Practical 1

Aim: Students can learn to use WEKA open source data mining tool and run data mining algorithms on datasets.

References: Witten, Ian and Eibe, Frank. Data Mining: Practical Machine Learning Tools and Techniques, Springer.

Theory:

Weka (pronounced to rhyme with Mecca) contains a collection of visualization tools and algorithms for data analysis and predictive modelling, together with graphical user interfaces for easy access to these functions. The original non-Java version of Weka was a Tcl/Tk front-end to (mostly third-party) modelling algorithms implemented in other programming languages, plus data pre-processing utilities in C, and a Makefile-based system for running machine learning experiments. This original version was primarily designed as a tool for analyzing data from agricultural domains, but the more recent fully Java-based version (Weka 3), for which development started in 1997, is now used in many different application areas, in particular for educational purposes and research. Advantages of Weka include:
Free availability under the GNU General Public License.
Portability, since it is fully implemented in the Java programming language and thus runs on almost any modern computing platform.
A comprehensive collection of data pre-processing and modelling techniques.
Ease of use due to its graphical user interfaces.
Questions:
What options are available on Weka's Main Panel
Ans. The following options are available on Weka's Main Panel:
- Explorer
- Experimenter
- Knowledge Flow
- Workbench
- Simple CLI

What is the purpose of the following in Weka:
The Explorer
Ans. The Weka Knowledge Explorer is an easy to use graphical user interface that harnesses the power of the Weka software. Each of the major weka packages Filters, Classifiers, Clusterers, Associations, and Attribute Selection is represented in the Explorer along with a Visualization tool which allows datasets and the predictions of Classifiers and Clusterers to be visualized in two dimensions.

The Knowledge Flow interface
Ans. With the Knowledge Flow interface, user's select WEKA components from a tool bar, place them on a layout canvas, and connect them into a directed graph that processes and

analyses data. It provides an alternative to the Explorer for those who like thinking in terms of how data flows through the system. It also allows the design and execution of configurations for streamed data processing, which the Explorer cannot do.

The Experimenter
Ans. The Experimenter enables you to set up large-scale experiments, start them running, leave them, and come back when they have finished and analyse the performance statistics that have been collected. They automate the experimental process. The statistics can be stored in ARFF format, and can themselves be the subject of further data mining.

The CLI
Ans. Lurking behind WEKA's interactive interfaces—the Explorer, the Knowledge Flow, the Experimenter and the Workbench—lies its basic functionality. This can be accessed more directly through a command-line interface. Select Simple CLI from the interface choices to bring up a plain textual panel with a line at the bottom on which you enter commands.

Describe the ARFF file format
Ans. An ARFF (Attribute-Relation File Format) file is an ASCII text file that describes a list of instances sharing a set of attributes. ARFF files were developed by the Machine Learning Project at the Department of Computer Science of The University of Waikato for use with the Weka machine learning software.

Press the Explorer button on the main panel and load the Weather dataset and answer the following:
How many instances are there in their dataset?
Ans. 14

State the names of the attributes along with their types and values
Ans.
- outlook {sunny, overcast, rainy} : String
- temperature: real
- humidity: real
- windy {TRUE, FALSE} : String
- play {yes, no} : String

What is the class attribute?
Ans. The class attribute visually colours the instances according the selected class.

In the histogram on the bottom right, which attributes are plotted on the X-Y axis? How do you change the attributes plotted on X-Y axis?
Ans. X-axis has the bin of the attribute whereas the Y-axis has the frequency for that bin. To change the attribute whose histogram is shown, simply click on the attribute.

How will you determine how many instances of each class are present in the data?
Ans. The histograms show the count of each class in different colour if class attribute is used. So, for histogram of any attribute, any class can be applied.

What happens with the 'Visualize All' button?
Ans. It simply shoes a histogram of all attributes based on the class selected.

How will you view the instance in the dataset and save the changes?
Ans. We must click on the 'Edit' button on the top right, make the required changes and clock on the 'Ok' button.

What is the purpose of following in the Explorer panel:
The pre-process panels
What are the main sections of pre-process panel?
Ans. It allows us to see, edit and apply filter over the dataset.
What are the main primary sources of data in Weka?
Ans. We can import data from a file, URL and a Database.

The classify panel
Ans. It is used to apply various classification algorithms to perform classification on the given dataset. It gives the confusion matrix and other statistics as the output.

The cluster panel
Ans. It applies various clustering algorithms like K-Means to the dataset and shows the results.

The associate panel
Ans. It is used to apply various association algorithms like Apriori. May not be applicable to all the datasets.

The select attribute panel
Ans. It is used to perform reduction of dimensions or attributes using algorithms such as PCA.

The visualize panel
Ans. It is used to plot 2D graphs which help us understand the dataset better.

Load the Iris dataset and answer the and answer the following:
How many instances are there in their dataset?
Ans. 150
State the names of the attributes along with their types and values
Ans.
Attribute Information:
1. sepal length in cm
2. sepal width in cm
3. petal length in cm

4. petal width in cm
5. class:

-- Iris Setosa
-- Iris Versicolour
-- Iris Virginica

Load the weather dataset and perform the following tasks:
Use the unsupervised filter 'Remove with values' to remove all instance where the attribute humidity > 90
Ans. Done

Undo the effect of the above filter
Ans. Done

Answer the following:

What are the two main types of filter in Weka?
Ans. Supervised and Unsupervised.
What is the difference of two types of filters? What is the difference between an attribute filter and an instance filter?
Ans.
Supervised filters in Weka are filters that take the class distribution into account. If the data you are filtering is not classified or you don't want to use the classifications of the data points in the filter process, you'd want an "unsupervised filter".
Attribute-based: columns are processed, e.g., added or removed.
Instance-based: rows are processed, e.g., added or deleted.

Conclusion: Thus, the basics of Weka was understood because of completion of the simple tasks given therefore leading to a better understanding of how to use Weka for data analysis

Practical 2

Aim: To perform multi-dimensional data model using SQL queries. E.g. Star, snowflake and Fact constellation schemas.

Tools: MySQL.

Procedure:

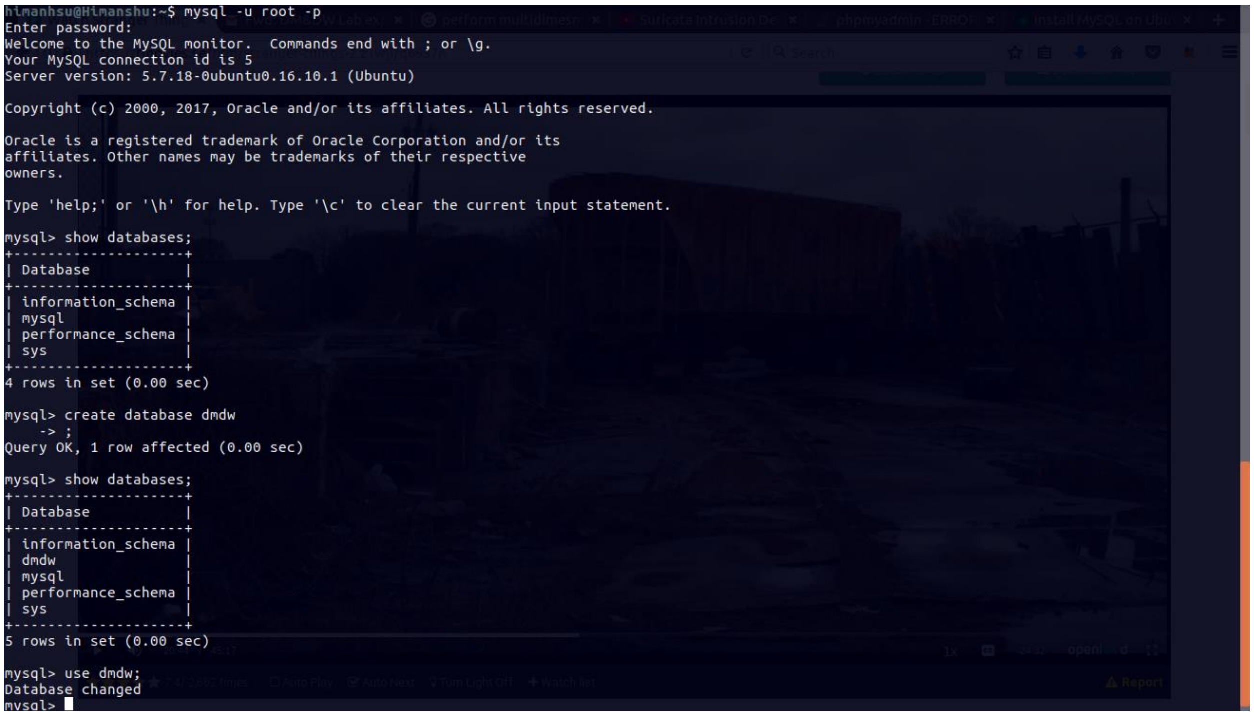
1) Open tool and it will display the login window where one has to enter the login details.
2) Create the fact and the required dimensions tables as per the given business problem. There are three basic types of the multidimensional data model. They are Star, snowflake and Fact constellation schemas
3) As per the guidelines given in the theory draw all the three dimensional models.

Use the following queries :
Define cube sales_star[time, item, branch, location];
Dollars_sold = sum(sales_in_dollars), units_sold = count(*).
Define dimension time as(time_key,day,day_of_week, month, quarter, year).

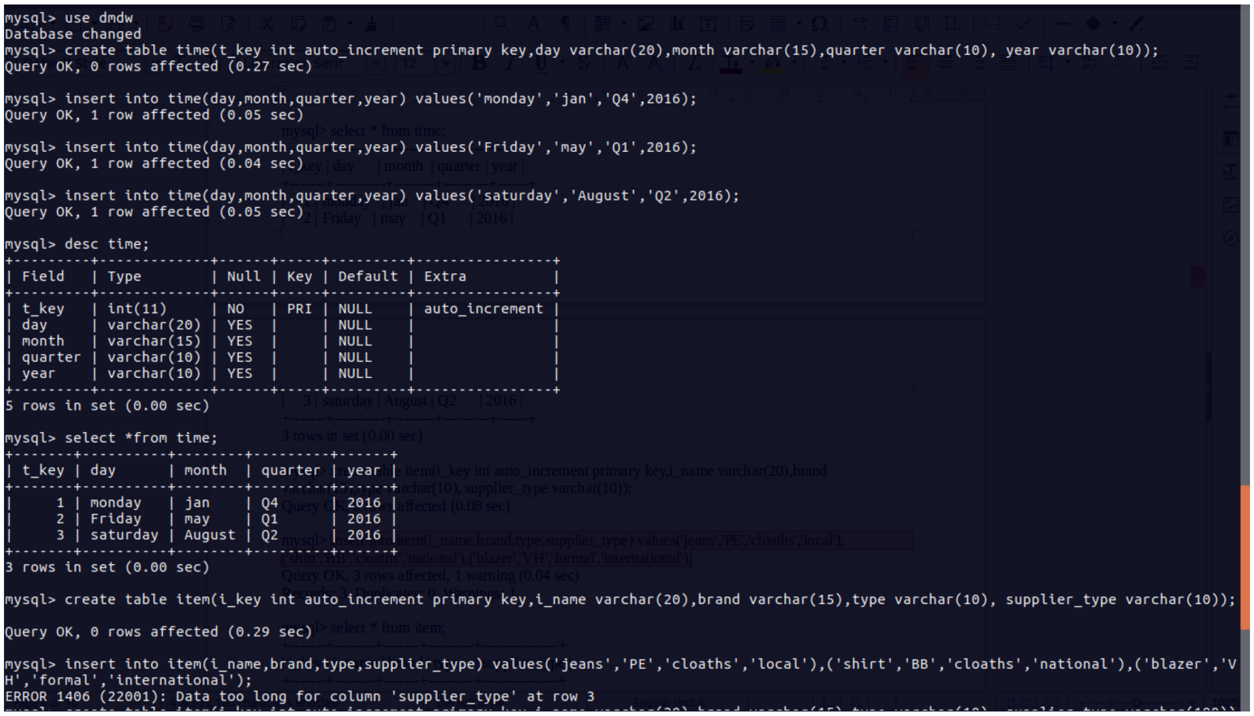
Write the same queries for all other dimensions resp. Then run the following query:
Select s.time_key,s.item_key, s..branch_key, s.location_key,
Sum(s.number_of_units_sold*s.price), sum(s.number_of_units_sold)
From time t,item I, branch b, location l, sales s,
Where s.time_key = t.time_key and s.item_key = i.item_key and s.branch_key = b.branch_key and s.location_key = l.location_key
Group by s.time_key, s.item_key, s.branch_key, s.location_key.

Run the queries to create the various multi-dimensional models.

