Jhulelal Institute of Technology
Nagpur

Jhulelal Institute of Technology

Department of Computer Science and Engineering

Off Koradi Road, Lonara, Nagpur Tele: +91-712-2668234, +91-712-2668235
Email: cse@jit.org.in Visit us at: http://www.jitnagpur.edu.in/

JHULELAL INSTITUTE OF TECHNOLOGY

Department of Computer Science & Engineering

Database Management System
Lab Manual

Name:
Branch:

College Vision

To become an eminent institution through knowledge and research.

College Mission

To produce world class engineers with academic and moral excellence who are not only equipped with cutting edge technology skills but also possess immense sense of social responsibility.

To inculcate awareness and acceptance of ethical values through co-curricular activities for overall development of students.

Department Vision

To become as a one of the best technology department through education, development of technical skills and collaborative research.

Department Mission

The mission of the department is,

1) To provide quality education to students.
2) To grow technically and give more knowledge for the betterment of mankind.
3) To develop e-awareness in students and society in general.

Department PEO's

1) To develop an ability to analyze the software, understand the technical specifications, design and provide novel engineering solutions and efficient product designs.

2) To develop professional skills that prepares them for immediate employment and for life-long learning in advanced areas of computer science and related fields.

3) To develop necessary communication skills to bridge the gap between advanced technology and end users in practice of computer science products.

4) To develop technical skills to adapt to an ever-changing professional environment

LAB COURSE OBJECTIVES:

- This course explains the concept of Database Management System.
- Implement data definition language for creating, altering and dropping table
- Learn to use the data manipulation language for inserting, selecting, updating and deleting.
- How to use Various Join operations
- Learn to handle failure and recovery under database recovery techniques.
- Use of SQL * Plus operations.

COURSE OUTCOMES: Database Management Systems

After completion of this course the students will be able -

SNO	DESCRIPTION	BLOOM'S TAXONOMY LEVEL
CO303P.1	Define data, understand the basic concepts of database management system, classify data modelling techniques, construct ER Model and to utilize the different database languages for constructing the database.	(Level 1, 2, 3)
CO303P.2	Define the concept of Relational data model, illustrate the Relational Operations from Set Theory, apply Relational Algebra and Relational Calculus to represent the SQL queries	(Level 1,2,3).
CO303P.3	Define the concept of index, classify the index, illustrate the Definition of Functional Dependencies, Apply, analyze and determine the different normalization techniques to design the normalized database.	(Level 1,2,3,4,5,6)
CO303P.4	Understand the complete process of Query processing, query evaluation plans, classify the query optimization techniques and to choose the best technique for optimizing the query.	(Level 1,2,3)
CO303P.5	Understand the concept of transactions, concurrency control, classify the different Database recovery and locking mechanism. apply the different locking mechanism on transaction. Understand the concept of Deadlock and compare the avoidance and prevention methods.	(Level 1,2,3)
CO303P.6	classify, solve and analyze the different failure classification, classify the advanced recovery mechanism. to understand the advanced concepts related to databases	(Level 1,2,3,4)

Lab Instructions:

- Make entry in the Log In out register once you enter in the Laboratory.
- Students are supposed to occupy the machines allotted to them and are not supposed to talk or make noise in the lab. The allocation is put up on the lab notice board
- All the students are supposed to enter the terminal number in the Log In out register.
- Do not change the terminal on which you are working.
- Strictly follow the instructions given by the Practical incharge / Lab. Instructor.
- Take permission before entering in the lab and keep your belongings in the racks outside the lab.
- NO FOOD, DRINK, IN ANY FORM is allowed in the lab.
- SILENT- CELL PHONES! If you need to use it, please keep it in bags.
- Do not misbehave in the computer laboratory. Work quietly.
- Keep your files in organized manner.
- Don’t change settings or password and surf safely.
- Do not reboot, turn off, or move any workstation or PC.
- Do not load any software on any lab computer (without prior permission of Faculty and Technical Support Personnel). Only Lab Operators and Technical Support Personnel are authorized to carry out these tasks.
- Do not reconfigure the cabling/equipment without prior permission.
- Do not play games on systems.
- Turn off the machine once you are done using it.
- Turn off lights and fans once lab is finished.
- Violation of the above rules and etiquette guidelines will result in disciplinary action.
- Equipment in the lab for the use of student community. Students need to maintain a proper decorum in the computer lab. Students must use the equipment with care. Any damage is caused is punishable.
- Lab can be used in free time / lunch hours by the students who need to use the systems should take prior permission from the lab in-charge

LIST OF PRACTICALS

Sr.No	Practicals	CO's	PO	PSO's
1	To Study of Relational Database Management System (RDBMS) Architecture.	CO1	PO1, PO2, PO3, PO9, PO11, PO12	PSO-1, PSO-2
2	TO Study SQL and their Features.	CO1	PO1, PO2, PO3, PO9, PO11, PO12	PSO-1, PSO-2
3	To use Data Definition Language for creating , altering and dropping the table object in a database.	CO1,CO2	PO1, PO2, PO3, PO9, PO11, PO12	PSO-1, PSO-2
4	To use the data manipulation language for inserting, selecting, updating anddeleting the data in the database table.	CO1,CO3	PO1, PO2, PO3, PO9, PO11, PO12	PSO-1, PSO-2
5	To use Order By, Group By and Having Clause in a database.	CO3	PO1, PO2, PO3, PO9, PO11, PO12	PSO-1, PSO-2
6	SQL *PLUS FUNCTIONS	CO3	PO1, PO2, PO3, PO9, PO11, PO12	PSO-1, PSO-2
7	To use comparison operators in SQL	CO4	PO1, PO2, PO3, PO9, PO11, PO12	PSO-1, PSO-2
7	To write and perform DCL commands	CO4	PO1, PO2, PO3, PO9, PO11, PO12	PSO-1, PSO-2
8	To use Group function in a database	CO5	PO1, PO2, PO3, PO9, PO11, PO12	PSO-1, PSO-2
9	To use Transaction control language(TCL) commands	CO5	PO1, PO2, PO3, PO9, PO11, PO12	PSO-1, PSO-2
10	To execute the use of Join commands	CO1	PO1, PO2, PO3, PO9, PO11, PO12	PSO-1, PSO-2
Additional Practicals				
11	To execute basic commands of Python	CO1	PO12	PSO1

Practical-01

AIM

To Study of Relational Database Management System (RDBMS) Architecture.

OBJECTIVE

- Know about databases and its implementation in real life.

THEORY

A DBMS is a software which is used to store, retrieve and manage data .It also provide data manipulation techniques like insertion, deletion ,modification and updation. It performs the operations like defining, revising, creating and controlling the database. It has to provide some uniform methods independent of a specific application for retrieving the information that is stored.

