

STELLA MARY'S COLLEGE OF ENGINEERING

(Approved by AICTE, New Delhi, Affiliated to Anna University, Chennai & Accredited by NAAC & NBA (CSE & MECH)

Aruthenganvilai, Kallukatti Junction, Azhikal Post Kanyakumari District – 629 202, Tamil Nadu



**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING**

CCS341 DATA WAREHOUSING

REGULATION - 2021

SEMESTER-VI

2023-2024 (EVEN)



MANUAL

Prepared by,

Mrs.S.Mamitha

Asst.Prof / Dept. of

CSE

INSTITUTE VISION

To be a beacon of academic excellence, empowering future innovators with technical mastery to harness technology for positive global change.

INSTITUTE MISSION

- To Cultivate a vibrant learning environment where students delve into the frontiers of technical knowledge, hone their problem-solving skills, and embrace innovation to transform ideas into solutions that address global challenges.
- To bridge the gap between technical brilliance and real-world impact by forging strong industry partnerships, fostering cutting-edge research, and nurturing entrepreneurial drive in our students, empowering them to build a better future through technology.
- To ignite the spark of intellectual curiosity within every student, equip them with the tools and knowledge. To become pioneers in their chosen fields, and guide them towards ethical and responsible use of technology for the betterment of humanity.

DEPARTMENT VISION

To be a leading department in computer science education and innovation, equipping students with the expertise to create and solve real-world challenges through ethical, responsible, and transformative technological solutions for global progress.

DEPARTMENT MISSION

- To provide a strong foundation in computer science principles, fostering innovative problem-solving and a deep-rooted commitment to developing socially responsible and impactful engineering solutions.
- To bridge academia and industry by fostering partnerships and research that equips students with practical skills and entrepreneurial drive.
- To inspire intellectual curiosity and ethical responsibility, guiding students to apply technology for the advancement of humanity.

Program Educational Objectives (PEOs)

PEO1:

Graduates will be competent in creating innovative technologies through inter- disciplinary research and comprehensive skills sets that are suitable for the global computing industry.

PEO2:

Graduates will be capable of managing leading positions with a broad understanding of the application of ethics in evolving computer-based solutions for the societal needs.

PEO3:

Graduates will imbibe entrepreneurial qualities and develop their career by upgrading their, communication, analytical and professional skills constantly.

Program Specific Outcomes (PSOs)

PSO1:

Use data management techniques and algorithmic thinking for Software Design and Development practices.

PSO2:

Develop reliable IT solutions based on the expertise in Distributed Applications Development, Web Designing and Networking for various societal needs and entrepreneurial practices ethically.

PSO3:

Manage multidisciplinary environments effectively through their interpersonal and analytical skills and be responsible members and leaders of the society.

PROGRAMME OUTCOMES (POs):

- 1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.
- 2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- 3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- 4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- 5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- 6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- 7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
- 8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

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

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

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

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

RUBRICS FOR ASSESSING LABORATORY

Sl. No.	Criteria	Marks	Excellent (3) 91% - 100%	Good (2) 71% - 90%	Average (1) 50% - 70%	Poor (0) <50%
1	Observation	3	Gives clear idea about the aim and having the capability of both recording & analyzing the data much easier. (3)	Capability of both recording & analyzing the data much easier but no proper clarification about the objective. (2)	Gives clear idea about the target and has less capability of both recording & analyzing the data. (1)	Gives indistinct idea about the target and has less capability of both recording & analyzing the data & who feel difficult to follow the objectives. (<1)
2	Assessment	3	Have executed the system in an efficient way & make credible and unbiased judgments regarding the conduct of the experiments. (3)	Executed the system with less difficulty & has partial judgements regarding the overall system. (2)	Executed the system with less efficiency and has no judgements regarding the system. (1)	Incomplete system execution & lack of judgments regarding the system. (<1)
3	Submission	4	Followed all the instructions given in the procedure and submitted the observation books in time. (4)	Followed all the instructions given in the procedure with some assisting (3)	Followed some of the instructions given in the procedure & late in submission of note books. (2)	Trying to follow the instructions given in the procedure & late in submission of note books. (<1)



STELLA MARY'S COLLEGE OF ENGINEERING

(Approved by AICTE, New Delhi, Affiliated to Anna University, Chennai, Accredited by NAAC & NBA (Mech& CSE))

Aruthenganvilai, Kallukatti Junction, Azhikal Post, Kanyakumari District - 629 202

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Academic Year: 2023-2024(EVEN)

CCS341-DATA WAREHOUSING

Course Objectives and Course Outcomes

COURSE OUTCOME:

- Design data warehouse architecture for various Problems
- Apply the OLAP Technology
- Analyse the partitioning strategy
- Critically analyze the differentiation of various schema for given problem
- Frame roles of process manager & system manager

COURSE OBJECTIVES:

- To know the details of data warehouse Architecture
- To understand the OLAP Technology
- To understand the partitioning strategy
- To differentiate various schema
- To understand the roles of process manager & system manager

Course Coordinator

CCS341-DATA WAREHOUSING

Exp. No.	Name of the Experiment
Ex.no:1	DATA EXPLORATION AND INTEGRATION WITH WEKA
Ex.no:2	APPLY WEKA TOOL FOR DATA VALIDATION
Ex.no:3	PLAN THE ARCHITECTURE FOR REAL TIME APPLICATION
Ex.no:4	WRITE THE QUERY FOR SCHEMA DEFINITION
Ex.no:5	DESIGN DATA WARE HOUSE FOR REAL TIME APPLICATION
Ex.no:6	ANALYSE THE DIMENSIONAL MODELING
Ex.no:7	CASE STUDY USING OLAP
Ex.no:8	CASE STUDY USING OLTP
Ex.no:9	CASE STUDY ON DATA WAREHOUSE TESTING
	CONTENT BEYOND SYLLABUS
Ex.No.10	CLUSTERING RULE PROCESS ON DATASET USING SIMPLE K- MEANS ALGORITHM

EX NO:1

DATA EXPLORATION AND INTEGRATION WITH WEKA

AIM:

To apply WEKA tool for data exploration and integration.

Installation steps of WEKA data mining toolkit.

1. Download the software as your requirements from the below given link. <http://www.cs.waikato.ac.nz/ml/weka/downloading.html>
2. The Java is mandatory for installation of WEKA so if you have already Java on your machine then download only WEKA else download the software with JVM.
3. Then open the file location and double click on the file.



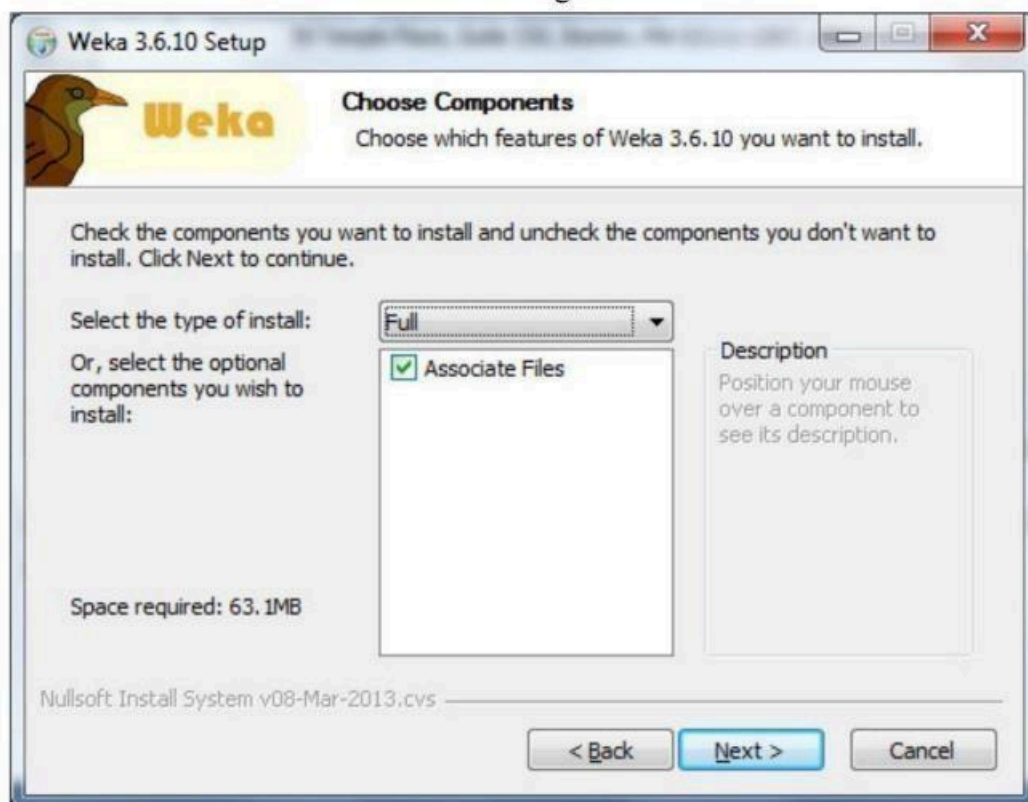
4. Click Next



5. Click I Agree.



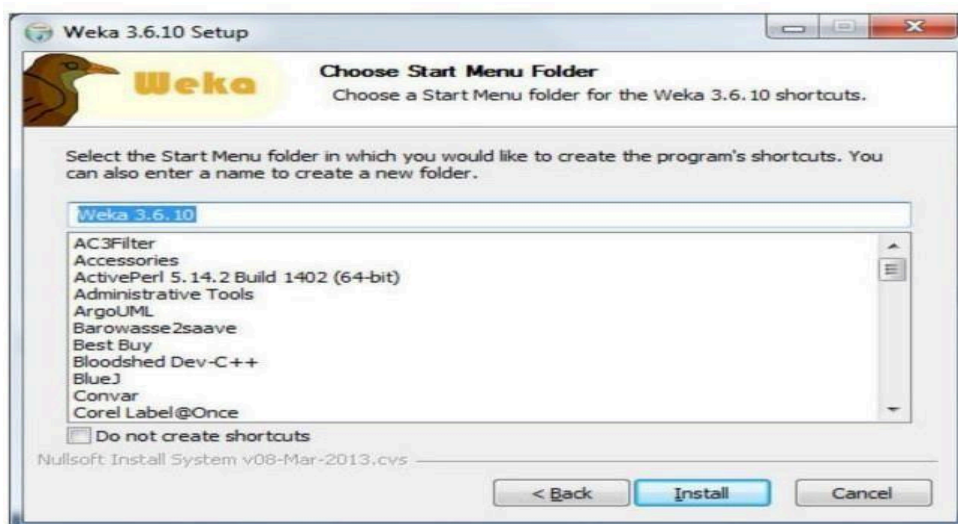
6. As your requirement do the necessary changes of settings and click Next.
Full and Associate files are the recommended settings.



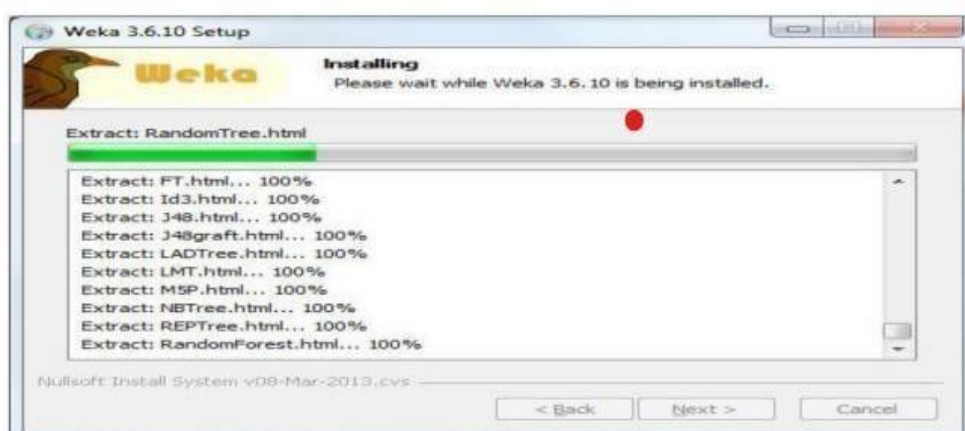
7. Change to your desired installation location.



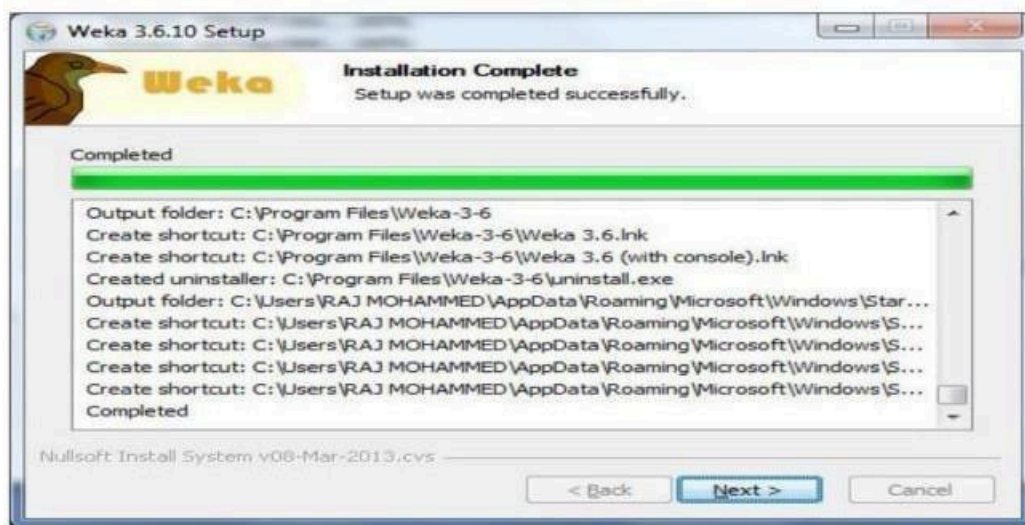
8. If you want a shortcut then check the box and click Install.



9. The Installation will start wait for a while it will finish within a minute.



10. After complete installation click on Next.



11. That's all click on the Finish and take a shovel and start Mining.





This is the GUI you get when started. You have 4 options Explorer, Experimenter, Knowledge Flow and Simple CLI.

The Graphical User Interface:

The Weka GUI Chooser (class `Weka.Gui.GUI Chooser`) provides a starting point for launching Weka's main GUI applications and supporting tools. If one prefers a MDI (multiple document interface) appearance, then this is provided by an alternative launcher called Main. The GUI Chooser consists of four buttons, one for each of the four major Weka applications, and four menus.

The buttons can be used to start the following applications:

- Explorer: An environment for exploring data with WEKA
- Experimenter: An environment for performing experiments and conducting statistical tests between learning schemes.
- 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.
- Simple CLI: 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.

1. Explorer

The Graphical user interface

1.1 Section Tabs

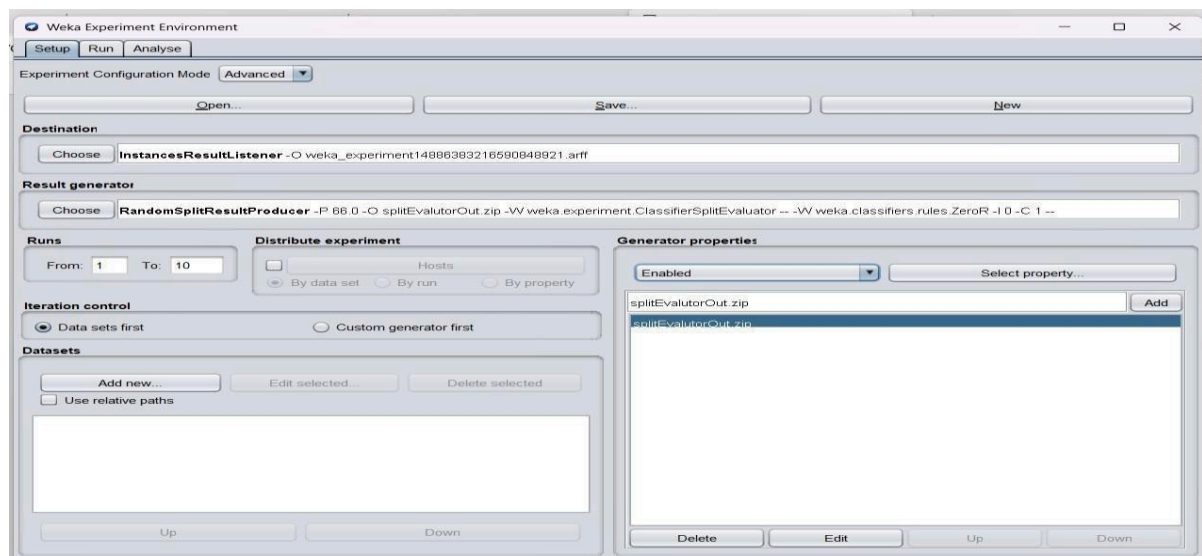
At the very top of the window, just below the title bar, is a row of tabs. When the Explorer is first started only the first tab is active; the others are greyed out. This is because it is necessary

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 & 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.

2. Weka Experimenter:

The Weka Experiment Environment enables the user to create, run, modify, and analyse experiments in a more convenient manner than is possible when processing the schemes individually. For example, the user can create an experiment that runs several schemes against a series of datasets and then analyse the results to determine if one of the schemes is (statistically) better than the other schemes.



The Experiment Environment can be run from the command line using the Simple CLI.

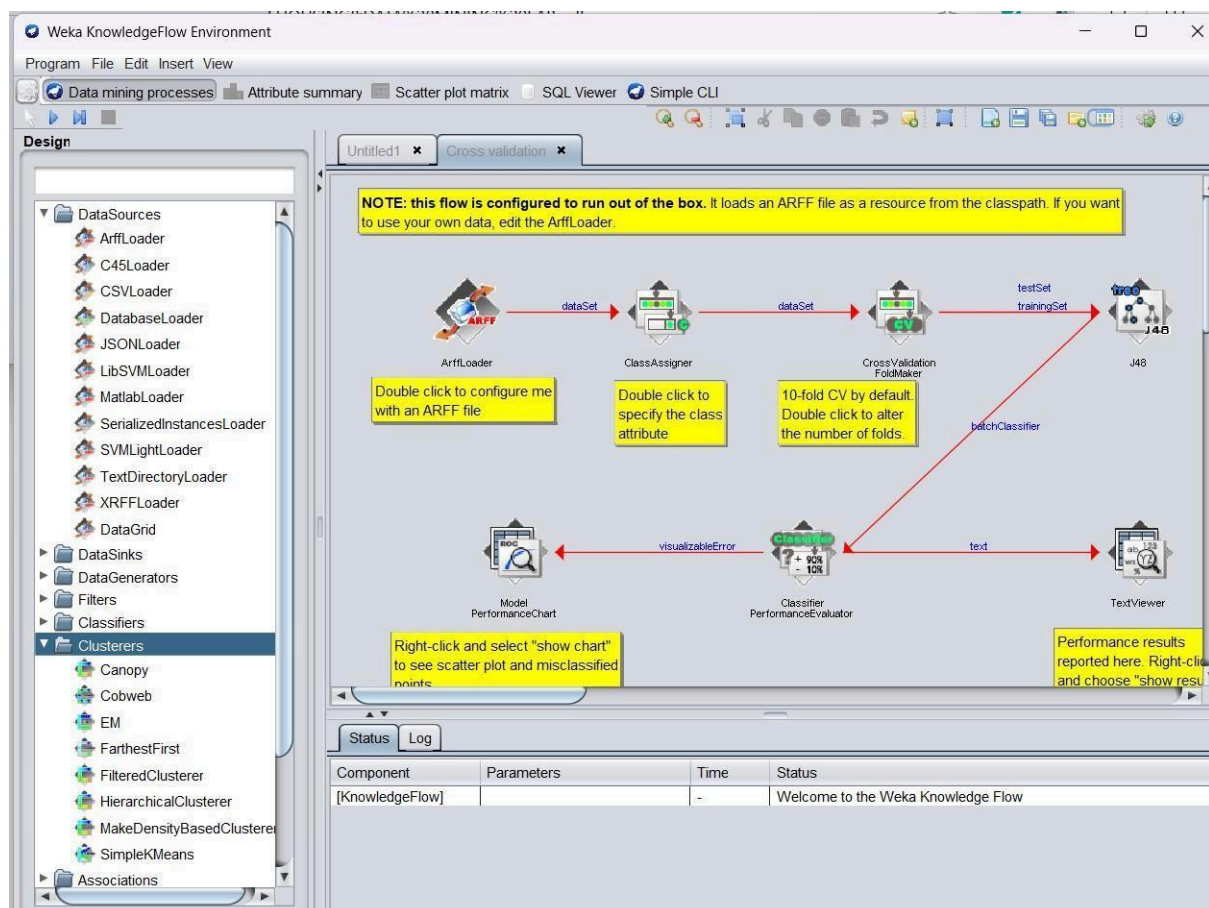
The Experimenter comes in two flavours, either with a simple interface that provides most of the functionality one needs for experiments, or with an interface with full access to the Experimenter's capabilities. You can choose between those two with the Experiment Configuration Mode radio buttons:

- Simple
- Advanced

Both setups allow you to setup standard experiments, that are run locally on a single machine, or remote experiments, which are distributed between several hosts. The distribution of experiments cut down the time the experiments will take until completion, but on the other hand the setup takes more time. The next section covers the standard experiments (both, simple and advanced), followed by the remote experiments and finally the analysing of the results.