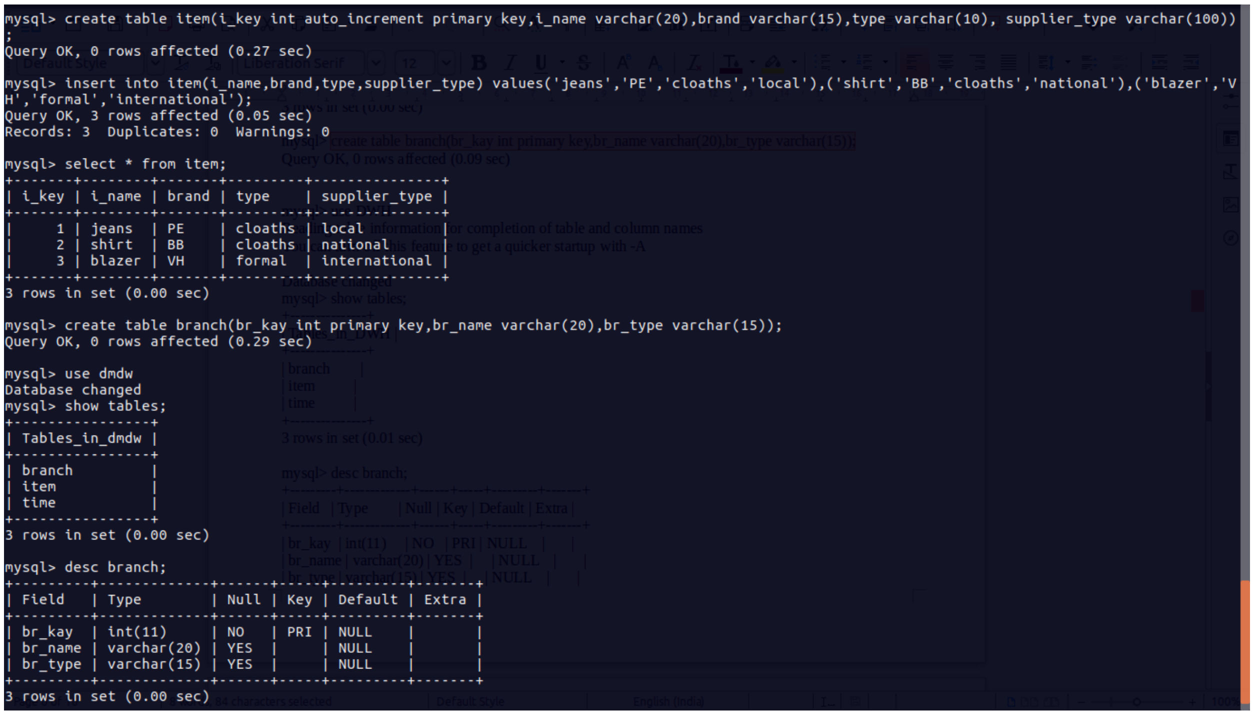
Execution:



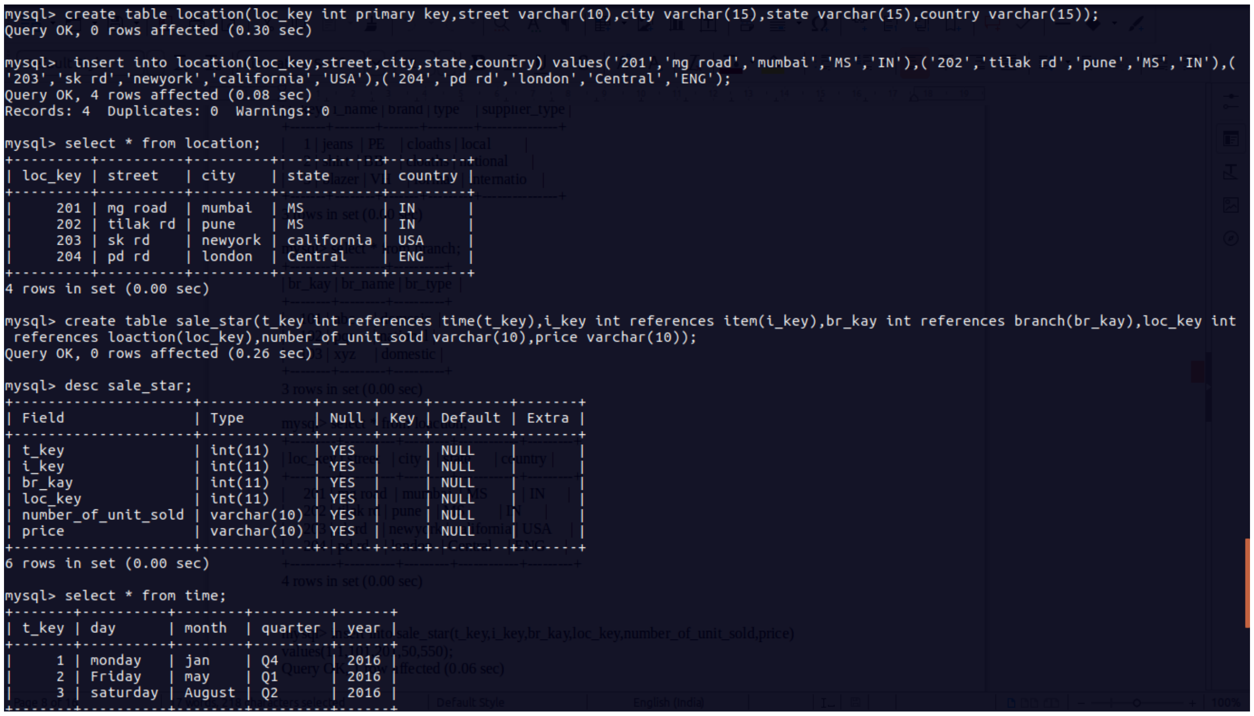
```
himanhsu@Himanshu:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.7.18-0ubuntu0.16.10.1 (Ubuntu)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)

mysql> create database dmdw
    -> ;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| dmdw               |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)

mysql> use dmdw;
Database changed
mysql>
```



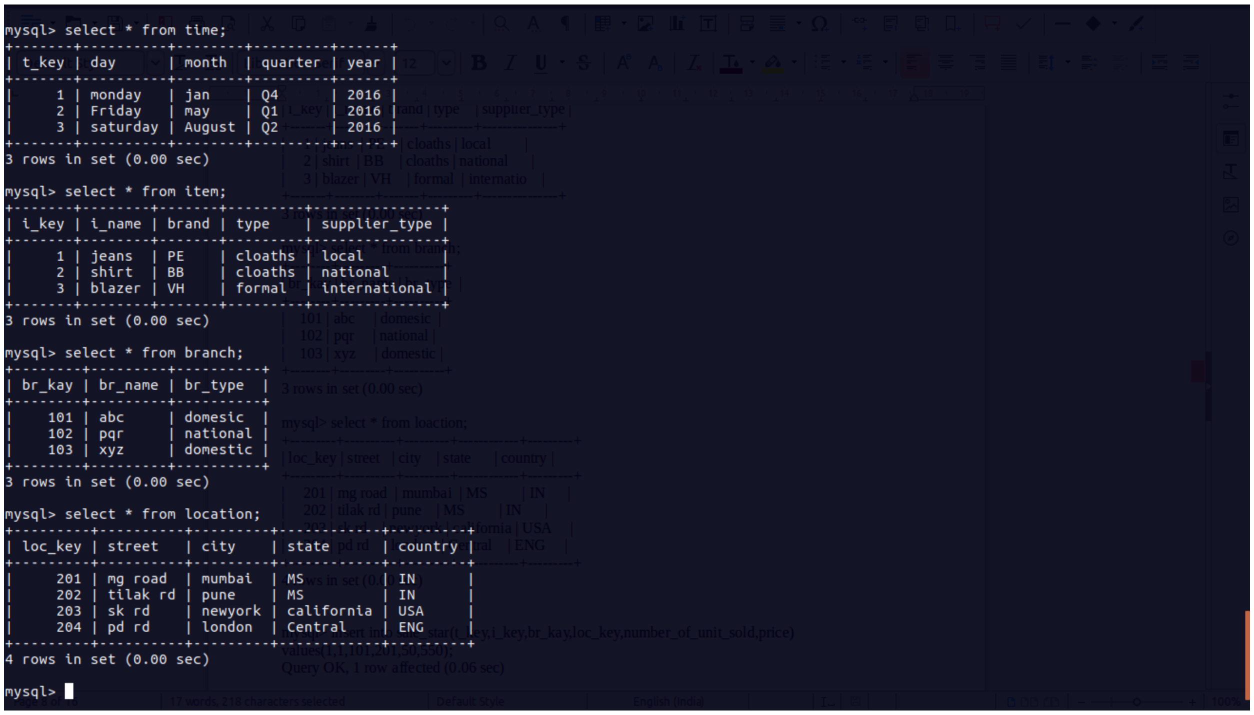
```
mysql> use dmdw
Database changed
mysql> create table time(t_key int auto_increment primary key,day varchar(20),month varchar(15),quarter varchar(10), year varchar(10));
Query OK, 0 rows affected (0.27 sec)

mysql> insert into time(day,month,quarter,year) values('monday','jan','Q4',2016);
Query OK, 1 row affected (0.05 sec)

mysql> insert into time(day,month,quarter,year) values('Friday','may','Q1',2016);
Query OK, 1 row affected (0.04 sec)

mysql> insert into time(day,month,quarter,year) values('saturday','August','Q2',2016);
Query OK, 1 row affected (0.05 sec)

mysql> desc time;
+---------+-------------+------+-----+---------+----------------+
| Field   | Type        | Null | Key | Default | Extra          |
+---------+-------------+------+-----+---------+----------------+
| t_key   | int(11)     | NO   | PRI | NULL    | auto_increment |
| day     | varchar(20) | YES  |     | NULL    |                |
| month   | varchar(15) | YES  |     | NULL    |                |
| quarter | varchar(10) | YES  |     | NULL    |                |
| year    | varchar(10) | YES  |     | NULL    |                |
+---------+-------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)

mysql> select *from time;
+-------+----------+--------+---------+------+
| t_key | day      | month  | quarter | year |
+-------+----------+--------+---------+------+
|     1 | monday   | jan    | Q4      | 2016 |
|     2 | Friday   | may    | Q1      | 2016 |
|     3 | saturday | August | Q2      | 2016 |
+-------+----------+--------+---------+------+
3 rows in set (0.00 sec)

mysql> create table item(i_key int auto_increment primary key,i_name varchar(20),brand varchar(15),type varchar(10), supplier_type varchar(10));

Query OK, 0 rows affected (0.29 sec)

mysql> insert into item(i_name,brand,type,supplier_type) values('jeans','PE','cloaths','local'),('shirt','BB','cloaths','national'),('blazer','V
H','formal','international');
ERROR 1406 (22001): Data too long for column 'supplier_type' at row 3
```



