edit

```
INSERT INTO consumer VALUES ('P5','Tushar Nankani','JAIPUR','9987453616')
```

1 row(s) inserted.

Statement **28**

Edit

```
INSERT INTO consumer VALUES ('P6','Rishabh Jain','MUMBAI','8898461028')
```

1 row(s) inserted.

Statement **29**

Edit

```
INSERT INTO consumer VALUES ('P7','Tej Mandani','DELHI','9136339446')
```

1 row(s) inserted.

Statement **30**

Edit

```
INSERT INTO consumer VALUES ('P8','Om Shidhaye','HYDERABAD','9619989159')
```

1 row(s) inserted.

Statement **31**

Edit

```
INSERT INTO t_time VALUES (DATE '2014-01-31',1,1,1,2014)
```

1 row(s) inserted.

Statement **32**

Edit

```
INSERT INTO t_time VALUES (DATE '2015-01-31',1,1,1,2015)
```

1 row(s) inserted.

Statement **33**

Edit

```
INSERT INTO t_time VALUES (DATE '2016-01-31',1,1,1,2016)
```

1 row(s) inserted.

Statement **34**

Edit

```
INSERT INTO t_time VALUES (DATE '2017-01-31',1,1,1,2017)
```

1 row(s) inserted.

Statement **35**

Edit

```
INSERT INTO t_time VALUES (DATE '2018-01-31',1,1,1,2018)
```

1 row(s) inserted.

Statement **36**

Edit

```
INSERT INTO t_time VALUES (DATE '2019-01-31',1,1,1,2019)
```

1 row(s) inserted.

Statement **37**

Edit

```
INSERT INTO t_time VALUES (DATE '2020-04-30',4,4,4,2020)
```

1 row(s) inserted.

Statement **38**

Edit

```
INSERT INTO t_time VALUES (DATE '2021-05-31',5,5,5,2021)
```

1 row(s) inserted.

Statement **39**

Edit

```
INSERT INTO category VALUES ('M01AB',4.02,33.04,159.09)
```

1 row(s) inserted.

Statement **40**

Edit

```
INSERT INTO category VALUES ('M01AE',3.56,25.17,112.8)
```

1 row(s) inserted.

Statement **41**

Edit

```
INSERT INTO category VALUES ('N02BA',2.06,23.05,124.8)
```

1 row(s) inserted.

Statement **42**

Edit

```
INSERT INTO category VALUES ('N02BE',34.5,45.6,178.38)
```

1 row(s) inserted.

Statement **43**

Edit

```
INSERT INTO category VALUES ('N05B',7.03,51.06,270.06)
```

1 row(s) inserted.

Statement **44**

Edit

```
INSERT INTO category VALUES ('N05C',2.06,23.05,124.8)
```

1 row(s) inserted.

Statement **45**

Edit

```
INSERT INTO category VALUES ('R03',34.5,45.6,178.38)
```

1 row(s) inserted.

Statement **46**

Edit

```
INSERT INTO category VALUES ('R06',7.03,51.06,270.06)
```

1 row(s) inserted.

Statement **47**

Edit

```
INSERT INTO fact VALUES ('1234','C1','P1','M01AB',DATE '2014-01-31',200,300)
```

1 row(s) inserted.

Statement **48**

Edit

```
INSERT INTO fact VALUES ('1234','C2','P2','M01AB',DATE '2014-01-31',400,100)
```

1 row(s) inserted.

Statement **49**

Edit

```
INSERT INTO fact VALUES ('1234','C3','P3','M01AB',DATE '2014-01-31',250,380)
```

1 row(s) inserted.

Statement **50**

Edit

```
INSERT INTO fact VALUES ('1112','C8','P8','M01AE',DATE '2015-01-31',200,300)
```

1 row(s) inserted.

Statement **51**

Edit

```
INSERT INTO fact VALUES ('5678','C5','P5','M01AE',DATE '2017-01-31',550,280)
```

1 row(s) inserted.

Statement **52**

Edit

```
INSERT INTO fact VALUES ('1112','C6','P6','N05B',DATE '2021-05-31',340,100)
```

1 row(s) inserted.

Statement **53**

Edit

```
INSERT INTO fact VALUES ('5678','C7','P7','N05B',DATE '2016-01-31',220,350)
```

1 row(s) inserted.

Statement **54**

Edit

```
INSERT INTO fact VALUES ('5678','C4','P4','M01AB',DATE '2014-01-31',110,600)
```

1 row(s) inserted.

Statement **55**

Edit

```
INSERT INTO fact VALUES ('1234','C1','P2','M01AB',DATE '2014-01-31',200,300)
```

1 row(s) inserted.

Statement **56**

Edit

```
INSERT INTO fact VALUES ('1234','C2','P3','M01AB',DATE '2014-01-31',400,100)
```

1 row(s) inserted.

Statement **57**

Edit

```
INSERT INTO fact VALUES ('1234','C3','P4','M01AB',DATE '2014-01-31',250,380)
```

1 row(s) inserted.

Statement **58**

Edit

```
INSERT INTO fact VALUES ('1112','C8','P1','M01AE',DATE '2015-01-31',200,300)
```

1 row(s) inserted.

Statement **59**

Edit

```
INSERT INTO fact VALUES ('5678','C5','P6','M01AE',DATE '2017-01-31',550,280)
```

1 row(s) inserted.

Statement **60**

Edit

```
INSERT INTO fact VALUES ('1112','C6','P7','N05B',DATE '2021-05-31',340,100)
```

1 row(s) inserted.

Statement **61**

Edit

```
INSERT INTO fact VALUES ('5678','C7','P8','N05B',DATE '2016-01-31',220,350)
```

1 row(s) inserted.

Statement **62**

Edit

```
INSERT INTO fact VALUES ('5678','C4','P5','M01AB',DATE '2014-01-31',110,600)
```

1 row(s) inserted.

Statement **63**

Edit

```
INSERT INTO fact VALUES ('9101','C1','P3','M01AB',DATE '2014-01-31',200,300)
```

1 row(s) inserted.

Statement **64**

Edit

```
INSERT INTO fact VALUES ('1234','C2','P5','M01AB',DATE '2014-01-31',400,100)
```

1 row(s) inserted.

Statement **65**

Edit

```
INSERT INTO fact VALUES ('1234','C3','P7','M01AB',DATE '2014-01-31',250,380)
```

1 row(s) inserted.

Statement **66**

Edit

```
INSERT INTO fact VALUES ('1112','C8','P2','M01AE',DATE '2015-01-31',200,300)
```

1 row(s) inserted.

Statement **67**

Edit

```
INSERT INTO fact VALUES ('5678','C5','P7','M01AE',DATE '2017-01-31',550,280)
```

1 row(s) inserted.

Statement **68**

Edit

```
INSERT INTO fact VALUES ('1112','C6','P8','N05B',DATE '2021-05-31',340,100)
```

1 row(s) inserted.

Statement **69**

Edit

```
INSERT INTO fact VALUES ('9101','C7','P1','M01AE',DATE '2016-01-31',220,350)
```

1 row(s) inserted.

Statement **70**

Edit

```
INSERT INTO fact VALUES ('5678','C4','P6','M01AB',DATE '2014-01-31',110,600)
```

1 row(s) inserted.

Statement **71**

Edit

```
INSERT INTO fact VALUES ('9101','C5','P5','N05B',DATE '2017-01-31',150,450)
```

1 row(s) inserted.

Statement **72**

Edit

```
INSERT INTO fact VALUES ('1214','C1','P8','N05C',DATE '2019-01-31',200,300)
```

1 row(s) inserted.

Statement **73**

Edit

```
INSERT INTO fact VALUES ('1214','C2','P5','N02BA',DATE '2020-04-30',400,100)
```

1 row(s) inserted.

Statement **74**

Edit

```
INSERT INTO fact VALUES ('1234','C3','P3','N02BA',DATE '2014-01-31',250,380)
```

1 row(s) inserted.

Statement **75**

Edit

```
INSERT INTO fact VALUES ('1112','C8','P8','N02BA',DATE '2015-01-31',200,300)
```

1 row(s) inserted.

Statement **76**

Edit

```
INSERT INTO fact VALUES ('5678','C5','P5','R03',DATE '2017-01-31',550,280)
```

1 row(s) inserted.

Statement **77**

Edit

```
INSERT INTO fact VALUES ('1112','C6','P6','R03',DATE '2021-05-31',340,100)
```

1 row(s) inserted.

Statement **78**

Edit

```
INSERT INTO fact VALUES ('5678','C7','P7','R06',DATE '2016-01-31',220,350)
```

1 row(s) inserted.

Statement **79**

Edit

```
INSERT INTO fact VALUES ('5678','C4','P4','M01AE',DATE '2014-01-31',110,600)
```

1 row(s) inserted.

Statement **80**

Edit

```
INSERT INTO fact VALUES ('1234','C1','P2','N02BA',DATE '2014-01-31',200,300)
```

1 row(s) inserted.

Statement **81**

Edit

```
INSERT INTO fact VALUES ('1234','C2','P3','M01AE',DATE '2014-01-31',400,100)
```

1 row(s) inserted.

Statement **82**

Edit

```
INSERT INTO fact VALUES ('7864','C3','P4','R06',DATE '2014-01-31',250,380)
```

1 row(s) inserted.

Statement **83**

Edit

```
INSERT INTO fact VALUES ('0657','C8','P1','R06',DATE '2015-01-31',200,300)
```

1 row(s) inserted.

Statement **84**

Edit

```
INSERT INTO fact VALUES ('7864','C5','P6','R03',DATE '2017-01-31',550,280)
```

1 row(s) inserted.

Statement **85**

Edit

```
INSERT INTO fact VALUES ('0657','C6','P7','N05C',DATE '2021-05-31',340,100)
```

1 row(s) inserted.

Statement **86**

Edit

```
INSERT INTO fact VALUES ('9101','C7','P8','N05B',DATE '2021-05-31',220,350)
```

1 row(s) inserted.

Statement **87**

Edit

```
INSERT INTO fact VALUES ('1214','C4','P5','N02BA',DATE '2017-01-31',110,600)
```

1 row(s) inserted.

Statement **88**

Edit

```
INSERT INTO fact VALUES ('7864','C1','P3','M01AE',DATE '2016-01-31',200,300)
```

1 row(s) inserted.

Statement **89**

Edit

```
INSERT INTO fact VALUES ('7864','C2','P5','M01AB',DATE '2015-01-31',400,100)
```

1 row(s) inserted.

Statement **90**

Edit

```
INSERT INTO fact VALUES ('7864','C3','P7','N05C',DATE '2018-01-31',250,380)
```

1 row(s) inserted.

Statement **91**

Edit

```
INSERT INTO fact VALUES ('0657','C8','P2','N05C',DATE '2019-01-31',200,300)
```

1 row(s) inserted.

Statement **92**

Edit

```
INSERT INTO fact VALUES ('5678','C5','P7','M01AE',DATE '2019-01-31',550,280)
```

1 row(s) inserted.

Statement **93**

Edit

```
INSERT INTO fact VALUES ('0657','C6','P8','N05B',DATE '2020-04-30',340,100)
```

1 row(s) inserted.

Statement **94**

Edit

```
INSERT INTO fact VALUES ('1214','C7','P1','N02BE',DATE '2021-05-31',220,350)
```

1 row(s) inserted.

Statement **95**

Edit

```
INSERT INTO fact VALUES ('1214','C4','P6','M01AE',DATE '2021-05-31',110,600)
```

1 row(s) inserted.

Statement **96**

Edit

```
INSERT INTO fact VALUES ('1214','C5','P5','N05C',DATE '2019-01-31',150,450)
```

1 row(s) inserted.

Statement **97**

Edit

```
INSERT INTO fact VALUES ('1234','C1','P6','M01AB',DATE '2014-01-31',200,300)
```

1 row(s) inserted.

Statement **98**

Edit

```
INSERT INTO fact VALUES ('1234','C2','P7','M01AB',DATE '2014-01-31',400,100)
```

1 row(s) inserted.

Statement **99**

Edit

```
INSERT INTO fact VALUES ('1234','C3','P8','M01AB',DATE '2014-01-31',250,380)
```

1 row(s) inserted.

Statement **100**

Edit

```
INSERT INTO fact VALUES ('1112','C8','P5','M01AE',DATE '2015-01-31',200,300)
```

1 row(s) inserted.

Statement **101**

Edit

```
INSERT INTO fact VALUES ('5678','C5','P2','M01AE',DATE '2017-01-31',550,280)
```

1 row(s) inserted.

Statement **102**

Edit

```
INSERT INTO fact VALUES ('1112','C6','P3','N05B',DATE '2021-05-31',340,100)
```

1 row(s) inserted.

Statement **103**

Edit

```
INSERT INTO fact VALUES ('5678','C7','P4','N05B',DATE '2016-01-31',220,350)
```

1 row(s) inserted.

Statement **104**

Edit

```
INSERT INTO fact VALUES ('5678','C4','P1','M01AB',DATE '2014-01-31',110,600)
```

1 row(s) inserted.

Statement **105**

Edit

```
INSERT INTO fact VALUES ('1234','C1','P4','M01AB',DATE '2014-01-31',200,300)
```

1 row(s) inserted.

Statement **106**

Edit

```
INSERT INTO fact VALUES ('1214','C2','P5','M01AB',DATE '2014-01-31',400,100)
```

1 row(s) inserted.

Statement **107**

Edit

```
INSERT INTO fact VALUES ('1234','C3','P6','M01AB',DATE '2014-01-31',250,380)
```

1 row(s) inserted.