RDBMS is a Relational Data Base Management System which is an advanced version of DBMS. This stores the data in the form of tables. with reference relationships between the them. The table contains rows called as tuples and columns called as domains

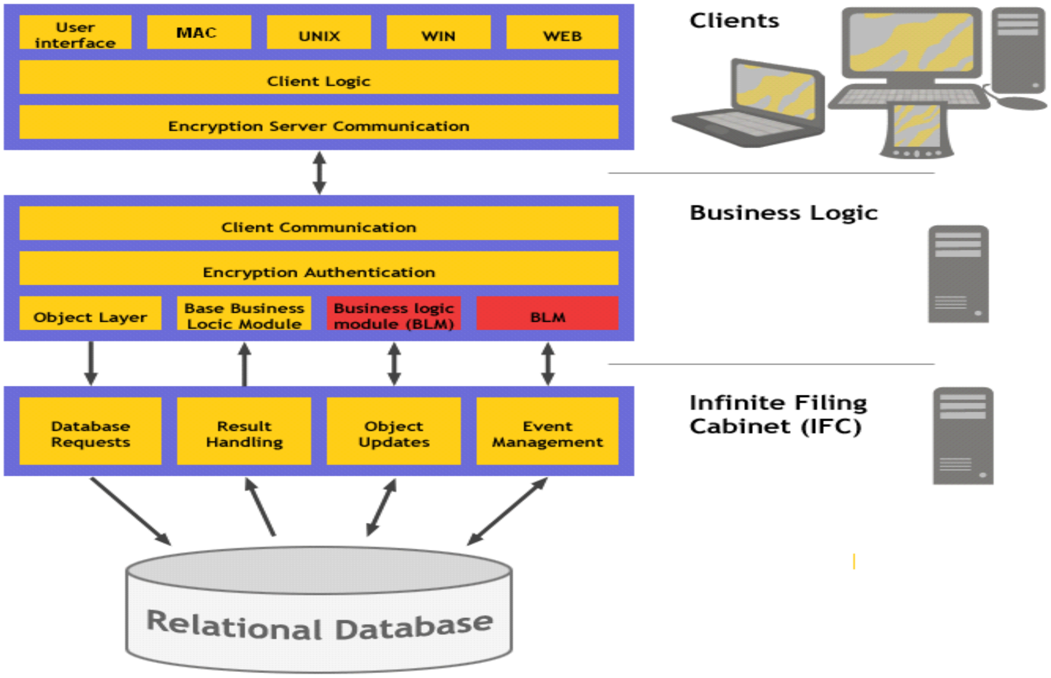


Figure: Architecture of Relational Database Management System (RDBMS)

Parameter	DBMS	RDBMS
Storage	It stores data in the form of file	It stores data in the form of tables.
Database Structure	It stores data in the hierarchical form.	It stores data in tabular form containing tuples and domains
Number of users	It supports Single User only.	It supports multiple users.
ACID	It may or may not follow ACID properties.	It has to follow ACID properties(Atomicity, Consistency,Isolation, Durability)
Hardware and software needs	It requires low Software and Hardware Requirements.	It requires high Software and Hardware Requirements.
Integrity constraints	Constraints are not imposed at the file level	It imposes constraints at the schema level.
Normalization	It does not support normalization	It supports normalization
Dr. E.F. Codd Rules	It supports 3 Codd’s rules	It supports 12 Codd’s rules
Examples	XML, Windows Registry	MySQL,Oracle,SQL Server

CONCLUSION

Thus we have studied and understood DBMS Architecture.

Viva Voce Questions

1. What is DBMS used for?
2. List the advantages of DBMS.
3. What is meant by a Database?
4. What are the different levels of abstraction in the DBMS?
5. What is RDBMS?

Practical-02

AIM: TO Study SQL and their Features.

OBJECTIVE

- To study Structured Query Language (SQL).

THEORY:

SQL (Structured Query Language) is a database query language designed for the storing and management of data in RDBMS, database schema creation, editing and manipulating. SQL is used to manage tuple and domain access control within a RDBMS, and database SQL was introduced by Dr. E.F. Codd.SQL is a non procedural programming language for querying and modifying data and managing databases. SQL allows the accessing, insertion, updating, and deletion of data.

A database management system also includes database management and database administrative functions. SQL uses command line interface. SQL is the general language used to interact with ralational database management systems.

SQL have two forms of languages DDL and DML.

Advantages of SQL :

- This allows users to retrieve data in relational database management.
- This allows users to explain the data.
- This allows to define and manipulate the data.
- This allows to create and delete databases and a tables.
- This allows users to create view and set permissions on tables.

HISTORY OF SQL :

1970-The first version of SQL was developed at IBM by Donald D. Chamberlin and Raymond F. Boyce in the early 1970s. This version, initially called SEQUEL, was designed to manipulate and retrieve data stored in IBM's original relational database product.

1974-Structured query language came into existance.

1978- System/R released following Codd’s rules.

1986-, IBM developed the first prototype of relational database and standardized by the American National Standards Institute (ANSI) as SQL-86.

SQL PROCESS:

While executing the SQL Command the following components play an important role-

- Query Dispatcher
- Optimization Engines
- Classic Query Language
- SQL Query Engine, etc

Common criticisms of SQL include a perceived lack of cross-platform portability between vendors, inappropriate handling of missing data (see Null (SQL)), and unnecessarily complex and occasionally ambiguous language grammar and semantics.

FEATURES OF SQL:

SQL is both an easy-to-understand language and a comprehensive tool for managing data. Some of the major features of SQL are

- Vendor independence
- Portability across computer systems
- SQL standards
- High-level, English-like structure
- Interactive, ad hoc queries
- Programmatic database access
- Multiple views of data
- Complete database language
- Dynamic data definition
- Client/server architecture
- Enterprise application support
- Extensibility and object technology
- Internet database access
- Java integration (JDBC)

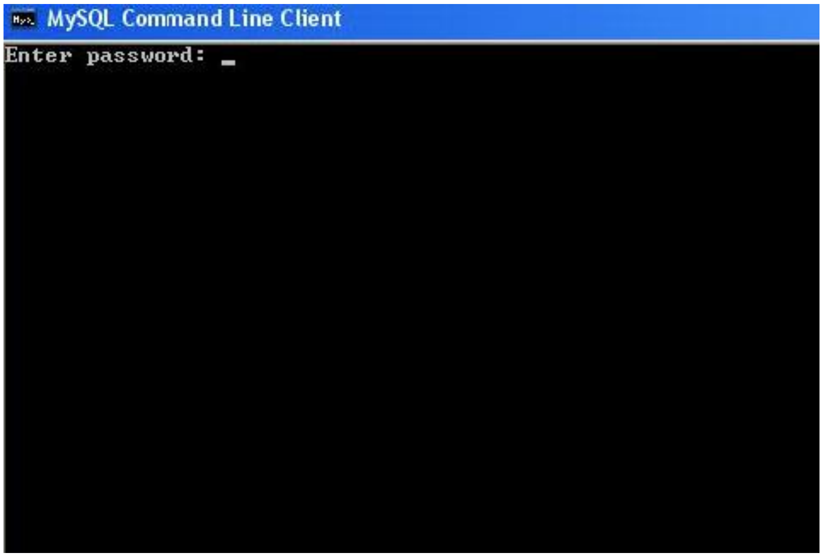
To start MySQL you would:

1. Select the Start button

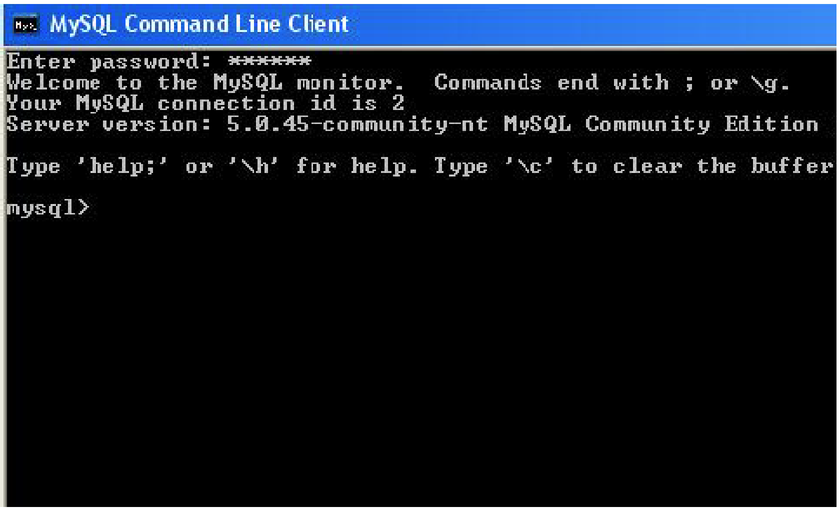
2. Select All Programs and then MySQL

3. Select MySQL Server 5.0

4. Click on the MySQL Command line client



```
Enter password: _
```



```
Enter password: *******
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.0.45-community-nt MySQL Community Edition

Type 'help;' or '\h' for help. Type '\c' to clear the buffer

mysql>
```

Once you have successfully logged on you will see the opening screen as shown in Figure 2. To work in MySQL requires you to type in commands. For example typing in the following will show you a list of help commands shown in Figure 3:

```
mysql> \h
```

CONCLUSION

Thus we have studied and understood SQL features.

Viva Voce Questions

- What is SQL?
- What are the features of SQL?
- What are the various SQL languages

Practical-03

AIM

To use Data Definition Language for creating , altering and dropping The table object in a database .

OBJECTIVE

- To understand the structure of databases
- To create ,alter and drop the tables

THEORY

Structured query language:-

SQL is a structured query language used for retrieving data in a relational database.

DDL- Data Definition Language:-

DDL commands are the SQL commands you use to create, alter, remove different database objects in an MySQL/ORACLE database.

Table Definition:-

A database object is something created and sorted in a databases. Tables, views, synonyms indexes, sequences, clusters are all types of database objects.

A table is a unit of the storage that holds data in the form of rows and columns.

1. CREATE TABLE

2. ALTER TABLE

3. DROP TABLE

4. TRUNCATE TABLE

Create table command:-is used to create the database, tables, views, procedures and triggers.

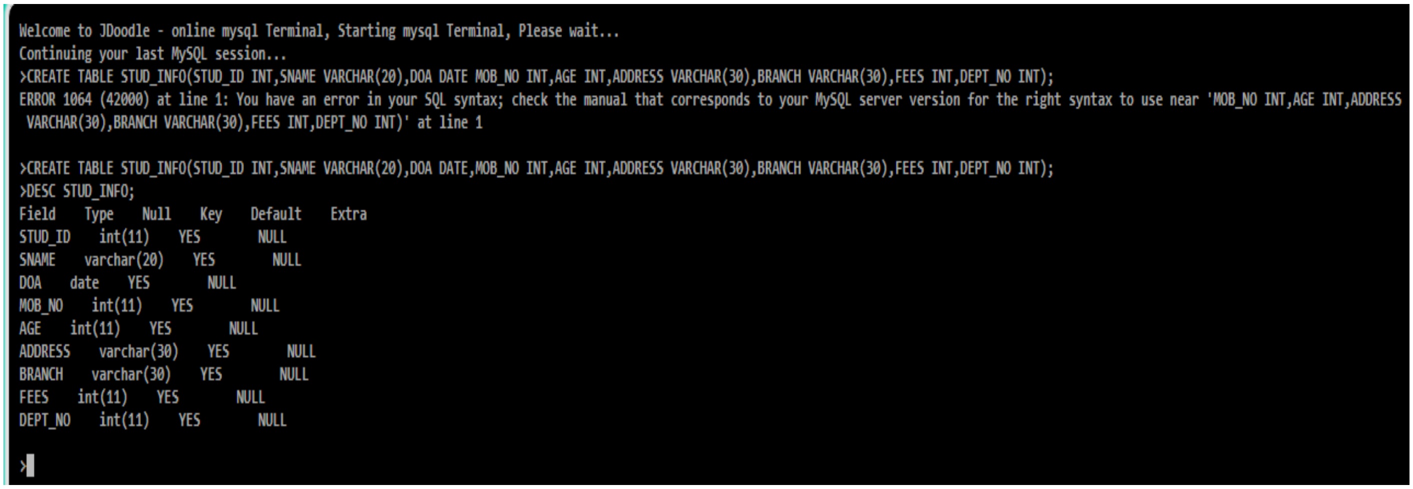
Syntax

```
Create table <  table_name>
( c1 datatype(size),
  c2 datatype(size),
   :               :
    cn datatype(size));
```

e.g. To create a table STUDENT_INFO with column STUD_ID, SNAME, DOA, MOB_NO,AGE,ADDRESS,BRANCH,FEES,DEPTNO)

```
Create table STUDE_INFO

(STUD_ID  INT, SNAME VARCHAR(20), DOA DATE, MOB_NO INT,

 AGE INT, ADDRESS VARCHAR(30),BRANCH VARCHAR(30),

FEES INT,DEPT_NO INT);
```

OUTPUT:



```
Welcome to JDoodle - online mysql Terminal, Starting mysql Terminal, Please wait...
Continuing your last MySQL session...
>CREATE TABLE STUD_INFO(STUD_ID INT,SNAME VARCHAR(20),DOA DATE MOB_NO INT,AGE INT,ADDRESS VARCHAR(30),BRANCH VARCHAR(30),FEES INT,DEPT_NO INT);
ERROR 1064 (42000) at line 1: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'MOB_NO INT,AGE INT,ADDRESS
 VARCHAR(30),BRANCH VARCHAR(30),FEES INT,DEPT_NO INT)' at line 1

>CREATE TABLE STUD_INFO(STUD_ID INT,SNAME VARCHAR(20),DOA DATE,MOB_NO INT,AGE INT,ADDRESS VARCHAR(30),BRANCH VARCHAR(30),FEES INT,DEPT_NO INT);
>DESC STUD_INFO;
Field    Type    Null    Key    Default    Extra
STUD_ID    int(11)    YES        NULL
SNAME    varchar(20)    YES        NULL
DOA    date    YES        NULL
MOB_NO    int(11)    YES        NULL
AGE    int(11)    YES        NULL
ADDRESS    varchar(30)    YES        NULL
BRANCH    varchar(30)    YES        NULL
FEES    int(11)    YES        NULL
DEPT_NO    int(11)    YES        NULL

>
```

The above SQL statement will create a table STUD_ID with the given columns to view the structure of the table created use the DESCRIBE COMMAND.

The result of the command is to see the column names and data types.