3. Knowledge Flow:

The Knowledge Flow provides an alternative to the Explorer as a graphical front end to WEKA's core algorithms. The Knowledge Flow presents a data-flow inspired interface to WEKA. The user can select WEKA components from a palette, place them on a layout canvas and connect them together in order to form a knowledge flow for processing and analysing data. At present, all of WEKA's classifiers, filters, clusters, associators, loaders and savers are available in the Knowledge Flow along with some extra tools.



The Knowledge Flow can handle data either incrementally or in batches (the Explorer handles batch data only).

4. Simple CLI:

The Simple CLI provides full access to all Weka classes, i.e., classifiers, filters, clusters,

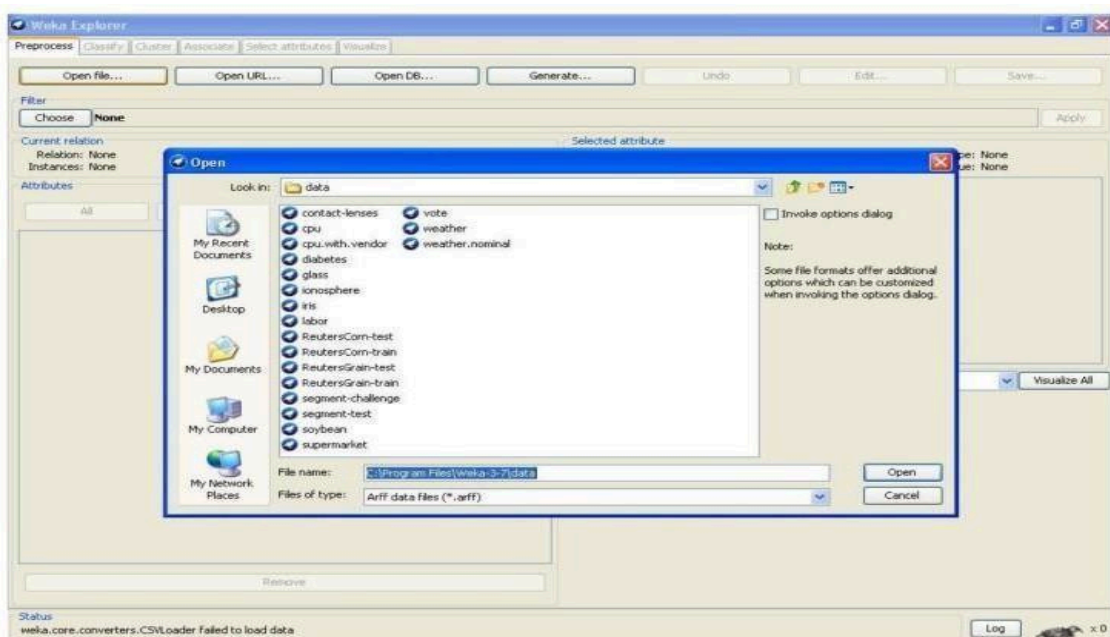
etc., but without the hassle of the CLASSPATH (it facilitates the one, with which Weka was started). It offers a simple Weka shell with separated command line and output.



Explore the available data sets in WEKA:

Perform data preprocessing tasks and demonstrate to categorical data and nominal data:

1. Open WEKA Tool.
2. Click on WEKA Explorer.
3. Click on open file button.
4. Choose WEKA folder in C drive.
5. Select and Click on data option button.



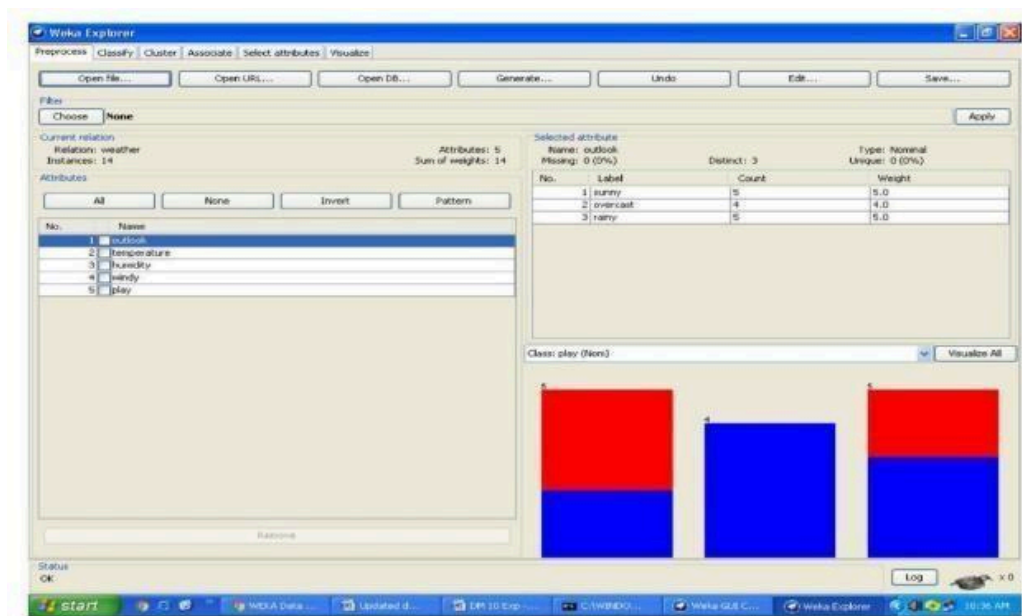
Sample Weka Data Sets

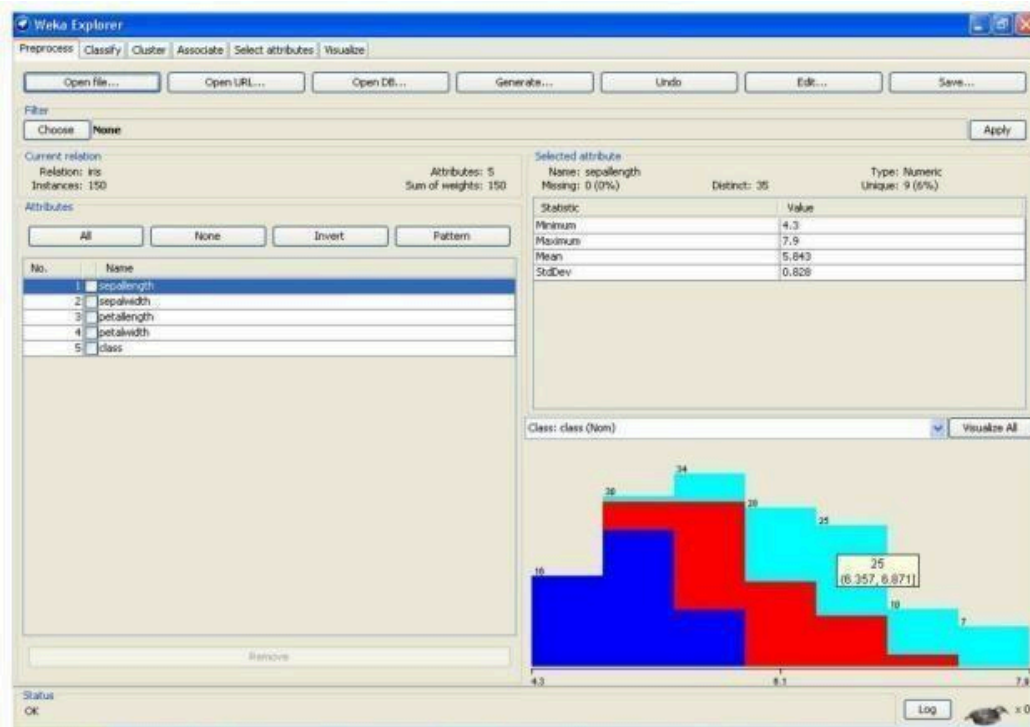
Below are some sample WEKA data sets, in arff format.

- contact-lens.arff
- cpu.arff
- cpu.with-vendor.arff
- diabetes.arff
- glass.arff
- ionospehre.arff
- iris.arff
- labor.arff
- ReutersCorn-train.arff
- segment-test.arff
- soybean.arff
- supermarket.arff
- vote.arff
- weather.arff
- weather.nominal.arff

Steps for load the Weather data set.

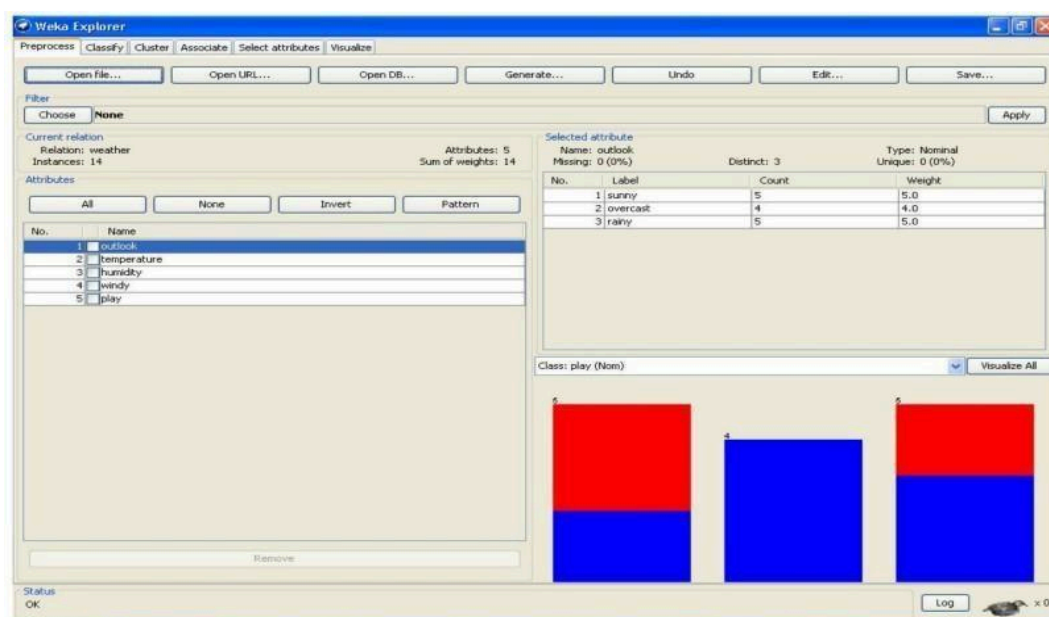
1. Open WEKA Tool.
2. Click on WEKA Explorer.
3. Click on open file button.
4. Choose WEKA folder in C drive.
5. Select and Click on data option button.
6. Choose Weather. arff file and open the file.





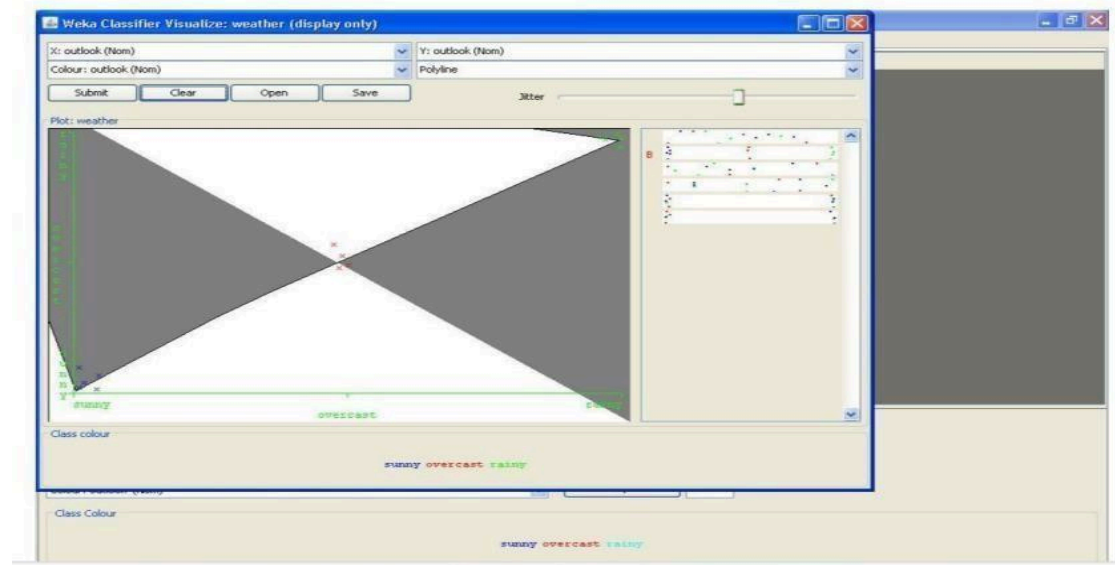
List out the attribute names:

1. outlook
2. temperature
3. humidity
4. windy
5. play



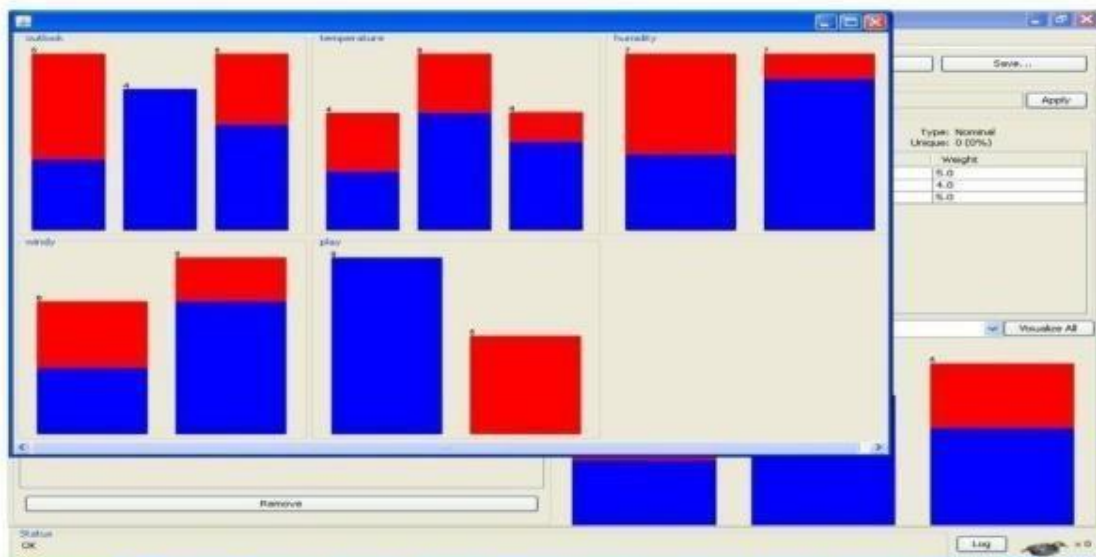
Steps to plot the histogram:

1. Open WEKA Tool.
2. Click on WEKA Explorer.
3. Click on Visualize button.
4. Click on right click button.
5. Select and Click on polyline option button.



To Visualize the data in various dimensions:

Click on Visualize All button in WEKA Explorer.



CONCLUSION:

Thus, the exploration and integration of data was done successfully with WEKA.

EX NO:2

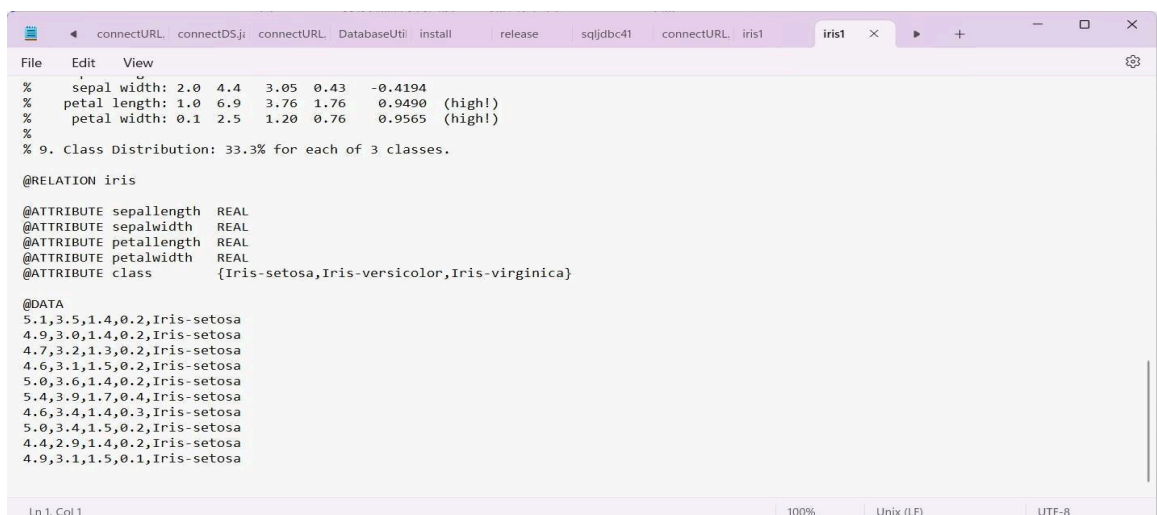
APPLY WEKA TOOL FOR DATA VALIDATION

AIM:

To apply WEKA tool for data validation by splitting a data set into training, testing and cross validating instances.

PROCEDURE:

1. Load a sample data set to be validated, iris.arff from WEKA dataset(Open C:\program files->WEKA 8.5->data->iris.arff) and save 10 instances in the file and save the file as iris1.arff in a desired location(D:\new folder).



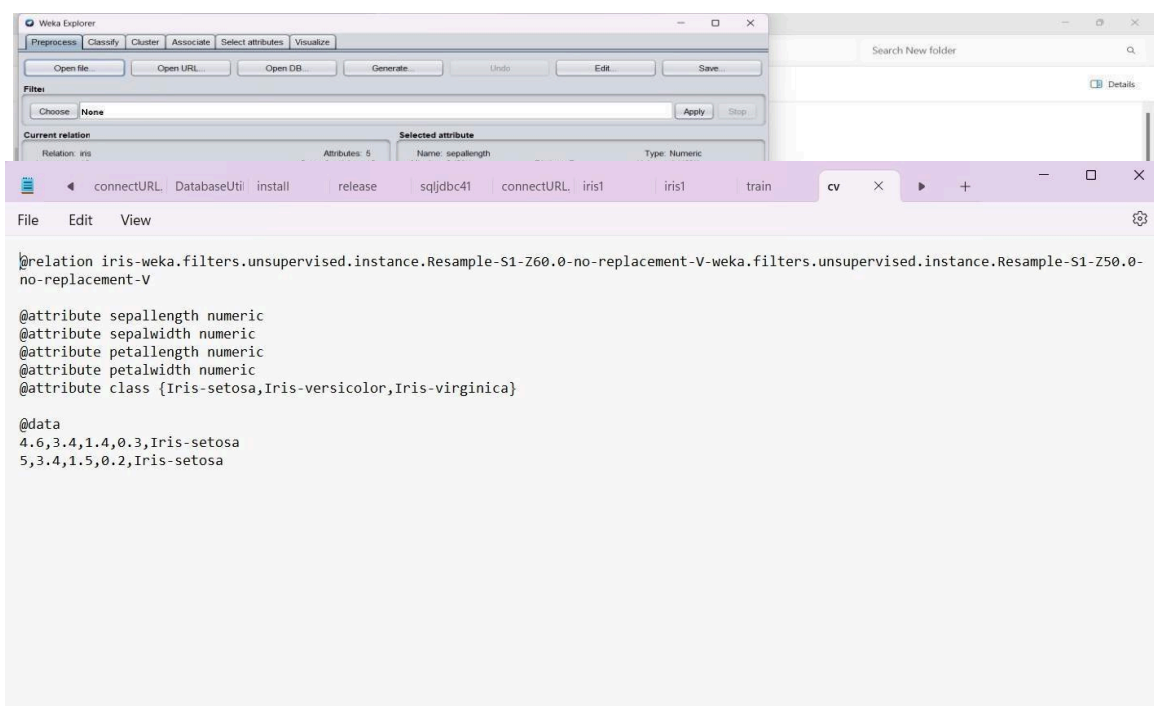
```
%      sepal width: 2.0  4.4  3.05  0.43  -0.4194
%      petal length: 1.0  6.9  3.76  1.76  0.9490  (high!)
%      petal width: 0.1  2.5  1.20  0.76  0.9565  (high!)
%
% 9. Class Distribution: 33.3% for each of 3 classes.

@RELATION iris

@ATTRIBUTE sepalwidth REAL
@ATTRIBUTE sepalwidth REAL
@ATTRIBUTE petallength REAL
@ATTRIBUTE petalwidth REAL
@ATTRIBUTE class {Iris-setosa,Iris-versicolor,Iris-virginica}

@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
```

2. Open WEKA Tool.
3. Click on WEKA Explorer.
4. In the Preprocessor tab, click on Open file, browse and load iris1.arff dataset (located at D:\new folder\iris1.arff) with 10 instances.