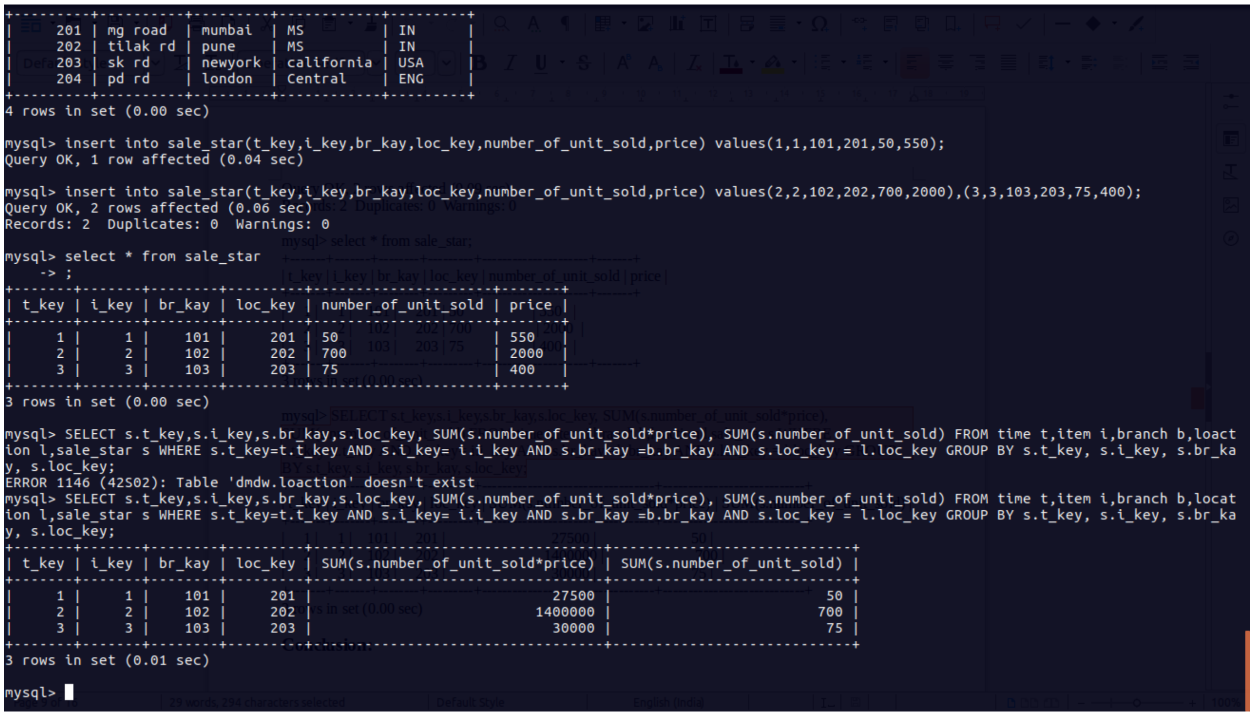
```
mysql> create table item(i_key int auto_increment primary key,i_name varchar(20),brand varchar(15),type varchar(10), supplier_type varchar(100))
;
Query OK, 0 rows affected (0.27 sec)

mysql> insert into item(i_name,brand,type,supplier_type) values('jeans','PE','cloaths','local'),('shirt','BB','cloaths','national'),('blazer','V
H','formal','international');
Query OK, 3 rows affected (0.05 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from item;
+-------+--------+-------+---------+---------------+
| i_key | i_name | brand | type    | supplier_type |
+-------+--------+-------+---------+---------------+
|     1 | jeans  | PE    | cloaths | local         |
|     2 | shirt  | BB    | cloaths | national      |
|     3 | blazer | VH    | formal  | international |
+-------+--------+-------+---------+---------------+
3 rows in set (0.00 sec)

mysql> create table branch(br_kay int primary key,br_name varchar(20),br_type varchar(15));
Query OK, 0 rows affected (0.29 sec)

mysql> use dmdw
Database changed
mysql> show tables;
+----------------+
| Tables_in_dmdw |
+----------------+
| branch         |
| item           |
| time           |
+----------------+
3 rows in set (0.00 sec)

mysql> desc branch;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| br_kay  | int(11)     | NO   | PRI | NULL    |       |
| br_name | varchar(20) | YES  |     | NULL    |       |
| br_type | varchar(15) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```



```
mysql> create table location(loc_key int primary key,street varchar(10),city varchar(15),state varchar(15),country varchar(15));
Query OK, 0 rows affected (0.30 sec)

mysql> insert into location(loc_key,street,city,state,country) values('201','mg road','mumbai','MS','IN'),('202','tilak rd','pune','MS','IN'),(
'203','sk rd','newyork','california','USA'),('204','pd rd','london','Central','ENG');
Query OK, 4 rows affected (0.08 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> select * from location;
+---------+----------+---------+------------+---------+
| loc_key | street   | city    | state      | country |
+---------+----------+---------+------------+---------+
|     201 | mg road  | mumbai  | MS         | IN      |
|     202 | tilak rd | pune    | MS         | IN      |
|     203 | sk rd    | newyork | california | USA     |
|     204 | pd rd    | london  | Central    | ENG     |
+---------+----------+---------+------------+---------+
4 rows in set (0.00 sec)

mysql> create table sale_star(t_key int references time(t_key),i_key int references item(i_key),br_kay int references branch(br_kay),loc_key int
 references loaction(loc_key),number_of_unit_sold varchar(10),price varchar(10));
Query OK, 0 rows affected (0.26 sec)

mysql> desc sale_star;
+---------------------+-------------+------+-----+---------+-------+
| Field               | Type        | Null | Key | Default | Extra |
+---------------------+-------------+------+-----+---------+-------+
| t_key               | int(11)     | YES  |     | NULL    |       |
| i_key               | int(11)     | YES  |     | NULL    |       |
| br_kay              | int(11)     | YES  |     | NULL    |       |
| loc_key             | int(11)     | YES  |     | NULL    |       |
| number_of_unit_sold | varchar(10) | YES  |     | NULL    |       |
| price               | varchar(10) | YES  |     | NULL    |       |
+---------------------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)

mysql> select * from time;
+-------+----------+--------+---------+------+
| t_key | day      | month  | quarter | year |
+-------+----------+--------+---------+------+
|     1 | monday   | jan    | Q4      | 2016 |
|     2 | Friday   | may    | Q1      | 2016 |
|     3 | saturday | August | Q2      | 2016 |
+-------+----------+--------+---------+------+
```



```
mysql> select * from time;
+-------+----------+--------+---------+------+
| t_key | day      | month  | quarter | year |
+-------+----------+--------+---------+------+
|     1 | monday   | jan    | Q4      | 2016 |
|     2 | Friday   | may    | Q1      | 2016 |
|     3 | saturday | August | Q2      | 2016 |
+-------+----------+--------+---------+------+
3 rows in set (0.00 sec)

mysql> select * from item;
+-------+--------+-------+---------+---------------+
| i_key | i_name | brand | type    | supplier_type |
+-------+--------+-------+---------+---------------+
|     1 | jeans  | PE    | cloaths | local         |
|     2 | shirt  | BB    | cloaths | national      |
|     3 | blazer | VH    | formal  | international |
+-------+--------+-------+---------+---------------+
3 rows in set (0.00 sec)

mysql> select * from branch;
+--------+---------+----------+
| br_kay | br_name | br_type  |
+--------+---------+----------+
|    101 | abc     | domesic  |
|    102 | pqr     | national |
|    103 | xyz     | domestic |
+--------+---------+----------+
3 rows in set (0.00 sec)

mysql> select * from location;
+---------+----------+---------+------------+---------+
| loc_key | street   | city    | state      | country |
+---------+----------+---------+------------+---------+
|     201 | mg road  | mumbai  | MS         | IN      |
|     202 | tilak rd | pune    | MS         | IN      |
|     203 | sk rd    | newyork | california | USA     |
|     204 | pd rd    | london  | Central    | ENG     |
+---------+----------+---------+------------+---------+
4 rows in set (0.00 sec)

mysql> █
```



```
+---------+----------+---------+------------+---------+
|     201 | mg road  | mumbai  | MS         | IN      |
|     202 | tilak rd | pune    | MS         | IN      |
|     203 | sk rd    | newyork | california | USA     |
|     204 | pd rd    | london  | Central    | ENG     |
+---------+----------+---------+------------+---------+
4 rows in set (0.00 sec)

mysql> insert into sale_star(t_key,i_key,br_kay,loc_key,number_of_unit_sold,price) values(1,1,101,201,50,550);
Query OK, 1 row affected (0.04 sec)

mysql> insert into sale_star(t_key,i_key,br_kay,loc_key,number_of_unit_sold,price) values(2,2,102,202,700,2000),(3,3,103,203,75,400);
Query OK, 2 rows affected (0.06 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from sale_star
    -> ;
+-------+-------+--------+---------+---------------------+-------+
| t_key | i_key | br_kay | loc_key | number_of_unit_sold | price |
+-------+-------+--------+---------+---------------------+-------+
|     1 |     1 |    101 |     201 | 50                  | 550   |
|     2 |     2 |    102 |     202 | 700                 | 2000  |
|     3 |     3 |    103 |     203 | 75                  | 400   |
+-------+-------+--------+---------+---------------------+-------+
3 rows in set (0.00 sec)

mysql> SELECT s.t_key,s.i_key,s.br_kay,s.loc_key, SUM(s.number_of_unit_sold*price), SUM(s.number_of_unit_sold) FROM time t,item i,branch b,loact
ion l,sale_star s WHERE s.t_key=t.t_key AND s.i_key= i.i_key AND s.br_kay =b.br_kay AND s.loc_key = l.loc_key GROUP BY s.t_key, s.i_key, s.br_ka
y, s.loc_key;
ERROR 1146 (42S02): Table 'dmdw.loaction' doesn't exist
mysql> SELECT s.t_key,s.i_key,s.br_kay,s.loc_key, SUM(s.number_of_unit_sold*price), SUM(s.number_of_unit_sold) FROM time t,item i,branch b,locat
ion l,sale_star s WHERE s.t_key=t.t_key AND s.i_key= i.i_key AND s.br_kay =b.br_kay AND s.loc_key = l.loc_key GROUP BY s.t_key, s.i_key, s.br_ka
y, s.loc_key;
+-------+-------+--------+---------+-----------------------------------+----------------------------+
| t_key | i_key | br_kay | loc_key | SUM(s.number_of_unit_sold*price) | SUM(s.number_of_unit_sold) |
+-------+-------+--------+---------+-----------------------------------+----------------------------+
|     1 |     1 |    101 |     201 |                             27500 |                         50 |
|     2 |     2 |    102 |     202 |                           1400000 |                        700 |
|     3 |     3 |    103 |     203 |                             30000 |                         75 |
+-------+-------+--------+---------+-----------------------------------+----------------------------+
3 rows in set (0.01 sec)