Statement **108**

Edit

```
INSERT INTO fact VALUES ('1112','C8','P3','M01AE',DATE '2015-01-31',200,300)
```

1 row(s) inserted.

Statement **109**

Edit

```
INSERT INTO fact VALUES ('5678','C5','P8','M01AE',DATE '2017-01-31',550,280)
```

1 row(s) inserted.

Statement **110**

Edit

```
INSERT INTO fact VALUES ('1112','C6','P1','N05B',DATE '2021-05-31',340,100)
```

1 row(s) inserted.

Statement **111**

Edit

```
INSERT INTO fact VALUES ('5678','C7','P2','N05B',DATE '2016-01-31',220,350)
```

1 row(s) inserted.

Statement **112**

Edit

```
INSERT INTO fact VALUES ('5678','C4','P7','M01AB',DATE '2014-01-31',110,600)
```

1 row(s) inserted.

Statement **113**

Edit

```
INSERT INTO fact VALUES ('9101','C1','P7','M01AB',DATE '2014-01-31',200,300)
```

1 row(s) inserted.

Statement **114**

Edit

```
INSERT INTO fact VALUES ('1234','C2','P8','M01AB',DATE '2014-01-31',400,100)
```

1 row(s) inserted.

Statement **115**

Edit

```
INSERT INTO fact VALUES ('1234','C3','P1','M01AB',DATE '2014-01-31',250,380)
```

1 row(s) inserted.

Statement **116**

Edit

```
INSERT INTO fact VALUES ('1112','C8','P6','M01AE',DATE '2015-01-31',200,300)
```

1 row(s) inserted.

Statement **117**

Edit

```
INSERT INTO fact VALUES ('5678','C5','P3','M01AE',DATE '2017-01-31',550,280)
```

1 row(s) inserted.

Statement **118**

Edit

```
INSERT INTO fact VALUES ('1112','C6','P4','N05B',DATE '2021-05-31',340,100)
```

1 row(s) inserted.

Statement **119**

Edit

```
INSERT INTO fact VALUES ('9101','C7','P5','M01AE',DATE '2016-01-31',220,350)
```

1 row(s) inserted.

Statement **120**

Edit

```
INSERT INTO fact VALUES ('5678','C4','P2','M01AB',DATE '2014-01-31',110,600)
```

1 row(s) inserted.

Statement **121**

Edit

```
INSERT INTO fact VALUES ('9101','C5','P3','N05B',DATE '2017-01-31',150,450)
```

1 row(s) inserted.

Statement **122**

Edit

```
INSERT INTO fact VALUES ('1214','C1','P2','N05C',DATE '2019-01-31',200,300)
```

1 row(s) inserted.

Statement **123**

Edit

```
INSERT INTO fact VALUES ('1214','C2','P8','N02BA',DATE '2020-04-30',400,100)
```

1 row(s) inserted.

Statement **124**

Edit

```
INSERT INTO fact VALUES ('1234','C3','P6','N02BA',DATE '2014-01-31',250,380)
```

1 row(s) inserted.

Statement **125**

Edit

```
INSERT INTO fact VALUES ('1112','C8','P2','N02BA',DATE '2015-01-31',200,300)
```

1 row(s) inserted.

Statement **126**

Edit

```
INSERT INTO fact VALUES ('5678','C5','P3','R03',DATE '2017-01-31',550,280)
```

1 row(s) inserted.

Statement **127**

Edit

```
INSERT INTO fact VALUES ('1112','C6','P4','R03',DATE '2021-05-31',340,100)
```

1 row(s) inserted.

Statement **128**

Edit

```
select * from fact
```

DRUG_ID_SK	COMPANY_ID_SK	CONSUMER_ID_SK	CATEGORY_ID_SK	DATE_SK	UNITPRICE	QUANTITY
1234	C1	P1	M01AB	31-JAN-14	200	300
1234	C2	P2	M01AB	31-JAN-14	400	100
1234	C3	P3	M01AB	31-JAN-14	250	380

Statement **129**

Edit

```
create materialized view drugview
as select drug_id_sk, count(*)
from   fact
group  by drug_id_sk
```

Statement processed.

Statement **130**

Edit

```
select * from drugview
```

DRUG_ID_SK	COUNT(*)
0657	4
1112	16
1214	9
1234	19
5678	21
7864	5
9101	7

Download CSV

7 rows selected.

Statement **131**

Edit

```
create table branch as select * from drugview
```

Table created.

Statement **132**

Edit

```
SELECT drug_id_sk, consumer_id_sk, SUM(Quantity)
FROM fact
GROUP BY ROLLUP (drug_id_sk, consumer_id_sk)
```

DRUG_ID_SK	CONSUMER_ID_SK	SUM(QUANTITY)
0657	P1	300
0657	P2	300
0657	P7	100
0657	P8	100
0657	-	800
1112	P1	400
1112	P2	600
1112	P3	400
1112	P4	200
1112	P5	300
1112	P6	500
1112	P7	100
1112	P8	700
1112	-	3200
1214	P1	350
1214	P2	300
1214	P5	1250
1214	P6	600
1214	P8	400

Statement 133

Edit

```
select count(*) from fact group by UnitPrice
```

COUNT(*)
10
20
9
10
10
9
10
3

Download CSV

8 rows selected.

Statement **134**

Edit

```
select count(*) from fact where substr(drug_id_sk,1,8)='1234'
```

COUNT(*)
19

Download CSV

Statement **135**

Edit

```
select count(distinct drug_id_sk) as drugid from branch
```

DRUGID
7

Statement **135**

Edit

```
select count(distinct drug_id_sk) as drugid from branch
```

DRUGID
7

Download CSV

Statement **136**

Edit

```
select count(*) as count
from fact,drug
where (drug.drug_id=fact.drug_id_sk) and substr(drug_id_sk,1,8)='1112'
```

COUNT
16

Download CSV

Experiment No 4

Date: 10/08/2021

Aim: Implementation of Bayesian algorithm.

Theory:

Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm. Naive Bayes classifiers have high accuracy and speed on large datasets.

Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features. For example, a loan applicant is desirable or not depending on his/her income, previous loan and transaction history, age, and location. Even if these features are interdependent, these features are still considered independently. This assumption simplifies computation, and that's why it is considered as naive. This assumption is called class conditional independence.

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- P(h): the probability of hypothesis h being true (regardless of the data). This is known as the prior probability of h.
- P(D): the probability of the data (regardless of the hypothesis). This is known as the prior probability.
- P(h|D): the probability of hypothesis h given the data D. This is known as posterior probability.

- P(D|h): the probability of data d given that hypothesis h was true. This is known as posterior probability.

How does Naive Bayes classifier work?

Let's understand the working of Naive Bayes through an example. Given an example of weather conditions and playing sports. You need to calculate the probability of playing sports. Now, you need to classify whether players will play or not, based on the weather condition.

Approach (In case of a single feature)

Naive Bayes classifier calculates the probability of an event in the following steps:

- Step 1: Calculate the prior probability for given class labels
- Step 2: Find Likelihood probability with each attribute for each class
- Step 3: Put these value in Bayes Formula and calculate posterior probability.
- Step 4: See which class has a higher probability, given the input belongs to the higher probability class.

For simplifying prior and posterior probability calculation you can use the two tables frequency and likelihood tables. Both of these tables will help you to calculate the prior and posterior probability. The Frequency table contains the occurrence of labels for all features. There are two likelihood tables. Likelihood Table 1 is showing prior probabilities of labels and Likelihood Table 2 is showing the posterior probability.

Whether	Play
Sunny	No
Sunny	No
Overcast	Yes
Rainy	Yes
Rainy	Yes
Rainy	No
Overcast	Yes
Sunny	No
Sunny	Yes
Rainy	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table

Whether	No	Yes
Overcast		4
Sunny	2	3
Rainy	3	2
Total	5	9

Likelihood Table 1

Whether	No	Yes		
Overcast		4	=4/14	0.29
Sunny	2	3	=5/14	0.36
Rainy	3	2	=5/14	0.36
Total	5	9		
	=5/14	=9/14		
	0.36	0.64		

Likelihood Table 2

Whether	No	Yes	Posterior Probability for No	Posterior Probability for Yes
Overcast		4	0/5=0	4/9=0.44
Sunny	2	3	2/5=0.4	3/9=0.33
Rainy	3	2	3/5=0.6	2/9=0.22
Total	5	9		

Now suppose you want to calculate the probability of playing when the weather is overcast.

Probability of playing:

P(Yes | Overcast) = P(Overcast | Yes) P(Yes) / P (Overcast) .....(1)

1. Calculate Prior Probabilities:
   P(Overcast) = 4/14 = 0.29
   P(Yes)= 9/14 = 0.64
1. Calculate Posterior Probabilities:
   P(Overcast |Yes) = 4/9 = 0.44
1. Put Prior and Posterior probabilities in equation (1)
   P (Yes | Overcast) = 0.44 * 0.64 / 0.29 = 0.98(Higher)

Similarly, you can calculate the probability of not playing

The probability of a 'Yes' class is higher. So you can determine here if the weather is overcast than players will play the sport.

Code and Output:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_csv("sales_data.csv")
data.head(10)
```

	M01AB	M01AE	N02BA	Target_reached
0	17.99	10.38	0.11840	0
1	20.57	17.77	0.08474	0
2	19.69	21.25	0.10960	0
3	11.42	20.38	0.14250	0
4	20.29	14.34	0.10030	0
5	12.45	15.70	0.12780	0
6	18.25	19.98	0.09463	0
7	13.71	20.83	0.11890	0
8	13.00	21.82	0.12730	0
9	12.46	24.04	0.11860	0

```
data["cat_M01AB"] = pd.cut(data["M01AB"].values, bins = 3, labels = [0,1,2])
data["cat_M01AE"] = pd.cut(data["M01AE"].values, bins = 3, labels = [0,1,2])
data["cat_N02BA"] = pd.cut(data["N02BA"].values, bins = 3, labels = [0,1,2])

data = data.drop(columns=["M01AB", "M01AE", "N02BA"])
data = data[["cat_M01AB",    "cat_M01AE", "cat_N02BA", "Target_reached"]]
data.head(10)
```

	cat_M01AB	cat_M01AE	cat_N02BA	Target_reached
0	1	0	1	0
1	2	1	0	0
2	2	1	1	0
3	0	1	2	0
4	2	0	1	0
5	0	0	2	0
6	1	1	0	0
7	0	1	1	0
8	0	1	2	0
9	0	2	1	0

```
X = data.drop(['Target_reached'],axis=1)
y = data['Target_reached']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=1)

prior = {}
for val in y_train.unique():
 val_count = len(y_train[y_train == val])
 prior[val] = (val_count)/(len(y_train))
print('Prior Probabilities :\n')
print(prior)

likelihood = {}
for col in X_train.columns:
  for X_val in X_train[col].unique():
    likelihood[X_val] = {}
    for y_val in y_train.unique():
      count = 0
      for row in zip(X_train[col],y_train):
        if row[0] == X_val and row[1] == y_val :
          count += 1
      y_count = len(y_train[y_train == y_val])
      likelihood[X_val][y_val] = (count)/(y_count)
```

```
print('Likelihood Probability :\n')
print(likelihood)

y_pred = []
for i in X_test.index:
  max_posterior = 0
  for y_val in y.unique():
    posterior = prior[y_val]
    for col in X_test.columns:
      X_val = X_test[col][i]
      posterior *= likelihood[X_val][y_val]
    if posterior >= max_posterior:
      max_posterior = posterior
      pred = y_val
  y_pred.append(pred)
print('Predicted Values :\n')
print(y_pred)

data = {'Actual': y_test, 'Predicted': y_pred}
results = pd.DataFrame(data,columns=['Actual','Predicted'])
print('Results :\n')
print(results)
```

```
Results :

    Actual  Predicted
91       0          0
82       0          0
31       0          0
27       0          0
77       0          0
17       0          0
97       1          0
80       1          0
67       1          0
62       0          0
44       0          0
32       0          0
52       1          0
48       1          0
46       1          0
56       0          0
78       0          0
59       1          0
65       0          0
10       0          0
```

```
correct_pred = len(results[results['Actual'] == results['Predicted']])
total = len(y_test)
tp = len(np.where((results['Actual'] == 1) & (results['Predicted'] == 1))[0])
tn = len(np.where((results['Actual'] == 0) & (results['Predicted'] == 0))[0])
fp = len(np.where((results['Actual'] == 0) & (results['Predicted'] == 1))[0])
fn = len(np.where((results['Actual'] == 1) & (results['Predicted'] == 0))[0])
accuracy = (correct_pred)/(total)
print('Correct Predictions : ',correct_pred)
print('Total : ',total)
print('Accuracy : ',accuracy)
```