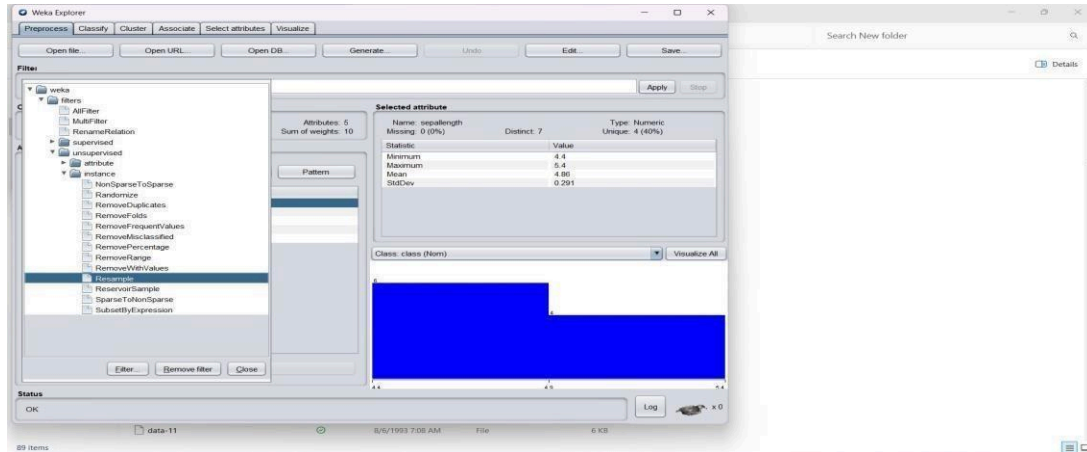


```
@relation iris-weka.filters.unsupervised.instance.Resample-S1-Z60.0-no-replacement-V-weka.filters.unsupervised.instance.Resample-S1-Z50.0-no-replacement-V

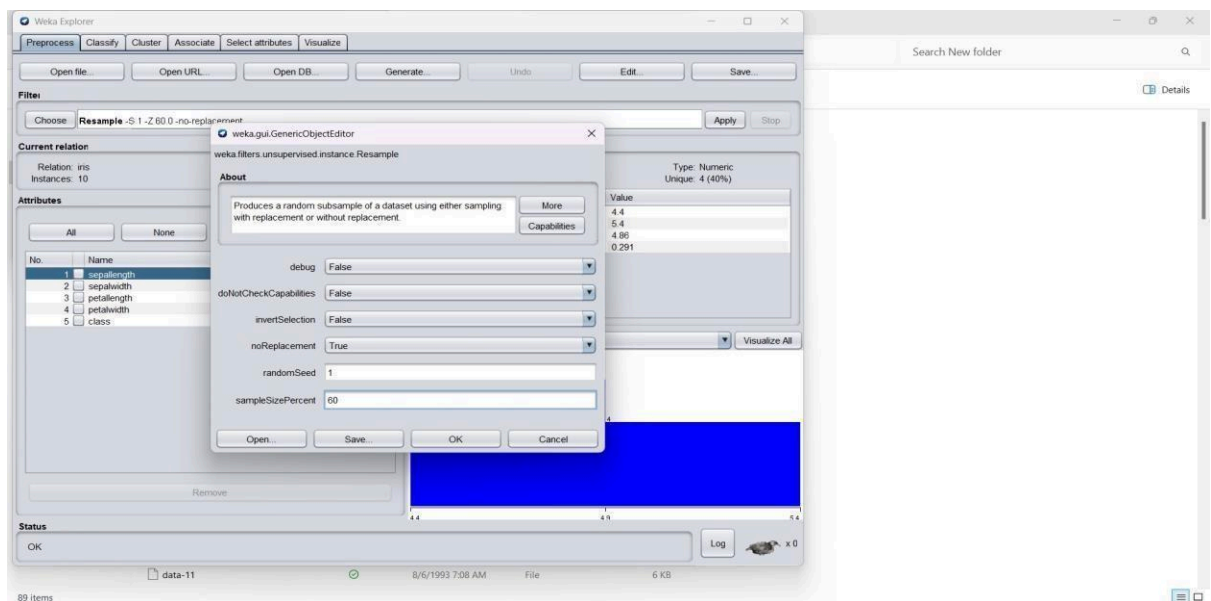
@attribute sepalwidth numeric
@attribute sepalwidth numeric
@attribute petallength numeric
@attribute petalwidth numeric
@attribute class {Iris-setosa,Iris-versicolor,Iris-virginica}

@data
4.6,3.4,1.4,0.3,Iris-setosa
5,3.4,1.5,0.2,Iris-setosa
```


5. For data validation, first split 60% of the dataset into training set with 6 instances. Click on Weka->select Filters->select Unsupervised ->select Instance->select Resample, then click Filter pane.

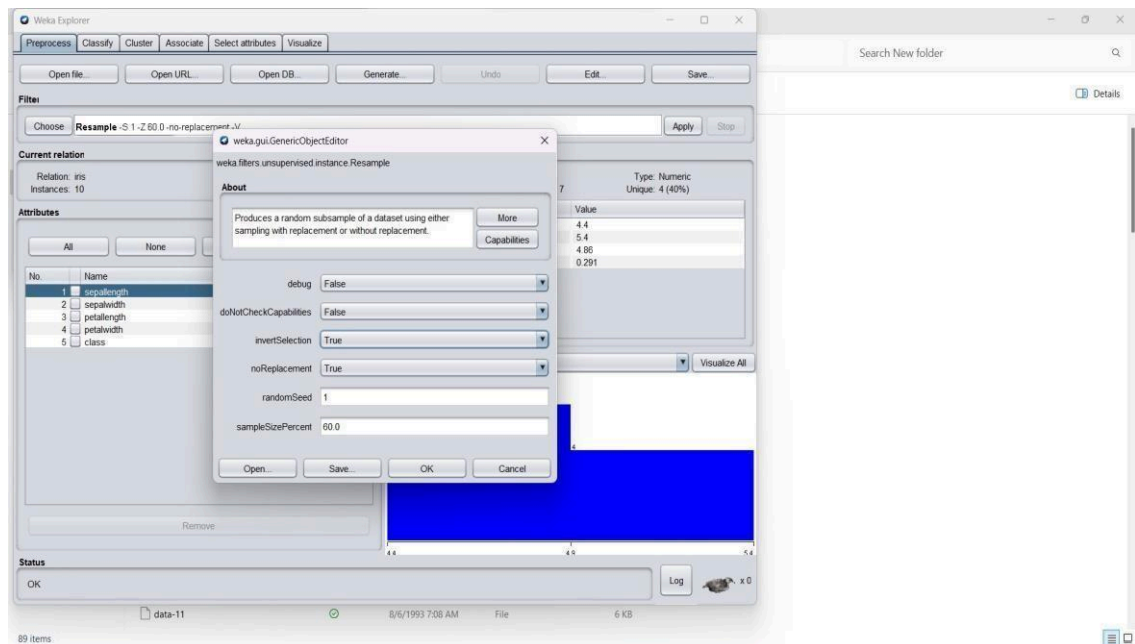


6. From the Filter pane , choose invertSelection – False, noReplacement- True, Sample size percentage-60, then click Ok->click Apply.

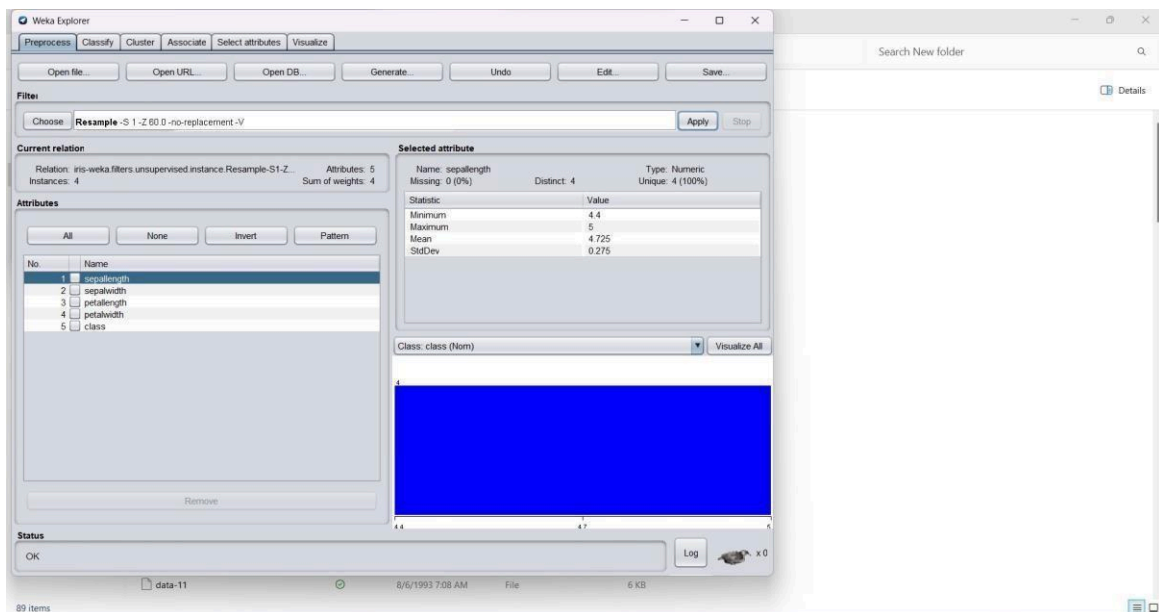


7. Sample dataset is filtered with 6 instances. Click on Save and save the file as train.arff in a desired location.

8. Click on Undo->click Resample from Filter pane, choose invertSelection – True, noReplacement- True, Sample size percentage-60, then click Ok->click Apply.

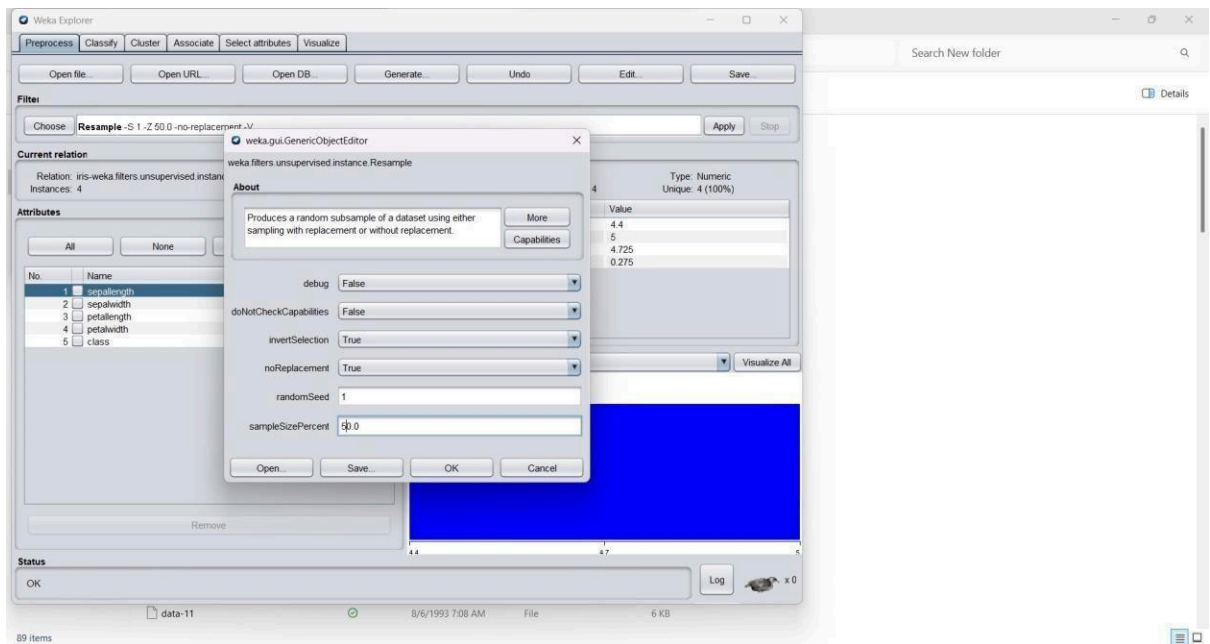


9. 4 instances are created.

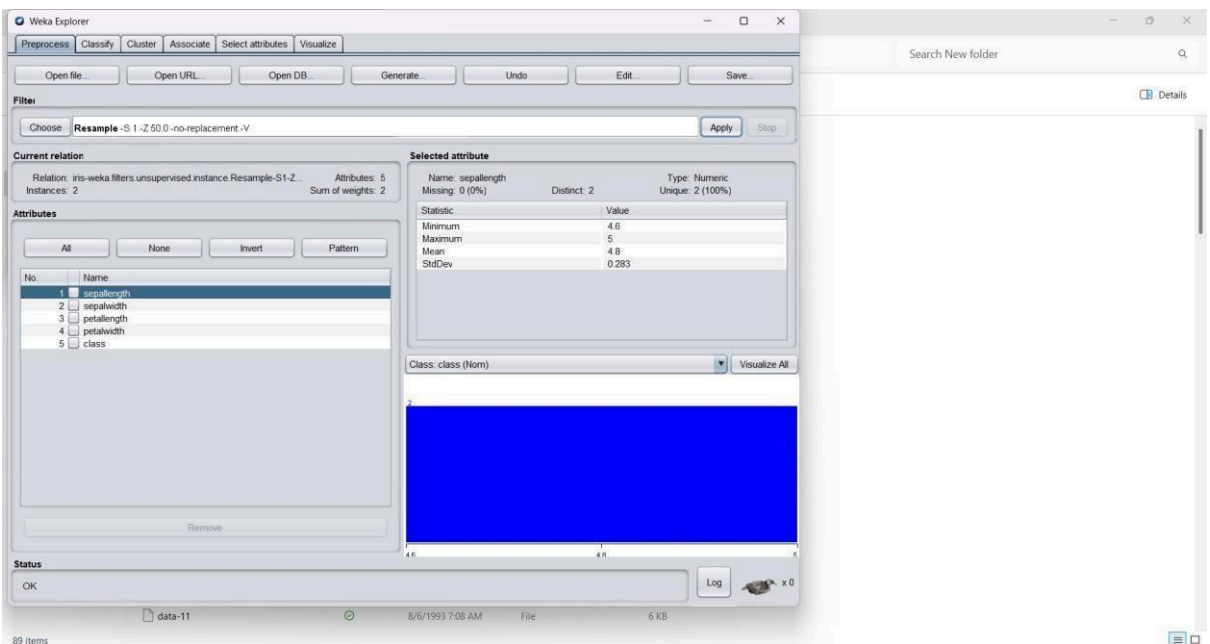


10. The 50% of remaining sample data set(4instances) is splitted into cross validated data and test data.

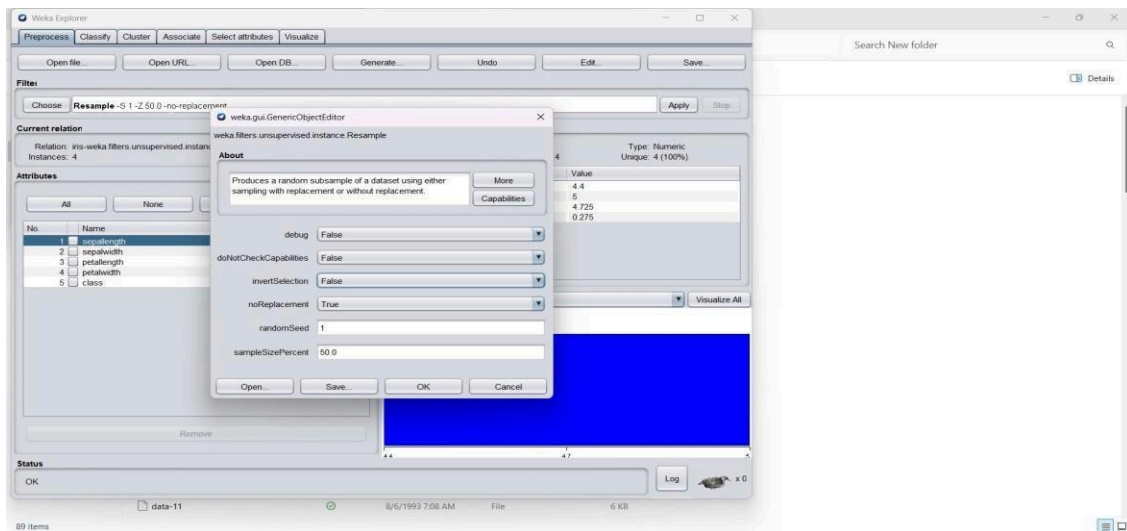
11. Click Resample from Filter pane, select invertSelection – True, noReplacement- True, Sample size percentage-50, then click Ok->click Apply.



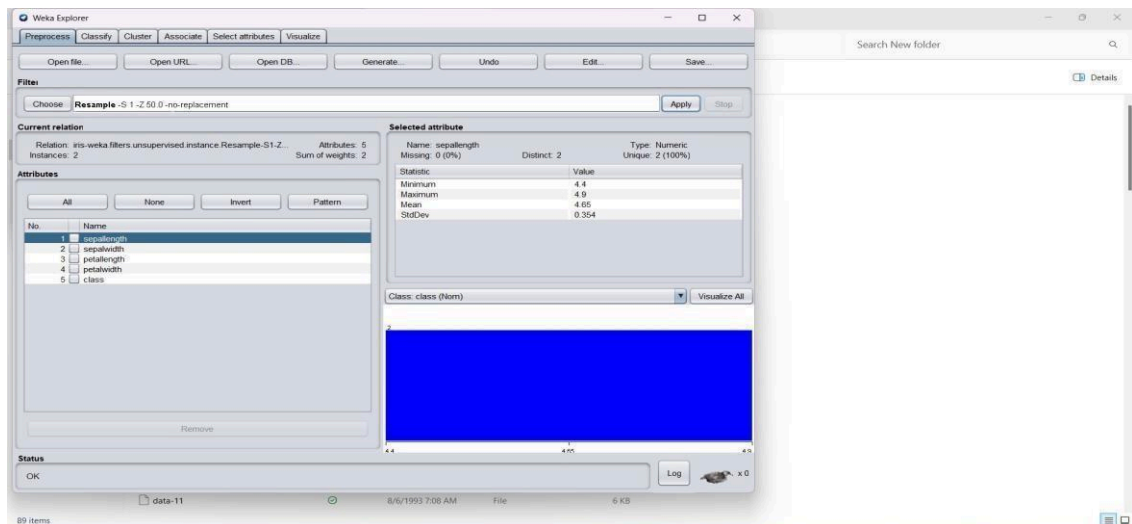
12. 2 instances are created. Click on Save and save the file as cv.arff in a desired location.