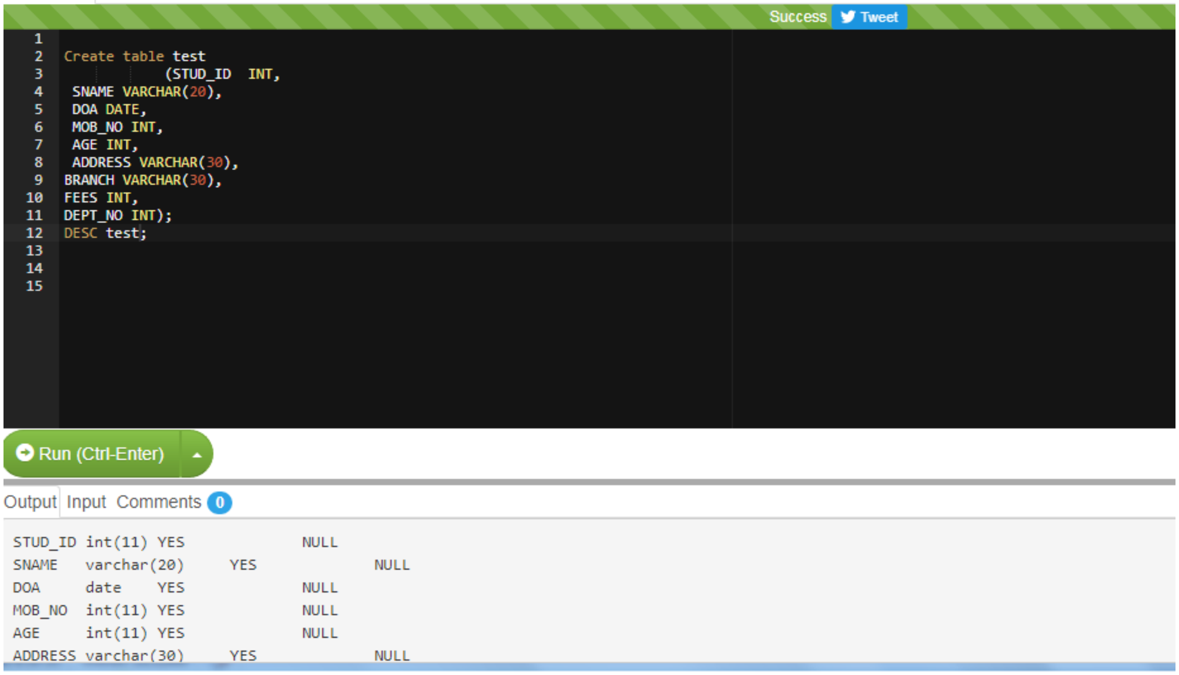
Syntax:-

```
DESC  <table_name>;
```

e.g.

```
DESC STUD_INFO;
```

OUTPUT :



```

Create table test
             (STUD_ID  INT,
 SNAME VARCHAR(20),
 DOA DATE,
 MOB_NO INT,
 AGE INT,
 ADDRESS VARCHAR(30),
BRANCH VARCHAR(30),
FEES INT,
DEPT_NO INT);
DESC test;
```

```
STUD_ID int(11) YES             NULL
SNAME   varchar(20)     YES             NULL
DOA     date    YES             NULL
MOB_NO  int(11) YES             NULL
AGE     int(11) YES             NULL
ADDRESS varchar(30)     YES             NULL
```

Restrictions for creating a table:

1.	Always start table name and column name with a letter
2.	Table names and column names can be 1 to 30 characters long
3.	Table names must contain only the characters A-Z, a-z,0-9,underscore _,$ and #.
4.	Table name should be unique
5.	Table name must not be an ORACLE reserved word
6.	Column name should be unique

Alter table command:

Syntax:-

```
Case 1:          alter table <table_name>

               add (c1 datatype, c2 datatype,

                        :            :

                        :            :

                     cn _namen datatype);



Case 2:        alter table<table_name>

               modify (c1 datatype,  c2 datatype,

                          :            :

                          :            :

                      cn _namen datatype);
```

After you create a table, you may need to change the table structures once you omitted a column or your column definition needs to be changed. Using the ALTER TABLE statement you can do it.

You can add columns to a table using the ALTER TABLE statement with the ADD clause.

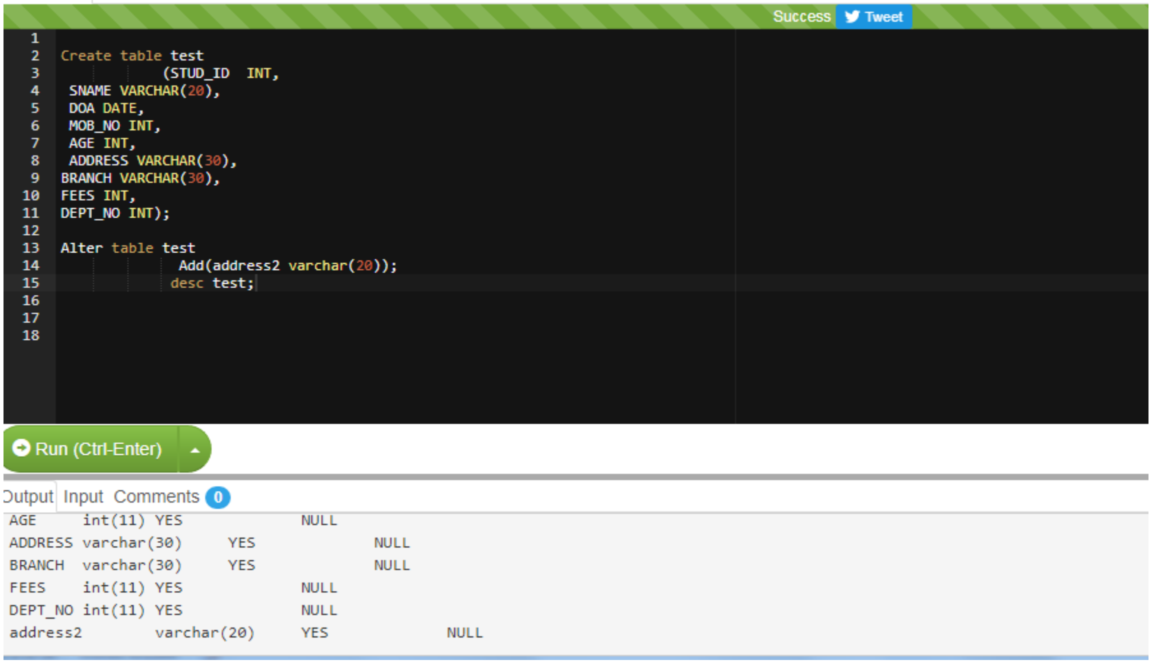
e.g

To add a column ADDRESS2 to the table STUDENT_INFO.