```
Correct Predictions :  13
Total :  20
Accuracy :  0.65
```

Experiment No 5

Date: 24/08/2021

Aim: Implementation of Data Discretization (any one) &Visualization (any one)

Theory:

A) DISCRETIZATION:

Data discretization refers to a method of converting a huge number of data values into smaller ones so that the evaluation and management of data become easy. In other words, data discretization is a method of converting attributes values of continuous data into a finite set of intervals with minimum data loss.

There are two forms of data discretization: first is supervised discretization, and the second is unsupervised discretization. Supervised discretization refers to a method in which the class data is used. Unsupervised discretization refers to a method depending upon the way which operation proceeds. It means it works on the top-down splitting strategy and bottom-up merging strategy.

SOME FAMOUS TECHNIQUES IN DATA DISCRETIZATION :

1. Histogram analysis
Histogram refers to a plot used to represent the underlying frequency distribution of a continuous data set. Histogram assists the data inspection for data distribution. For example, Outliers, skewness representation, normal distribution representation, etc.

2. Binning
Binning refers to a data smoothing technique that helps to group a huge number of continuous values into smaller values. For data discretization and the development of idea hierarchy, this technique can also be used.

3. Cluster Analysis
Cluster analysis is a form of data discretization. A clustering algorithm is executed by dividing the values of x numbers into clusters to isolate a computational feature of x.

4. Data discretization using decision tree analysis
Data discretization refers to a decision tree analysis in which a top-down slicing technique is used. It is done through a supervised procedure. In a numeric attribute discretization, first, you need to select the attribute that has the least entropy, and then you need to run it with the help of a recursive process. The recursive process divides it into various discretized disjoint intervals, from top to bottom, using the same splitting criterion.

5. Data discretization using correlation analysis
Discretizing data by linear regression technique, you can get the best neighboring interval, and then the large intervals are combined to develop a larger overlap to form the final 20 overlapping intervals. It is a supervised procedure.

B) DATA VISUALIZATION:

Data visualization is the graphical representation of information and data. By using visual elements like chats, graphs and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.
In the world of Big Data, data visualization tools and technologies are essential to analyze massive amounts of information and make data-driven decisions.

Types of data visualization charts

Now that we understand how data visualization can be used, let's apply the different types of data visualization to their uses. There are numerous tools available to help create data visualizations. Some are more manual and some are automated, but either way they should allow you to make any of the following types of visualizations.

1. Line chart
A line chart illustrates changes over time. The x-axis is usually a period of time, while the y-axis is quantity. So, this could illustrate a company's sales for the year broken down by month or how many units a factory produced each day for the past week.

2. Area chart
An area chart is an adaptation of a line chart where the area

under the line is filled in to emphasize its significance. The color fill for the area under each line should be somewhat transparent so that overlapping areas can be discerned.

3. Bar chart
A bar chart also illustrates changes over time. But if there is more than one variable, a bar chart can make it easier to compare the data for each variable at each moment in time. For example, a bar chart could compare the company's sales from this year to last year.

4. Histogram
A histogram looks like a bar chart, but measures frequency rather than trends over time. The x-axis of a histogram lists the "bins" or intervals of the variable, and the y-axis is frequency, so each bar represents the frequency of that bin. For example, you could measure the frequencies of each answer to a survey question. The bins would be the answer: "unsatisfactory," "neutral," and "satisfactory." This would tell you how many people gave each answer.

5. Scatter plot
Scatter plots are used to find correlations. Each point on a scatter plot means "when x = this, then y equals this." That way, if the points trend a certain way (upward to the left, downward to the right, etc.) there is a relationship between them. If the plot is truly scattered with no trend at all, then the variables do not affect each other at all.

6. Bubble chart
A bubble chart is an adaptation of a scatter plot, where each point is illustrated as a bubble whose area has meaning in in addition to its placement on the axes. A pain point associated with bubble charts is the limitations on sizes of bubbles due to the limited space within the axes. So, not all data will fit effectively in this type of visualization.

7. Pie chart
A pie chart is the best option for illustrating percentages, because it shows each element as part of a whole. So, if your data explains a breakdown in percentages, a pie chart will clearly present the pieces in the proper proportions.

8. Gauge
A gauge can be used to illustrate the distance between intervals. This can be presented as a round clock-like gauge or as a tube type gauge resembling a liquid thermometer. Multiple gauges can be shown next to each other to illustrate the difference between multiple intervals.

9. Map
Much of the data dealt with in businesses has a location element, which makes it easy to illustrate on a map. An example of a map visualization is mapping the number of purchases customers made in each state in the U.S. In this example, each state would be shaded in and states with less purchases would be a lighter shade, while states with more purchases would be darker shades. Location information can also be very valuable for business leadership to understand, making this an important data visualization to use.

10. Heat map
A heat map is basically a color-coded matrix. A formula is used to color each cell of the matrix and is shaded to represent the relative value or risk of that cell. Usually heat map colors range from green to red, with green being a better result and red being worse. This type of visualization is helpful because colors are quicker to interpret than numbers.

11. Frame diagram
Frame diagrams are basically tree maps which clearly show hierarchical relationship structure. A frame diagram consists of branches, which each have more branches connecting to them with each level of the diagram consisting of more and more branches

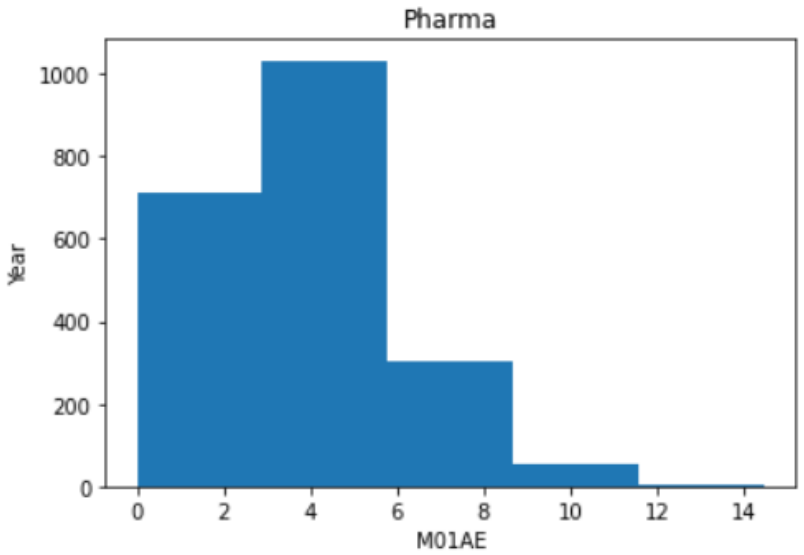
Code and Output:

+ Code + Text

```
import pandas as pd
df = pd.read_csv("salesdaily.csv")
print(df.head(3))  #Print first three observations
```

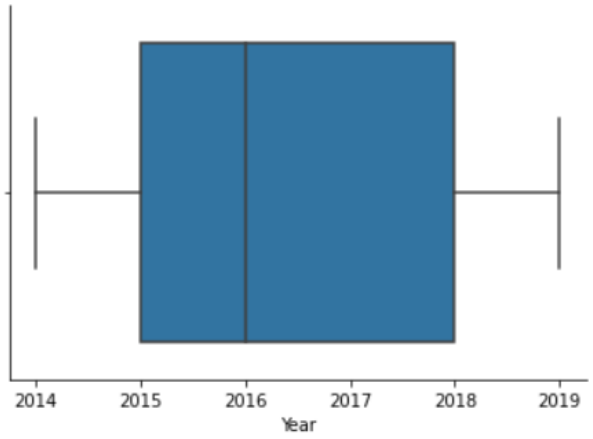
```
      datum  M01AB  M01AE  N02BA  N02BE  ...  R06  Year  Month  Hour  Weekday Name
0  1/2/2014    0.0   3.67    3.4  32.40  ...  2.0  2014      1   248      Thursday
1  1/3/2014    8.0   4.00    4.4  50.60  ...  4.0  2014      1   276        Friday
2  1/4/2014    2.0   1.00    6.5  61.85  ...  1.0  2014      1   276      Saturday

[3 rows x 13 columns]
```

```
[ ]  #histogram
     import matplotlib.pyplot as plt
     fig=plt.figure()
     ax = fig.add_subplot(1,1,1)
     ax.hist(df['M01AE'],bins = 5)
     plt.title('Pharma')
     plt.xlabel('M01AE')
     plt.ylabel('Year')
     plt.show()
```

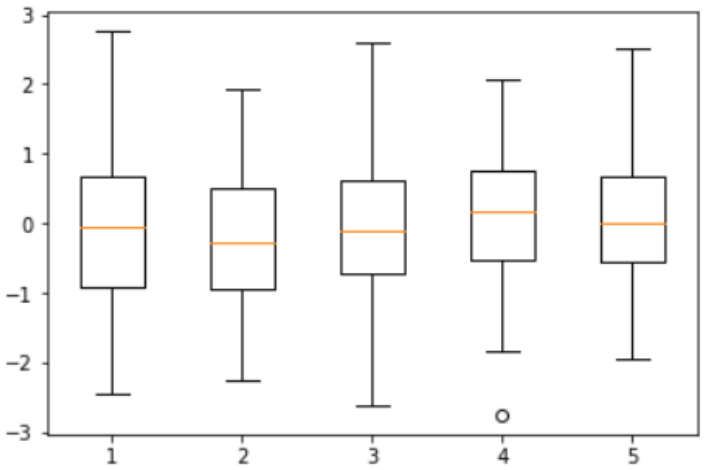


```
#boxplot
import seaborn as sns
sns.boxplot(df['Year'])
sns.despine()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarni
  FutureWarning
```

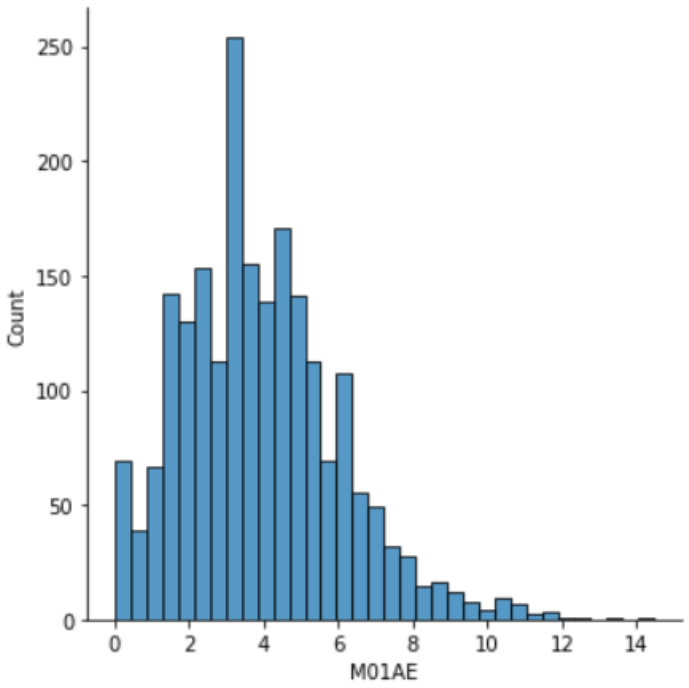



```
[ ]  import numpy as np
     import matplotlib.pyplot as plt

     data = np.random.randn(100, 5)

     plt.boxplot(data)
     plt.show()
```



```
#scatterplot
sns.set()
cols = ['datum', 'M01AE', 'M01AB', 'N02BA', 'N02BE', 'R06',
'Year']
sns.pairplot(df[cols], height = 2.5)
plt.show();
```

```
[ ]  #histogram
     import seaborn as sns
     import pandas as pd
     df = pd.read_csv("salesdaily.csv")
     sns.displot(df['M01AE']);
```

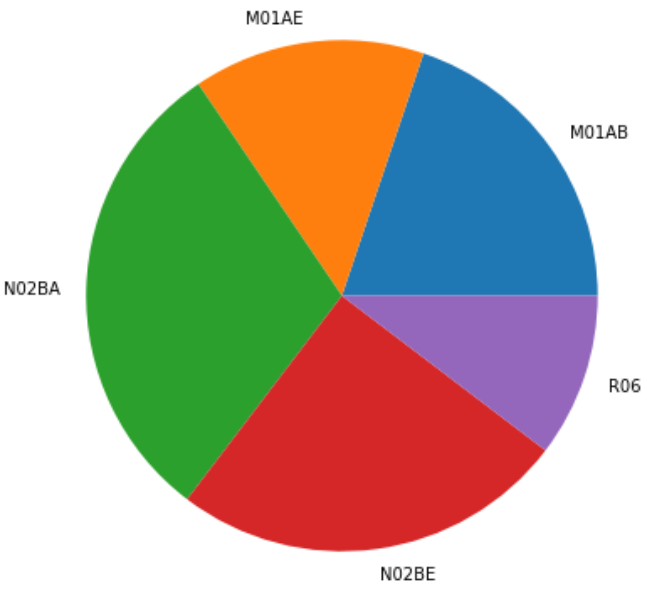


+ Code + Text

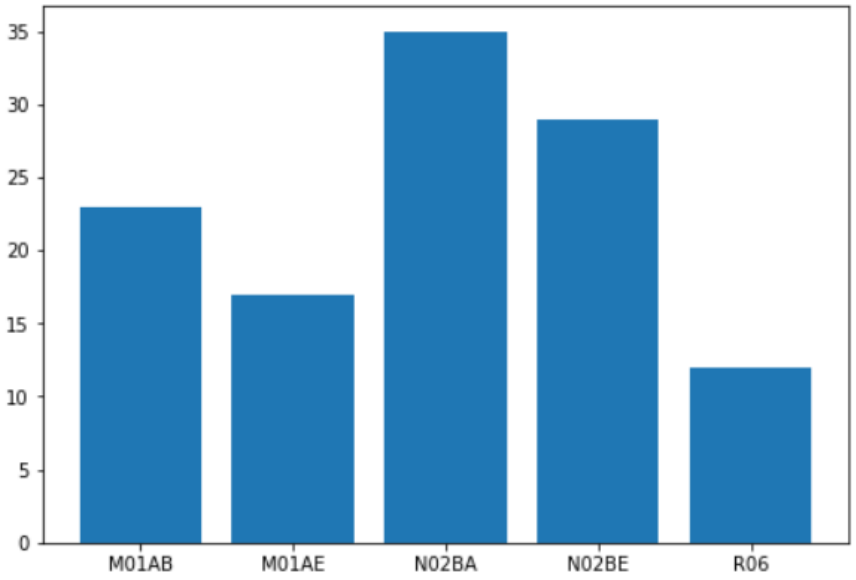
```
[ ]  #min, max, mean, median
     df['M01AE'].describe()
```