13. Click on Undo->click Resample from Filter pane, select, choose invertSelection – False, noReplacement- True, Sample size percentage-50, then click Ok->click Apply.

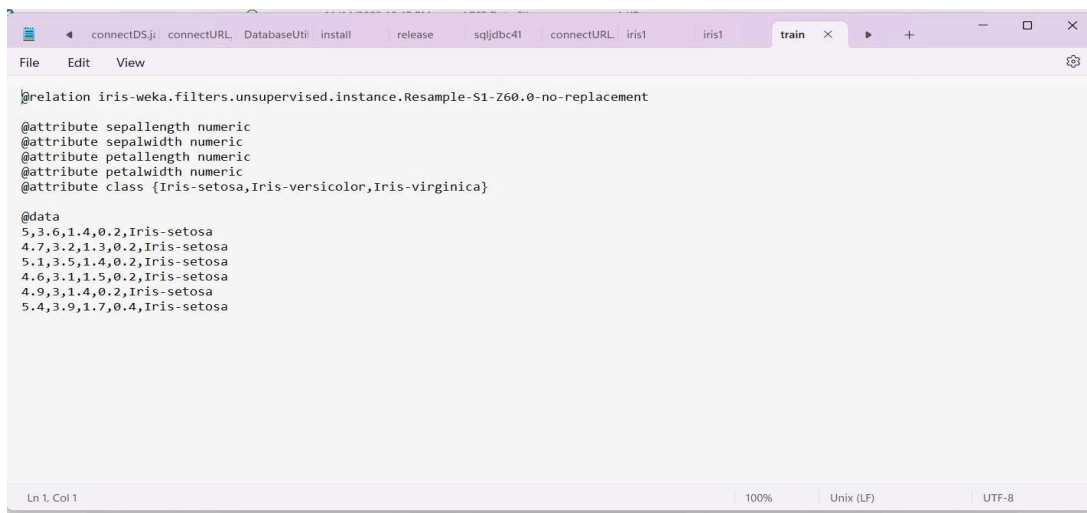


13. 2 instances are created. Click on Save and save the file as test.arff in a desired location.



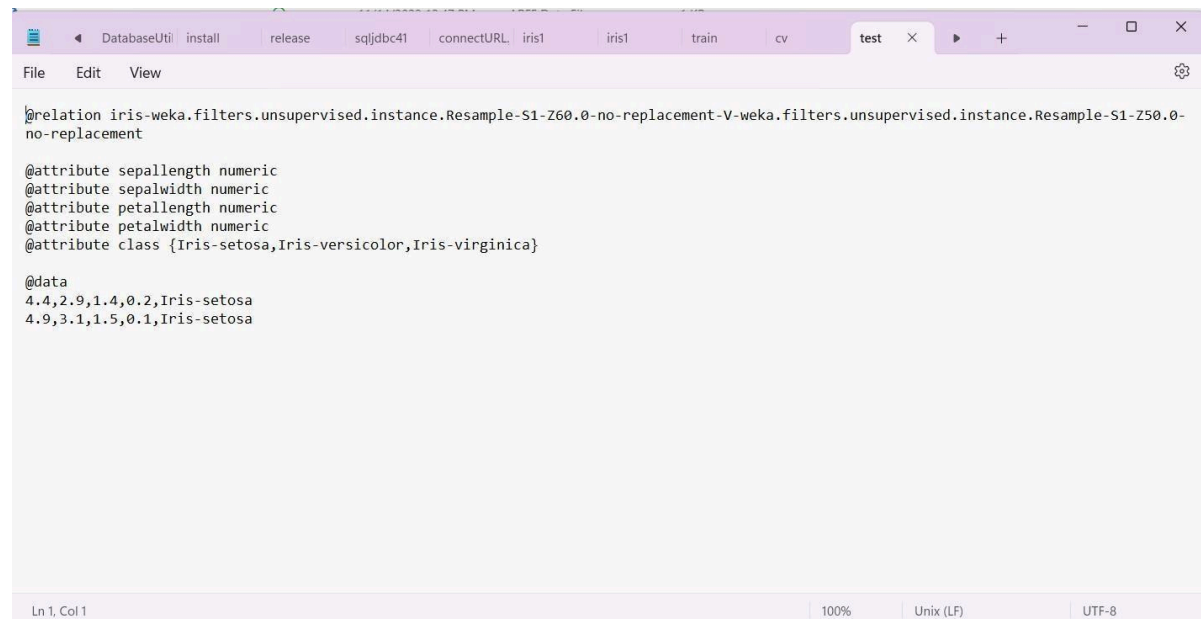
14. To check the instances created in training, cross validating and testing data right click and open the files in notepad, the files will be:

train.arff:



cv.arff:

test.arff:



```
@relation iris-weka.filters.unsupervised.instance.Resample-S1-Z60.0-no-replacement-V-weka.filters.unsupervised.instance.Resample-S1-Z50.0-no-replacement

@attribute sepallength numeric
@attribute sepalwidth numeric
@attribute petallength numeric
@attribute petalwidth numeric
@attribute class {Iris-setosa,Iris-versicolor,Iris-virginica}

@data
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
```

CONCLUSION:

Thus, the data validation by splitting a data set into training, testing and cross validating instances was done successfully with WEKA.

EX NO 3:

PLAN THE ARCHITECTURE FOR REAL TIME APPLICATION

AIM:

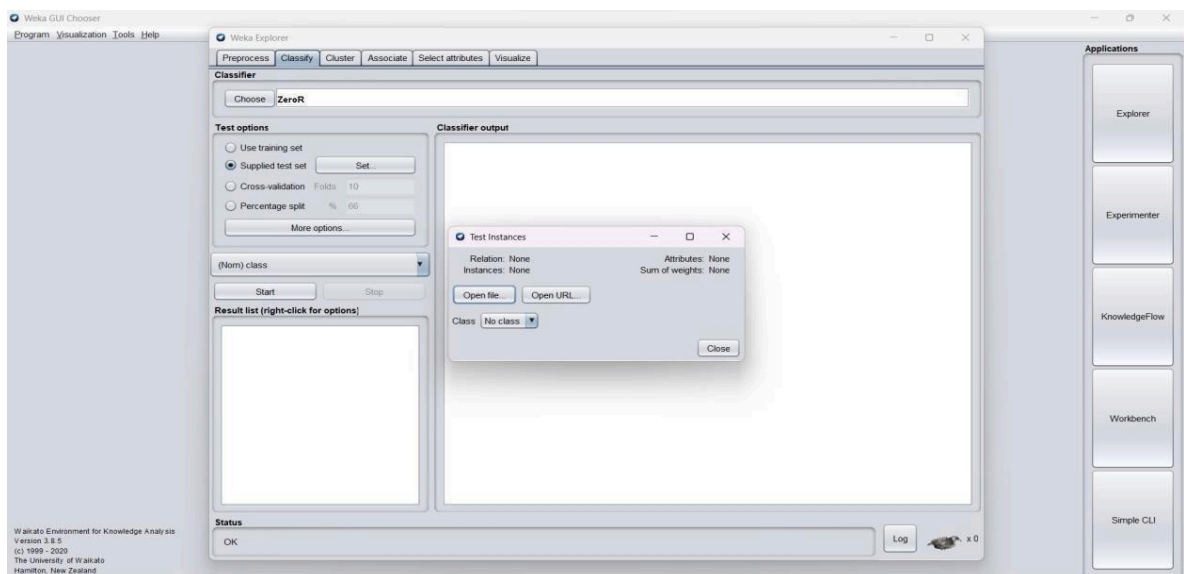
To design an architecture for classifying, cross validating, and splitting dataset into training and testing dataset and thereby computing the accuracy by applying decision tree rule.

Steps to plan the architecture:

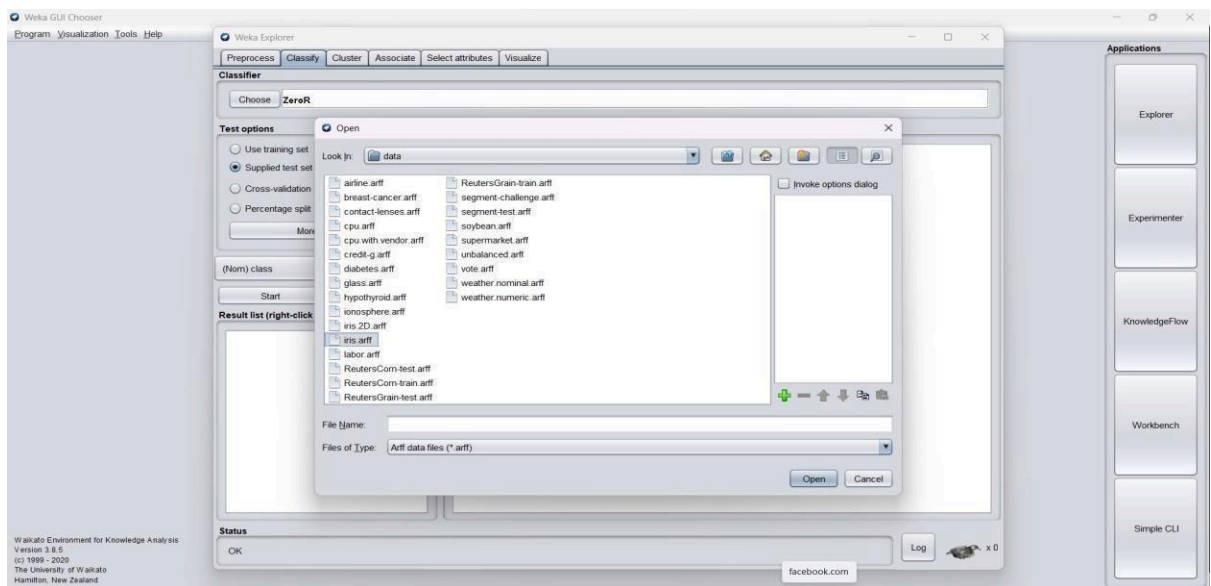
1. Define the problem
2. Data collection and preprocessing
3. Choose the appropriate Weka algorithms
4. Real-time data streaming
5. Model training and evaluation.
6. Integration and deployment
7. Monitoring and maintenance

PROCEDURE:

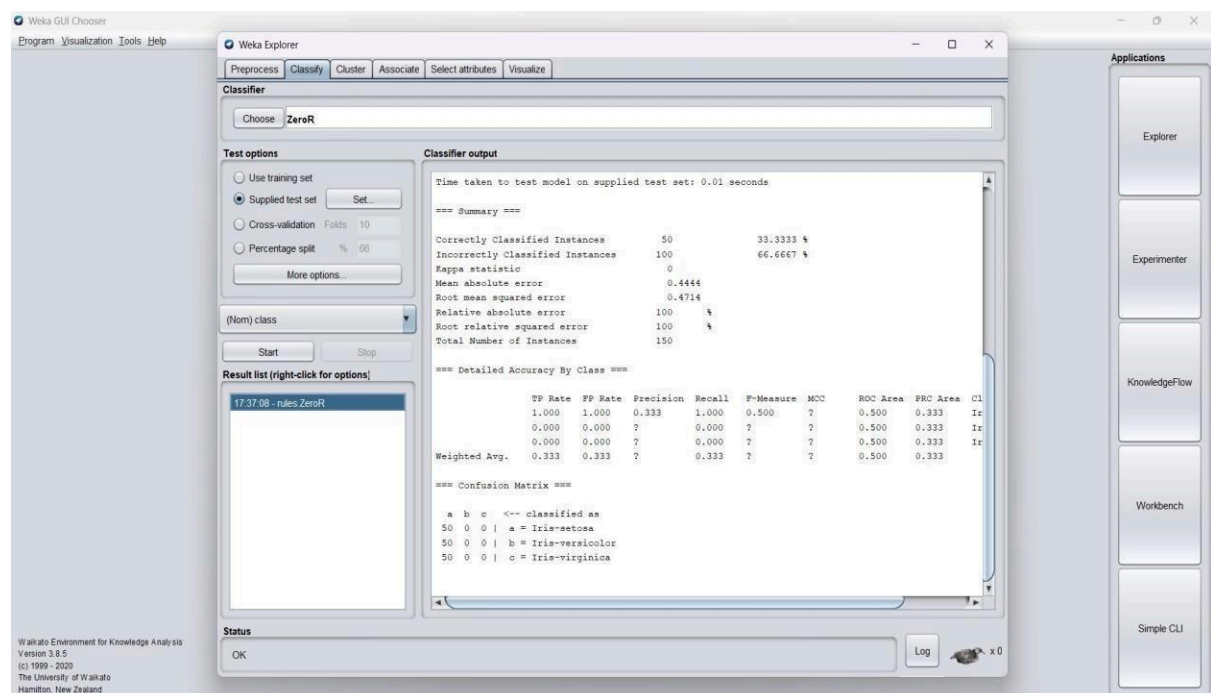
1. Open WEKA Tool.
2. Click on WEKA Explorer.
3. Click on Test option->Supplied Test set ->Set
4. Click on Open file



5. Load iris. arff data set from data set in WEKA,



6. Click on Classify and then Start option. The output will be:

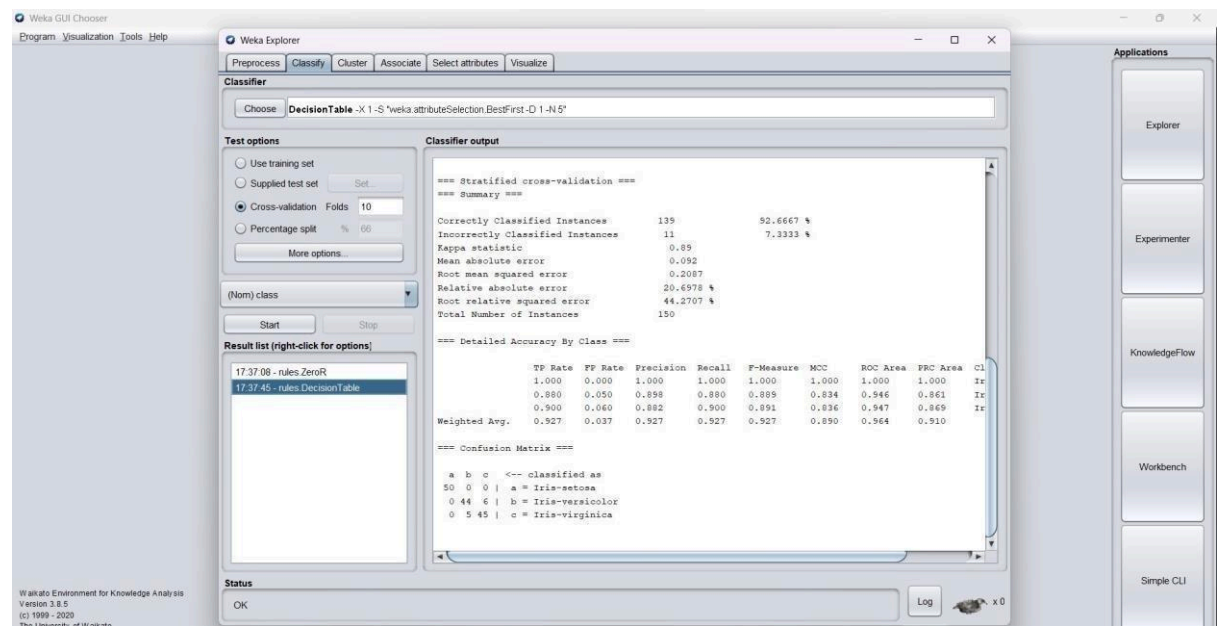


To improve the accuracy,

7. Click on Choose and select Decision table rule

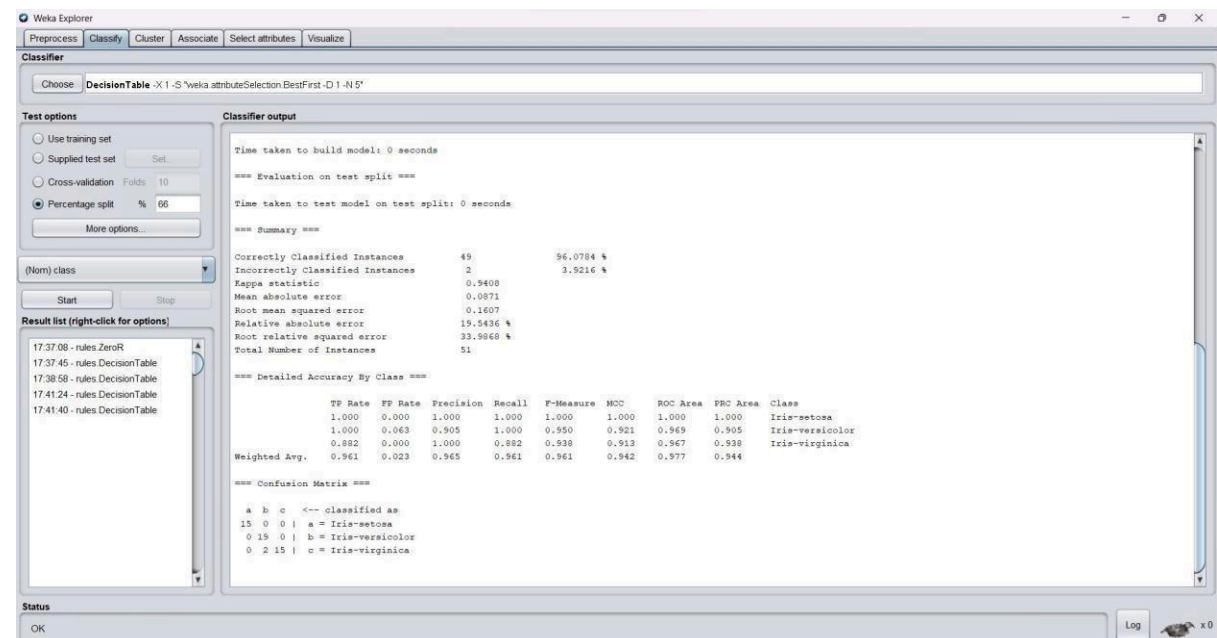
8. Choose Cross Validation folds ->10 from Test options and then click on Start.

The output will be:



To further improve the accuracy,

- Choose Percentage split % 66 option from Test option and Click on Start. The output will be:



CONCLUSION:

Thus, the architecture for classifying and testing a real time application (data set) was designed successfully.