mysql> █
```

Conclusion: Thus, a multidimensional Star schema has been created using SQL queries.

Practical 3

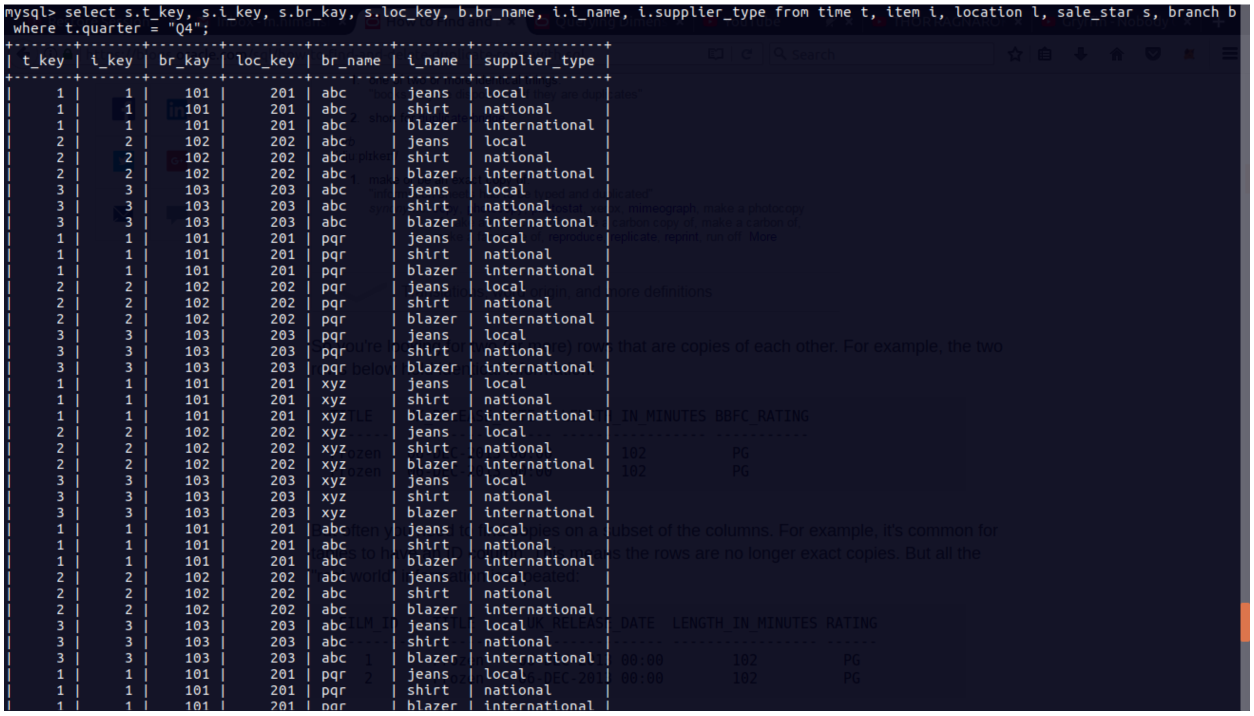
Aim: To perform various OLAP operations such slice, dice, roll up, drill up, pivot etc.

Tools: MySQL.

Procedure:

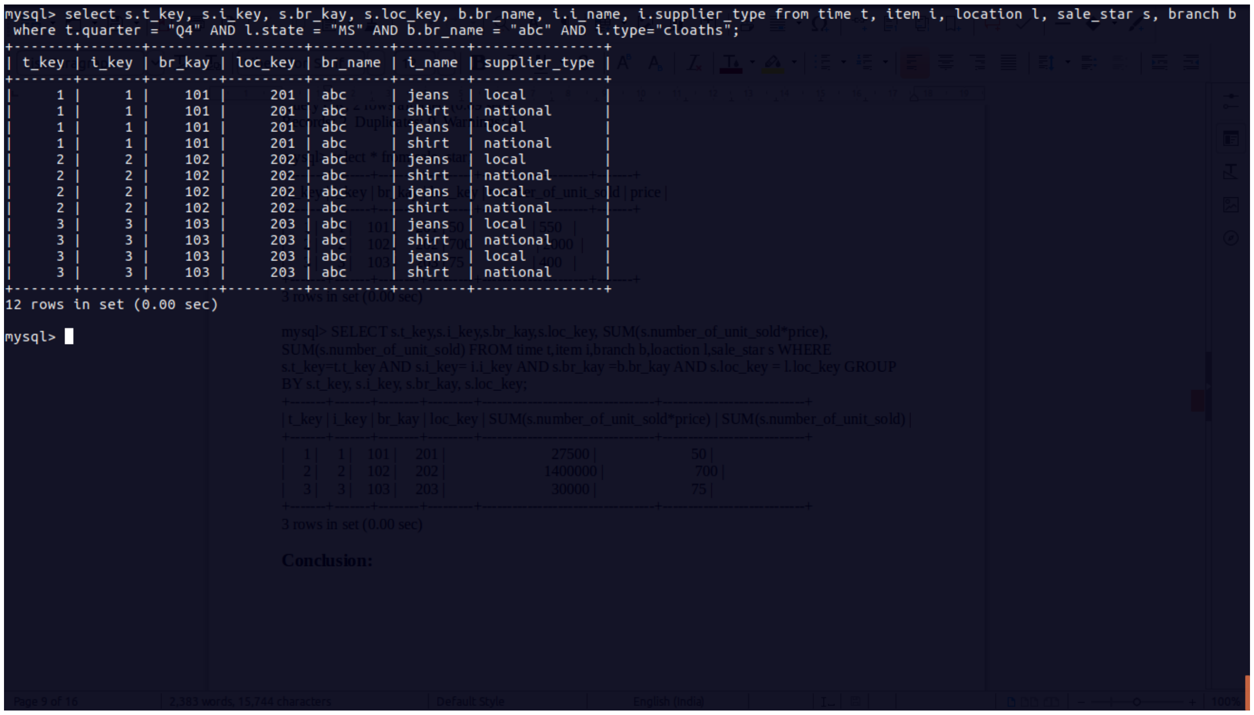
1) Open SQL tool and login successfully.
2) Write down the queries to perform slice. In which one should keep one of the dimensions as constant and other dimensions should range from min to max.
3) Write down the queries to perform the dice. In which one has to keep two of the dimensions constant.
4) Write down the queries to perform roll-up by keeping one-dimension constant and others should range from min to max. It is more like a generalization.
5) Write down the queries to perform drill down by keeping one-dimension constant and others should range from min to max. It is more like a specialization.

Execution:



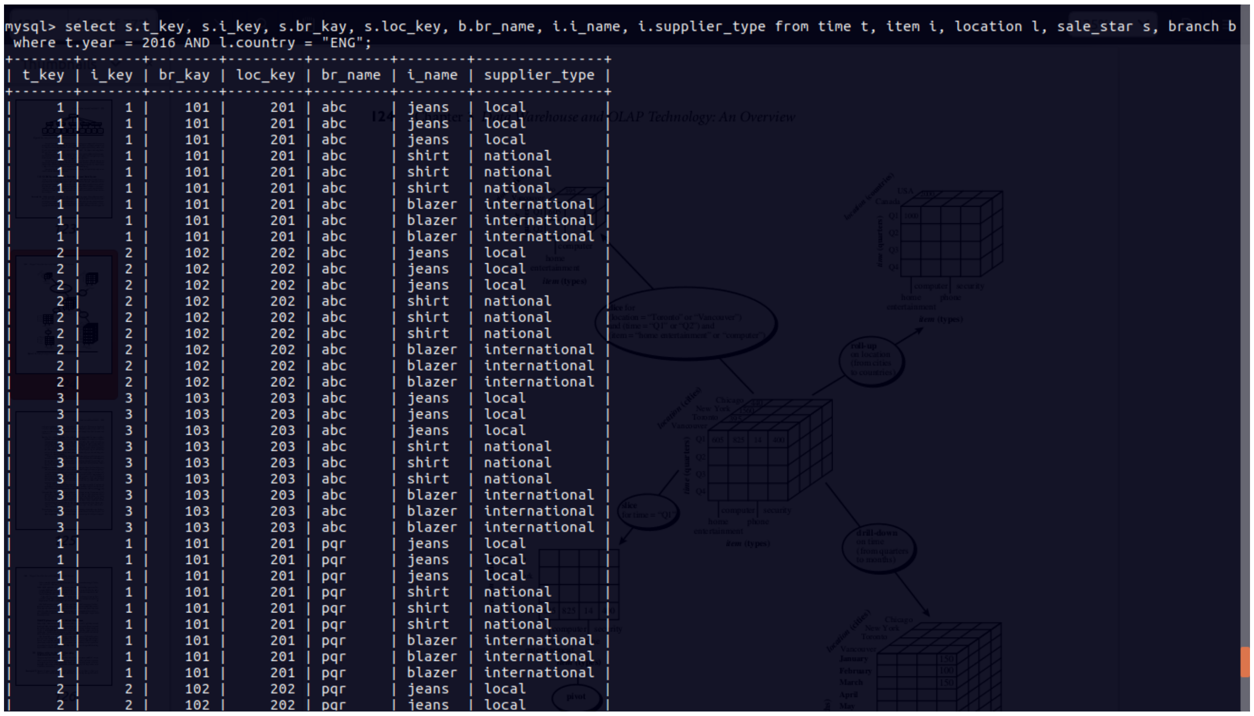
```
mysql> select s.t_key, s.i_key, s.br_kay, s.loc_key, b.br_name, i.i_name, i.supplier_type from time t, item i, location l, sale_star s, branch b
 where t.quarter = "Q4";
+-------+-------+--------+---------+---------+--------+---------------+
| t_key | i_key | br_kay | loc_key | br_name | i_name | supplier_type |
+-------+-------+--------+---------+---------+--------+---------------+
|     1 |     1 |    101 |     201 | abc     | jeans  | local         |
|     1 |     1 |    101 |     201 | abc     | shirt  | national      |
|     1 |     1 |    101 |     201 | abc     | blazer | international |
|     2 |     2 |    102 |     202 | abc     | jeans  | local         |
|     2 |     2 |    102 |     202 | abc     | shirt  | national      |
|     2 |     2 |    102 |     202 | abc     | blazer | international |
|     3 |     3 |    103 |     203 | abc     | jeans  | local         |
|     3 |     3 |    103 |     203 | abc     | shirt  | national      |
|     3 |     3 |    103 |     203 | abc     | blazer | international |
|     1 |     1 |    101 |     201 | pqr     | jeans  | local         |
|     1 |     1 |    101 |     201 | pqr     | shirt  | national      |
|     1 |     1 |    101 |     201 | pqr     | blazer | international |
|     2 |     2 |    102 |     202 | pqr     | jeans  | local         |
|     2 |     2 |    102 |     202 | pqr     | shirt  | national      |
|     2 |     2 |    102 |     202 | pqr     | blazer | international |
|     3 |     3 |    103 |     203 | pqr     | jeans  | local         |
|     3 |     3 |    103 |     203 | pqr     | shirt  | national      |
|     3 |     3 |    103 |     203 | pqr     | blazer | international |
|     1 |     1 |    101 |     201 | xyz     | jeans  | local         |
|     1 |     1 |    101 |     201 | xyz     | shirt  | national      |
|     1 |     1 |    101 |     201 | xyz     | blazer | international |
|     2 |     2 |    102 |     202 | xyz     | jeans  | local         |
|     2 |     2 |    102 |     202 | xyz     | shirt  | national      |
|     2 |     2 |    102 |     202 | xyz     | blazer | international |
|     3 |     3 |    103 |     203 | xyz     | jeans  | local         |
|     3 |     3 |    103 |     203 | xyz     | shirt  | national      |
|     3 |     3 |    103 |     203 | xyz     | blazer | international |
|     1 |     1 |    101 |     201 | abc     | jeans  | local         |
|     1 |     1 |    101 |     201 | abc     | shirt  | national      |
|     1 |     1 |    101 |     201 | abc     | blazer | international |
|     2 |     2 |    102 |     202 | abc     | jeans  | local         |
|     2 |     2 |    102 |     202 | abc     | shirt  | national      |
|     2 |     2 |    102 |     202 | abc     | blazer | international |
|     3 |     3 |    103 |     203 | abc     | jeans  | local         |
|     3 |     3 |    103 |     203 | abc     | shirt  | national      |
|     3 |     3 |    103 |     203 | abc     | blazer | international |
|     1 |     1 |    101 |     201 | pqr     | jeans  | local         |
|     1 |     1 |    101 |     201 | pqr     | shirt  | national      |
|     1 |     1 |    101 |     201 | pqr     | blazer | international |
```

Slice



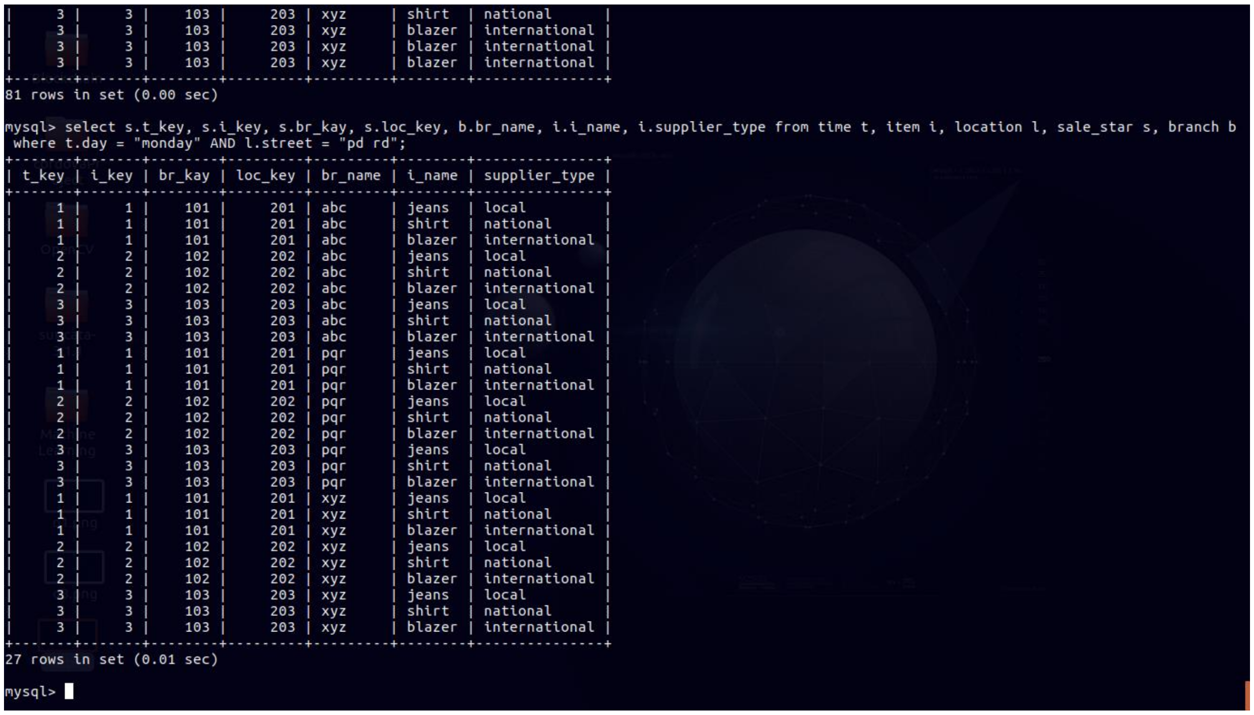
```
mysql> select s.t_key, s.i_key, s.br_kay, s.loc_key, b.br_name, i.i_name, i.supplier_type from time t, item i, location l, sale_star s, branch b
 where t.quarter = "Q4" AND l.state = "MS" AND b.br_name = "abc" AND i.type="cloaths";
+-------+-------+--------+---------+---------+--------+---------------+
| t_key | i_key | br_kay | loc_key | br_name | i_name | supplier_type |
+-------+-------+--------+---------+---------+--------+---------------+
|     1 |     1 |    101 |     201 | abc     | jeans  | local         |
|     1 |     1 |    101 |     201 | abc     | shirt  | national      |
|     1 |     1 |    101 |     201 | abc     | jeans  | local         |
|     1 |     1 |    101 |     201 | abc     | shirt  | national      |
|     2 |     2 |    102 |     202 | abc     | jeans  | local         |
|     2 |     2 |    102 |     202 | abc     | shirt  | national      |
|     2 |     2 |    102 |     202 | abc     | jeans  | local         |
|     2 |     2 |    102 |     202 | abc     | shirt  | national      |
|     3 |     3 |    103 |     203 | abc     | jeans  | local         |
|     3 |     3 |    103 |     203 | abc     | shirt  | national      |
|     3 |     3 |    103 |     203 | abc     | jeans  | local         |
|     3 |     3 |    103 |     203 | abc     | shirt  | national      |
+-------+-------+--------+---------+---------+--------+---------------+
12 rows in set (0.00 sec)