```
count    2106.000000
mean        3.895830
std         2.133337
min         0.000000
25%         2.340000
50%         3.670000
75%         5.138000
max        14.463000
Name: M01AE, dtype: float64
```

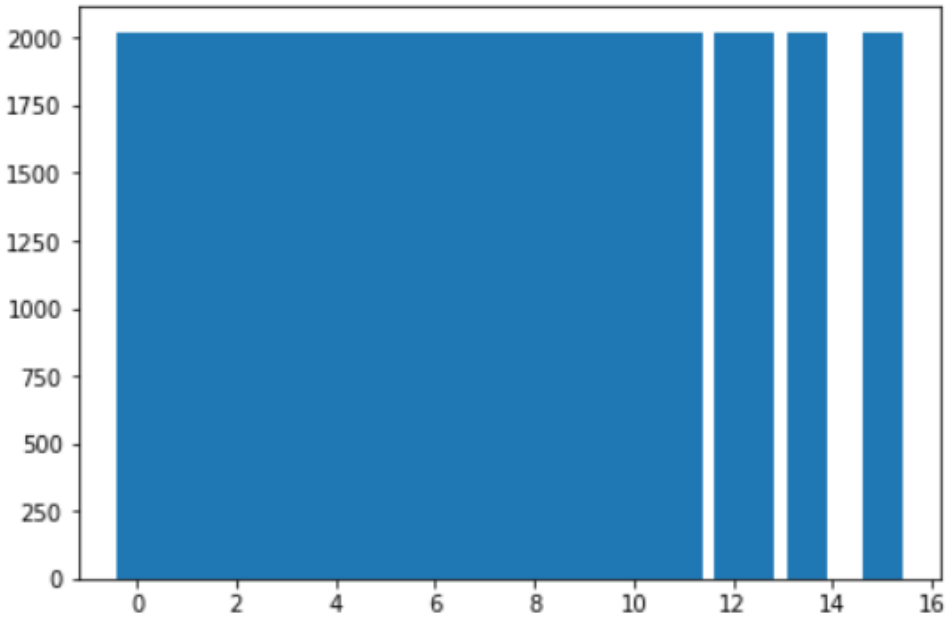
```
[ ]  from matplotlib import pyplot as plt
     import numpy as np
     pharma = ['M01AB', 'M01AE', 'N02BA',
             'N02BE', 'R06']
     data = [23, 17, 35, 29, 12]
     fig = plt.figure(figsize =(10, 7))
     plt.pie(data, labels = pharma)
     plt.show()
```



```
[ ]  import matplotlib.pyplot as plt
     fig = plt.figure()
     ax = fig.add_axes([0,0,1,1])
     pharma = ['M01AB', 'M01AE', 'N02BA',
             'N02BE', 'R06']
     data = [23, 17, 35, 29, 12]
     ax.bar(pharma,data)
     plt.show()
```



```
[ ]  import matplotlib.pyplot as plt
     fig = plt.figure()
     ax = fig.add_axes([0,0,0.9,0.9])
     data = pd.read_csv(r"salesdaily.csv")
     data.head()
     df = pd.DataFrame(data)
     ax.bar(df['R06'],df['Year'])
     plt.show()
```



Experiment No 6

Date: 07/09/2021

Aim: Perform data Pre-processing task and Demonstrate performing Classification,Clustering, Association algorithm on data sets using data mining tool (WEKA/R tool)

Theory:

The Weka GUI Chooser (class weka.gui.GUIChooser) provides a starting point for launching Weka‘s main GUI applications and supporting tools. If one prefersa MDI (—multiple document interfacell) appearance, then this is provided by an alternative launcher called —Mainll (class weka.gui.Main). The GUI Chooser consists of four buttons—one for each of the four majorWeka applications— and four menus.

The buttons can be used to start the following applications:
Explorer - An environment for exploring data with WEKA
a) Click on —explorer button to bring up the explorer window.
b) Make sure the —preprocess tab is highlighted.
c) Open a new file by clicking on —Open New file‖ and choosing a file with —.arff‖ extension from the —Data‖ directory.
d) Attributes appear in the window below.
e) Click on the attributes to see the visualization on the right.
f) Click —visualize all‖ to see them all.

Experimenter - An environment for performing experiments and conducting statistical tests between learning schemes.
a) Experimenter is for comparing results.
b) Under the —set up tab click —New.
c) Click on —Add New under —Data frame. Choose a couple of arff format files from —Data‖ directory one at a time.
d) Click on —Add New under —Algorithm frame. Choose several algorithms, one at a time by clicking —OK in the window and —Add New.
e) Under the —Run tab click —Start‖
f) Wait for WEKA to finish.
g) Under —Analyses tab click on —Experiment to see results.

Knowledge Flow - This environment supports essentially the same functions as the Explorer but with a drag-and-drop interface. One advantage is that it supports incremental learning.

SimpleCLI - Provides a simple command-line interface that allows direct execution of WEKA commands for operating systems that do not provide their own command line interface. Navigate the options available in the WEKA (ex. Select attributes panel, Preprocess panel, classify panel, Cluster

panel, Associate panel and Visualize panel)

When the Explorer is first started only the first tab is active; the others are greyed out. This is because it is necessary to open (and potentially pre-process) a data set before starting to explore the data.

The tabs are as follows:
1. Preprocess. Choose and modify the data being acted on.
2. Classify. Train and test learning schemes that classify or perform regression.
3. Cluster. Learn clusters for the data.
4. Associate. Learn association rules for the data.
5. Select attributes. Select the most relevant attributes in the data.
6. Visualize. View an interactive 2D plot of the data. Once the tabs are active, clicking on them flicks between different screens, on which the respective actions can be performed. The bottom area of the window (including the status box, the log button, and the Weka bird) stays visible regardless of which section you are in

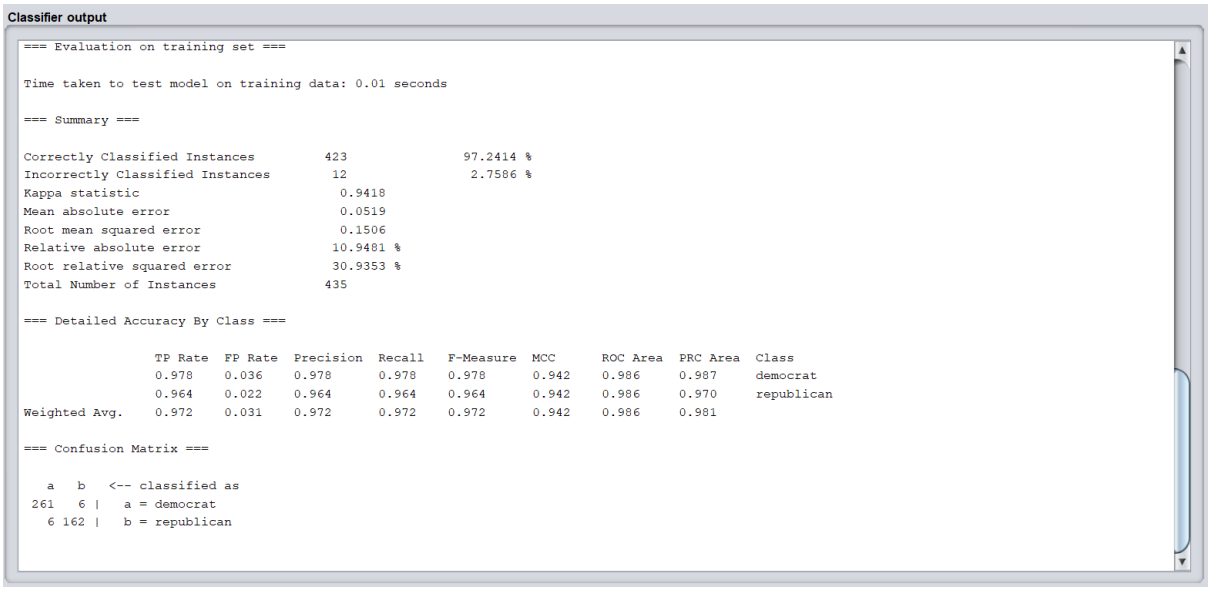
Loading Data:
The first four buttons at the top of the preprocess section enable you to load data into WEKA:
1. Open file.... Brings up a dialog box allowing you to browse for the datafile on the local file system.
2. Open URL.... Asks for a Uniform Resource Locator address for where the data is stored.
3. Open DB.....Reads data from a database. (Note that to make this work you might have to edit the file in weka/experiment/DatabaseUtils.props.)
4. Generate.... Enables you to generate artificial data from a variety ofDataGenerators.

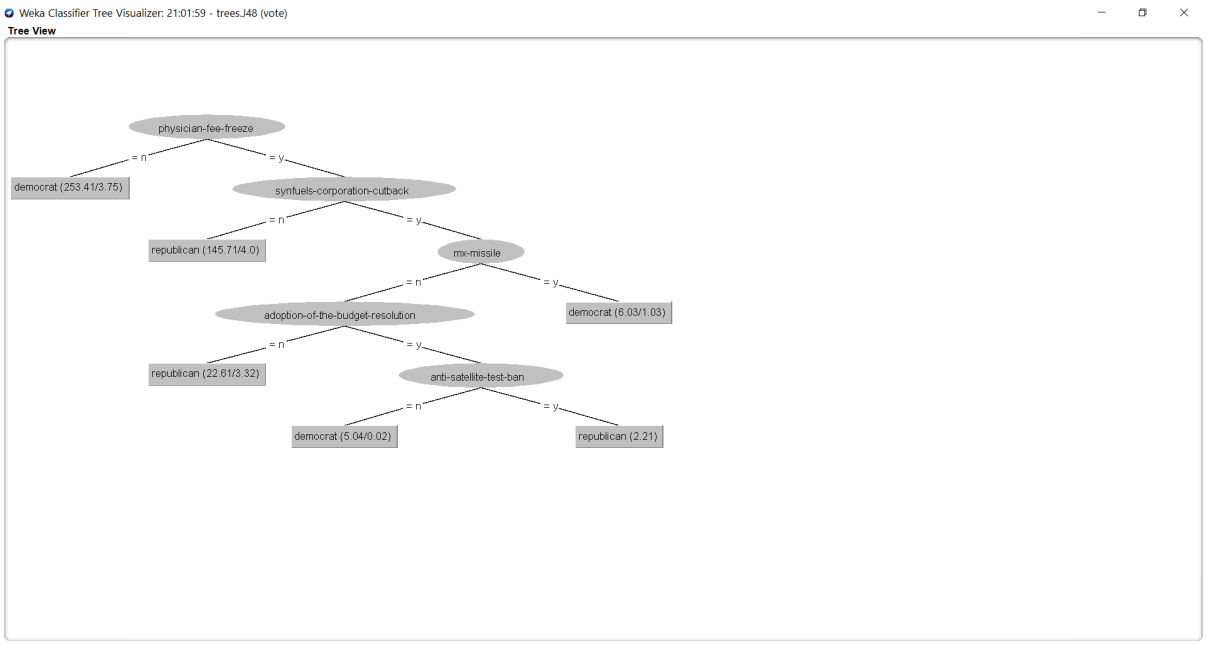
Using the Open file ...button you can read files in a variety of formats:
WEKA‘s ARFF format, CSV format, C4.5 format, or serialized Instances format. ARFF files typically have a .arff extension, CSV files a .csv extension,C4.5 files a .data and .names extension, and serialized Instances objects a .bsiextension.

CSV files a .csv extension,C4.5 files a .data and .names extension, and serialized Instances objects a .bsiextension.

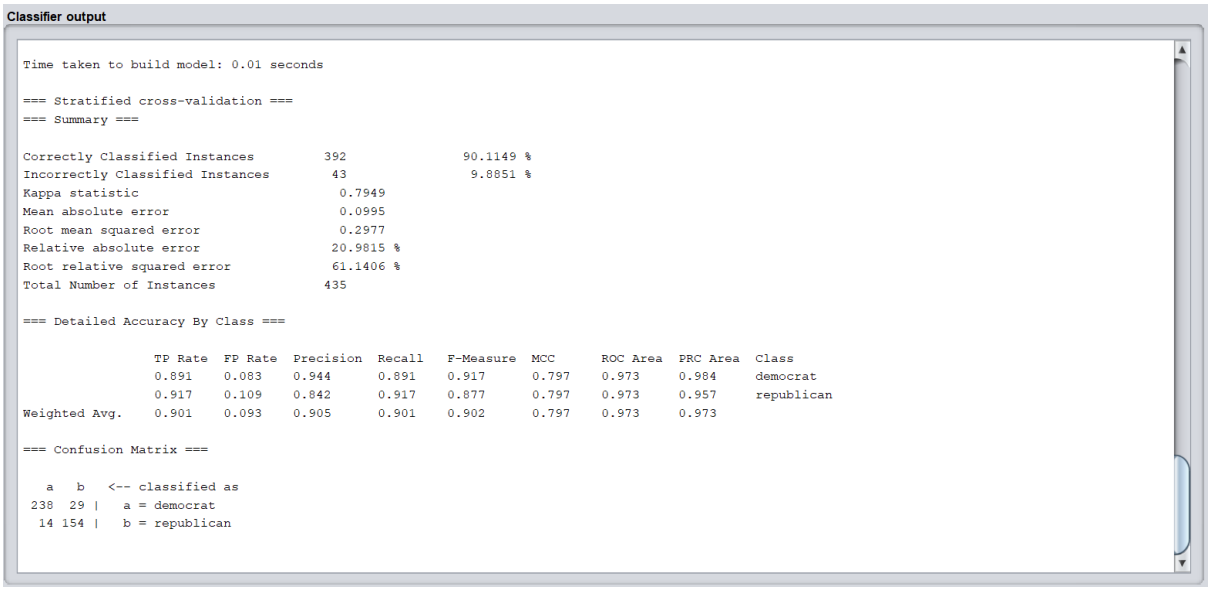
Output:

Classification: J48





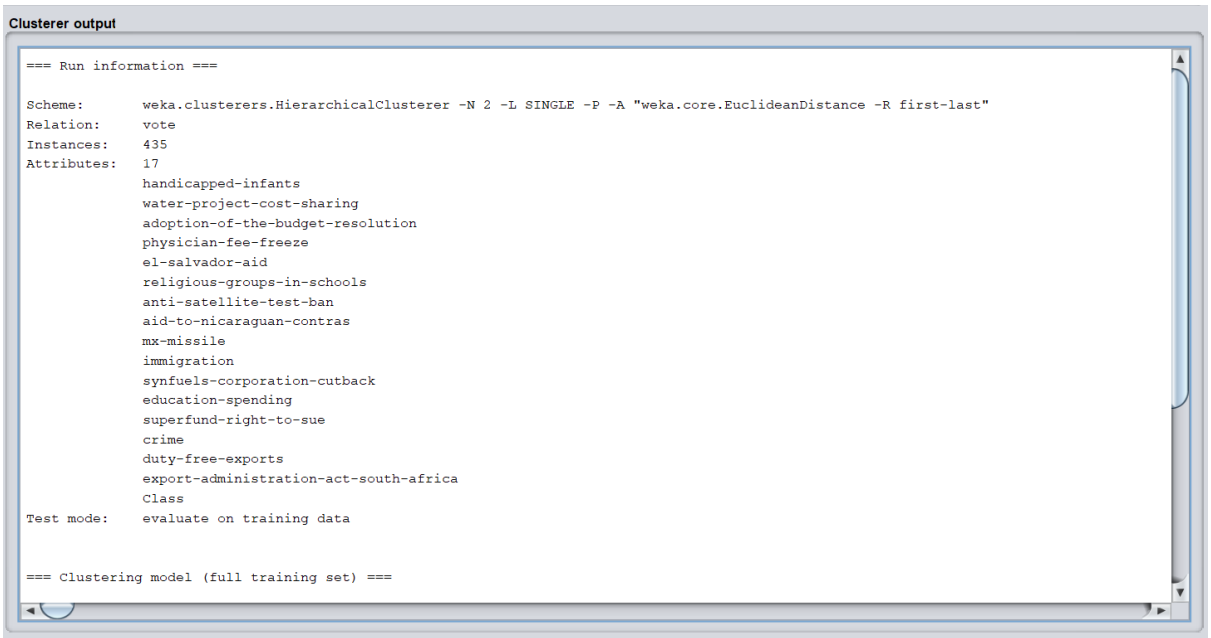
Classification: Naive-Bayes



Classifier output

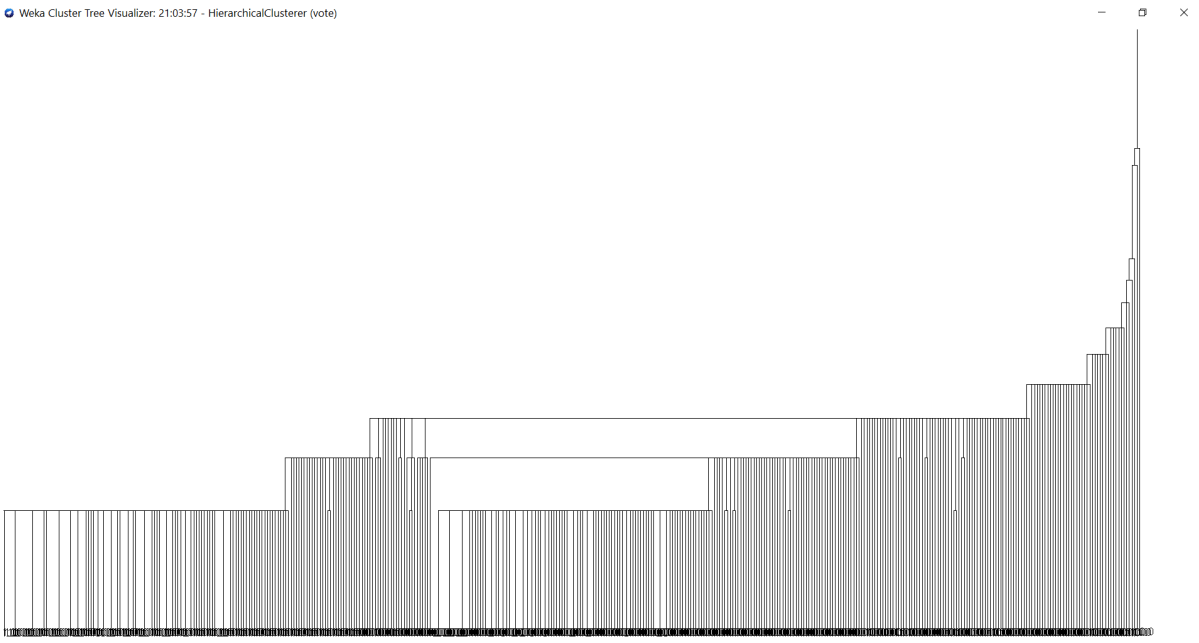
```
Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         392               90.1149 %
Incorrectly Classified Instances        43                9.8851 %
Kappa statistic                          0.7949
Mean absolute error                      0.0995
Root mean squared error                  0.2977
Relative absolute error                 20.9815 %
Root relative squared error             61.1406 %
Total Number of Instances              435