EX NO: 4

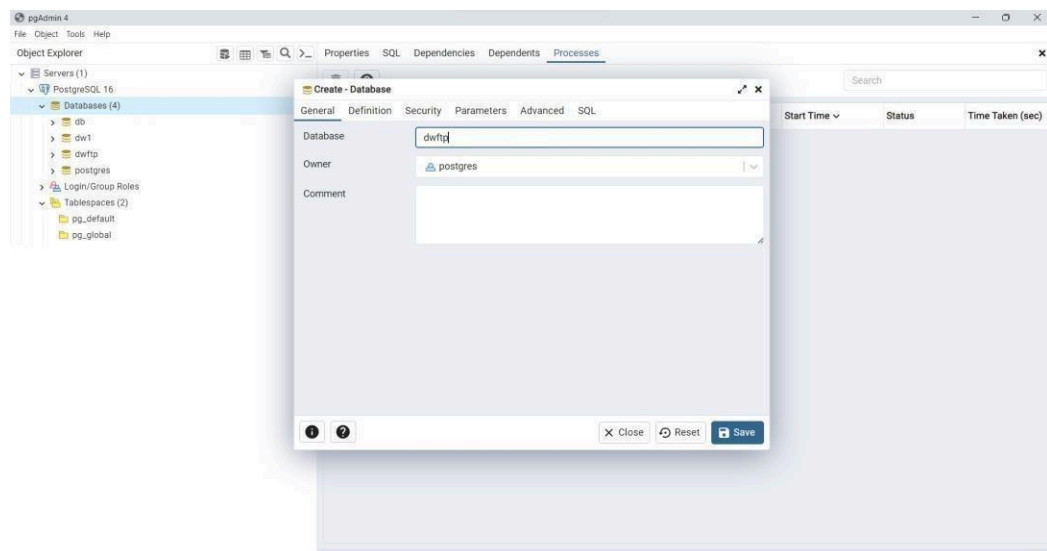
WRITE THE QUERY FOR SCHEMA DEFINITION

AIM:

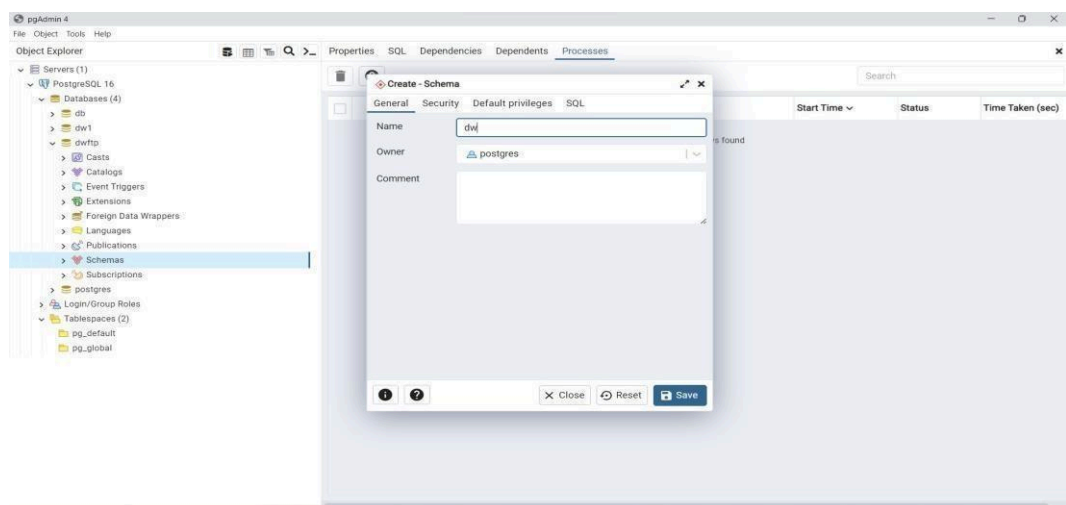
To write the query for schema definition using PostgreSQL tool.

PROCEDURE:

1. Click Start- AllPrograms -PostgreSQL 16 - Open pgAdmin4.
2. Click this icon, enter name, host and password as postgres.
3. Double click PostgreSQL 16.
4. Right click databases (1) and choose Create and type database name as dwftp and Save.



5. Double click dwftp and click schemas (1) - Right click and select Create and type schema name as dw and Save.



6. Double click dw- right click Tables -select Query Tool and run the queries for creating the tables:

- (1) location
- (2) phonerate
- (3) timedim
- (4) facts

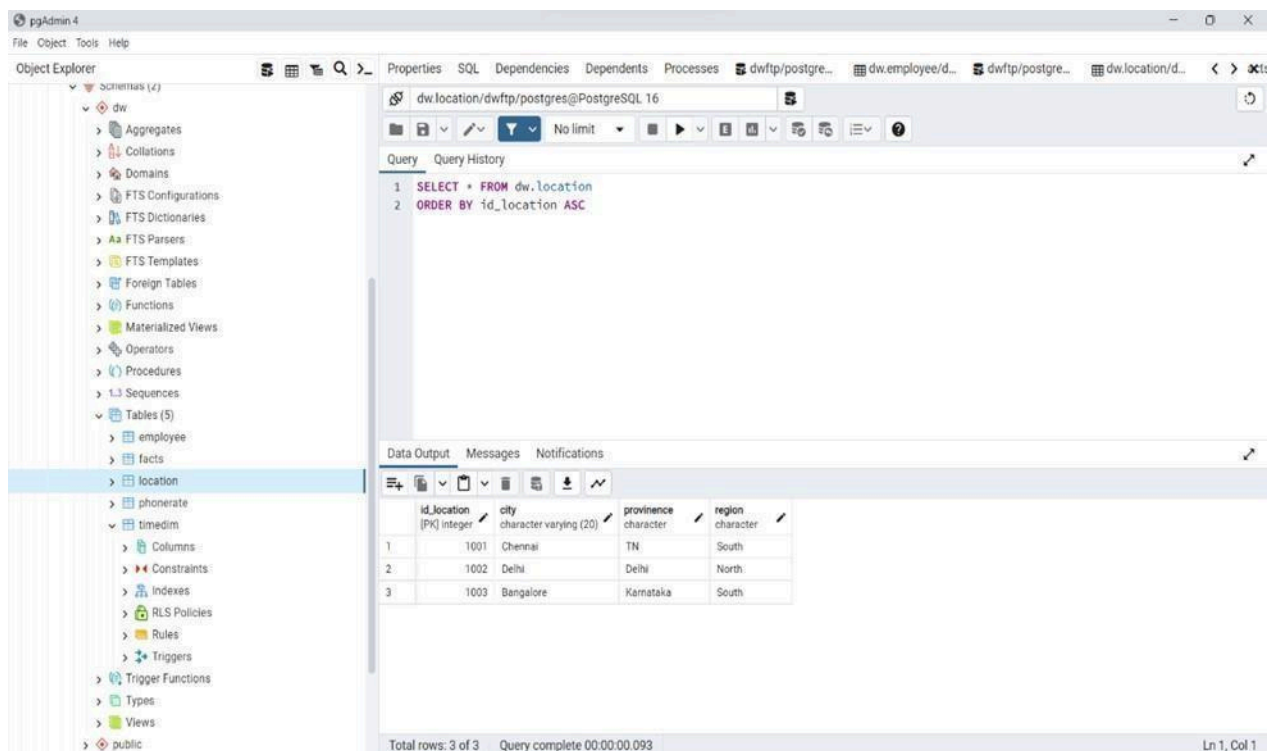
CREATE TABLE

dw.location (

**id_location integer NOT
NULL, city character
VARCHAR (20), province
VARCHAR (20),
region VARCHAR (20),
PRIMARY KEY
(id_location)**

);

To insert the values into the tables right click the table(location) and type the values displayed in Data Output pane.

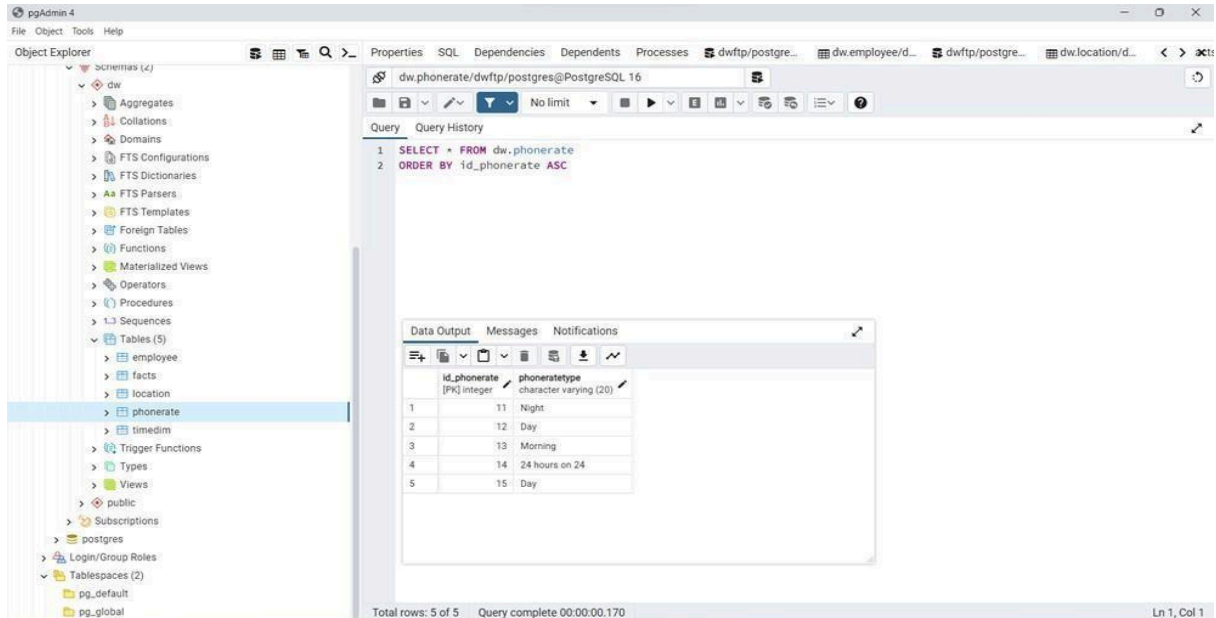


The screenshot displays the pgAdmin 4 interface. On the left, the Object Explorer shows the 'dw' schema with a table named 'location'. The table is selected, and its structure is shown in the Properties pane. The table has four columns: 'id_location' (integer, primary key), 'city' (character varying (20)), 'province' (character), and 'region' (character). The Data Output pane shows the results of a query: 'SELECT * FROM dw.location ORDER BY id_location ASC'. The results are as follows:

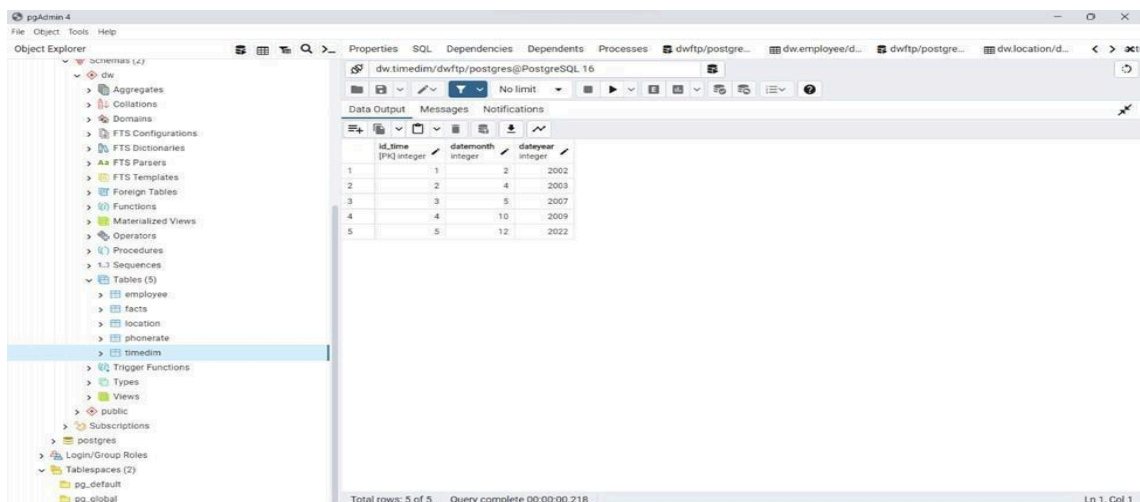
	id_location	city	province	region
1	1001	Chennai	TN	South
2	1002	Delhi	Delhi	North
3	1003	Bangalore	Karnataka	South

Total rows: 3 of 3 Query complete 00:00:00.093 Ln 1, Col 1


```
CREATE TABLE dw.phonerate(
ID_phoneRate INTEGER NOT NULL,
phoneRateType VARCHAR (20),
PRIMARY KEY(ID_phoneRate)
);
```



```
CREATE TABLE dw.timedim(
ID_time INTEGER NOT NULL,
dateMonth INTEGER NOT NULL,
dateYear INTEGER NOT NULL,
PRIMARY KEY(ID_time)
);
```



CREATE TABLE

dw.facts (

ID_time INTEGER NOT NULL,

ID_phoneRate INTEGER NOT NULL,

ID_location_Caller INTEGER NOT NULL,

ID_location_Receiver INTEGER NOT NULL,

Price FLOAT NOT NULL,

NumberOfCalls INTEGER NOT NULL,

PRIMARY KEY (ID_time, ID_phoneRate, ID_location_Caller,
ID_location_Receiver), FOREIGN KEY(ID_time) REFERENCES dw.timedim
(ID_time),

FOREIGN KEY(ID_phoneRate) REFERENCES dw.phonerate(ID_phoneRate),

FOREIGN KEY(ID_location_Caller) REFERENCES dw.location (ID_location),

FOREIGN KEY(ID_location_receiver) REFERENCES dw.location
(ID_location)

);

The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer displays the database structure, including schemas, tables, and constraints. The 'dw' schema is expanded, showing tables like 'facts', 'location', 'phonerate', and 'timedim'. The 'facts' table is selected. In the center, the SQL query editor shows the following query:

```
1 SELECT * FROM dw.facts
2 ORDER BY id_time ASC, id_phonerate ASC, id_location_caller ASC, id_location_receiver ASC
```

Below the query editor, the Data Output tab displays the results of the query. The results are shown in a table with 7 columns: id_time, id_phonerate, id_location_caller, id_location_receiver, price, and numberofcalls. The data is as follows:

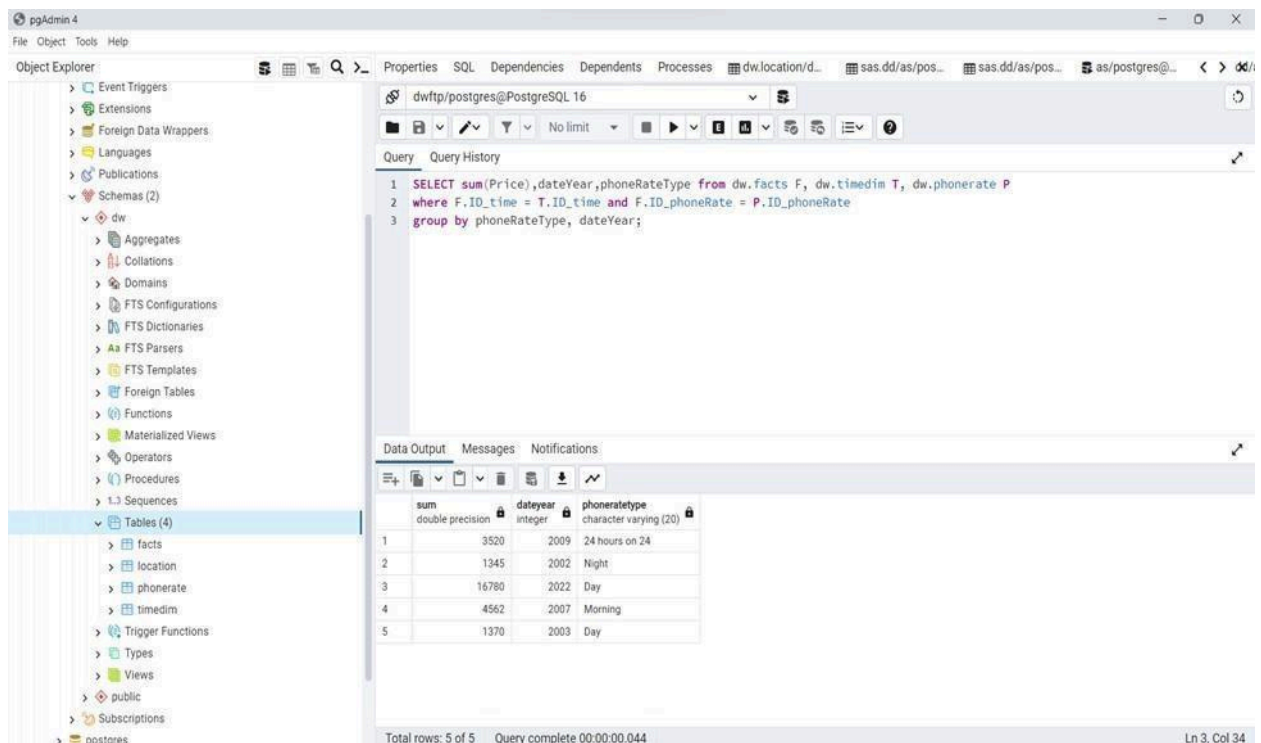
	id_time [PK] integer	id_phonerate [PK] integer	id_location_caller [PK] integer	id_location_receiver [PK] integer	price double precision	numberofcalls integer
1	1	11	1001	1002	1345	12
2	2	12	1002	1003	1370	32
3	3	13	1003	1001	4562	24
4	4	14	1001	1002	3520	15
5	5	15	1002	1003	16780	22

At the bottom, the status bar indicates 'Total rows: 5 of 5' and 'Query complete 00:00:00.388'.

- (1) To display sum of the prices, year and phone rate type(hint:use facts, timedim and phonerate tables, group by phoneratetype and dateyear)

```
SELECT sum(Price),dateYear,phoneRateType from dw.facts F, dw.timedim T, dw.phonerate P
where F.ID_time = T.ID_time and F.ID_phoneRate = P.ID_phoneRate
group by phoneRateType, dateYear;
```

OUTPUT:



The screenshot shows the pgAdmin 4 interface. The left pane displays the database structure, including the 'dw' schema and its tables. The central pane shows the SQL query: `SELECT sum(Price),dateYear,phoneRateType from dw.facts F, dw.timedim T, dw.phonerate P where F.ID_time = T.ID_time and F.ID_phoneRate = P.ID_phoneRate group by phoneRateType, dateYear;`. The bottom pane displays the query results in a table format.

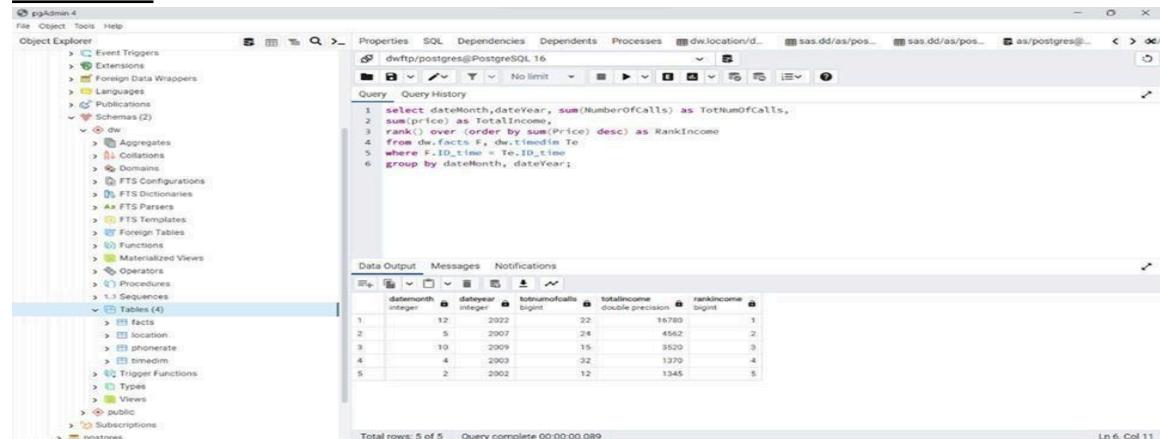
	sum double precision	dateyear integer	phoneratetype character varying (20)
1	3520	2009	24 hours on 24
2	1345	2002	Night
3	16780	2022	Day
4	4562	2007	Morning
5	1370	2003	Day

Total rows: 5 of 5 Query complete 00:00:00.044 Ln 3, Col 34

- (2) To display month, year, Total number of calls, Total Income and Rank Income:
(hint: use facts and timedim tables, Rename Sum of Number of calls as Total number of calls, Sum of Price as Total income, Rank(sum of Price) as Rank Income, group by datemonth and dateyear)

```
select dateMonth,dateYear, sum(NumberOfCalls) as TotNumOfCalls,
sum(price) as TotalIncome,
rank() over (order by sum(Price) desc) as RankIncome
from dw.facts F, dw.timedim Te
where F.ID_time = Te.ID_time
group by dateMonth, dateYear;
```

OUTPUT:



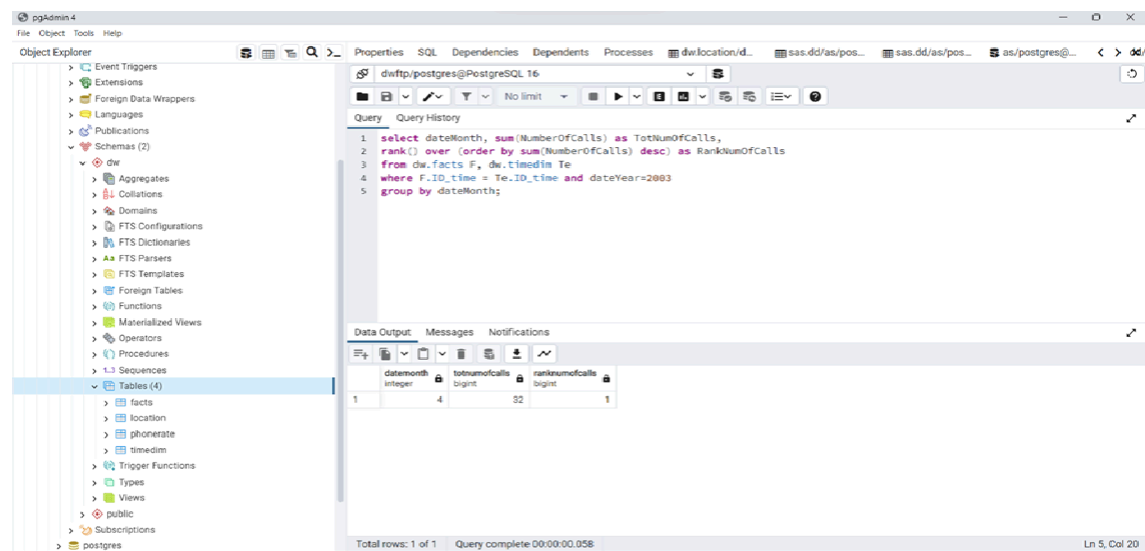
The screenshot shows the pgAdmin 4 interface. The left pane displays the database structure, including schemas, tables, and views. The central pane shows a SQL query that filters data for the year 2003 and ranks it by the number of calls. The bottom pane displays the query results in a table format.

dateMonth	dateYear	totnumofcalls	totalIncome	rankIncome
12	2002	22	16780	1
5	2007	24	4562	2
10	2009	15	3520	3
4	2003	32	1270	4
2	2002	12	1345	5

(3) To display month, total num of calls, Rank of num of calls where year=2003, grp by dateMonth.(hint: use facts and timedim tables, Rename Sum of Number of calls as TotNumOfCalls, Rank(sum of NumberOfCalls) as RankNumOfCalls)

```
select dateMonth, sum (NumberOfCalls) as TotNumOfCalls,  
rank () over (order by sum (NumberOfCalls) desc) as RankNumOfCalls  
from dw.facts F, dw.timedim Te  
where F.ID_time = Te.ID_time and dateYear=2003  
group by dateMonth;
```

OUTPUT:



The screenshot shows the pgAdmin 4 interface. The left pane displays the database structure. The central pane shows a SQL query that filters data for the year 2003 and ranks it by the number of calls. The bottom pane displays the query results in a table format.

dateMonth	totnumofcalls	ranknumofcalls
4	32	1

CONCLUSION:

Thus, the query for schema definition using PostgreSQL tool was executed successfully.