Alter table STUDENT_INFO

Add(address2 varchar(20));

OUTPUT :



```
Create table test
            (STUD_ID  INT,
 SNAME VARCHAR(20),
 DOA DATE,
 MOB_NO INT,
 AGE INT,
 ADDRESS VARCHAR(30),
BRANCH VARCHAR(30),
FEES INT,
DEPT_NO INT);

Alter table test
              Add(address2 varchar(20));
             desc test;
```

```
AGE     int(11) YES             NULL
ADDRESS varchar(30)     YES             NULL
BRANCH  varchar(30)     YES             NULL
FEES    int(11) YES             NULL
DEPT_NO int(11) YES             NULL
address2        varchar(20)     YES             NULL
```

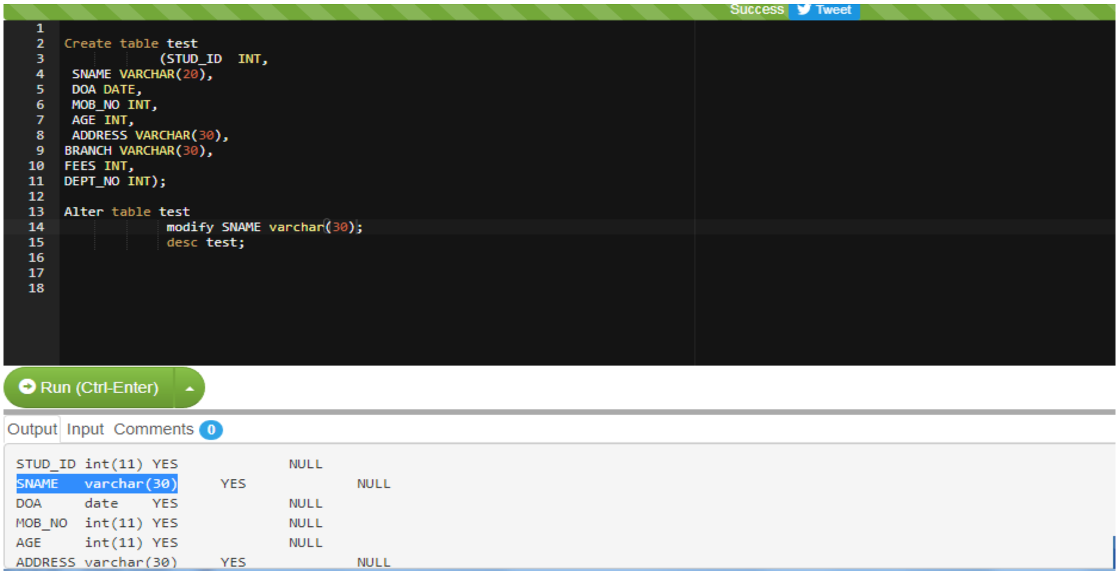
MODIFY Clause is used with the ALTER TABLE statement.

e.g To modify the length of the SNAME field to 30 in the STUDENT_INFO table.

Alter table STUDENT_INFO

modify (SNAME varchar(30));

OUTPUT :



```

Create table test
            (STUD_ID  INT,
 SNAME VARCHAR(20),
 DOA DATE,
 MOB_NO INT,
 AGE INT,
 ADDRESS VARCHAR(30),
BRANCH VARCHAR(30),
FEES INT,
DEPT_NO INT);

Alter table test
             modify SNAME varchar(30);
             desc test;
```

```
STUD_ID int(11) YES              NULL
SNAME   varchar(30)     YES              NULL
DOA     date    YES              NULL
MOB_NO  int(11) YES              NULL
AGE     int(11) YES              NULL
ADDRESS varchar(30)     YES              NULL
```

Result of the above commands can be seen by describing the table.

Restrictions:

•You can add or modify columns, but you cannot drop them from a table.
•The new column becomes last column by default.
•You can increase the width or precision of numeric column.
•You can change the datatype if the column contains only null values or if the table has no rows.
•Decrease the width of a column if the column contains null values or if the table has no rows.
•You can convert a CHAR column to the VARCHAR2 datatype or convert

VARCHAR column to CHAR datatype if the column contains null values or if you do not change the size.

Drop table command

The drop table command removes the definition(structure) of an Oracle table. When drop table command is used , the database loses all the data in the table and all the indexes concerned with it.

Syntax:-

```
drop table<table_name>;
```

e.g.

To change the name of the table STUDENT_INFO to STUDENTTBL

```
rename STUDENT_INFO to STUDENTTBL;
```

CONCLUSION

Thus we have studied and performed the above mysql commands.

Viva Voce Questions

1. Define instance and schema.

2. What is conceptual schema?

3. What is relationship? Give examples

4. Define weak and strong entity sets

Practical-04

AIM

To use the data manipulation language for inserting, selecting, updating and deleting the data in the database table.

OBJECTIVE

- Use of insert, select, update & delete.

THEORY

Data manipulation commands are

- Insert
- Select
- Update
- Delete

Insert command

Syntax:-

```
Insert into<table_name>
   (c1, c 2,   :        :        :        :     c n)
          values
                (exp1,  exp 2,  :    :    :    :       exp n);
```

Example:

To insert a row into the STUDENT_INFO table created in the previous experiment.

The above SQL statement will insert a single of data in the STUDENT_INFO table. In order

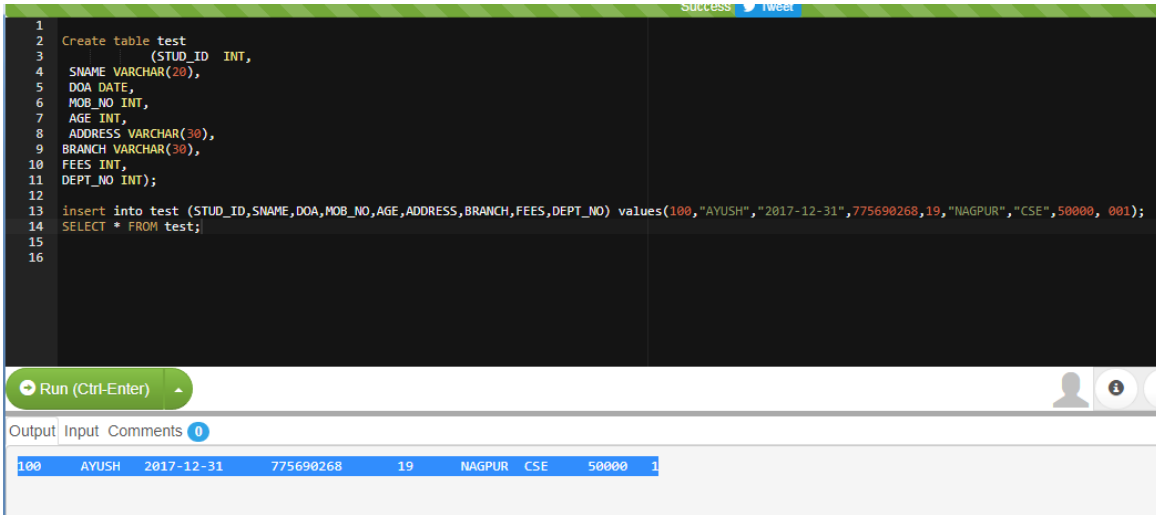
To add more data to the STUDENT_INFO table, the insert command can be used with a new set Of values each time.