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               0.891     0.083     0.944      0.891    0.917      0.797    0.973     0.984     democrat
               0.917     0.109     0.842      0.917    0.877      0.797    0.973     0.957     republican
Weighted Avg.  0.901     0.093     0.905      0.901    0.902      0.797    0.973     0.973

=== Confusion Matrix ===

   a   b   <-- classified as
 238  29 |   a = democrat
  14 154 |   b = republican
```

Clustering: Hierarchical Clustering



Clusterer output

```
=== Run information ===

Scheme:       weka.clusterers.HierarchicalClusterer -N 2 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"
Relation:     vote
Instances:    435
Attributes:   17
              handicapped-infants
              water-project-cost-sharing
              adoption-of-the-budget-resolution
              physician-fee-freeze
              el-salvador-aid
              religious-groups-in-schools
              anti-satellite-test-ban
              aid-to-nicaraguan-contras
              mx-missile
              immigration
              synfuels-corporation-cutback
              education-spending
              superfund-right-to-sue
              crime
              duty-free-exports
              export-administration-act-south-africa
              Class
Test mode:    evaluate on training data


=== Clustering model (full training set) ===
```



Clustering: K-Means

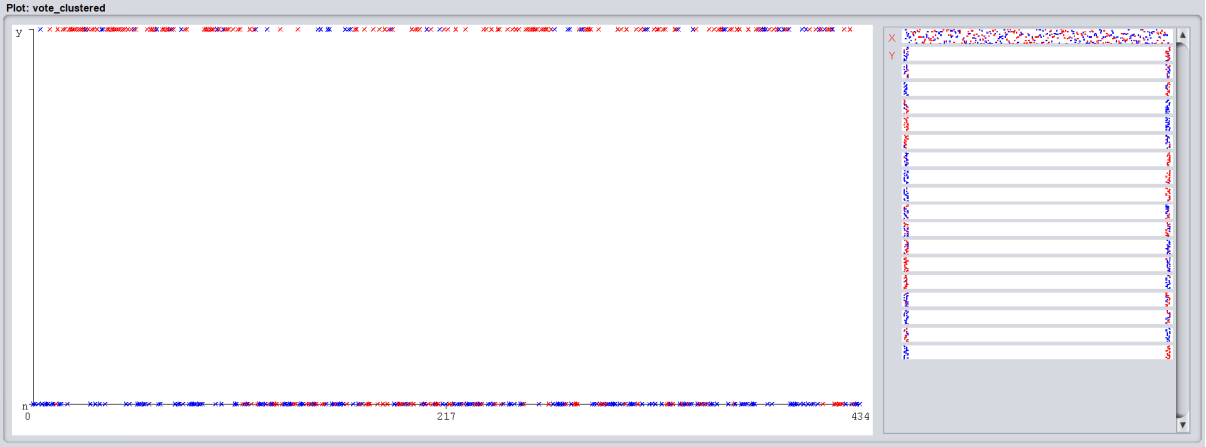
Clusterer output

```
kMeans
======

Number of iterations: 3
Within cluster sum of squared errors: 1510.0

Initial starting points (random):

Cluster 0: n,n,y,y,y,n,n,y,n,n,n,y,y,y,democrat
Cluster 1: n,n,y,n,y,n,y,y,y,n,n,n,n,y,n,y,democrat

Missing values globally replaced with mean/mode

Final cluster centroids:
                                                Cluster#
Attribute                          Full Data           0           1
                                     (435.0)     (214.0)     (221.0)
=================================================================================
handicapped-infants                        n           n           y
water-project-cost-sharing                 y           y           n
adoption-of-the-budget-resolution          y           n           y
physician-fee-freeze                       n           y           n
el-salvador-aid                            y           y           n
religious-groups-in-schools                y           y           n
anti-satellite-test-ban                    y           n           y
aid-to-nicaraquan-contras                  y           n           y
```

Clusterer output

```
physician-fee-freeze                              n          y          n
el-salvador-aid                                   y          y          n
religious-groups-in-schools                       y          y          n
anti-satellite-test-ban                           y          n          y
aid-to-nicaraguan-contras                         y          n          y
mx-missile                                        y          n          y
immigration                                       y          y          y
synfuels-corporation-cutback                      n          n          n
education-spending                                n          y          n
superfund-right-to-sue                            y          y          n
crime                                             y          y          n
duty-free-exports                                 n          n          y
export-administration-act-south-africa            y          y          y
Class                                      democrat republican   democrat