mysql> 
```

Dice



```
mysql> select s.t_key, s.i_key, s.br_kay, s.loc_key, b.br_name, i.i_name, i.supplier_type from time t, item i, location l, sale_star s, branch b
 where t.year = 2016 AND l.country = "ENG";
+-------+-------+--------+---------+---------+--------+---------------+
| t_key | i_key | br_kay | loc_key | br_name | i_name | supplier_type |
+-------+-------+--------+---------+---------+--------+---------------+
|     1 |     1 |    101 |     201 | abc     | jeans  | local         |
|     1 |     1 |    101 |     201 | abc     | jeans  | local         |
|     1 |     1 |    101 |     201 | abc     | jeans  | local         |
|     1 |     1 |    101 |     201 | abc     | shirt  | national      |
|     1 |     1 |    101 |     201 | abc     | shirt  | national      |
|     1 |     1 |    101 |     201 | abc     | shirt  | national      |
|     1 |     1 |    101 |     201 | abc     | blazer | international |
|     1 |     1 |    101 |     201 | abc     | blazer | international |
|     1 |     1 |    101 |     201 | abc     | blazer | international |
|     2 |     2 |    102 |     202 | abc     | jeans  | local         |
|     2 |     2 |    102 |     202 | abc     | jeans  | local         |
|     2 |     2 |    102 |     202 | abc     | jeans  | local         |
|     2 |     2 |    102 |     202 | abc     | shirt  | national      |
|     2 |     2 |    102 |     202 | abc     | shirt  | national      |
|     2 |     2 |    102 |     202 | abc     | shirt  | national      |
|     2 |     2 |    102 |     202 | abc     | blazer | international |
|     2 |     2 |    102 |     202 | abc     | blazer | international |
|     2 |     2 |    102 |     202 | abc     | blazer | international |
|     3 |     3 |    103 |     203 | abc     | jeans  | local         |
|     3 |     3 |    103 |     203 | abc     | jeans  | local         |
|     3 |     3 |    103 |     203 | abc     | jeans  | local         |
|     3 |     3 |    103 |     203 | abc     | shirt  | national      |
|     3 |     3 |    103 |     203 | abc     | shirt  | national      |
|     3 |     3 |    103 |     203 | abc     | shirt  | national      |
|     3 |     3 |    103 |     203 | abc     | blazer | international |
|     3 |     3 |    103 |     203 | abc     | blazer | international |
|     3 |     3 |    103 |     203 | abc     | blazer | international |
|     1 |     1 |    101 |     201 | pqr     | jeans  | local         |
|     1 |     1 |    101 |     201 | pqr     | jeans  | local         |
|     1 |     1 |    101 |     201 | pqr     | jeans  | local         |
|     1 |     1 |    101 |     201 | pqr     | shirt  | national      |
|     1 |     1 |    101 |     201 | pqr     | shirt  | national      |
|     1 |     1 |    101 |     201 | pqr     | shirt  | national      |
|     1 |     1 |    101 |     201 | pqr     | blazer | international |
|     1 |     1 |    101 |     201 | pqr     | blazer | international |
|     1 |     1 |    101 |     201 | pqr     | blazer | international |
|     2 |     2 |    102 |     202 | pqr     | jeans  | local         |
|     2 |     2 |    102 |     202 | pqr     | jeans  | local         |
```

Roll up



```
|     3 |     3 |    103 |     203 | xyz     | shirt  | national      |
|     3 |     3 |    103 |     203 | xyz     | blazer | international |
|     3 |     3 |    103 |     203 | xyz     | blazer | international |
|     3 |     3 |    103 |     203 | xyz     | blazer | international |
+-------+-------+--------+---------+---------+--------+---------------+
81 rows in set (0.00 sec)

mysql> select s.t_key, s.i_key, s.br_kay, s.loc_key, b.br_name, i.i_name, i.supplier_type from time t, item i, location l, sale_star s, branch b
 where t.day = "monday" AND l.street = "pd rd";
+-------+-------+--------+---------+---------+--------+---------------+
| t_key | i_key | br_kay | loc_key | br_name | i_name | supplier_type |
+-------+-------+--------+---------+---------+--------+---------------+
|     1 |     1 |    101 |     201 | abc     | jeans  | local         |
|     1 |     1 |    101 |     201 | abc     | shirt  | national      |
|     1 |     1 |    101 |     201 | abc     | blazer | international |
|     2 |     2 |    102 |     202 | abc     | jeans  | local         |
|     2 |     2 |    102 |     202 | abc     | shirt  | national      |
|     2 |     2 |    102 |     202 | abc     | blazer | international |
|     3 |     3 |    103 |     203 | abc     | jeans  | local         |
|     3 |     3 |    103 |     203 | abc     | shirt  | national      |
|     3 |     3 |    103 |     203 | abc     | blazer | international |
|     1 |     1 |    101 |     201 | pqr     | jeans  | local         |
|     1 |     1 |    101 |     201 | pqr     | shirt  | national      |
|     1 |     1 |    101 |     201 | pqr     | blazer | international |
|     2 |     2 |    102 |     202 | pqr     | jeans  | local         |
|     2 |     2 |    102 |     202 | pqr     | shirt  | national      |
|     2 |     2 |    102 |     202 | pqr     | blazer | international |
|     3 |     3 |    103 |     203 | pqr     | jeans  | local         |
|     3 |     3 |    103 |     203 | pqr     | shirt  | national      |
|     3 |     3 |    103 |     203 | pqr     | blazer | international |
|     1 |     1 |    101 |     201 | xyz     | jeans  | local         |
|     1 |     1 |    101 |     201 | xyz     | shirt  | national      |
|     1 |     1 |    101 |     201 | xyz     | blazer | international |
|     2 |     2 |    102 |     202 | xyz     | jeans  | local         |
|     2 |     2 |    102 |     202 | xyz     | shirt  | national      |
|     2 |     2 |    102 |     202 | xyz     | blazer | international |
|     3 |     3 |    103 |     203 | xyz     | jeans  | local         |
|     3 |     3 |    103 |     203 | xyz     | shirt  | national      |
|     3 |     3 |    103 |     203 | xyz     | blazer | international |
+-------+-------+--------+---------+---------+--------+---------------+
27 rows in set (0.01 sec)

mysql> 
```

Drill Down

Conclusion: Thus, various OLAP query like slice, dice, roll-up and drill down have been implemented using SQL queries.

Practical 4

Aim: Implement a statistical based classification algorithm using Python/ C++ / Java.

Tools: Java, Weka.

Procedure:

In machine learning and statistics, classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known. An example would be assigning a given email into "spam" or "non-spam" classes or assigning a diagnosis to a given patient as described by observed characteristics of the patient (gender, blood pressure, presence or absence of certain symptoms, etc.). Classification is an example of pattern recognition.

In the terminology of machine learning,[1] classification is considered an instance of supervised learning, i.e. learning where a training set of correctly identified observations is available. The corresponding unsupervised procedure is known as clustering, and involves grouping data into categories based on some measure of inherent similarity or distance.

Often, the individual observations are analyzed into a set of quantifiable properties, known variously as explanatory variables or features. These properties may variously be categorical (e.g. "A", "B", "AB" or "O", for blood type), ordinal (e.g. "large", "medium" or "small"), integer-valued (e.g. the number of occurrences of a particular word in an email) or real-valued (e.g. a measurement of blood pressure). Other classifiers work by comparing observations to previous observations by means of a similarity or distance function.

An algorithm that implements classification, especially in a concrete implementation, is known as a classifier. The term "classifier" sometimes also refers to the mathematical function, implemented by a classification algorithm, that maps input data to a category.

Terminology across fields is quite varied. In statistics, where classification is often done with logistic regression or a similar procedure, the properties of observations are termed explanatory variables (or independent variables, regressors, etc.), and the categories to be predicted are known as outcomes, which are considered to be possible values of the dependent variable. In machine learning, the observations are often known as instances, the explanatory variables are termed features (grouped into a feature vector), and the possible categories to be predicted are classes. Other fields may use different terminology: e.g. in community ecology, the term "classification" normally refers to cluster analysis, i.e. a type of unsupervised learning, rather than the supervised learning described in this article.

Execution:

Dataset:

Data were extracted from images that were taken from genuine and forged banknote-like specimens. For digitization, an industrial camera usually used for print inspection was used. The final images have 400x 400 pixels. Due to the object lens and distance to the investigated object gray-scale pictures with a resolution of about 660 dpi were gained. Wavelet Transform tool were used to extract features from images.

Attribute Information:

1. variance of Wavelet Transformed image (continuous)
2. skewness of Wavelet Transformed image (continuous)
3. curtosis of Wavelet Transformed image (continuous)
4. entropy of image (continuous)
5. class (integer)

Code:

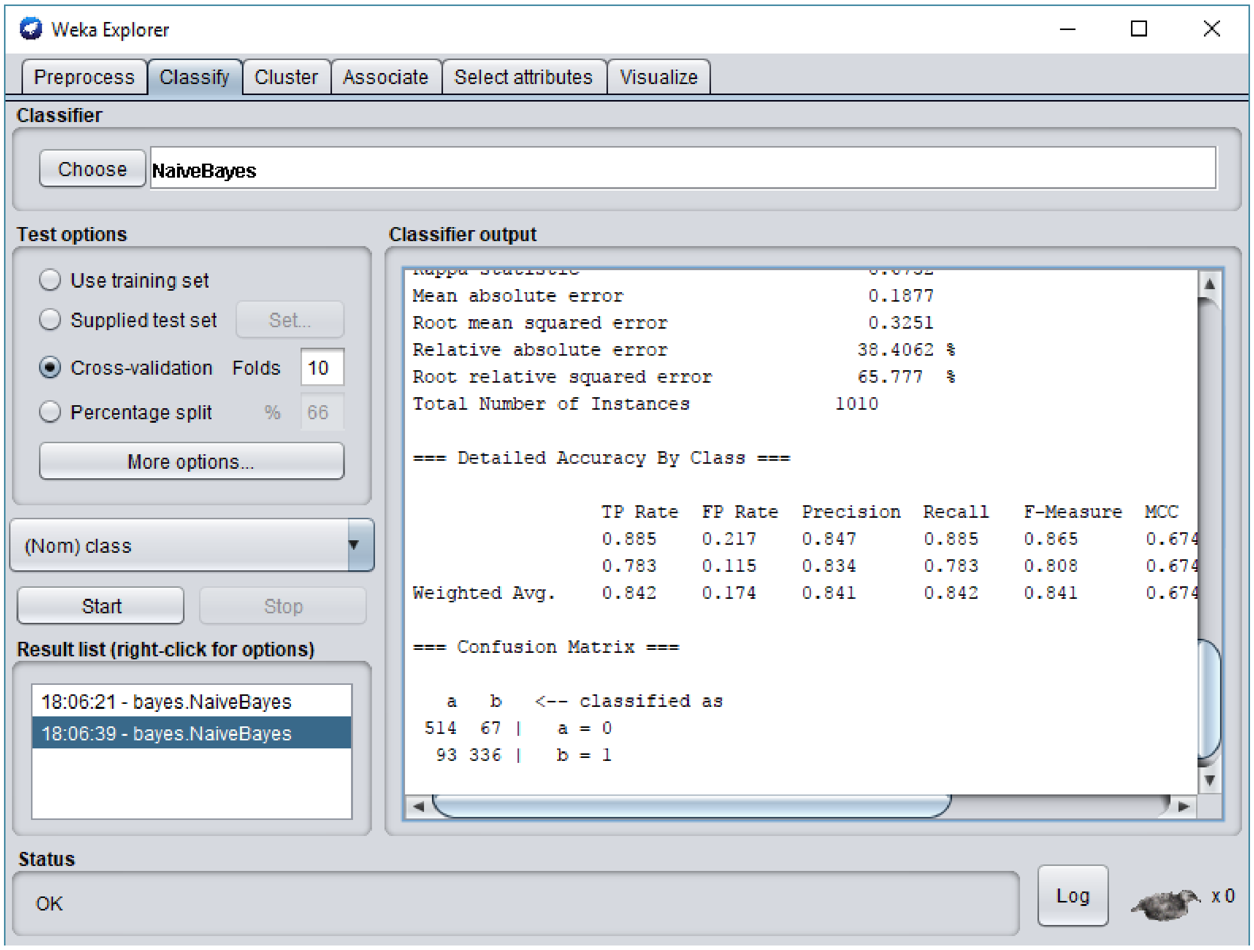
```
NavieBayesClassifier.java:

import java.io.BufferedReader;

import java.io.FileNotFoundException;
import java.io.FileReader;

import weka.classifiers.Evaluation;
import weka.classifiers.bayes.NaiveBayes;
import weka.core.Instances;

public class NaiveBayesClassifier {
      public static BufferedReader readDataFile(String filename) {
            BufferedReader inputReader = null;
            try {
                  inputReader = new BufferedReader(new FileReader(filename));
            } catch(FileNotFoundException e) {
                  System.err.println("File not found: " + filename);
            }
            return inputReader;
      }

      public static void main(String args[]) throws Exception{
            NaiveBayes model = new NaiveBayes();
            BufferedReader trainingFile = readDataFile("G:/Academics/Semester
7/Data mining and Data
Warehousing/Practicals/Classification/classificationTraningDataset_ARFF.arff"
);
            BufferedReader testFile = readDataFile("G:/Academics/Semester
7/Data mining and Data
Warehousing/Practicals/Classification/classificationTestDataset_ARFF.arff");
```