Case 1: insert into STUDENT_INFO (STUD_ID,SNAME,DOA,MOB_NO,AGE,ADDRESS,BRANCH,FEES,DEPTNO)

values (101, “KARAN” , “2017-12-18” ,775690269,18, “NASHIK”, “CSE” ,50000, 001);

Case 2: insert into STUDENT_INFO values(100, “AYUSH” , “2017-12-31” ,775690268,19, “NAGPUR” , “CSE” ,50000, 001);

OUTPUT :



```

Create table test
            (STUD_ID  INT,
 SNAME VARCHAR(20),
 DOA DATE,
 MOB_NO INT,
 AGE INT,
 ADDRESS VARCHAR(30),
BRANCH VARCHAR(30),
FEES INT,
DEPT_NO INT);

insert into test (STUD_ID,SNAME,DOA,MOB_NO,AGE,ADDRESS,BRANCH,FEES,DEPT_NO) values(100,"AYUSH","2017-12-31",775690268,19,"NAGPUR","CSE",50000, 001);
SELECT * FROM test;


```

```
100     AYUSH    2017-12-31       775690268        19       NAGPUR   CSE      50000    1
```

Case 3: insert into STUDENT_INFO values(&STUD_ID, “&SNAME”, “&DOA” , &MOB_NO, &AGE, “&ADDRESS” , “&BRANCH”,&FEES, &DEPTNO);

Enter value for STUD_ID :103

Enter value for SNAME: SAMIR

Enter value for DOA: 16-10-2000

Enter value for MOB_NO: 7756902690

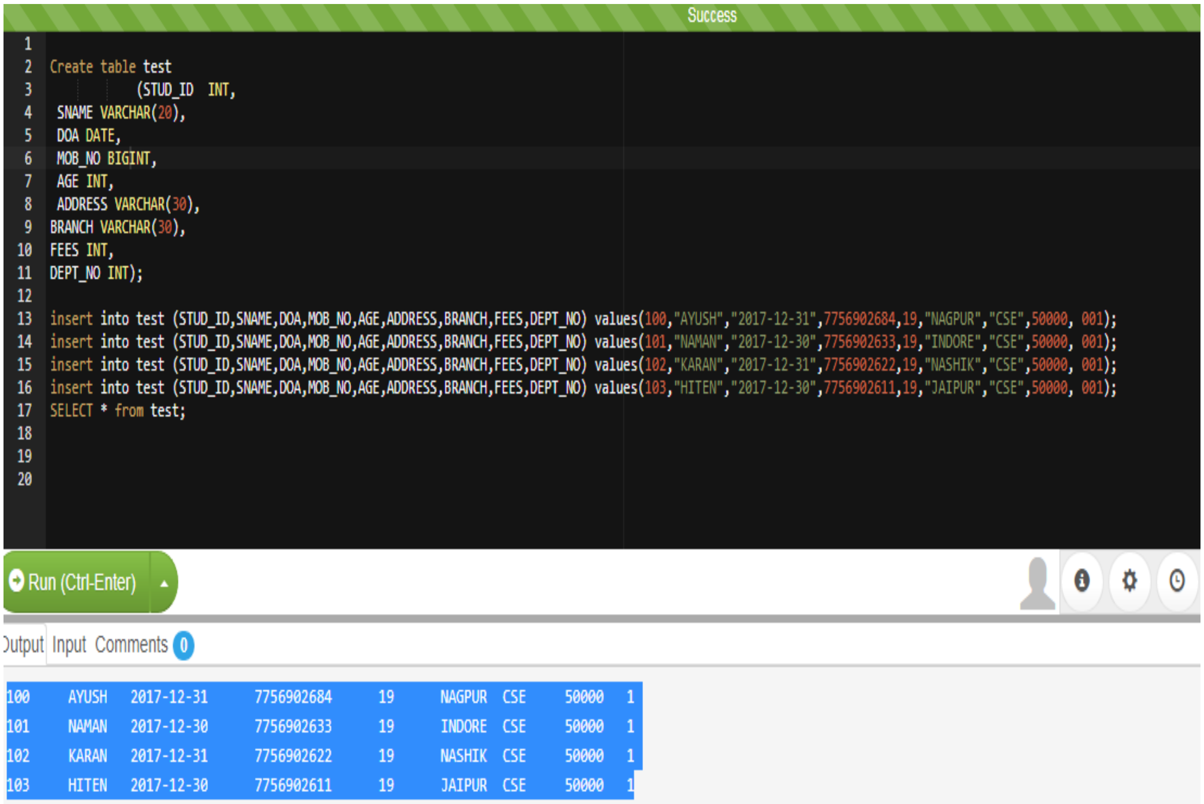
Enter value for AGE: 17

Enter value for ADDRESS: NAGPUR

Enter value for BRANCH: CSE

Enter value for FEES: 10000

Enter value for DEPTNO: 30



```

Create table test
		(STUD_ID  INT,
 SNAME VARCHAR(20),
 DOA DATE,
 MOB_NO BIGINT,
 AGE INT,
 ADDRESS VARCHAR(30),
BRANCH VARCHAR(30),
FEES INT,
DEPT_NO INT);

insert into test (STUD_ID,SNAME,DOA,MOB_NO,AGE,ADDRESS,BRANCH,FEES,DEPT_NO) values(100,"AYUSH","2017-12-31",7756902684,19,"NAGPUR","CSE",50000, 001);
insert into test (STUD_ID,SNAME,DOA,MOB_NO,AGE,ADDRESS,BRANCH,FEES,DEPT_NO) values(101,"NAMAN","2017-12-30",7756902633,19,"INDORE","CSE",50000, 001);
insert into test (STUD_ID,SNAME,DOA,MOB_NO,AGE,ADDRESS,BRANCH,FEES,DEPT_NO) values(102,"KARAN","2017-12-31",7756902622,19,"NASHIK","CSE",50000, 001);
insert into test (STUD_ID,SNAME,DOA,MOB_NO,AGE,ADDRESS,BRANCH,FEES,DEPT_NO) values(103,"HITEN","2017-12-30",7756902611,19,"JAIPUR","CSE",50000, 001);
SELECT * from test;
```

```
100    AYUSH   2017-12-31    7756902684    19    NAGPUR  CSE    50000   1
101    NAMAN   2017-12-30    7756902633    19    INDORE  CSE    50000   1
102    KARAN   2017-12-31    7756902622    19    NASHIK  CSE    50000   1
103    HITEN   2017-12-30    7756902611    19    JAIPUR  CSE    50000   1
```

Select command:

```
Syntax:-   Select col_name 1,
                  Col_name 2
           from <table_name>;
                      OR
           Select *
           from  <table_name>;
```

Note: Meta character asterisk(*) means it gives all the entries from the associatedtable.

e.g. To see the contents of table STUDENT_INFO

```
Select SNAME,DOA,MOB_NO,AGE,ADDRESS,BRANCH,FEES,DEPTNO
form STUDENT_INFO;
                                   OR
 Select *
 from STUDENT_INFO
```