Time taken to build model (full training data) : 0.02 seconds

=== Model and evaluation on training set ===

Clustered Instances

0      214 ( 49%)
1      221 ( 51%)
```

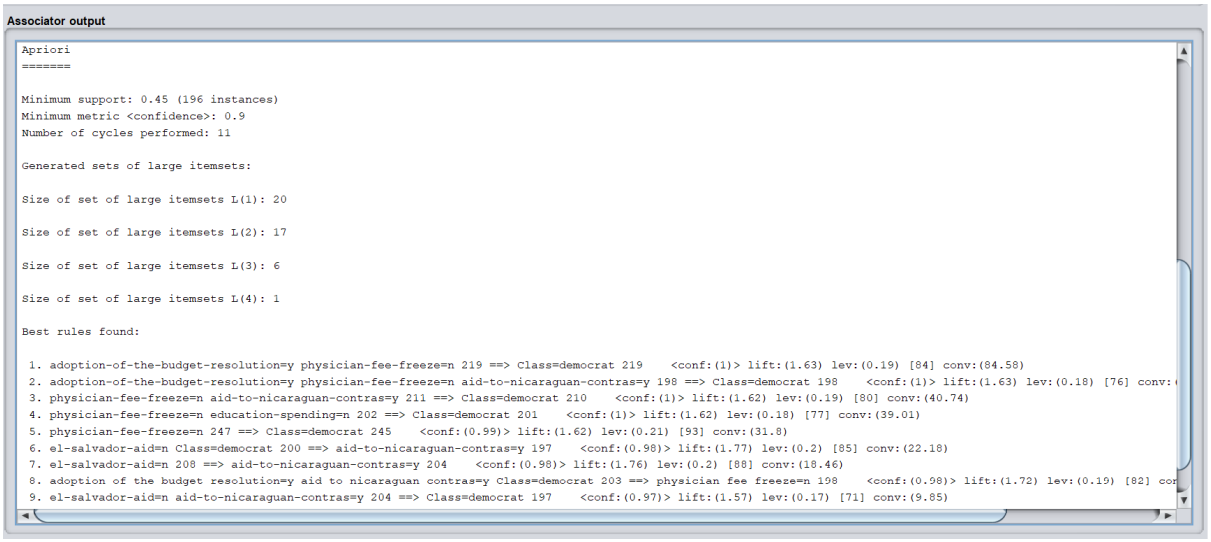
Plot: vote_clustered



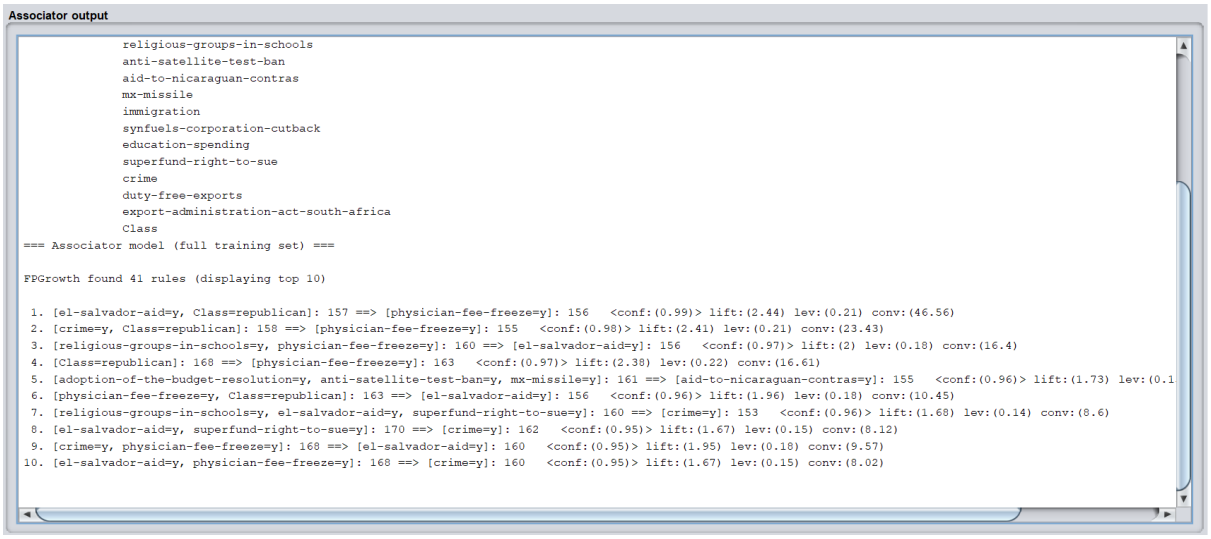
Class colour

cluster0 cluster1

Associate: Apriori



Associate: FP Growth



Experiment No 7

Date: 21/09/2021

Aim: Implementation of Clustering algorithm (K-means / K-medoids)

Theory:

Clustering is the task of grouping together a set of objects in a way that objects in the same cluster are more similar to each other than to objects in other clusters. Similarity is a metric that reflects the strength of relationship between two data objects. Clustering is mainly used for exploratory data mining. It has manifold usage in many fields such as machine learning, pattern recognition, image analysis, information retrieval, bio-informatics, data compression, and computer graphics.

K-Means falls under the category of centroid-based clustering. A centroid is a data point (imaginary or real) at the center of a cluster. In centroid-based clustering, clusters are represented by a central vector or a centroid. This centroid might not necessarily be a member of the dataset. Centroid-based clustering is an iterative algorithm in which the notion of similarity is derived by how close a data point is to the centroid of the cluster.

The inner workings of the K-Means clustering algorithm:

To do this, you will need a sample dataset (training set):

Objects	X	Y	Z
OB-1	1	4	1
OB-2	1	2	2
OB-3	1	4	2
OB-4	2	1	2
OB-5	1	1	1
OB-6	2	4	2
OB-7	1	1	2
OB-8	2	1	1

The sample dataset contains 8 objects with their X, Y and Z coordinates. Your task is to cluster these objects into two clusters (here you define the value of K (of K-Means) in essence to be 2).

So, the algorithm works by:

- Taking any two centroids or data points (as you took 2 as K hence the number of centroids also 2) in its account initially.
- After choosing the centroids, (say C1 and C2) the data points (coordinates here) are assigned to any of the Clusters (let's take centroids = clusters for the time being) depending upon the distance between them and the centroids.
- Assume that the algorithm chose OB-2 (1,2,2) and OB-6 (2,4,2) as centroids and cluster 1 and cluster 2 as well.
- For measuring the distances, you take the following distance measurement function (also termed as similarity measurement function):

$$d=|x2–x1|+|y2–y1|+|z2–z1|$$

$$d=|x2–x1|+|y2–y1|+|z2–z1|$$

This is also known as the **Taxicab distance** or **Manhattan distance**, where d is distance measurement between two objects, (x1,y1,z1) and (x2,y2,z2) are the X, Y and Z coordinates of any two objects taken for distance measurement.

Code and Output:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# For better readablity;
import matplotlib
matplotlib.rcParams['font.size'] = 16
matplotlib.rcParams['figure.figsize'] = (12, 6)
matplotlib.rcParams['figure.facecolor'] = '#00000000'

df = pd.read_csv('salesdaily.csv')
df
```

	datum	M01AB	M01AE	N02BA	N02BE	N05B	N05C	R03	R06	Year	Month	Hour	Weekday Name
0	1/2/2014	0.00	3.670	3.40	32.40	7.0	0.0	0.0	2.00	2014	1	248	Thursday
1	1/3/2014	8.00	4.000	4.40	50.60	16.0	0.0	20.0	4.00	2014	1	276	Friday
2	1/4/2014	2.00	1.000	6.50	61.85	10.0	0.0	9.0	1.00	2014	1	276	Saturday
3	1/5/2014	4.00	3.000	7.00	41.10	8.0	0.0	3.0	0.00	2014	1	276	Sunday
4	1/6/2014	5.00	1.000	4.50	21.70	16.0	2.0	6.0	2.00	2014	1	276	Monday
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2101	10/4/2019	7.34	5.683	2.25	22.45	13.0	0.0	1.0	1.00	2019	10	276	Friday
2102	10/5/2019	3.84	5.010	6.00	25.40	7.0	0.0	0.0	0.33	2019	10	276	Saturday
2103	10/6/2019	4.00	11.690	2.00	34.60	6.0	0.0	5.0	4.20	2019	10	276	Sunday
2104	10/7/2019	7.34	4.507	3.00	50.80	6.0	0.0	10.0	1.00	2019	10	276	Monday
2105	10/8/2019	0.33	1.730	0.50	44.30	20.0	2.0	2.0	0.00	2019	10	190	Tuesday

2106 rows × 13 columns

dataset = df[['M01AB', 'Month']]
dataset

[ ]

	M01AB	Month
0	0.00	1
1	8.00	1
2	2.00	1
3	4.00	1
4	5.00	1
...	...	...
2101	7.34	10
2102	3.84	10
2103	4.00	10
2104	7.34	10
2105	0.33	10

2106 rows × 2 columns

import random

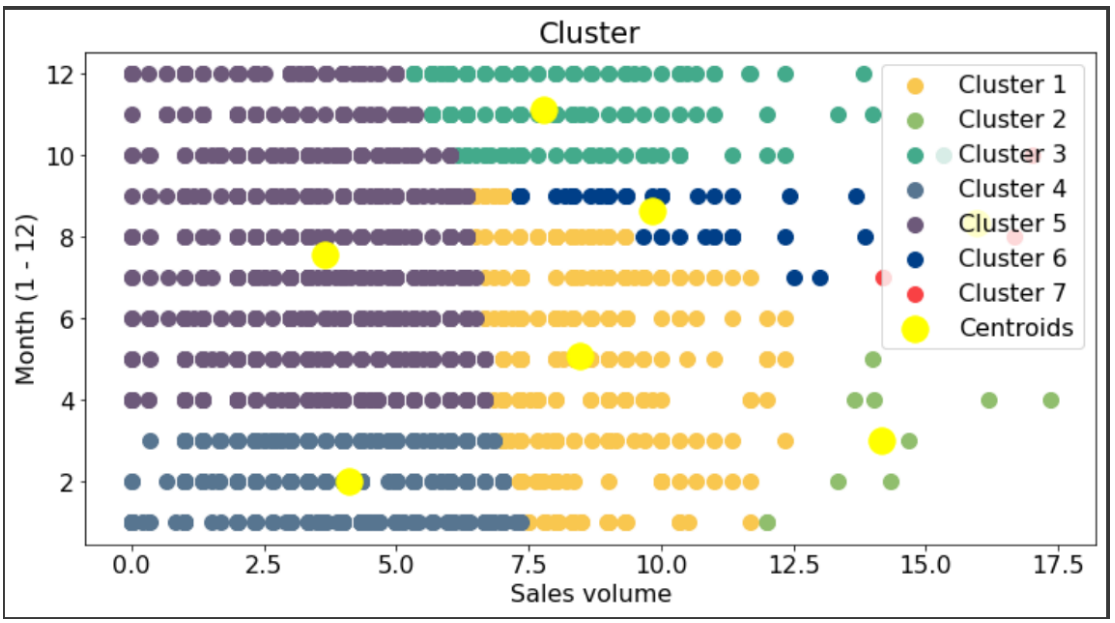
```
def init_centroids(k,dataset):
    centroids = []
    for i in range(0,k):
        point = []
        for col in dataset.columns:
            point.append(random.uniform(min(dataset[col]),max(dataset[col])))
        centroids.append(point)

    return centroids

import math

def calcdist(dataset,cluster):
    dist = 0
    for idx in range(len(dataset.columns)-1):
        dist += (dataset[dataset.columns[idx]]-cluster[idx])**2
    dist = dist**(1/2)
    return dist

def kmeans(k,dataset):
    centroids = init_centroids(k,dataset)
    dataset['Cluster'] = 0
    original = dataset['Cluster']
    while True:
        dist = pd.Series([math.inf] * len(dataset))
        for idx in range(len(centroids)):
            point = centroids[idx]
            dataset.loc[calcdist(dataset,point)<=dist,['Cluster']] = idx
            dist = pd.concat([dist, calcdist(dataset,point)], axis=1).min(axis=1)
        for idx in range(len(centroids)):
            centroids[idx] =
list(dataset[dataset['Cluster']==idx][dataset.columns[0:-1]].mean(axis=0))
        if dataset['Cluster'].eq(original, axis=0).all():
            return dataset,centroids
        else:
            original = dataset['Cluster']


ct = init_centroids(k,dataset)

k = int(input('Enter number of clusters : '))
result,centroids = kmeans(k,dataset)
result
```

```
Enter number of clusters : 7
```

	M01AB	Month	Cluster
0	0.00	1	3
1	8.00	1	0
2	2.00	1	3
3	4.00	1	3
4	5.00	1	3
...	...	...	...
2101	7.34	10	2
2102	3.84	10	4
2103	4.00	10	4
2104	7.34	10	2
2105	0.33	10	4

2106 rows × 3 columns

```
def getX(lst):
    return [item[0] for item in lst]

def getY(lst):
    return [item[1] for item in lst]

color = ['#F9C74F', '#90BE6D', '#43AA8B', '#577590',
'#6D597A','#003F88', '#F94144', '#F3722C', '#F8961E', '#FDC500']
for i in range(k):
    if(i < 2):
        plt.scatter(result[result['Cluster'] == i][result.columns[0]],
result[result['Cluster'] == i][result.columns[1]], s = 100, c = color[i], label =
f'Cluster {i + 1}')
    elif(k > i):
        plt.scatter(result[result['Cluster'] == i][result.columns[0]],
result[result['Cluster'] == i][result.columns[1]], s = 100, c = color[i], label =
f'Cluster {i + 1}')

plt.scatter(getX(centroids), getY(centroids), s = 300, c = 'yellow', label =
'Centroids')
plt.title('Cluster')
plt.xlabel('Sales volume')
plt.ylabel('Month (1 - 12)')
plt.legend()
plt.show()
```


Cluster
Month (1 - 12)
Sales volume
12
10
8
6
4
2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Centroids

Experiment No 8

Date: 28/09/2021

Aim: Implementation of anyone Hierarchical Clustering method

Theory:

Theory of Hierarchical Clustering

There are two types of hierarchical clustering: Agglomerative and Divisive. In the former, data points are clustered using a bottom-up approach starting with individual data points, while in the latter top-down approach is followed where all the data points are treated as one big cluster and the clustering process involves dividing the one big cluster into several small clusters.

Steps to Perform Hierarchical Clustering

Following are the steps involved in agglomerative clustering:

- At the start, treat each data point as one cluster. Therefore, the number of clusters at the start will be K, while K is an integer representing the number of data points.
- Form a cluster by joining the two closest data points resulting in K-1 clusters.
- Form more clusters by joining the two closest clusters resulting in K-2 clusters.
- Repeat the above three steps until one big cluster is formed.
- Once the single cluster is formed, dendrograms are used to divide into multiple clusters depending upon the problem. We will study the concept of dendrogram in detail in an upcoming section.

There are different ways to find distance between the clusters. The distance itself can be Euclidean or Manhattan distance. Following are some of the options to measure distance between two clusters:

- Measure the distance between the closes points of two clusters.
- Measure the distance between the farthest points of two clusters.
- Measure the distance between the centroids of two clusters.
- Measure the distance between all possible combination of points between the two clusters and take the mean.