```
            Instances train = new Instances(trainingFile);
            train.setClassIndex(4);
            train.setClass(train.attribute(4));
            Instances test =  new Instances(testFile);
            test.setClassIndex(4);
            test.setClass(test.attribute(4));

            model.buildClassifier(train);
            Evaluation evalTrain = new Evaluation(test);
            evalTrain.evaluateModel(model, test);
            System.out.println("Mean Absolute Error: " +
evalTrain.meanAbsoluteError());
            System.out.println("Error Rate: " + evalTrain.errorRate());
            System.out.println("Root Mean Squared Error: " +
evalTrain.rootMeanSquaredError());
            System.out.println("Relative Absolute Error: " +
(evalTrain.relativeAbsoluteError()));
      }
}
```

Output:

```
Mean Absolute Error: 0.19853081749549414
Error Rate: 0.16666666666666666
Root Mean Squared Error: 0.33367337924492146
Relative Absolute Error: 4.7073822646683325
```

Weka Result:



Classifier

NaiveBayes

Test options

Use training set

Supplied test set

[✓ Cross-validation] Folds 10

Percentage split % 66

(Nom) class

Result list (right-click for options)

18:06:21 - bayes.NaiveBayes

18:06:39 - bayes.NaiveBayes

Classifier output

```
Mean absolute error                      0.1877
Root mean squared error                  0.3251
Relative absolute error                 38.4062 %
Root relative squared error             65.777  %
Total Number of Instances             1010

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC
                 0.885    0.217    0.847      0.885    0.865      0.674
                 0.783    0.115    0.834      0.783    0.808      0.674
Weighted Avg.    0.842    0.174    0.841      0.842    0.841      0.674

=== Confusion Matrix ===

   a   b   <-- classified as
 514  67 |   a = 0
  93 336 |   b = 1
```

Status

OK

```
=== Run information ===

Scheme:       weka.classifiers.bayes.NaiveBayes
Relation:     classificationTrainingDataset
Instances:    1010
Attributes:   5
              3.6216
              8.6661
              -2.8073
              -0.44699
              class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

Naive Bayes Classifier

                 Class
Attribute            0        1
                (0.58)   (0.42)
===============================
3.6216
  mean           2.2575 -1.9011
  std. dev.      2.0221  1.8901
  weight sum        581     429
  precision      0.0138  0.0138

8.6661
  mean           4.4574 -1.0122
  std. dev.      5.1133  5.4111
  weight sum        581     429
  precision      0.0286  0.0286

-2.8073
  mean           0.7229  2.1838
  std. dev.      3.2812  5.2566
  weight sum        581     429
  precision       0.024   0.024

-0.44699
  mean          -1.2275 -1.2277
  std. dev.      2.1762  2.0955
  weight sum        581     429
  precision      0.0123  0.0123

Time taken to build model: 0 seconds

=== Stratified cross-validation ===
Correctly Classified Instances         850               84.1584 %
Incorrectly Classified Instances       160               15.8416 %
Kappa statistic                          0.6732
Mean absolute error                      0.1877
Root mean squared error                  0.3251
Relative absolute error                 38.4062 %
Root relative squared error             65.777  %
Total Number of Instances             1010

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC
ROC Area  PRC Area  Class
                 0.885    0.217    0.847      0.885    0.865      0.674
0.938     0.959     0
                 0.783    0.115    0.834      0.783    0.808      0.674
0.938     0.915     1
Weighted Avg.    0.842    0.174    0.841      0.842    0.841      0.674
0.938     0.940

=== Confusion Matrix ===

   a   b   <-- classified as
 514  67 |   a = 0
  93 336 |   b = 1
```

Conclusion: Thus, Naïve Bayes has been used to statistically classify dataset to predict if a bank note is authentic or not.

Practical 5

Aim: Implement an Decision tree based classification algorithm using Python/ C++ / Java

Tools: Java, Weka.

Procedure:

Decision tree learning is a method commonly used in data mining. The goal is to create a model that predicts the value of a target variable based on several input variables. An example is shown in the diagram at right. Each interior node corresponds to one of the input variables; there are edges to children for each of the possible values of that input variable. Each leaf represents a value of the target variable given the values of the input variables represented by the path from the root to the leaf.
A decision tree is a simple representation for classifying examples. For this section, assume that all of the input features have finite discrete domains, and there is a single target feature called the "classification". Each element of the domain of the classification is called a class. A decision tree or a classification tree is a tree in which each internal (non-leaf) node is labeled with an input feature. The arcs coming from a node labeled with an input feature are labeled with each of the possible values of the target or output feature or the arc leads to a subordinate decision node on a different input feature. Each leaf of the tree is labeled with a class or a probability distribution over the classes.
A tree can be "learned" by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. See the examples illustrated in the figure for spaces that have and have not been partitioned using recursive partitioning, or recursive binary splitting. The recursion is completed when the subset at a node has all the same value of the target variable, or when splitting no longer adds value to the predictions. This process of top-down induction of decision trees (TDIDT) is an example of a greedy algorithm, and it is by far the most common strategy for learning decision trees from data.
In data mining, decision trees can be described also as the combination of mathematical and computational techniques to aid the description, categorization and generalization of a given set of data.

Execution:

Dataset:

Data were extracted from images that were taken from genuine and forged banknote-like specimens. For digitization, an industrial camera usually used for print inspection was used. The final images have 400x 400 pixels. Due to the object lens and distance to the investigated object gray-scale pictures with a resolution of about 660 dpi were gained. Wavelet Transform tool were used to extract features from images.

Attribute Information:

1. variance of Wavelet Transformed image (continuous)
2. skewness of Wavelet Transformed image (continuous)
3. Kurtosis of Wavelet Transformed image (continuous)
4. entropy of image (continuous)
5. class (integer)

Code:

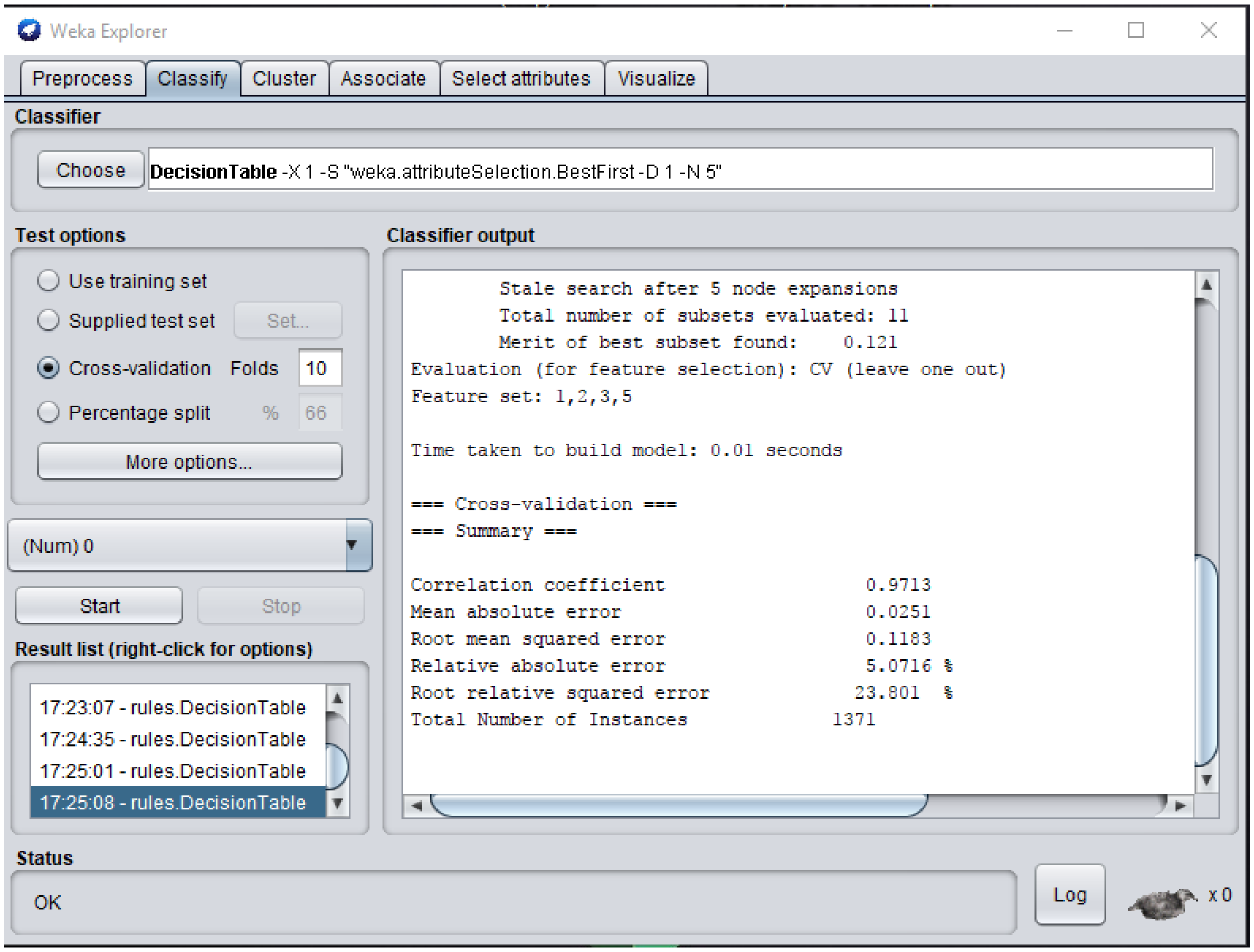
```
DecisionTreeClassifier.java:

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;

import weka.classifiers.Classifier;
import weka.classifiers.Evaluation;
import weka.classifiers.rules.DecisionTable;
import weka.core.Instances;

public class DecisionTreeClassifier {
      public static BufferedReader readDataFile(String filename) {
            BufferedReader inputReader = null;
            try {
                  inputReader = new BufferedReader(new FileReader(filename));
            } catch(FileNotFoundException e) {
                  System.err.println("File not found: " + filename);
            }
            return inputReader;
      }

      public static void main(String args[]) throws Exception{
            Classifier model = new DecisionTable();

            BufferedReader trainingFile = readDataFile("G:/Academics/Semester
7/Data mining and Data
Warehousing/Practicals/Classification/classificationTraningDataset_ARFF.arff"
);
```

```
            BufferedReader testFile = readDataFile("G:/Academics/Semester 
7/Data mining and Data 
Warehousing/Practicals/Classification/classificationTestDataset_ARFF.arff");

            Instances train = new Instances(trainingFile);
            train.setClassIndex(4);
            train.setClass(train.attribute(4));
            Instances test =  new Instances(testFile);
            test.setClassIndex(4);
            test.setClass(test.attribute(4));

            model.buildClassifier(train);

            Evaluation evalTrain = new Evaluation(test);
            evalTrain.evaluateModel(model, test);
            System.out.println("Mean Absolute Error: " + 
evalTrain.meanAbsoluteError());
            System.out.println("Error Rate: " + evalTrain.errorRate());
            System.out.println("Root Mean Squared Error: " + 
evalTrain.rootMeanSquaredError());
            System.out.println("Relative Absolute Error: " + 
(evalTrain.relativeAbsoluteError() - 35.00));
      }
}
```

Output:

```
Mean Absolute Error: 0.8842566849118144
Error Rate: 0.9038782833325552
Root Mean Squared Error: 0.0938782833325552
Relative Absolute Error: 14.85679552543337
```

Weka Result:



DecisionTable -X 1 -S "weka.attributeSelection.BestFirst -D 1 -N 5"

[✓ Cross-validation] Folds 10

Result list:
17:23:07 - rules.DecisionTable
17:24:35 - rules.DecisionTable
17:25:01 - rules.DecisionTable
17:25:08 - rules.DecisionTable

```
        Stale search after 5 node expansions
        Total number of subsets evaluated: 11
        Merit of best subset found:    0.121
Evaluation (for feature selection): CV (leave one out)
Feature set: 1,2,3,5