EX NO: 5

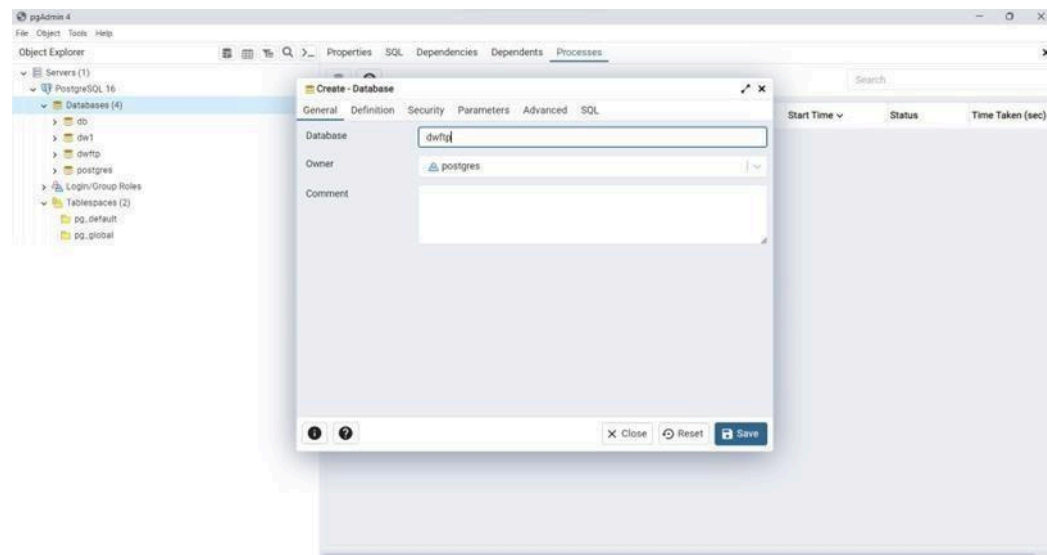
DESIGN DATA WARE HOUSE FOR REAL TIME APPLICATION

AIM:

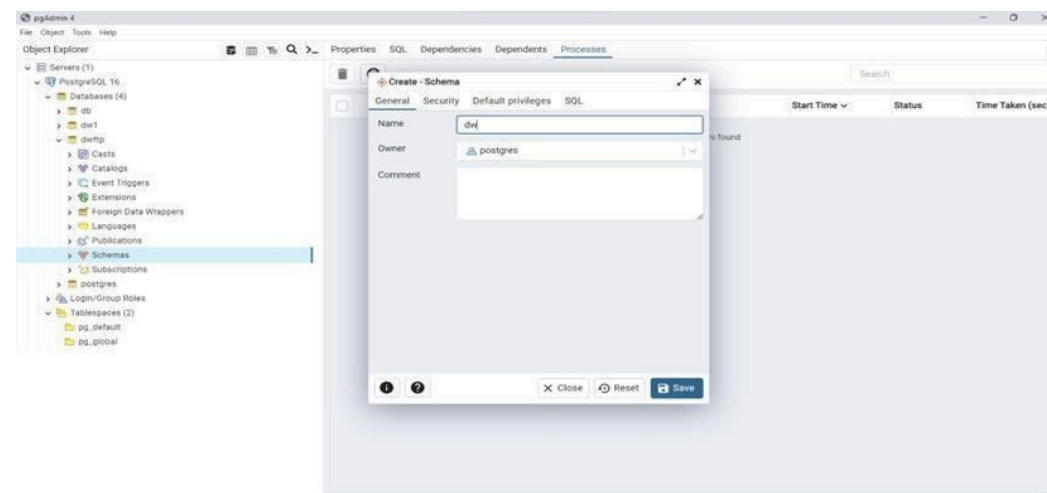
To design a data ware house for a real time application using PostgreSQL tool.

PROCEDURE:

1. Click Start- AllPrograms -PostgreSQL 16 - Open pgAdmin4.
2. Click this icon, enter name, host and password as postgres.
3. Double click PostgreSQL 16.
4. Right click databases (1) and choose Create and type database name as dwftp and Save.

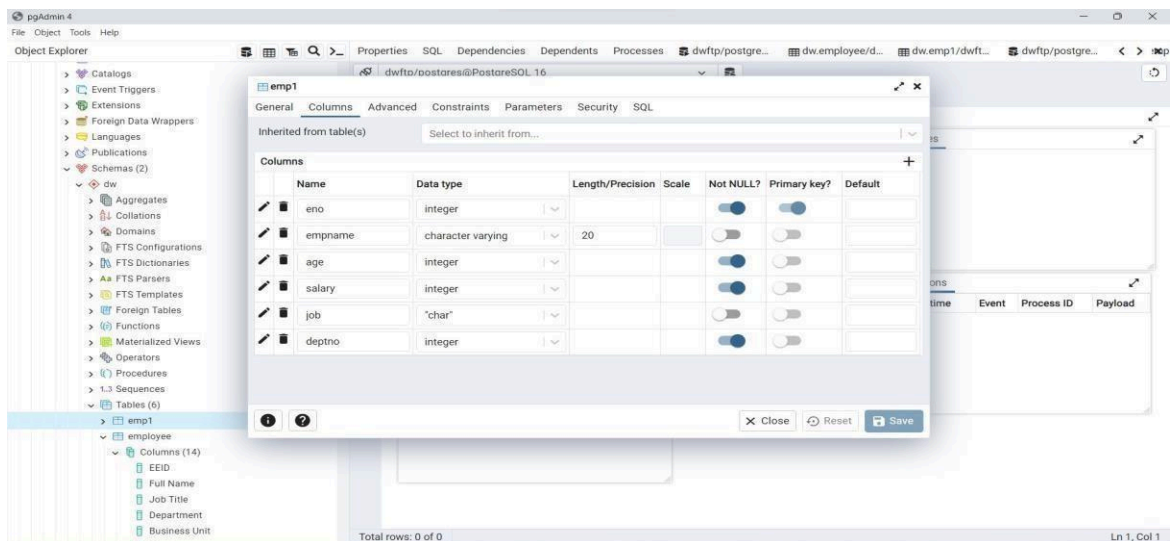


5. Double click dwftp and click schemas (1) - Right click and select Create and type schema name as dw and Save.

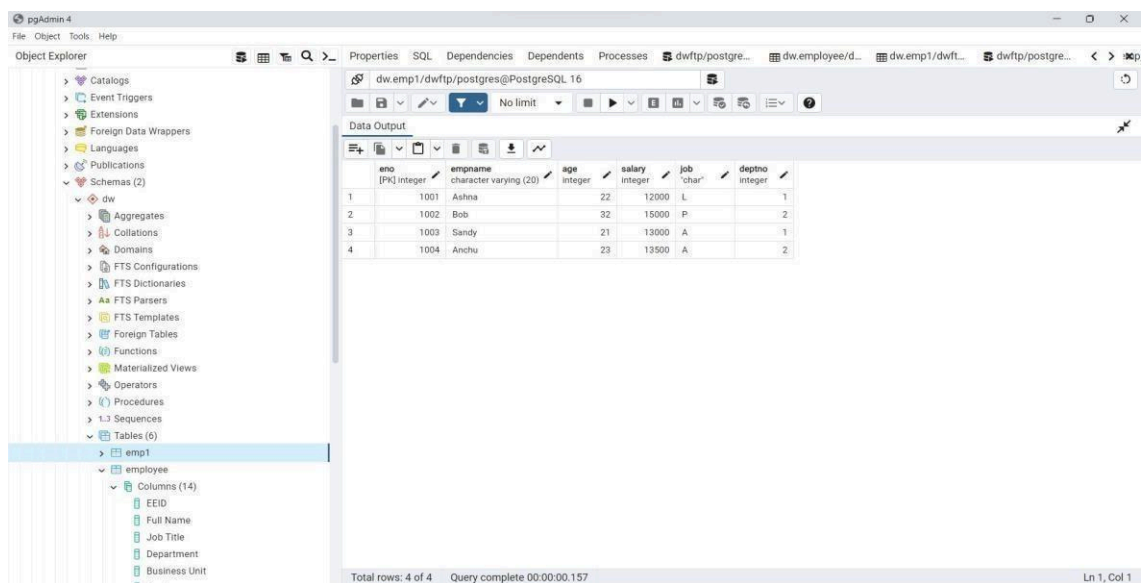


6. Double click dw- right click Tables -select Table create table for Employee as emp1 with columns:

- Eno integer PRIMARY KEY
- Empname VARCHAR(20)
- Age integer
- Salary integer
- Job Char
- Deptno integer and Save



7. To insert values into table right click the table emp1 select View/edit data -> All rows and then add the number of rows, insert values by double clicking each attribute and Save.

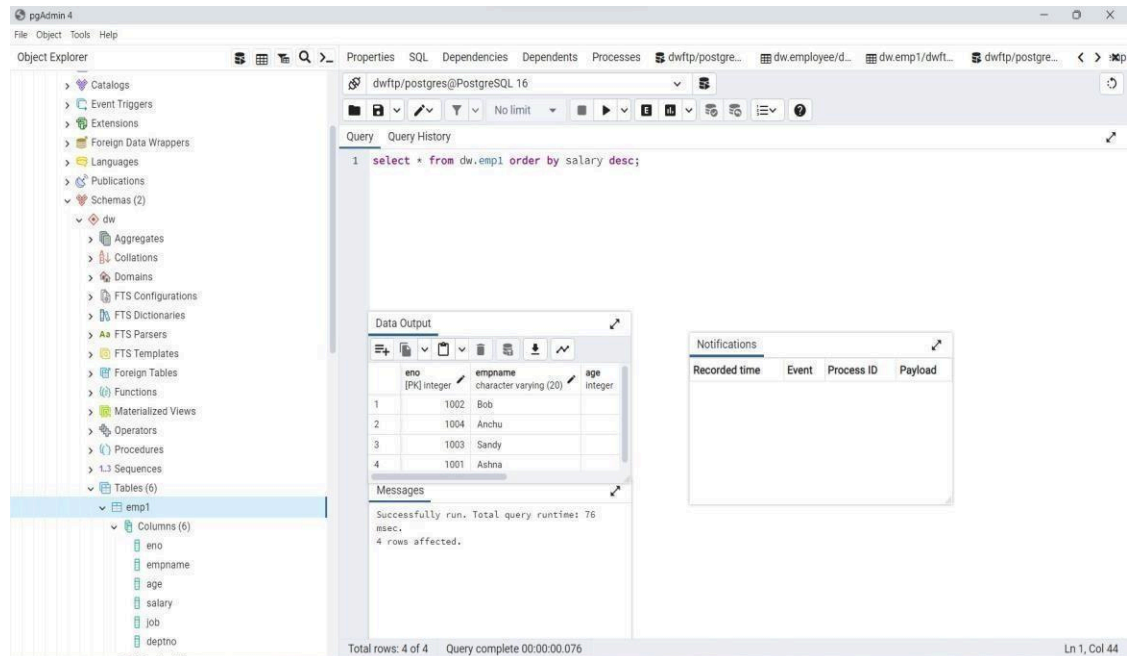


8. Right click on table emp1, select Query tool and perform the query operations:

(a) To list the records in the emp1 table order by salary in descending order.

select * from dw.emp1 order by salary

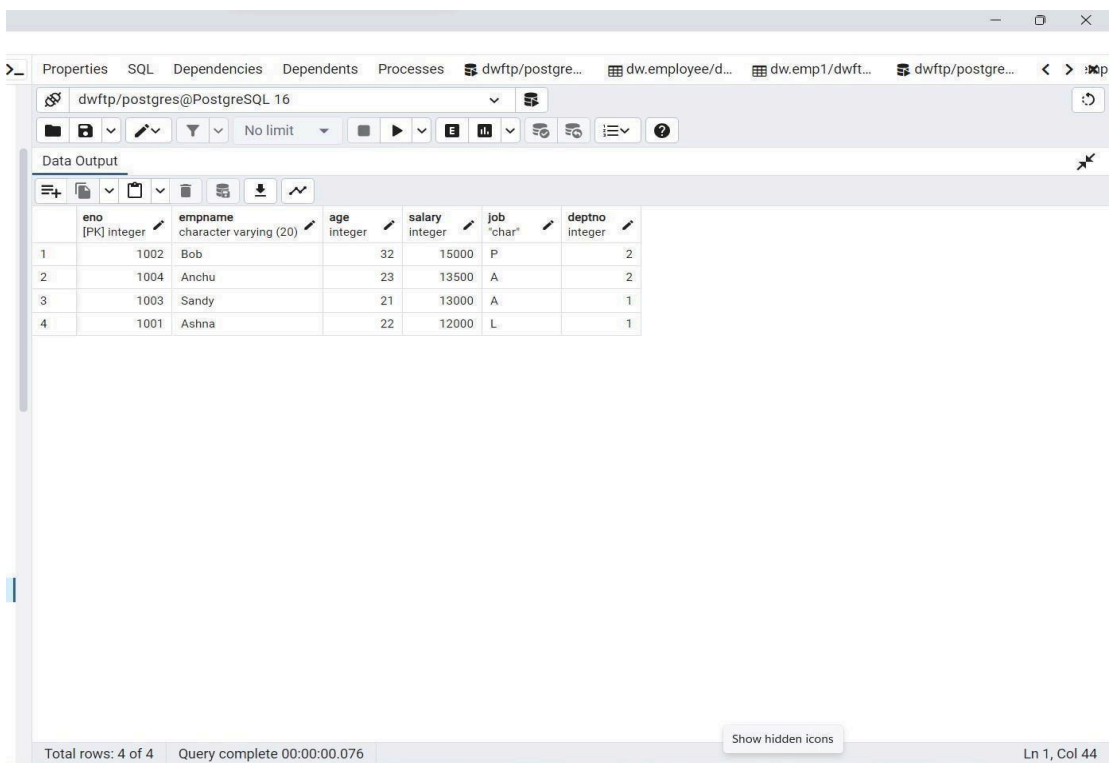
desc; OUTPUT:



The screenshot shows the pgAdmin 4 interface. The left pane displays the database structure, with the 'emp1' table selected under the 'dw' schema. The central pane shows the SQL query: `select * from dw.emp1 order by salary desc;`. The right pane displays the 'Data Output' window, which contains the following data:

eno	empname	age
1002	Bob	
1004	Anchu	
1003	Sandy	
1001	Ashna	

Below the data output, a 'Messages' window shows the status: 'Successfully run. Total query runtime: 76 msec. 4 rows affected.' The status bar at the bottom indicates 'Total rows: 4 of 4' and 'Query complete 00:00:00.076'.



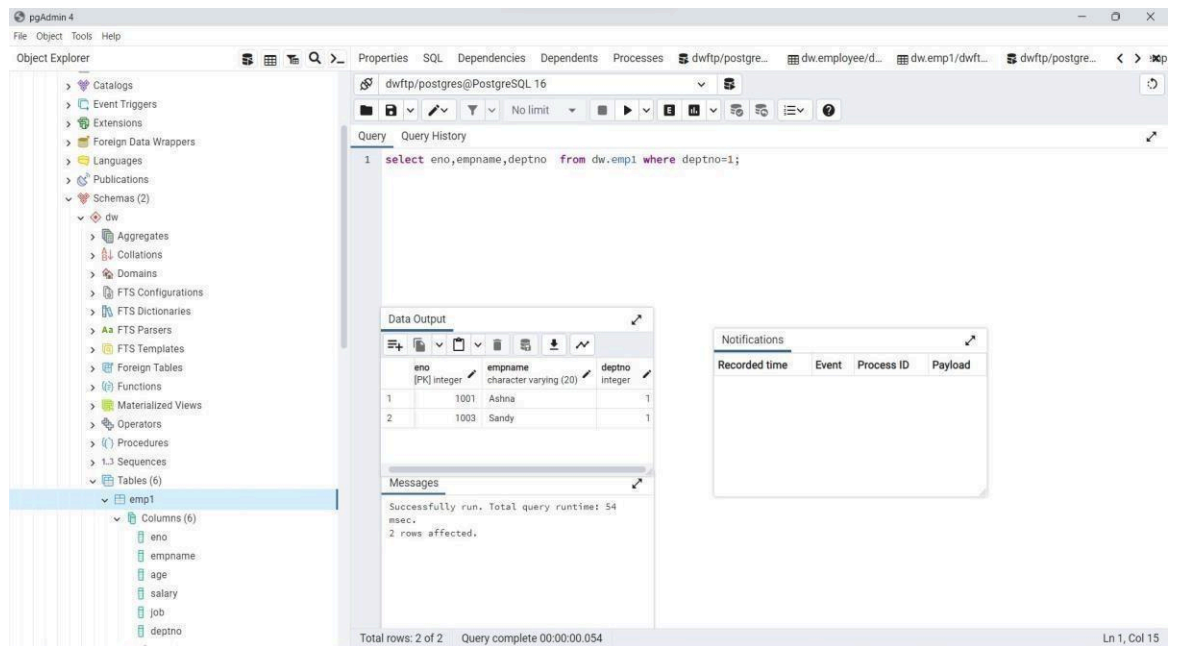
The screenshot shows the pgAdmin 4 interface. The left pane displays the database structure, with the 'emp1' table selected under the 'dw' schema. The central pane shows the SQL query: `select eno,empname,deptno from dw.emp1 where deptno=1;`. The right pane displays the 'Data Output' window, which contains the following data:

eno	empname	age	salary	job	deptno
1002	Bob	32	15000	P	2
1004	Anchu	23	13500	A	2
1003	Sandy	21	13000	A	1
1001	Ashna	22	12000	L	1

Below the data output, a 'Messages' window shows the status: 'Successfully run. Total query runtime: 76 msec. 4 rows affected.' The status bar at the bottom indicates 'Total rows: 4 of 4' and 'Query complete 00:00:00.076'.