Code and Output:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# For better readablity;
import matplotlib
matplotlib.rcParams['font.size'] = 16
matplotlib.rcParams['figure.figsize'] = (12, 6)
matplotlib.rcParams['figure.facecolor'] = '#00000000'

df = pd.read_csv('daily.csv')
df
```

	datum	M01AB	M01AE	N02BA	N02BE	N05B	N05C	R03	R06	Year	Month	Hour	Weekday Name
0	1/2/2014	0.00	3.67	3.4	32.40	7	0	0	2	2014	1	248	Thursday
1	1/3/2014	8.00	4.00	4.4	50.60	16	0	20	4	2014	1	276	Friday
2	1/4/2014	2.00	1.00	6.5	61.85	10	0	9	1	2014	1	276	Saturday
3	1/5/2014	4.00	3.00	7.0	41.10	8	0	3	0	2014	1	276	Sunday
4	1/6/2014	5.00	1.00	4.5	21.70	16	2	6	2	2014	1	276	Monday
5	1/7/2014	0.00	0.00	0.0	0.00	0	0	0	0	2014	1	276	Tuesday
6	1/8/2014	5.33	3.00	10.5	26.40	19	1	10	0	2014	1	276	Wednesday
7	1/9/2014	7.00	1.68	8.0	25.00	16	0	3	2	2014	1	276	Thursday
8	1/10/2014	5.00	2.00	2.0	53.30	15	2	0	2	2014	1	276	Friday

```
dataset = df[['M01AB', 'Month']]
dataset
```

	M01AB	Month
0	0.00	1
1	8.00	1
2	2.00	1
3	4.00	1
4	5.00	1
5	0.00	1
6	5.33	1
7	7.00	1
8	5.00	1

```
X = np.array(dataset)
X

class Distance_computation_grid(object):
    def __init__(self):
        pass

    def compute_distance(self,samples):
        Distance_mat = np.zeros((len(samples),len(samples)))
        for i in range(Distance_mat.shape[0]):
            for j in range(Distance_mat.shape[0]):
                if i!=j:
                                                        Distance_mat[i,j]    =
float(self.distance_calculate(samples[i],samples[j]))
                else:
                    Distance_mat[i,j] = 10**4
        return Distance_mat

    #For two samples
    def distance_calculate(self,sample1,sample2):
```

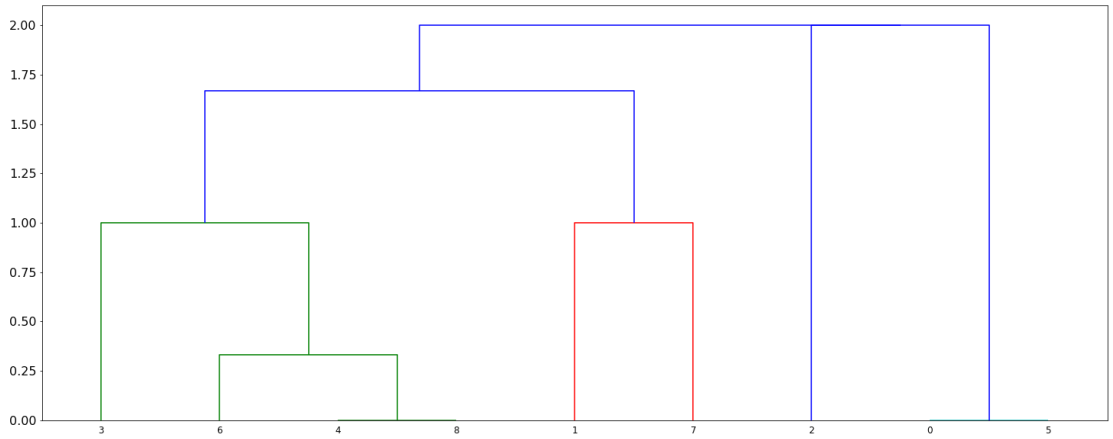
```
        dist = []
        for i in range(len(sample1)):
            for j in range(len(sample2)):
                try:

dist.append(np.linalg.norm(np.array(sample1[i])-np.array(sample2[j])))
                except:
                    dist.append(self.intersampledist(sample1[i],sample2[j]))
        return min(dist)

    #For one sample and one cluster
    def intersampledist(self,s1,s2):
        if str(type(s2[0]))!='<class \'list\'>':
            s2=[s2]
        if str(type(s1[0]))!='<class \'list\'>':
            s1=[s1]
        m = len(s1)
        n = len(s2)
        dist = []
        if n>=m:
            for i in range(n):
                for j in range(m):
                    if (len(s2[i])>=len(s1[j])) and str(type(s2[i][0])!='<class \'list\'>'):
                        dist.append(self.interclusterdist(s2[i],s1[j]))
                    else:
                        dist.append(np.linalg.norm(np.array(s2[i])-np.array(s1[j])))
        else:
            for i in range(m):
                for j in range(n):
                    if (len(s1[i])>=len(s2[j])) and str(type(s1[i][0])!='<class \'list\'>'):
                        dist.append(self.interclusterdist(s1[i],s2[j]))
                    else:
                        dist.append(np.linalg.norm(np.array(s1[i])-np.array(s2[j])))
        return min(dist)

    #For two clusters
    def interclusterdist(self,cl,sample):
        if sample[0]!='<class \'list\'>':
            sample = [sample]
        dist   = []
        for i in range(len(cl)):
            for j in range(len(sample)):
                dist.append(np.linalg.norm(np.array(cl[i])-np.array(sample[j])))
        return min(dist)

progression = [[i] for i in range(X.shape[0])]
samples     = [[list(X[i])] for i in range(X.shape[0])]
m = len(samples)
```

```
distcal  = Distance_computation_grid()

while m>1:
   print('Sample size before clustering    :- ',m)
   Distance_mat      = distcal.compute_distance(samples)
   sample_ind_needed = np.where(Distance_mat==Distance_mat.min())[0]
   value_to_add      = samples.pop(sample_ind_needed[1])
   samples[sample_ind_needed[0]].append(value_to_add)

   print('Cluster Node 1                :-',progression[sample_ind_needed[0]])
   print('Cluster Node 2                :-',progression[sample_ind_needed[1]])


progression[sample_ind_needed[0]].append(progression[sample_ind_need
ed[1]])
                                        progression[sample_ind_needed[0]]           =
[progression[sample_ind_needed[0]]]
   v = progression.pop(sample_ind_needed[1])
   m = len(samples)

   print('Progression(Current Sample)      :-',progression)
   print('Cluster attained                 :-',progression[sample_ind_needed[0]])
   print('Sample size after clustering     :-',m)
   print('\n')
```

```
Sample size before clustering    :-  8
Cluster Node 1                   :- [[0, [4]]]
Cluster Node 2                   :- [[0, [4]]]
Progression(Current Sample)      :- [[1], [2], [3], [5], [6], [7], [8]]
Cluster attained                 :- [1]
Sample size after clustering     :- 7


Sample size before clustering    :-  7
Cluster Node 1                   :- [1]
Cluster Node 2                   :- [8]
Progression(Current Sample)      :- [[[1, [8]]], [2], [3], [5], [6], [7]]
Cluster attained                 :- [[1, [8]]]
Sample size after clustering     :- 6


Sample size before clustering    :-  6
Cluster Node 1                   :- [[1, [8]]]
Cluster Node 2                   :- [6]
Progression(Current Sample)      :- [[[[1, [8]], [6]]], [2], [3], [5], [7]]
Cluster attained                 :- [[[1, [8]], [6]]]
Sample size after clustering     :- 5
```

```
Sample size before clustering    :-  4
Cluster Node 1                   :- [2]
Cluster Node 2                   :- [5]
Progression(Current Sample)      :- [[[2, [5]]], [3], [7]]
Cluster attained                 :- [[2, [5]]]
Sample size after clustering     :- 3


Sample size before clustering    :-  3
Cluster Node 1                   :- [[2, [5]]]
Cluster Node 2                   :- [3]
Progression(Current Sample)      :- [[[[2, [5]], [3]]], [7]]
Cluster attained                 :- [[[2, [5]], [3]]]
Sample size after clustering     :- 2


Sample size before clustering    :-  2
Cluster Node 1                   :- [[[2, [5]], [3]]]
Cluster Node 2                   :- [7]
Progression(Current Sample)      :- [[[[[2, [5]], [3]], [7]]]]
Cluster attained                 :- [[[[2, [5]], [3]], [7]]]
Sample size after clustering     :- 1
```

```
from scipy.cluster.hierarchy import dendrogram, linkage
from matplotlib import pyplot as plt
Z = linkage(X, 'single')
fig = plt.figure(figsize=(25, 10))
dn = dendrogram(Z)
```



Experiment No 9

Date: 05/10/2021

Aim: Implementation of Association Rule Mining algorithm(Apriori)

Theory: Association rule mining is a technique to identify underlying relations between different items. There are many methods to perform association rule mining. The Apriori algorithm is the most simple and straightforward approach. However, since it's the fundamental method, there are many different improvements that can be applied to it.

Concepts of Apriori

Support

Fraction of transactions that contain an itemset.

For example, the support of item I is defined as the number of transactions containing I divided by the total number of transactions.

$$support(I) = \frac{Number\ of\ transactions\ containing\ I}{Total\ number\ of\ transactions}$$

Confidence

Measures how often items in Y appear in transactions that contain X

Confidence is the likelihood that item Y is also bought if item X is bought. It's calculated as the number of transactions containing X and Y divided by the number of transactions containing X.

$$confidence(X \rightarrow Y) = \frac{Number\ of\ transactions\ containing\ X\ and\ Y}{Number\ of\ transactions\ containing\ X}$$

Frequent Item Set

An itemset whose support is greater than or equal to a minSup threshold

Frequent itemsets or also known as frequent pattern simply means all the itemsets that the support satisfies the minimum support threshold.

Code and output:

```
!pip install apyori
```

```
Collecting apyori
  Downloading apyori-1.1.2.tar.gz (8.6 kB)
Building wheels for collected packages: apyori
  Building wheel for apyori (setup.py) ... done
  Created wheel for apyori: filename=apyori-1.1.2-py3-none-any.whl size=5974 sha256=bf7745a2dff414ec23d0
  Stored in directory: /root/.cache/pip/wheels/cb/f6/e1/57973c631d27efd1a2f375bd6a83b2a616c4021f24aab840
Successfully built apyori
Installing collected packages: apyori
Successfully installed apyori-1.1.2
```

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from apyori import apriori
```

```
[ ] store_data = pd.read_csv('salesdaily.csv')
    store_data.head()
```

	datum	M01AB	M01AE	N02BA	N02BE	N05B	N05C	R03	R06	Year	Month	Hour	Weekday Name
0	1/2/2014	0.0	3.67	3.4	32.40	7.0	0.0	0.0	2.0	2014	1	248	Thursday
1	1/3/2014	8.0	4.00	4.4	50.60	16.0	0.0	20.0	4.0	2014	1	276	Friday
2	1/4/2014	2.0	1.00	6.5	61.85	10.0	0.0	9.0	1.0	2014	1	276	Saturday
3	1/5/2014	4.0	3.00	7.0	41.10	8.0	0.0	3.0	0.0	2014	1	276	Sunday
4	1/6/2014	5.0	1.00	4.5	21.70	16.0	2.0	6.0	2.0	2014	1	276	Monday

```
[ ] records = []
    for i in range(0, 1000):
        records.append([str(store_data.values[i,j]) for j in range(0, 10)])
```

```
[ ] association_rules = apriori(records, min_support=0.0045, min_confidence=0.2, min_lift=3, min_length=2)
    association_results = list(association_rules)
```

```
print(len(association_results))
```

```
61
```

+ Code + Text

```python
for item in association_results:

    # first index of the inner list
    # Contains base item and add item
    pair = item[0]
    items = [x for x in pair]
    print("Rule: " + items[0] + " -> " + items[1])

    #second index of the inner list
    print("Support: " + str(item[1]))

    #third index of the list located at 0th
    #of the third index of the inner list

    print("Confidence: " + str(item[2][0][2]))
    print("Lift: " + str(item[2][0][3]))
    print("====================================")
```

```
Rule: 21.0 -> 11.0
Support: 0.007
Confidence: 0.31818181818181823
Lift: 3.3492822966507183
====================================
Rule: 2.68 -> 13.0
Support: 0.005
Confidence: 0.23809523809523808
Lift: 3.720238095238095
====================================
Rule: 11.0 -> 21.0
Support: 0.005
Confidence: 0.2272727272727273
Lift: 3.0303030303030307
====================================
Rule: 13.0 -> 2.68
Support: 0.005
Confidence: 0.23809523809523808
Lift: 4.859086491739553
====================================
Rule: 5.36 -> 2014
Support: 0.005
Confidence: 1.0
Lift: 3.4722222222222223
====================================
Rule: 2016 -> 0.0
Support: 0.006
```

```
[ ]
```

```
Rule: 2016 -> 0.0
Support: 0.006
Confidence: 0.6666666666666667
Lift: 3.0441400304414006
====================================
Rule: 7.1 -> 2016
Support: 0.005
Confidence: 0.7142857142857143
Lift: 3.2615786040443577
====================================
Rule: 26.6 -> 3.0
Support: 0.005
Confidence: 1.0
Lift: 3.115264797507788
====================================
Rule: 1.0 -> 2014
Support: 0.005
Confidence: 0.7142857142857143
Lift: 3.417634996582365
====================================
Rule: 1.0 -> 2014
Support: 0.005
Confidence: 0.7142857142857143
Lift: 3.417634996582365
====================================
Rule: 1.0 -> 2016
Support: 0.006
Confidence: 0.6666666666666667
Lift: 5.847953216374269
====================================
Rule: 1.0 -> 35.0
Support: 0.005
Confidence: 0.5555555555555556
Lift: 4.340277777777778
```

Experiment No 10

Date: 20/10/2021

AIM: Implementation of Page rank/HITS algorithm

THEORY:

Hyperlink Induced Topic Search (HITS) is an algorithm used in link analysis. It could discover and rank the webpages relevant for a particular search. The idea of this algorithm originated from the fact that an ideal website should link to other relevant sites and also being linked by other important sites.

HITS uses hubs and authorities to define a recursive relationship between web pages.

- Authority: A node is high-quality if many high-quality nodes link to it
- Hub: A node is high-quality if it links to many high-quality nodes

Algorithm Steps

- Initialize the hub and authority of each node with a value of 1
- For each iteration, update the hub and authority of every node in the graph
- The new authority is the **sum of the hub** of its parents
- The new hub is the **sum of the authority** of its children
- Normalize the new authority and hub

The first step that the algorithm takes is to retrieve the data of the search query. This is the information people type on search engines to obtain a specific result. Then, it performs a computation only

regarding these results, without taking into consideration other websites.