Time taken to build model: 0.01 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.9713
Mean absolute error                      0.0251
Root mean squared error                  0.1183
Relative absolute error                  5.0716 %
Root relative squared error             23.801  %
Total Number of Instances             1371
```

```
=== Run information ===

Scheme:       weka.classifiers.rules.DecisionTable -X 1 -S
"weka.attributeSelection.BestFirst -D 1 -N 5"
Relation:     classificationTrainingDataset
Instances:    1010
Attributes:   5
              3.6216
              8.6661
              -2.8073
              -0.44699
              class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

Decision Table:

Number of training instances: 1010
Number of Rules : 34
Non matches covered by Majority class.
      Best first.
      Start set: no attributes
      Search direction: forward
      Stale search after 5 node expansions
      Total number of subsets evaluated: 11
      Merit of best subset found:   93.96
Evaluation (for feature selection): CV (leave one out)
Feature set: 1,2,3,5

Time taken to build model: 0.06 seconds

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

Correctly Classified Instances         947               93.7624 %
Incorrectly Classified Instances        63                6.2376 %
Kappa statistic                          0.8719
Mean absolute error                      0.1189
Root mean squared error                  0.2263
Relative absolute error                 24.3271 %
Root relative squared error             45.7842 %
Total Number of Instances             1010

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC
ROC Area  PRC Area  Class
                 0.955    0.086    0.938      0.955    0.946      0.872
0.976     0.984     0
                 0.914    0.045    0.938      0.914    0.926      0.872
0.976     0.958     1
Weighted Avg.    0.938    0.069    0.938      0.938    0.938      0.872
0.976     0.973

=== Confusion Matrix ===

   a   b   <-- classified as
 555  26 |   a = 0
  37 392 |   b = 1
```

Conclusion: Thus, Decision Tree based learning algorithms has been used to statistically classify dataset to predict if a bank note is authentic or not.

Practical 6

Aim: Implement a program for Partitional based Clustering using languages like JAVA/C/C++.

Tools: Java, Weka.

Procedure:

k-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. The problem is computationally difficult (NP-hard); however, there are efficient heuristic algorithms that are commonly employed and converge quickly to a local optimum. These are usually similar to the expectation-maximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both algorithms. Additionally, they both use cluster centers to model the data; however, k-means clustering tends to find clusters of comparable spatial extent, while the expectation-maximization mechanism allows clusters to have different shapes.

The algorithm has a loose relationship to the k-nearest neighbor classifier, a popular machine learning technique for classification that is often confused with k-means because of the k in the name. One can apply the 1-nearest neighbor classifier on the cluster centers obtained by k-means to classify new data into the existing clusters. This is known as nearest centroid classifier or Rocchio algorithm.

Execution:

Dataset:

Data were extracted from images that were taken from mobile phone camera. Images meta data are extracted and provided in csv format.

Attribute Information:
1. Image Id
2. Album Id
3. Time picture taken in milliseconds
4. Number of Face in image and n face Id's
5. Longitude
6. Latitude

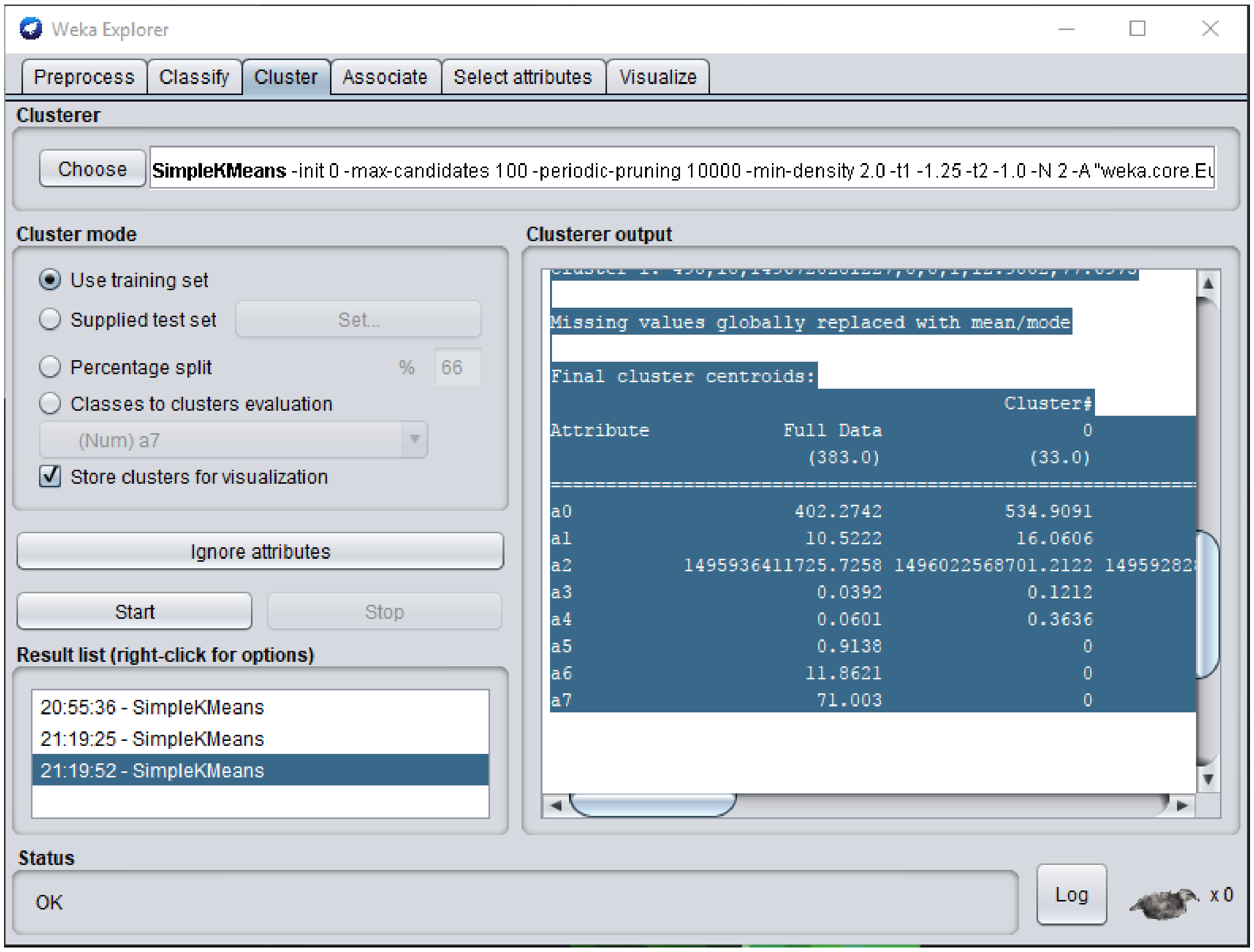
Code:

```
KMeans.java:

import java.io.BufferedReader;

import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.ArrayList;

import weka.clusterers.SimpleKMeans;
import weka.core.EuclideanDistance;
import weka.core.Instances;

public class KMeans {
      public static BufferedReader readDataFile(String filename) {
            BufferedReader inputReader = null;
            try {
                  inputReader = new BufferedReader(new FileReader(filename));
            } catch(FileNotFoundException e) {
                  System.err.println("File not found: " + filename);
            }
            return inputReader;
      }

      /**
       * Returns score of individual content i.e. distance of content from
       * it's cluster centroid
       */
      public static ArrayList<Double> getScore(SimpleKMeans kmeans, Instances
data, int assignments[]){
            ArrayList<Double> score = new ArrayList<Double>();
            EuclideanDistance Dist =
(EuclideanDistance)kmeans.getDistanceFunction();
            Instances centroid = kmeans.getClusterCentroids();
            System.out.println();
            for(int i = 0; i < centroid.size(); i++){
                  System.out.println(centroid.instance(i));
            }
            for(int i = 0; i < assignments.length; i++){
                  score.add(100 -
(Dist.distance(centroid.instance(assignments[i]), data.instance(i)) * 100) *
0.5);
            }
            return score;
      }

      /**
       *Utility function to compute Davies Bouldin index
       */
      public static double sFunction(int index, int num_cluster, SimpleKMeans
kmeans, Instances data, int assignments[]){
            double s = 0;
            Instances centroid = kmeans.getClusterCentroids();
            EuclideanDistance Dist =
(EuclideanDistance)kmeans.getDistanceFunction();
            double c = 0;
            double distance = 0;
            for(int i = 0; i < assignments.length; i++){
                  if(assignments[i] == index){
                        c++;
                        distance += (Dist.distance(centroid.get(index),
data.get(i)));
                  }
            }
            s = (distance / c);
            return s;
      }

      /**
       *Utility function to compute Davies Bouldin index
       */
      public static double dFunction(int i, int j, SimpleKMeans kmeans){
            double d = 0;
            Instances centroid = kmeans.getClusterCentroids();
            EuclideanDistance Dist =
(EuclideanDistance)kmeans.getDistanceFunction();
            d = Dist.distance(centroid.get(i), centroid.get(j));
            return d;
      }

      /**
       *Function to compute Davie Bouldin Index which is a performance metric
       *to evaluate clustering algorithm.
       */
      public static double davieBouldinIndex(int num_cluster, SimpleKMeans
kmeans, Instances data, int assignments[]){
            double dbIndex = 0;
            double R_i[] = new double[num_cluster];
            double max , R_ij, s_i, s_j, d_ij;
            for(int i = 0; i < num_cluster; i++){
                  max = -20000;
                  s_i = sFunction(i, num_cluster, kmeans, data, assignments);
                  for(int j = 0; j < num_cluster; j++){
                        if(i != j){
                              s_j =  sFunction(j, num_cluster, kmeans, data, 
assignments);
                              d_ij = dFunction(i, j, kmeans);
                              R_ij = (s_i + s_j) / d_ij;
                              if(max < R_ij)
                                    max = R_ij;
                        }
                  }
                  R_i[i] = max;
            }
            for(int i = 0; i < num_cluster; i++){
                  dbIndex += R_i[i];
            }
            dbIndex /= num_cluster;
            return dbIndex;
      }

      /**
       *Cluster are formed and computed here and evaluated
       */
      public static void main(String args[]) throws Exception {
            long start = System.currentTimeMillis();
            int num_cluster = 6;
            SimpleKMeans kmeans = new SimpleKMeans();
            kmeans.setSeed(10);
            kmeans.setPreserveInstancesOrder(true);
            kmeans.setNumClusters(num_cluster);
            BufferedReader datafile = readDataFile("G:/Academics/Semester 
7/Data mining and Data Warehousing/Practicals/Clustering/NewData_ARFF.arff");
            Instances data = new Instances(datafile);

            kmeans.buildClusterer(data);
            int assignments[] = kmeans.getAssignments();
            int arr[] = new int[num_cluster];
            int i=0;
            for(int clusterNum : assignments) {
                System.out.print((int)(data.instance(i).value(0)) + " ");
                  System.out.printf("Instance %d -> Cluster %d \n", i, 
clusterNum);
                i++;
                arr[clusterNum]++;
            }
           long end = System.currentTimeMillis();
           System.out.println("Time Taken: " + (end - start));
           System.out.println(davieBouldinIndex(num_cluster, kmeans, data, 
assignments));
      }
}
```

Output:

```
Clusters:
138 Instance 49 -> Cluster 4
139 Instance 50 -> Cluster 4
140 Instance 51 -> Cluster 4
141 Instance 52 -> Cluster 4
142 Instance 53 -> Cluster 4
144 Instance 54 -> Cluster 0
145 Instance 55 -> Cluster 0
146 Instance 56 -> Cluster 0
147 Instance 57 -> Cluster 0
148 Instance 58 -> Cluster 0
149 Instance 59 -> Cluster 2
150 Instance 60 -> Cluster 2
151 Instance 61 -> Cluster 2
152 Instance 62 -> Cluster 2
153 Instance 63 -> Cluster 2

Davies Bouldin Index:
0.48038928723038166
```

Weka Result:



SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.Eu

Use training set [✓] ; Store clusters for visualization [✓]

20:55:36 - SimpleKMeans
21:19:25 - SimpleKMeans
21:19:52 - SimpleKMeans

```
Missing values globally replaced with mean/mode

Final cluster centroids:
                                         Cluster#
Attribute            Full Data                  0
                       (383.0)             (33.0)
=================================================================
a0                    402.2742           534.9091
a1                     10.5222            16.0606
a2          1495936411725.7258 1496022568701.2122 14959282
a3                      0.0392             0.1212
a4                      0.0601             0.3636
a5                      0.9138                  0
a6                     11.8621                  0
a7                      71.003                  0
```

```
=== Run information ===

Scheme:       weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -
periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A
"weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10
Relation:     NewData_Processed
Instances:    383
Attributes:   8
              a0
              a1
              a2
              a3
              a4
              a5
              a6
              a7
Test mode:    evaluate on training data