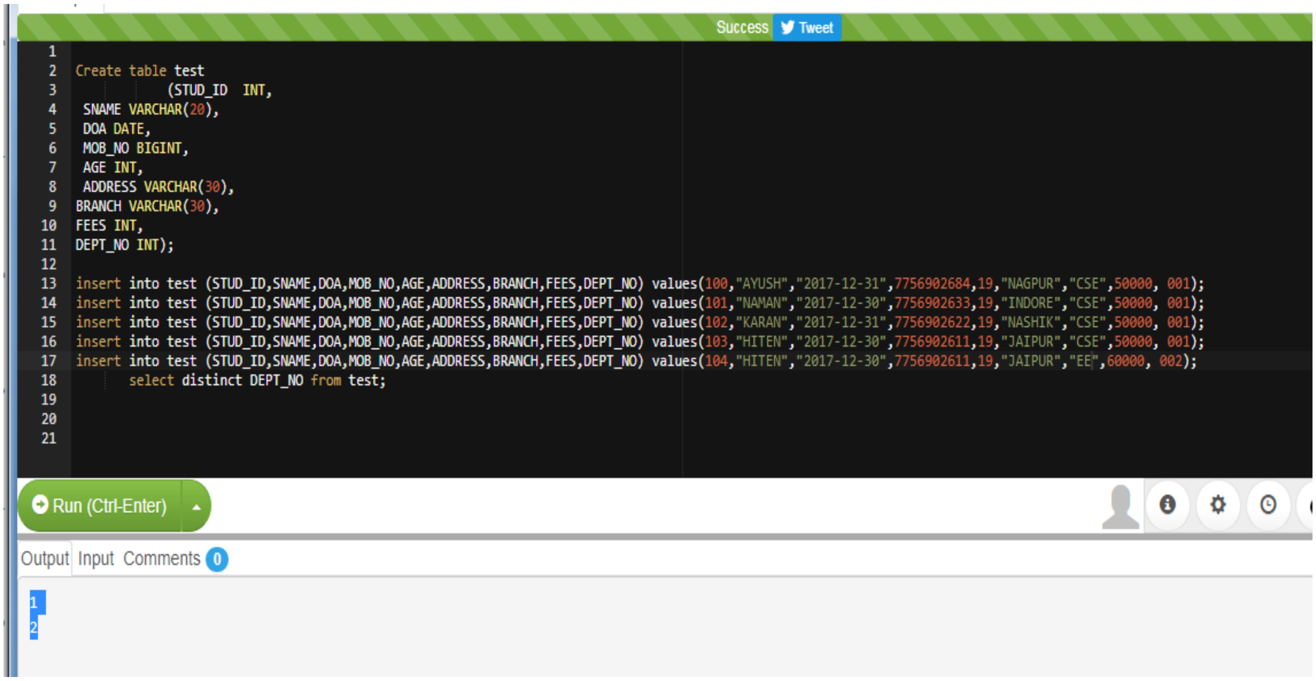
Selecting distinct rows

To prevent the selection of duplicate/redundant rows, we can use distinct clause in the select command.

e.g.

```
select distinct DEPTNO from STUDENT_INFO;
```

OUTPUT :



```

Create table test
            (STUD_ID  INT,
 SNAME VARCHAR(20),
 DOA DATE,
 MOB_NO BIGINT,
 AGE INT,
 ADDRESS VARCHAR(30),
BRANCH VARCHAR(30),
FEES INT,
DEPT_NO INT);

insert into test (STUD_ID,SNAME,DOA,MOB_NO,AGE,ADDRESS,BRANCH,FEES,DEPT_NO) values(100,"AYUSH","2017-12-31",7756902684,19,"NAGPUR","CSE",50000, 001);
insert into test (STUD_ID,SNAME,DOA,MOB_NO,AGE,ADDRESS,BRANCH,FEES,DEPT_NO) values(101,"NAMAN","2017-12-30",7756902633,19,"INDORE","CSE",50000, 001);
insert into test (STUD_ID,SNAME,DOA,MOB_NO,AGE,ADDRESS,BRANCH,FEES,DEPT_NO) values(102,"KARAN","2017-12-31",7756902622,19,"NASHIK","CSE",50000, 001);
insert into test (STUD_ID,SNAME,DOA,MOB_NO,AGE,ADDRESS,BRANCH,FEES,DEPT_NO) values(103,"HITEN","2017-12-30",7756902611,19,"JAIPUR","CSE",50000, 001);
insert into test (STUD_ID,SNAME,DOA,MOB_NO,AGE,ADDRESS,BRANCH,FEES,DEPT_NO) values(104,"HITEN","2017-12-30",7756902611,19,"JAIPUR","EE",60000, 002);
       select distinct DEPT_NO from test;
```

```
1
2
```

To retrieve a specific columns from table

Syntax:

Select col_name 1,

Col_name 2

From <table_name>;

e.g. To select only the SNAME & DOA columns from the STUDENT_INFO table, following command is used.

Select SNAME,DOA

from STUDENT_INFO;

To retrieve selected rows & all columns from a table :-

ORACLE provides the option of using a 'WHERE' clause in an SQL sentence to apply a filter on the rows in the table. When WHERE clause is added to the SQL statement, ORACLE compares each record from the table with the condition specified in the 'WHERE' clause.

Syntax:-

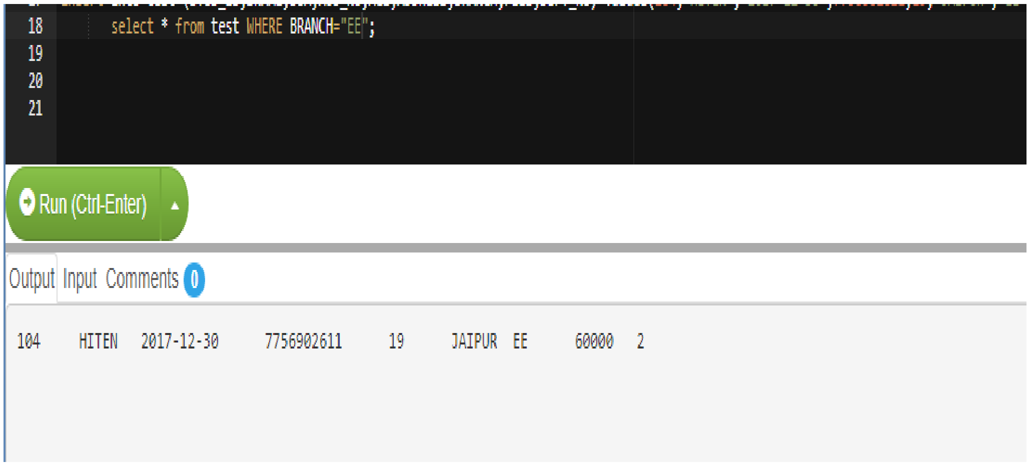
```
Select *
from <table_name>
where<condition>;
```

e.g.

To select only those rows from the STUDENT_INFO table where the 'BRANCH' is 'CSE',, following command is used.

```
  Select *

from  STUDENT_INFO

where BRANCH= 'EE';
```

OUTPUT :



```
select * from test WHERE BRANCH="EE";
```

```
104    HITEN   2017-12-30     7756902611     19     JAIPUR  EE     60000   2
```

Update command

updating of all rows:-

Syntax:-

```
Update <table_name>
 set col_name1= exp1,
        col_name1= exp2,
             :                        :
        col1= exp;
```

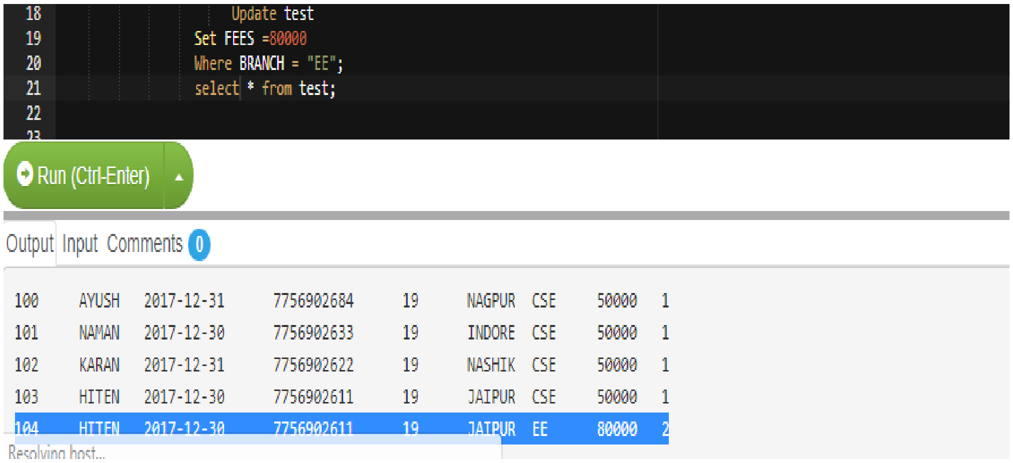
The update command is used to change or modify data values in a table.

e.g.