(b) Display only those employees whose deptno is 1.

select eno,empname,deptno from dw.emp1 where deptno=1;



CONCLUSION:

Thus, the data ware house for a real time application such as Employee details was designed successfully.

EX NO: 6

ANALYSE THE DIMENSIONAL MODELING

AIM:

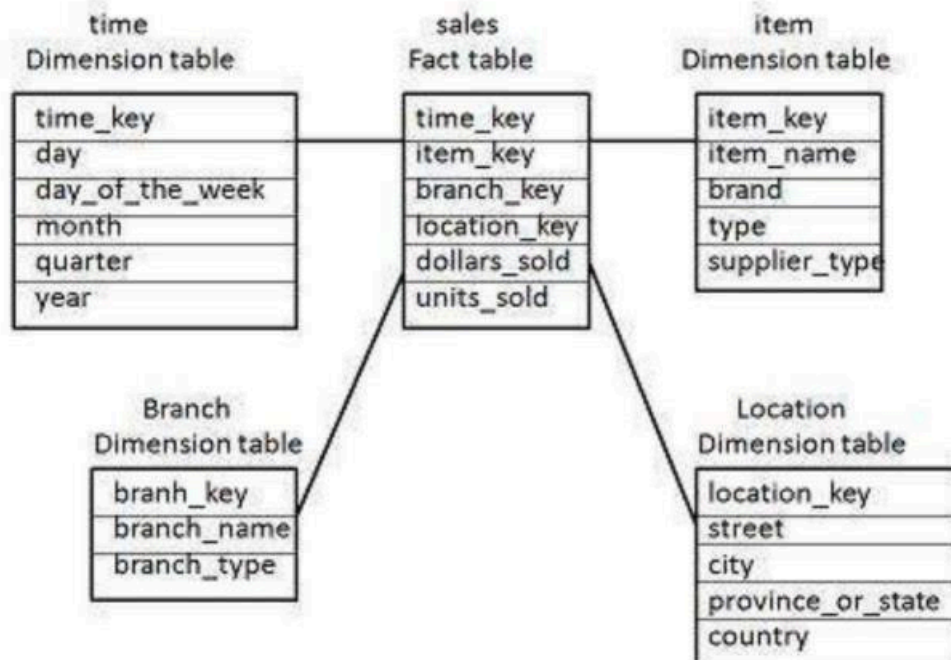
To analyse the dimensional modelling for applications and designing the star, snowflake and fact constellation schemas.

PROCEDURE:

Multidimensional schema is defined using Data Mining Query Language (DMQL). The two primitives, cube definition and dimension definition, can be used for defining the data warehouses and data marts.

Star Schema

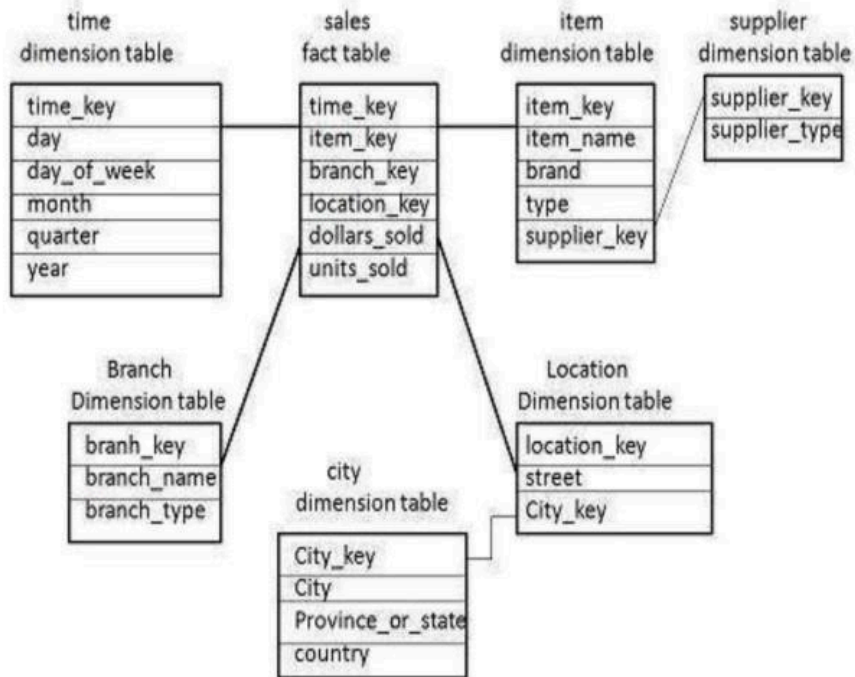
- Each dimension in a star schema is represented with only one-dimension table.
- This dimension table contains the set of attributes.
- The following diagram shows the sales data of a company with respect to the four dimensions, namely time, item, branch, and location.
- There is a fact table at the centre. It contains the keys to each of four dimensions.
- The fact table also contains the attributes, namely dollars sold and units sold.



Snowflake Schema

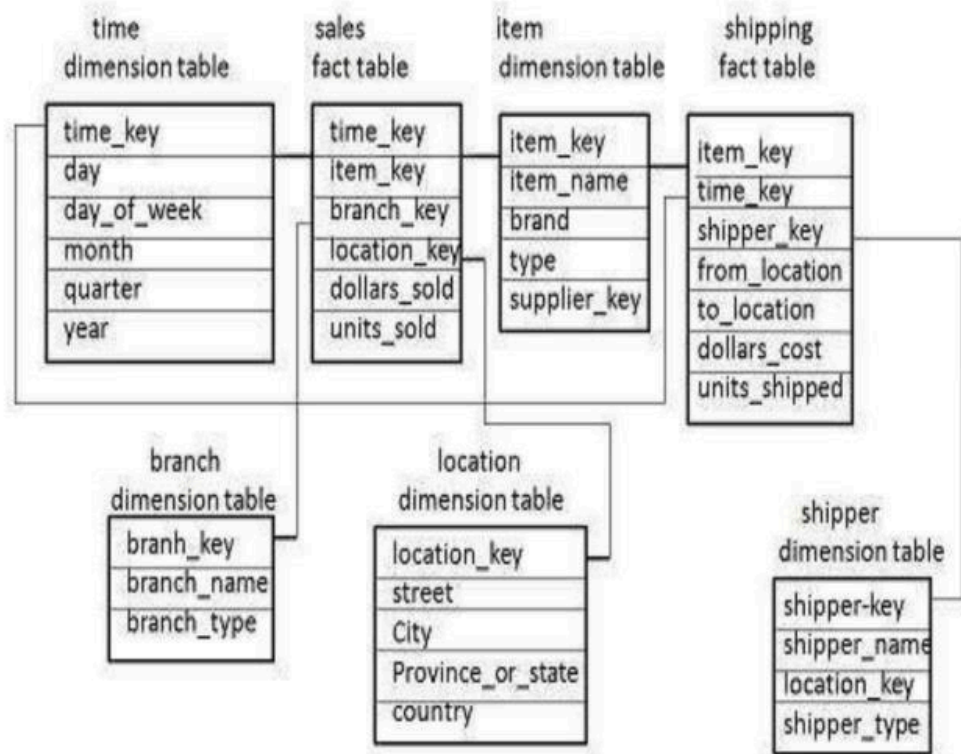
- Some dimension tables in the Snowflake schema are normalized.
- The normalization splits up the data into additional tables.

- Unlike Star schema, the dimensions table in a snowflake schema is normalized. For example, the item dimension table in star schema is normalized and split into two dimension tables, namely item and supplier table.
- Now the item dimension table contains the attributes item_key, item_name, type, brand, and supplier-key.
- The supplier key is linked to the supplier dimension table. The supplier dimension table contains the attributes supplier_key and supplier_type.



Fact Constellation Schema

- A fact constellation has multiple fact tables. It is also known as galaxy schema.
- The following diagram shows two fact tables, namely sales and shipping.
- The sales fact table is same as that in the star schema.
- The shipping fact table has the five dimensions, namely item_key, time_key, shipper_key, from_location, to_location.
- The shipping fact table also contains two measures, namely dollars sold and units sold.
- It is also possible to share dimension tables between fact tables. For example, time, item, and location dimension tables are shared between the sales and shipping fact table.



CONCLUSION:

Thus, the dimensional modelling was analysed and the star, snowflake and fact constellation schemas were designed successfully.

AIM:

To study the different OLAP operations.

OLAP Operations:

Since OLAP servers are based on multidimensional view of data, we will discuss OLAP operations in multidimensional data.

The different OLAP operations are:

1. Roll-up (Drill-up)
2. Drill-down
3. Slice and dice
4. Pivot (rotate)

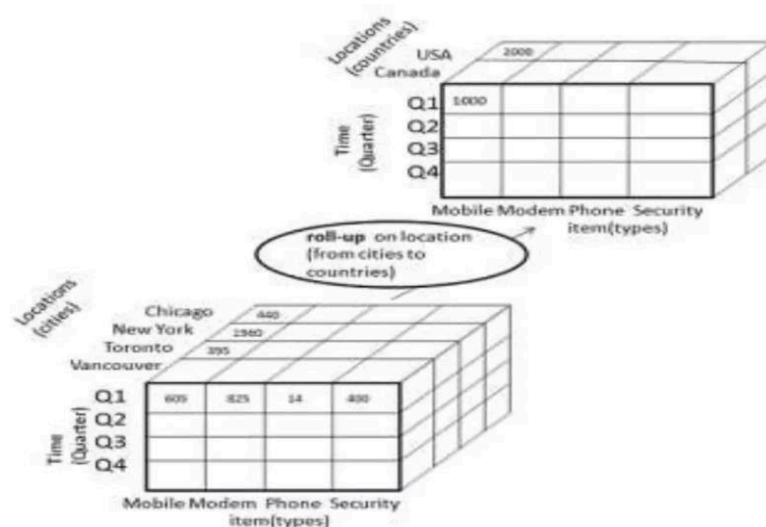
1. Roll-up (Drill-up):

Roll-up performs aggregation on a data cube in any of the following ways:

- By climbing up a concept hierarchy for a dimension
- By dimension reduction

Roll-up is performed by climbing up a concept hierarchy for the dimension location.

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

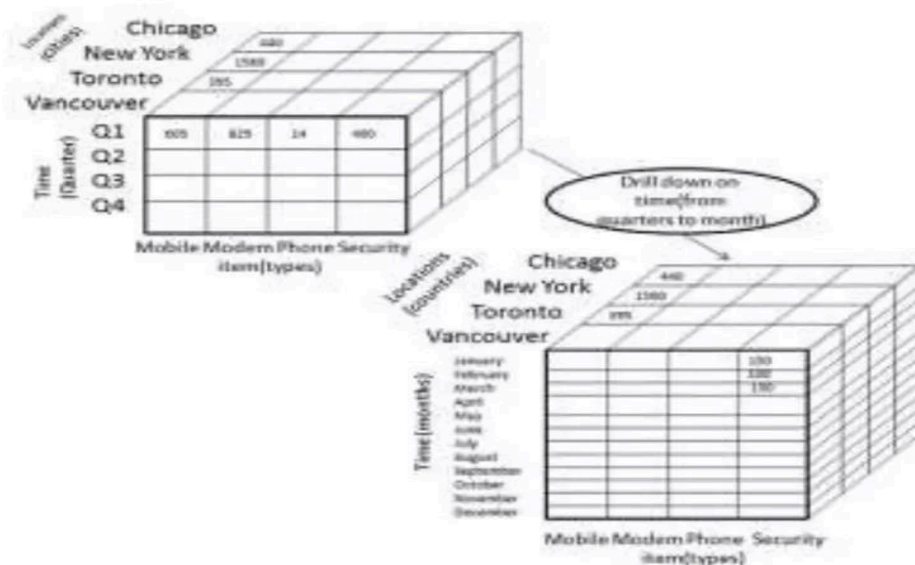


2. Drill-down:

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

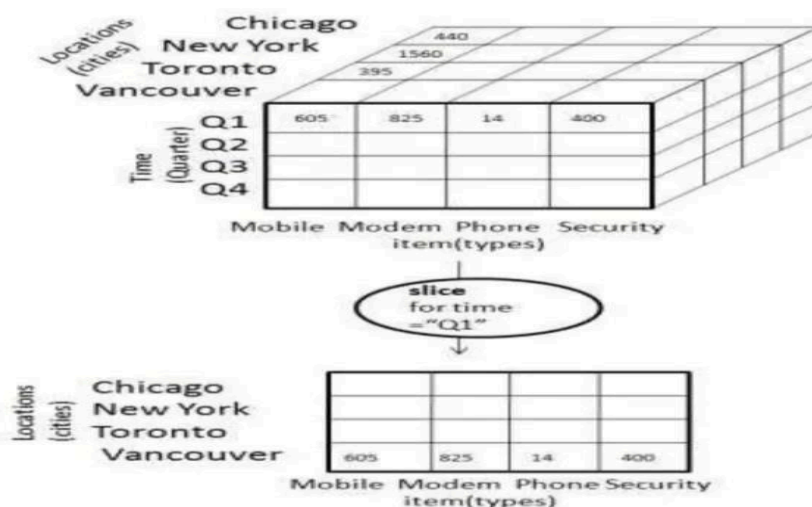
- By stepping down a concept hierarchy for a dimension.
- By introducing a new dimension.

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



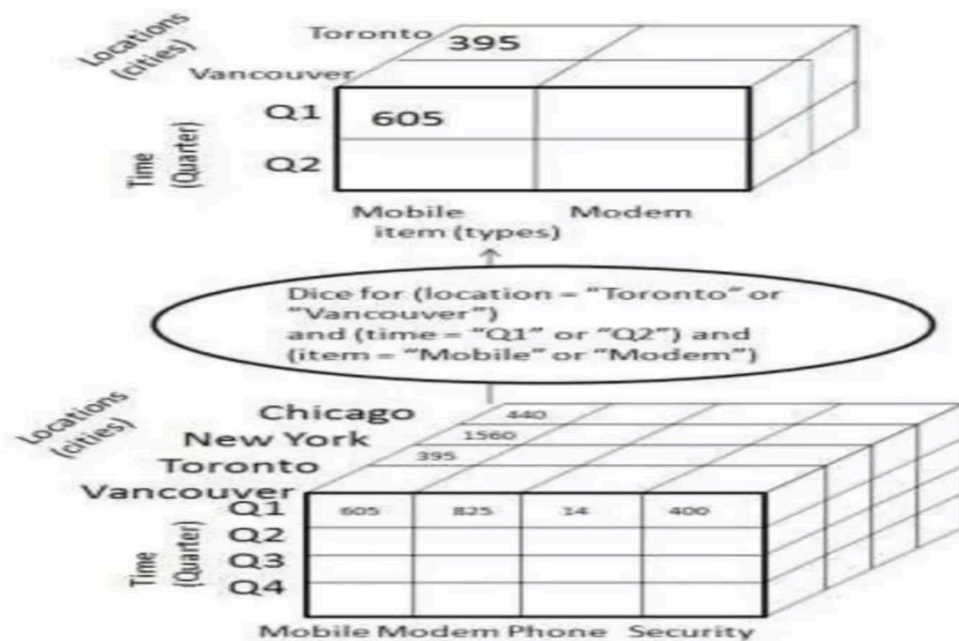
3. Slice:

The slice operation selects one particular dimension from a given cube and provides a new sub-cube.



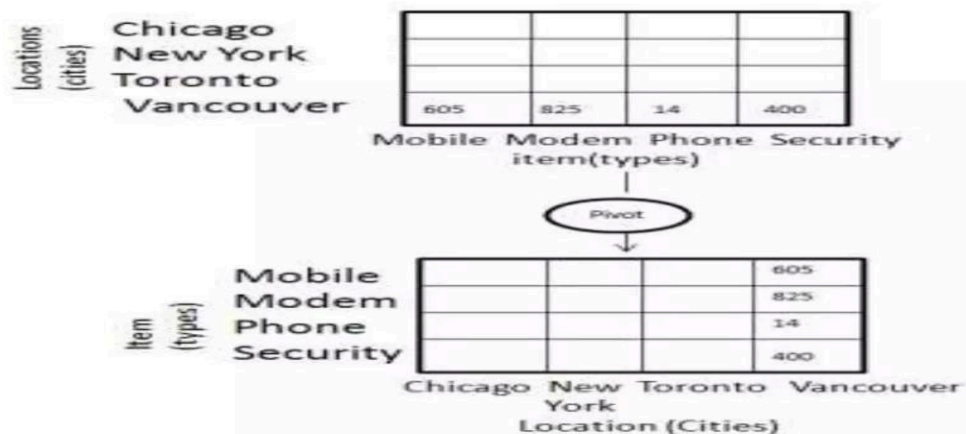
4. Dice:

Dice selects two or more dimensions from a given cube and provides a new sub-cube.



5. Pivot (rotate):

The pivot operation is also known as rotation. It rotates the data axes in view in order to provide an alternative presentation of data.



CONCLUSION:

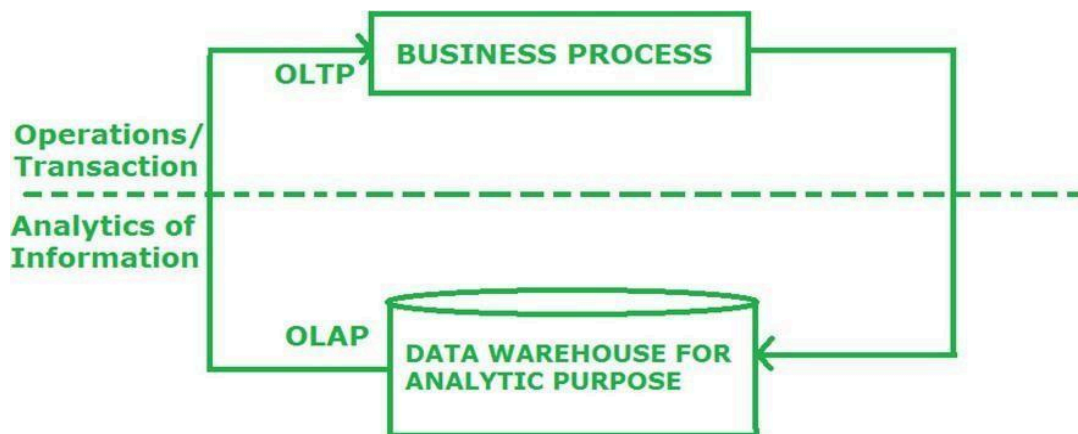
Thus, the case study of different OLAP operations was done successfully.

AIM:

To study the various aspects of OLTP operations.

OLTP:

Online Transaction Processing (OLTP) is a type of database system that is optimized for high-speed data processing and rapid transaction execution in real-time. OLTP is an operational system that supports transaction-oriented applications in a 3-tier architecture. OLTP is basically focused on query processing, maintaining data integrity in multi-access environments as well as effectiveness that is measured by the total number of transactions per second.

**OLTPArchitecture:**

An OLTP system uses a 3-tier architecture:

- The presentation layer: This layer is the front end or user interface where transactions are generated.
- The logic layer: Also called the business logic or application layer, this layer processes transaction data based on predefined rules.
- The data or data store layer: This is where each transaction and related data are stored and indexed. It includes the database management system (DBMS) and the database server.

Key characteristics of OLTP:

OLTP systems have four critical characteristics:

1. Fast query processing:

OLTP systems are used for high-speed query processing. They handle transactions in real-time, meaning that transactions are executed as soon as they are received, with little or no delay.

2. High concurrency:

OLTP systems use algorithms that allow many concurrent users to perform transactions simultaneously. Each transaction is executed independently of the others and in the proper order.

3. ACID properties:

To ensure data integrity, consistency, and reliability, OLTP transactions comply with the ACID (Atomicity, Consistency, Isolation, Durability) properties. These are:

- **Atomicity:** Transactions in OLTP systems are atomic, meaning they are treated as a single, indivisible unit of work. If any part of a transaction fails, the entire transaction is rolled back, so the database is left in its original state.
- **Consistency:** OLTP databases are designed to maintain data consistency, despite failures or errors. Every transaction change table in predefined and predictable ways, and the database will always be in a valid state.
- **Isolation:** Transactions in OLTP systems are isolated from each other. This means that when multiple users read and write data simultaneously, they are executed independently. It keeps the database in a consistent state.
- **Durability:** When a transaction is successfully executed in an OLTP database, the changes to the data are permanent and will survive any subsequent failures or errors, like system crashes or power outages.