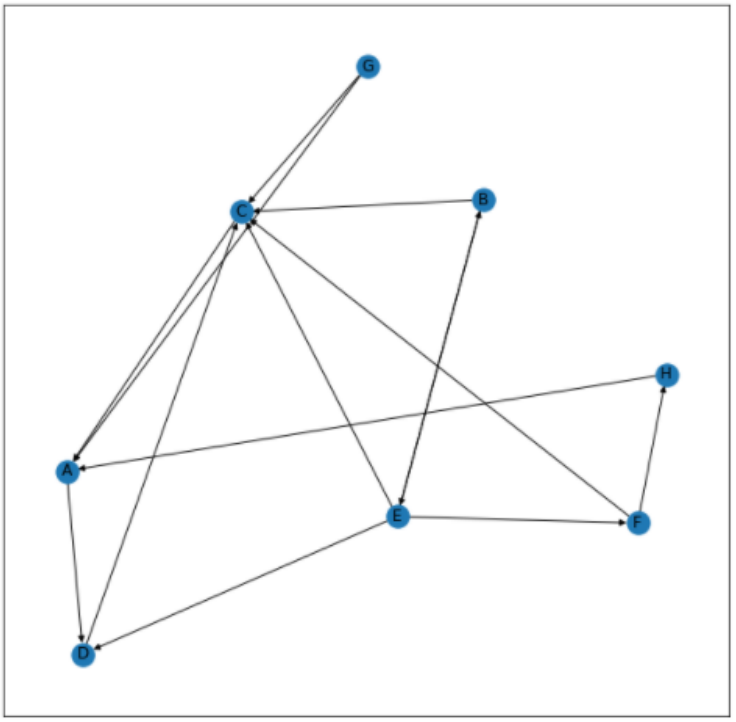
After that, authoritative and hub values are defined, and a process of iteration begins. In the iteration process, two updates are done: the authority update and the hub update. For HITS, an authoritative website is a site that has valuable content. These types of websites tend to rank higher on the search engine results page because they are considered ‘expert’ pages. Much like PageRank (another algorithm that identifies and ranks sites), HITS takes the linkage of documents on the web into account.

However, HITS differentiates itself from PageRank on a few aspects:

- It is executed at query time, not at indexing time. The hub and authority scores assigned to a page are query-specific. This means that the ranking will always consider the keyword or content that people are searching for.
- Whereas algorithms like PageRank compute one score per document, HITS compute two.
- It is processed on a small subset of documents, instead of all documents, like PageRank does.
- Search engines do not commonly use it.

```python
import networkx as nx
import matplotlib.pyplot as plt

graph = nx.DiGraph()

graph.add_edges_from([('A', 'D'), ('B', 'C'), ('B', 'E'), ('C', 'A'),
                      ('D', 'C'), ('E', 'D'), ('E', 'B'), ('E', 'F'),
                      ('E', 'C'), ('F', 'C'), ('F', 'H'), ('G', 'A'),
                      ('G', 'C'), ('H', 'A')])

plt.figure(figsize =(10, 10))
nx.draw_networkx(graph, with_labels = True)

hubs, authorities = nx.hits(graph, max_iter = 10, normalized = True)

print("Hub Scores: ", hubs)
print("Authority Scores: ", authorities)
```

```
Hub Scores:  {'A': 0.04642540403219995, 'D': 0.13366037526115382, 'B': 0.15763599442967322, 'C': 0.037389
Authority Scores:  {'A': 0.10864044011724344, 'D': 0.13489685434358, 'B': 0.11437974073336446, 'C': 0.388
```

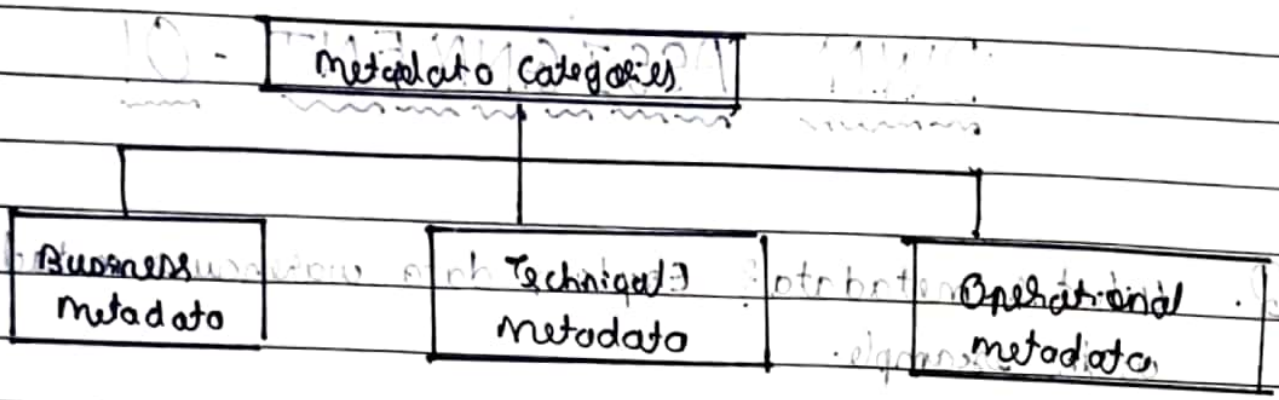


Tushar Nankani
1902112

DWM ASSIGNMENT - 01

Q. What is metadata? Explain data warehouse metadata with example.

A → • Metadata is simply defined as data about data. The data that is used to represent other data is known as metadata.
Eg:- The index of a book serves as metadata for the contents in the book. In other words, we can say that metadata is the summarized data that leads us to detailed data.
• In terms of data warehouse, we create metadata for data names and definitions of the given data. It acts like a roadmap to the data warehouse and defines the warehouse objects. It also acts as a directory which helps in the decision support system to locate the contents of a data warehouse.
• Metadata can be categorized into 3 categories:
⇒ Business metadata:- It has the data ownership information, business definition and changing policies.
⇒ Technical metadata:- It includes database system names, table and column names and sizes, data types and allowed values. Technical metadata also includes structural info like primary, foreign keys, attributes and indices.
⇒ Operational metadata:- Includes currency of data and data lineage. Currency of data means whether the data is active, archived or purged.



- Role of Metadata :-

⇒ It acts as a directory.

⇒ This directory helps the decision support system to locate the contents of the data warehouse.

⇒ Metadata helps in decision support system for mapping of data when data is transformed from operational environment to data warehouse environment.

⇒ It helps in summarization between current detailed data and highly summarized data.

⇒ Metadata also helps in summarization between lightly detailed data and highly summarized data.

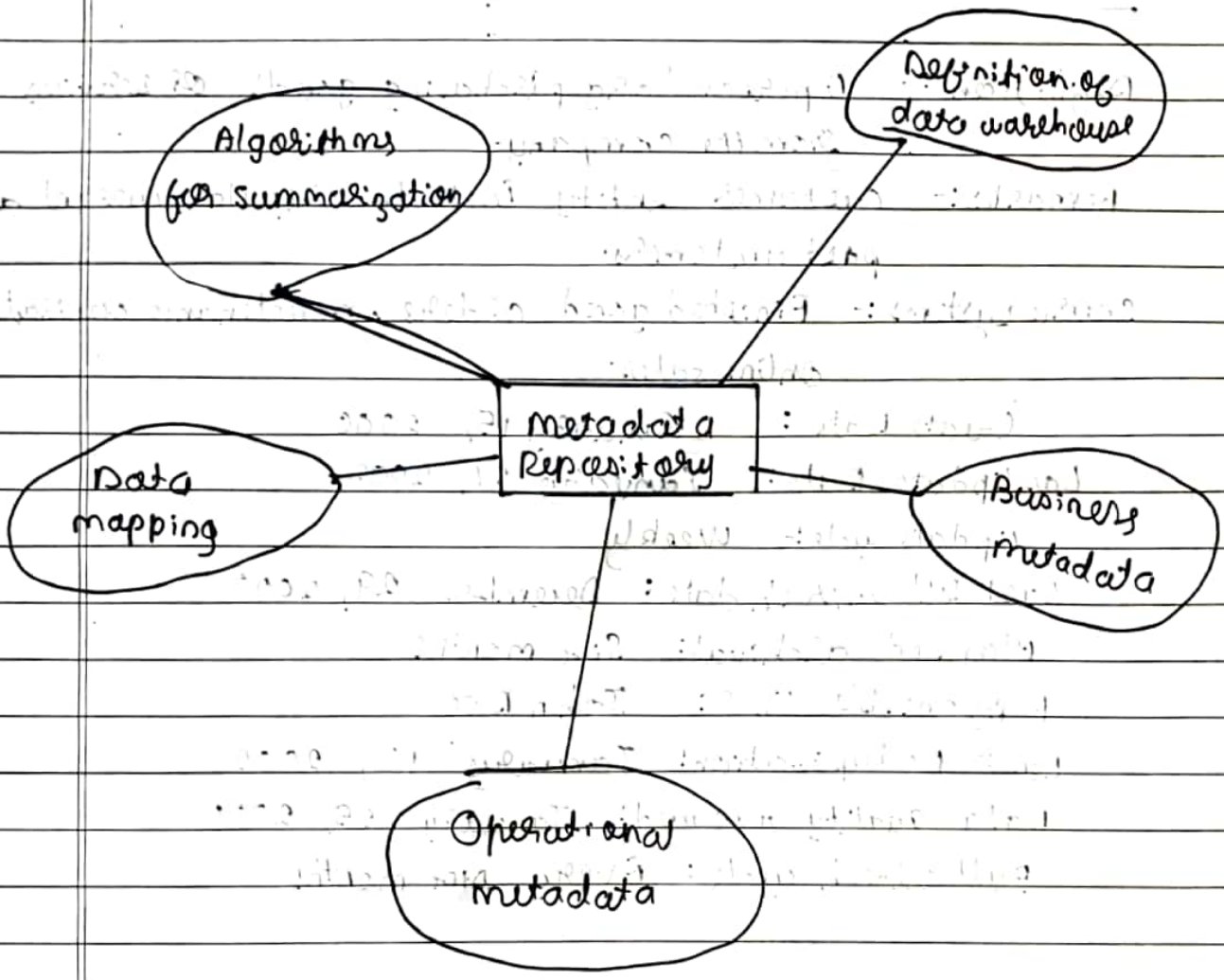
⇒ Metadata is used for query tools.

⇒ Metadata is used in extraction and cleaning tools.

⇒ Metadata is used for transformation tools.

⇒ Plays an important role in loading functions.

Metadata Repository is an integral part of a data warehouse system. It has following metadata:-



Example:

Metadata for a 'Customer' entity:

Definition :- A person / org purchasing goods or services from the company.

Remarks :- Customer entity includes regular, current and past customers.

Source systems :- Finished good orders, maintenance contracts, online sales.

Create Date : January 15, 2000

Last Update Date : January 21, 2002.

Update Cycle :- Weekly

Last full refresh date : December 29, 2000

Planned archival : Six months

Responsible User : John Doe

Last Deduplication : January 10, 2002

Data quality reviewed : January 25, 2001

Full refresh cycle : Every six months

PAGE No
DATE / /

ASSIGNMENT-2

Q1. Write short notes on:

a) Multilevel Association rules and Multidimensional association rules:-

→ 1. Association rules generated from mining data at multiple levels of abstraction are called multiple level or multilevel association rules.
2. Multilevel association rules can be mined efficiently using concept hierarchies under a support-confidence.
3. Rules at high concept level may add to common sense while rules at low concepts level may not be useful always.
4. Using uniform minimum support for all levels:
- When a uniform min support threshold is used, the search procedure is simplified.
- The method is so simple, in that users are required to specify only one min support threshold.
- The same minimum support threshold is used when mining at each level of abstraction.
- In reduced minimum supports at lower ends, each level of abstraction has its own minimum support threshold.

Multidimensional association rules:

1. In multidimensional association rules:
- Attributes can be categorical or quantitive.

PAGE No.
DATE / /

- Initia Quantitative attributes are numeric and incorporates hierarchy.
- Numeric attributes must be discretized.
- Multi-dimensional association rule consists of more than one dimension:
Eg. buys (X, "Buys IBM laptop") buys (X, "HB Printer")

2. Three approaches in mining multidimensional association rules:

1) Using static discretization of quantative attributes:
- Discretization is static and happens preceding mining.
- Discretized ascribes are treated as unimitigated.
- Use apriori calculation to locate all k-regular predicate sets. Each subset of it should be continuous.

2) Use powerful discretization of quantitive traits:
- Known as mining Quantitative Association Rules.
- Numeric properties are progressively discretized.

3) Grid FOR TUPLES:
- This id dynamic discretization measure that considers the distance between info focused.
- It includes a two stage mining measure:
  - Perform bucking to discover the time period included
  - Get affiliation rules via looking for gatherings of groups that happen together.

b) Web Usage Mining:

→ Web usage mining, a subset of Data Mining, is the extraction of various types of interesting data that is readily available and accessible in the ocean of huge web pages, Internet - or formally known as World wide web (WWW). Being one of the applications of data mining technique, it has helped to analyze user activities on different web pages and track them over a period of time.

The 3 main types of web data are:

1) Web Content Data: HTML, web pages, images, audio-video, content, JSP, PHP, etc are included in this category.
2) Web Structure Data: Relationship/links describing the connection between webpages is structured data.
3) Web Usage Data: It involves log data which is collected by the Web Server and Application Server. The data in this type can be mainly categorized into 3 types based on its source:
   - Server side
   - Client side
   - Proxy side

There are other additional data sources also which include cookies, demographics, etc.

PAGE No.
DATE / /

Types of Web Usage Mining based upon the Usage Data:

1. Web Server Data: It generally includes IP address, browser logs, proxy server logs, user profiles, etc. The user logs are collected by the web server data.

2. Application Server Data: An added feature on the commercial application servers is to build applications on it. Tracking various business events and logging them into application server logs is mainly what application servaer data consists of.

3. Application-Level Data: There are various new kinds of events that can be there in an application. The logging feature enabled in them helps us get the past record of events.

Applications of Web Usage Mining:

- Personalization of Web content
- E-commerce
- Prefetching and Catching