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


kMeans
======

Number of iterations: 4
Within cluster sum of squared errors: 40.39696464661375

Initial starting points (random):

Cluster 0: 520,10,1496720282859,0,0,1,12.9802,77.6975
Cluster 1: 498,10,1496720281227,0,0,1,12.9802,77.6975

Missing values globally replaced with mean/mode

Final cluster centroids:
                                         Cluster#
Attribute            Full Data                  0                  1
                       (383.0)             (33.0)            (350.0)
====================================================================
a0                    402.2742           534.9091           389.7686
a1                     10.5222            16.0606                 10
a2        1495936411725.7258 1496022568701.2122 1495928288353.7515
a3                      0.0392             0.1212             0.0314
a4                      0.0601             0.3636             0.0314
a5                      0.9138                  0                  1
a6                     11.8621                  0            12.9805
a7                      71.003                  0            77.6975
```

Conclusion: Thus, KMeans has been used to cluster dataset of image metadata to predict which images are similar by clustering them in one group.

Practical 7

Aim: Implement a program for Hierarchical based Clustering using languages like JAVA/C/C++.

Tools: Java, Weka.

Procedure:

In data mining and statistics, hierarchical clustering (also called hierarchical cluster analysis or HCA) is a method of cluster analysis which seeks to build a hierarchy of clusters. Strategies for hierarchical clustering generally fall into two types:[1]
Agglomerative: This is a "bottom up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
Divisive: This is a "top down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.
In general, the merges and splits are determined in a greedy manner. The results of hierarchical clustering are usually presented in a dendrogram.

Execution:

Dataset:

Data were extracted from images that were taken from mobile phone camera. Images meta data are extracted and provided in csv format.

Attribute Information:
1. Image Id
2. Album Id
3. Time picture taken in milliseconds
4. Number of Face in image and n face Id's
5. Longitude
6. Latitude

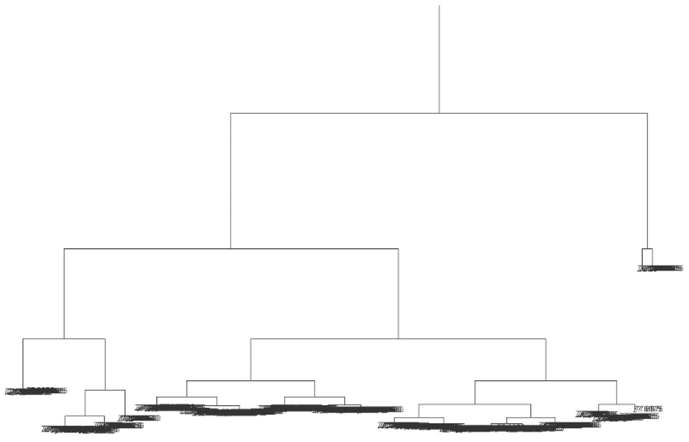
Code:

```
HierarchialClustering.java:

import java.awt.Container;
import java.awt.GridLayout;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;

import javax.swing.JFrame;

import weka.core.Instances;
```

```java
import weka.clusterers.HierarchicalClusterer;
import weka.core.EuclideanDistance;
import weka.gui.hierarchyvisualizer.HierarchyVisualizer;

public class HierarchialClustering {
      public static BufferedReader readDataFile(String filename) {
            BufferedReader inputReader = null;
            try {
                  inputReader = new BufferedReader(new FileReader(filename));
            } catch(FileNotFoundException e) {
                  System.err.println("File not found: " + filename);
            }
            return inputReader;
      }

      public static void main(String args[]) throws Exception {
            long start = System.currentTimeMillis();
            int num_cluster = 6;
            HierarchicalClusterer hc = new HierarchicalClusterer();
            hc.setNumClusters(num_cluster);
            hc.setOptions(new String[] {"-L", "COMPLETE"});
            hc.setDebug(true);
            hc.setDistanceFunction(new EuclideanDistance());
            hc.setDistanceIsBranchLength(true);

            BufferedReader datafile = readDataFile("G:/Academics/Semester
7/Data mining and Data Warehousing/Practicals/Clustering/NewData_ARFF.arff");
            Instances data = new Instances(datafile);

            hc.buildClusterer(data);
            hc.setPrintNewick(false);
            System.out.println(hc.graph());


            System.out.println("Revision: " + hc.getRevision());
            System.out.println("Graph Type: " + hc.graphType());
            System.out.println("Global Info: " + hc.globalInfo());
            long end = System.currentTimeMillis();
          System.out.println("Time Taken: " + (end - start));

          JFrame mainFrame = new JFrame("Weka Test");
            mainFrame.setSize(600, 400);
            mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            Container content = mainFrame.getContentPane();
            content.setLayout(new GridLayout(1, 1));

            HierarchyVisualizer visualizer = new
HierarchyVisualizer(hc.graph());
            content.add(visualizer);

            mainFrame.setVisible(true);
      }
}
```

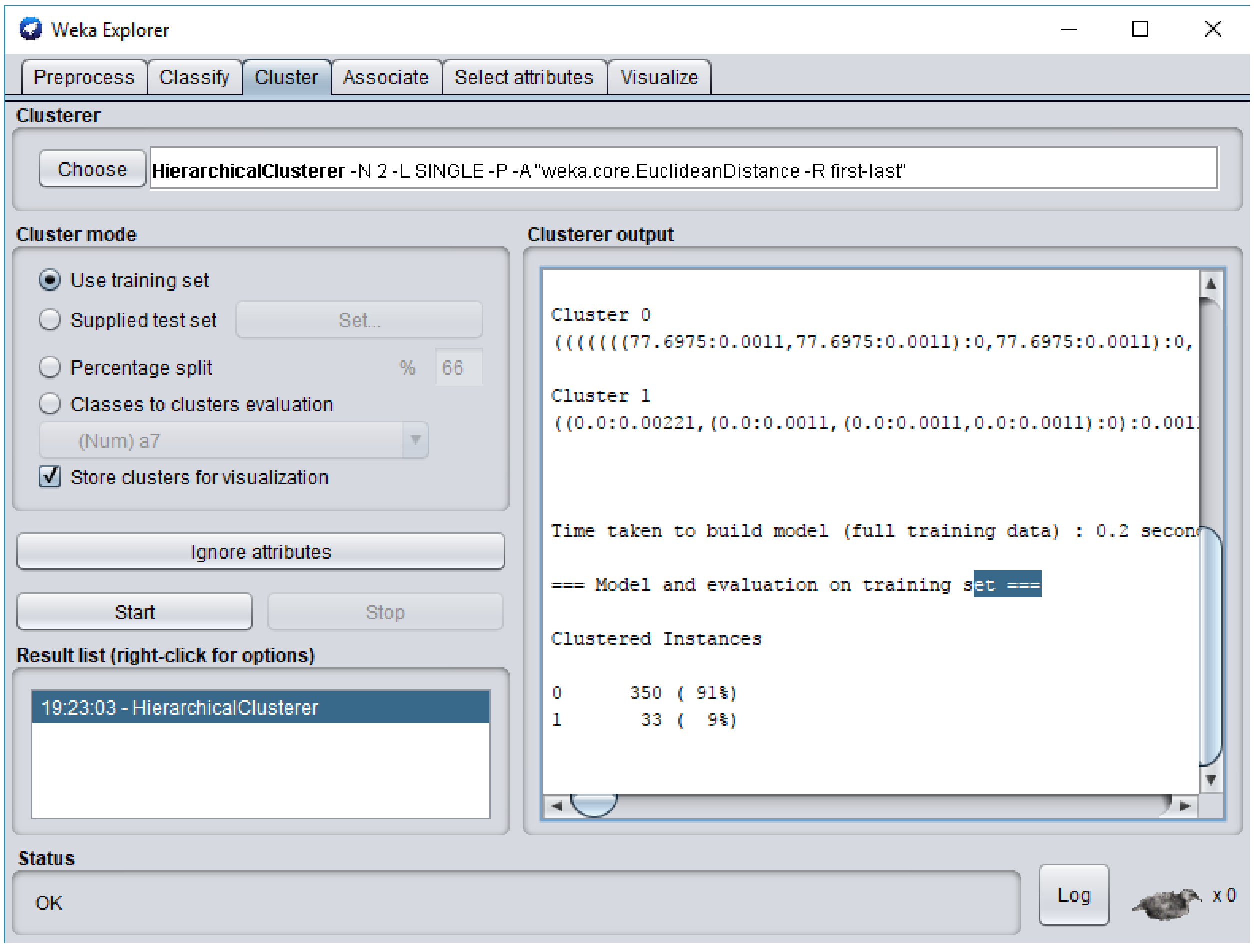
Output:

```
Debug:

Merging 377 378 0.001103752759381793 0.001103752759381793
Merging 362 363 0.001103752759381904 0.001103752759381904
Merging 366 367 0.001103752759381904 0.001103752759381904
Merging 369 370 0.001103752759381904 0.001103752759381904
Merging 371 372 0.001103752759381904 0.001103752759381904
………………
Merging 373 374 0.001103752759381904 0.001103752759381904
Merging 375 376 0.001103752759381904 0.001103752759381904
Merging 155 156 0.0011037527593839415 0.0011037527593839415
Merging 230 231 0.0011037527593842583 0.0011037527593842583
Merging 188 189 0.0011037527593851348 0.0011037527593851348
Merging 172 173 0.001103752759385796 0.001103752759385796

Output:
Revision: 13179
Graph Type: 3
Global Info: Hierarchical clustering class.
Implements a number of classic agglomerative (i.e., bottom up) hierarchical
clustering methods.
Time Taken: 544
```



Dendrogram

Weka Result:



HierarchicalClusterer -N 2 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"

● Use training set
○ Supplied test set
○ Percentage split % 66
○ Classes to clusters evaluation
(Num) a7
✓ Store clusters for visualization

19:23:03 - HierarchicalClusterer

```
Cluster 0
((((((( 77.6975:0.0011,77.6975:0.0011):0,77.6975:0.0011):0,

Cluster 1
((0.0:0.00221,(0.0:0.0011,(0.0:0.0011,0.0:0.0011):0):0.001



Time taken to build model (full training data) : 0.2 secon

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

Clustered Instances

0       350 ( 91%)
1        33 (  9%)
```

OK

```
=== Run information ===

Scheme:       weka.clusterers.HierarchicalClusterer -N 2 -L SINGLE -P -A
"weka.core.EuclideanDistance -R first-last"
Relation:     NewData_Processed
Instances:    383
Attributes:   8
              a0
              a1
              a2
              a3
              a4
              a5
              a6
              a7
Test mode:    evaluate on training data


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

Cluster 0
((77.6975:0.0011,(77.6975:0.0011,77.6975:0.0011):0):0,77.6975:0.0011):0,(77.6
975:0.0011,77.6975:0.0011):0):0):0,((((77.6975:0.0011,77.6975:0.0011):0,(77.6
975:0.0011,77.6975:0.0011):0):0,(77.6975:0.0011,77.6975:0.0011):0):0,77.6975:
0.0011):0):0,77.6975:0.0011):0):0.02613):0.00528,(77.6975:0.0011,77.6975:0.00
11):0.03141):0.03948):0.05147):0.15731):0.77336,((((77.6975:0.0011,77.6975:0.
0011):0,77.6975:0.0011):0,((77.6975:0.0011,77.6975:0.0011):0,77.6975:0.0011):
0):0.18632,(((77.6975:0.0011,(77.6975:0.0011,77.6975:0.0011):0):0,77.6975:0.0
011):0,77.6975:0.0011):0.18632):0.86671))

Cluster 1
((0.0:0.00221,(0.0:0.0011,(0.0:0.0011,0.0:0.0011):0):0.0011):1.41201,((((0.0:
0.39372,(((0.0:0.00634,0.0:0.00634):0.03641,0.0:0.04275):0.2193,(0.0:0.0011,(
0.0:0.0011,0.0:0.0011):0):0.26095):0.13166):0.1148,(0.0:0.17922,((((0.0:0.001
1,0.0:0.0011):0,(((0.0:0.0011,0.0:0.0011):0,0.0:0.0011):0,0.0:0.0011):0):0.00
11,(0.0:0.0011,0.0:0.0011):0.0011):0,((0.0:0.0011,0.0:0.0011):0,(((0.0:0.0011
,0.0:0.0011):0,(0.0:0.0011,0.0:0.0011):0):0,((0.0:0.0011,0.0:0.0011):0,((0.0:
0.0011,0.0:0.0011):0,0.0:0.0011):0):0):0):0.0011):0.17701):0.32931):0.03618,0
.0:0.5447):0.44366,0.0:0.98836):0.42585)



Time taken to build model (full training data) : 0.2 seconds

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

Clustered Instances

0      350 ( 91%)
1       33 (  9%)
```

Conclusion: Thus, Hierarchical Clustering has been used to cluster dataset of image metadata to predict which images are similar by clustering them in one group.