4. Support for simple transactions:

OLTP systems support specific applications or business processes like order processing, inventory management, or customer service. They are typically not used for complex queries, data analysis, or reporting tasks.

OLTP use cases:

Online transaction processing systems are used in applications where the primary goal is to manage and process many transactions in real time. Some standard use cases include:

E-commerce systems:

E-commerce applications use OLTP to manage customer orders, payments, and inventory in real time. This allows them to provide exceptional customer service, boost

customer loyalty, and drive growth. For example, an OLTP database can help maintain up-to-date and accurate inventory data, allowing e-commerce companies to fulfill orders promptly.

Banking and financial services:

Banks and other financial services use OLTP to process financial transactions in real-time, manage customer data, and enable customers to make deposits, withdraw money, transfer funds, and access other services quickly. OLTP solutions for financial transaction systems, like online banking, must have secure and reliable data management practices, multi-currency support, and custom reporting options. ATMs are the most common example of an OLTP system used in the financial industry.

Reservation systems:

OLTP drives online reservation systems in the travel and hospitality industry. It is used in applications that manage bookings, flights, payments, and related services. An online transaction processing system in the travel and hospitality industry must integrate with external systems, like airline reservation or car rental applications, and have multi-language and multi-currency support. OLTP also enables efficient customer data management and helps provide personalized recommendations that create seamless experiences for travellers.

Customer relationship management (CRM):

Customer relationship management (CRM) platforms use OLTP to manage customer data, interactions, and transactions. An OLTP system can centralize customer data and record interactions across multiple channels, like phone calls, emails, and chat messages. OLTP helps CRM applications automate many sales and marketing processes, including lead generation and campaign management, so companies can focus on nurturing customer relationships and increasing sales.

Designing an effective OLTP system:

There are three key factors to consider when building an OLTP solution:

Best practices for schema design

A schema outlines how data is organized in a relational database. In OLTP, the database schema is designed to process high data volumes. Here are some best practices when creating an OLTP schema:

- **Normalization:** Normalization involves breaking down a large table into smaller, more manageable tables to minimize data duplication. This ensures data consistency and helps maintain data integrity.

- Choose appropriate indexes: Indexing is a method used to provide quick access to database files. Indexed data can speed up query processing and improve performance. This involves identifying the most frequently used queries and creating indexes on the relevant columns.
- Use the correct keys: Keys are used to uniquely identify the rows in a table. They establish relationships between different tables and ensure that there are no duplicate rows in the database. There are three types of keys:
 - * Primary key
 - * Composite key
 - * Foreign key
- Monitor and optimize: Regularly monitor and optimize the schema to ensure it meets performance metrics and can handle the required transaction volumes.

Scalability consideration:

Scalability determines the OLTP system's ability to handle increasing transaction volumes as the business grows. Factors that improve scalability are:

1. Sharding (horizontal scaling): Horizontal scaling involves adding more servers or nodes to the database cluster to distribute the workload and improve performance.
2. Vertical scaling: Vertical scaling involves increasing the capacity of the hardware or infrastructure that the database is running on. This can include adding more memory, CPUs, or storage to the server.
3. Cloud-based solutions: Cloud-based solutions, such as Amazon RDS, Azure SQL Database, and Google Cloud SQL, are scalable and highly available OLTP databases that don't need infrastructure maintenance.

Transaction and concurrency management:

Transaction and concurrency management are critical components of OLTP solutions. They ensure database consistency and prevent conflicts between multiple users accessing and modifying data. ACID compliance is the first consideration for transaction management in OLTP. Here are three other crucial mechanisms:

- Locking: Locking is a technique that ensures that multiple transactions do not access the same data simultaneously. Locks can be implemented at the database or table level. They can be either shared or exclusive. Shared locks allow multiple transactions to read the same data at the same time, while exclusive locks prevent other transactions from accessing the data until the lock is released.

- Isolation level: An isolation level defines how transactions interact and how they see changes made by other transactions. Most OLTP systems offer four different isolation levels - Read Uncommitted, Read Committed, Repeatable Read, and Serializable.
- Deadlock handling: Deadlocks occur when two or more transactions are waiting for each other to release locked resources and cannot proceed. Deadlock prevention techniques, like timeout mechanisms and prioritizing transactions, are used to detect and resolve deadlocks before they occur.

CONCLUSION:

Thus, the case study of Online Transaction Processing was done successfully.

Ex.no:9 CASE STUDY ON DATA WAREHOUSE TESTING

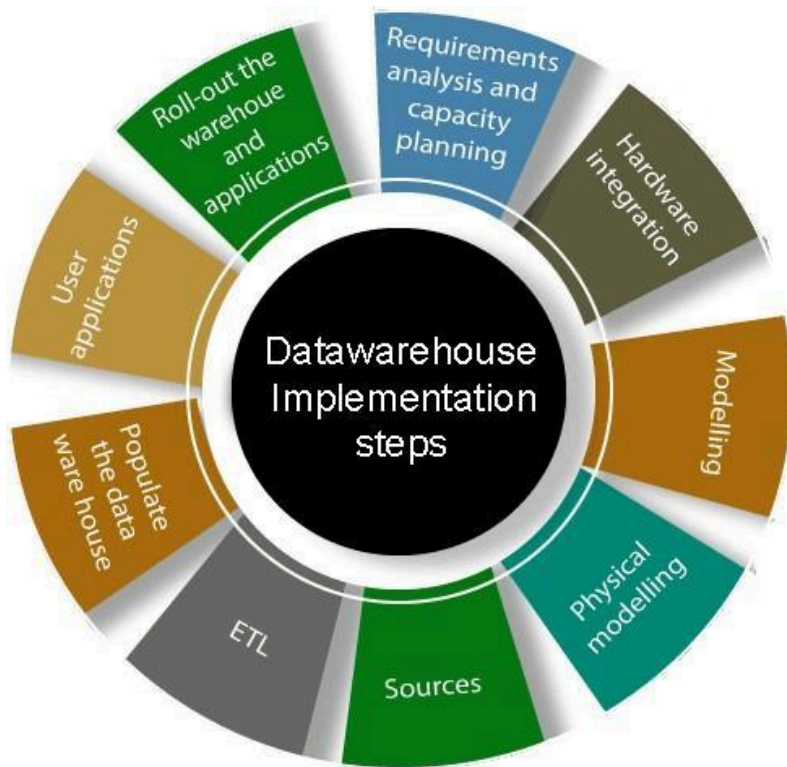
AIM : To study about various implementation in data warehouse testing.

DATA WAREHOUSE TESTING

Organizations are focusing testing on the ETL (extraction, transformation, load) process, business intelligence infrastructures, and applications that rely on data warehouses. Different stages of the data warehouse implementation, source data profiling, data warehouse design, ETL development, data loading and transformations, and so on, require the testing team's participation and expertise. A key element contributing to the success of the data warehouse solution is the ability of the test team to plan, design, and execute a set of effective tests that help identify multiple issues. A data warehouse implementation needs end-to-end testing. The QA team must test loads at key points, from the identification of source data to report and portal functions. All data-load programs and the resulting data loads should be verified throughout the end-to-end QA process. Testing data and systems systematically for errors, bugs, and inconsistencies before production is vital.

There are various implementation in data warehouses which are as follows

1. **Requirements analysis and capacity planning:** The first process in data warehousing involves defining enterprise needs, defining architectures, carrying out capacity planning, and selecting the hardware and software tools. This step will contain be consulting senior management as well as the different stakeholder.
2. **Hardware integration:** Once the hardware and software has been selected, they require to be put by integrating the servers, the storage methods, and the user software tools.
3. **Modeling:** Modelling is a significant stage that involves designing the warehouse schema and views. This may contain using a modeling tool if the data warehouses are sophisticated.
4. **Physical modeling:** For the data warehouses to perform efficiently, physical modeling is needed. This contains designing the physical data warehouse organization, data placement, data partitioning, deciding on access techniques, and indexing.
5. **Sources:** The information for the data warehouse is likely to come from several data sources. This step contains identifying and connecting the sources using the gateway, ODBC drives, or another wrapper.
6. **ETL:** The data from the source system will require to go through an ETL phase. The process of designing and implementing the ETL phase may contain defining a suitable ETL tool vendors and purchasing and implementing the tools. This may contains customize the tool to suit the need of the enterprises.
7. **Populate the data warehouses:** Once the ETL tools have been agreed upon, testing the tools will be needed, perhaps using a staging area. Once everything is working adequately, the ETL tools may be used in populating the warehouses given the schema and view definition.
8. **User applications:** For the data warehouses to be helpful, there must be end-user applications. This step contains designing and implementing applications required by the end-users.
9. **Roll-out the warehouses and applications:** Once the data warehouse has been populated and the end-client applications tested, the warehouse system and the operations may be rolled out for the user's community to use.



Implementation Guidelines:



1. **Build incrementally:** Data warehouses must be built incrementally. Generally, it is recommended that a data marts may be created with one particular project in mind, and once it is implemented, several other sections of the enterprise may also want to implement similar systems. An enterprise data warehouses can then be implemented in an iterative manner allowing all data marts to extract information from the data warehouse.

2. **Need a champion:** A data warehouses project must have a champion who is active to carry out considerable researches into expected price and benefit of the project. Data warehousing projects requires inputs from many units in an enterprise and therefore needs to be driven by someone who is needed for interacting with people in the enterprises and can actively persuade colleagues.

3. **Senior management support:** A data warehouses project must be fully supported by senior management. Given the resource-intensive feature of such project and the time they can take to implement, a warehouse project signal for a sustained commitment from senior management.

4. **Ensure quality:** The only record that has been cleaned and is of a quality that is implicit by the organizations should be loaded in the data warehouses.

5. **Corporate strategy:** A data warehouse project must be suitable for corporate strategies and business goals. The purpose of the project must be defined before the beginning of the projects.

Business plan: The financial costs (hardware, software, and peopleware), expected advantage, and a project plan for a data warehouses project must be clearly outlined and understood by all stakeholders. Without such understanding, rumors about expenditure and benefits can become the only sources of data, subversion the projects.

7. **Training:** Data warehouses projects must not overlook data warehouses training requirements. For a data warehouses project to be successful, the customers must be trained to use the warehouses and to understand its capabilities.

Adaptability: The project should build in flexibility so that changes may be made to the data warehouses if and when required. Like any system, a data warehouse will require to change, as the needs of an enterprise change.

9. **Joint management:** The project must be handled by both IT and business professionals in the enterprise. To ensure that proper communication with the stakeholder and which the project is the target for assisting the enterprise's business, the business professional must be involved in the project along with technical professionals.

CONCLUSION:

Thus the case study of data warehouse testing was done successfully.

CONTENT BEYOND SYLLABUS

Clustering rule process on dataset using simple k- means Algorithm

Aim: This experiment illustrates the use of simple k-mean clustering with Weka explorer. The sample data set used for this example is based on the student data available in ARFF format. This document assumes that appropriate pre-processing has been performed.

Steps :

Step 1: Run the Weka explorer and load the data file student.arff in pre-processing interface.

Step 2: In order to perform clustering select the 'cluster' tab in the explorer and click on the Choose button. This step results in a dropdown list of available clustering algorithms.

Step 3: In this case we select 'simple k-means'.

Step 4: Next click in text button to the right of the choose button to get popup window shown in the screenshots. In this window we enter six on the number of clusters and we leave the value of the seed on as it is. The seed value is used in generating a random number which is used for making the internal assignments of instances of clusters.

Step 5 : Once of the option have been specified. We run the clustering algorithm there we must make sure that they are in the 'cluster mode' panel. The use of training set option is selected and then we click 'start' button. This process and resulting window are shown in the following screenshots.

Step 6 : The result window shows the centroid of each cluster as well as statistics on the number and the percent of instances assigned to different clusters. Here clusters centroid are means vectors for each clusters. This clusters can be used to characterized the cluster.

Step 7: Another way of understanding characterstics of each cluster through visualization ,we can do this, try right clicking the result set on the result. List panel and selecting the visualize cluster assignments.

Interpretation of the above visualization

From the above visualization, we can understand the distribution of age and instance number in each cluster. For instance, for each cluster is dominated by age. In this case by changing the color dimension to other attributes we can see their distribution with in each of the cluster.

Step 8: We can assure that resulting dataset which included each instance along with its assign cluster. To do so we click the save button in the visualization window and save the result student k-mean .The top portion of this file is shown in the following figure.

Dataset student .arff

@relation student

@attribute age {<30,30-40,>40}

@attribute income {low,medium,high}

@attribute student {yes,no}

@attribute credit-rating {fair,excellent}

@attribute buyspc {yes,no}

@data

%

<30, high, no, fair, no

<30, high, no, excellent, no

30-40, high, no, fair, yes

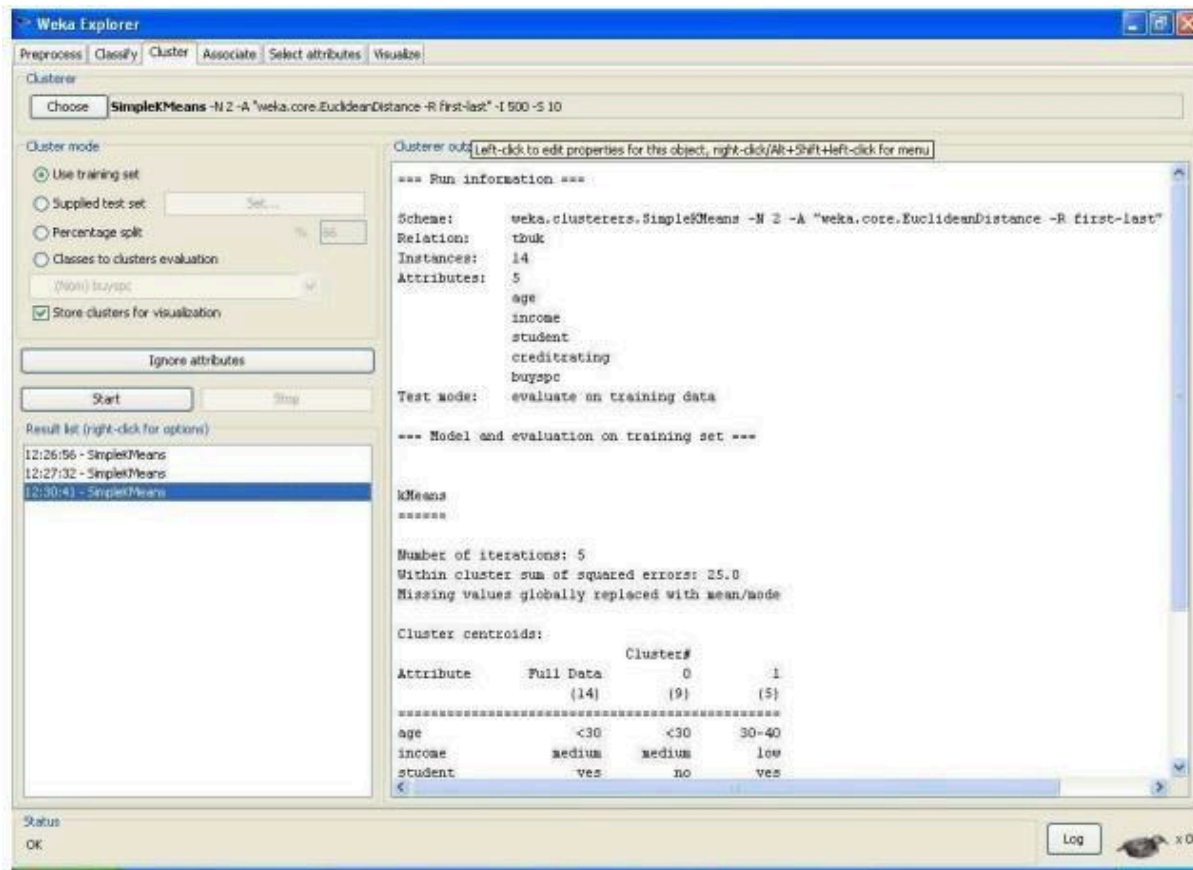
>40, medium, no, fair, yes

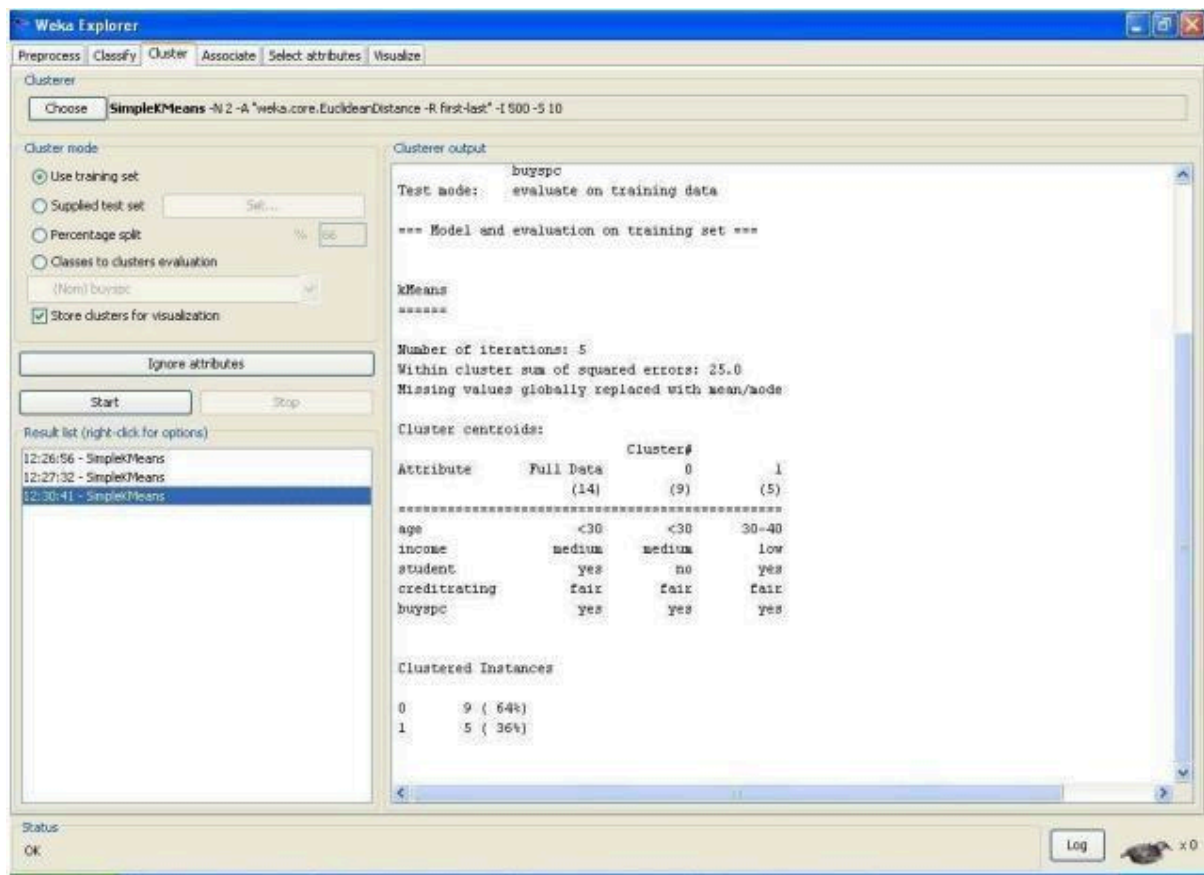
>40, low, yes, fair, yes

>40, low, yes, excellent, no

30-40, low, yes, excellent, yes
 <30, medium, no, fair, no
 <30, low, yes, fair, no
 >40, medium, yes, fair, yes
 <30, medium, yes, excellent, yes
 30-40, medium, no, excellent,
 yes 30-40, high, yes, fair, yes
 >40, medium, no, excellent, no
 %

The following screenshot shows the clustering rules that were generated when simple kmeans algorithm is applied on the given dataset.





Result:

Thus the experiment demonstration of clustering rule process on dataset student.arff using simple k-means was done successfully.