To change the fees of student whose branch is ‘ETC’, following command is used.

```
Update STUDENT_INFO

Set FEES =50000

Where BRANCH= ‘ETC’;
```

OUTPUT :



```
      Update test
    Set FEES =80000
    Where BRANCH = "EE";
    select * from test;
```

```
100    AYUSH    2017-12-31    7756902684    19    NAGPUR   CSE    50000   1
101    NAMAN    2017-12-30    7756902633    19    INDORE   CSE    50000   1
102    KARAN    2017-12-31    7756902622    19    NASHIK   CSE    50000   1
103    HITEN    2017-12-30    7756902611    19    JAIPUR   CSE    50000   1
104    HITEN    2017-12-30    7756902611    19    JAIPUR   EE     80000   2
Resolving host...
```

Delete command

Removal of all rows:-

Syntax:-

```
Delete from <table_name>;
```

Removal of specified rows:-

Syntax:-

```
Delete from <table_name>

Where <condition>;
```

e.g

To see whether rows have been deleted from STUDENT_INFO table use select command

```
select *

from STUDENT_INFO;
```

CONCLUSION

Thus we have studied and performed the above mysql commands.

Practical 05

AIM

To use Order By, Group By and Having Clause in a database.

OBJECTIVE

- Sort the data in the resulting query
- Apply SQL aggregate functions

THEORY

Sorting of Oracle table data

When you execute a simple query then the order of rows is undefined. To prevent this form happening, the ORDRE BY clause is used.

Grouping rows in a table.

The rows of data in a oracle table can be divided into groups by using the GROUP BY clause.

Having clause:-

The HAVING clause can be used to restrict groups from being displayed.

ORDER BY clause:-

Syntax:-

```
select <expr>

from <table_name>

[where condition(s)]
```

[order by {col, expr} [asc/ desc]];

Where

order by- specifies the order in which the rows are displayed.

asc - orders the rows in ascending order.

desc – orders the rows in descending order

e.g.

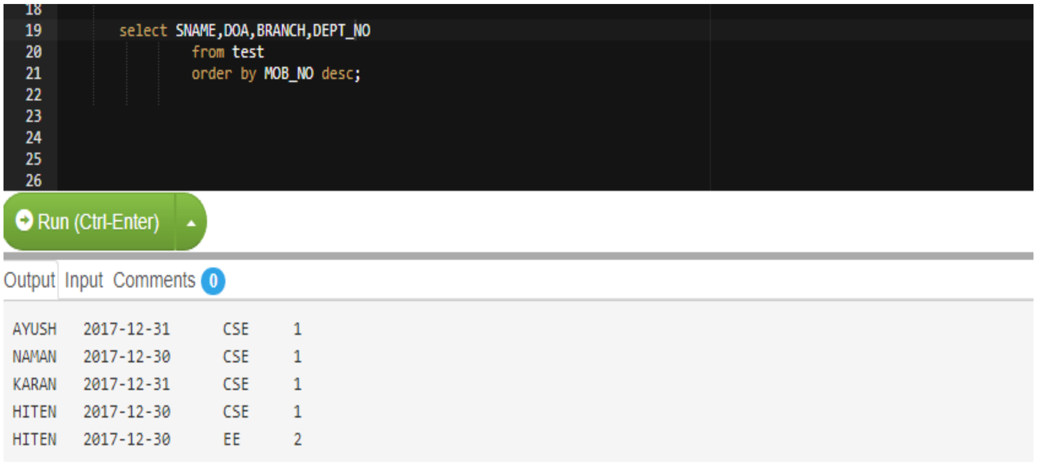
select SNAME,DOA,BRANCH,DEPTNO

from STUDENT_INFO

order by DOA desc;

The above example sorts the result by the students date of admission.

OUTPUT :



```
select SNAME,DOA,BRANCH,DEPT_NO
        from test
        order by MOB_NO desc;
```

```
AYUSH   2017-12-31      CSE     1
NAMAN   2017-12-30      CSE     1
KARAN   2017-12-31      CSE     1
HITEN   2017-12-30      CSE     1
HITEN   2017-12-30      EE      2
```

GROUP BY clause.

Syntax:-

select column, group_function

from <table_name>

[where condition]

[group by group_by_exp]

[order by col];

Where group by expression - specifies columns whose values determine the basis for grouping rows.

The GROUP BY clause can be used to divided the rows in a table into groups.   We can then use the group functions to return summary information for each group.

The GROUP BY column does not have to be in SELECT list.

In the above e.g. GROUOP BY column deptno is not in the GROUP BY clause on

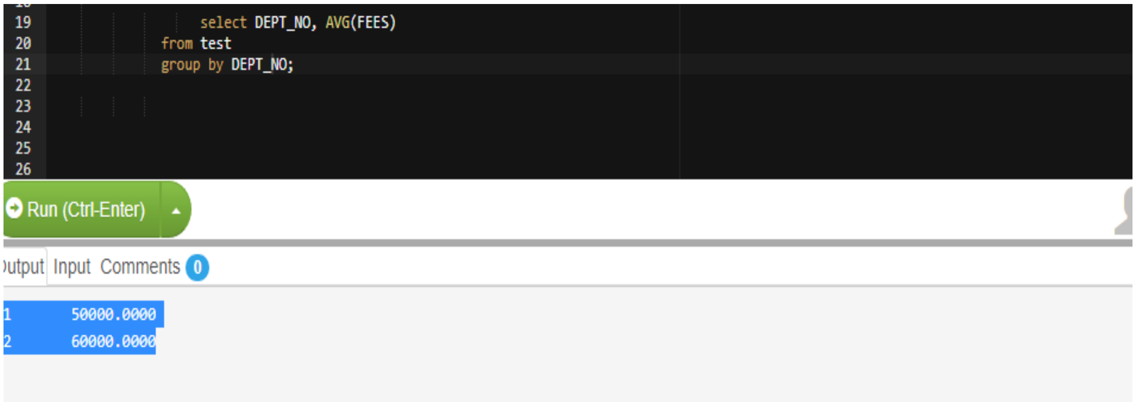
multiple columns.

select DEPTNO, AVG(FEES)

from STUDENT_INFO

group by DEPTNO;

OUTPUT :



```
        select DEPT_NO, AVG(FEES)
    from test
    group by DEPT_NO;
```

```
1       50000.0000
2       60000.0000
```

HAVING clause:-

Syntax:-

```
select col, group_function
from <table_name>
[ group by group_by_exp]
[ having group_condition]
[order by  col];
```

Where

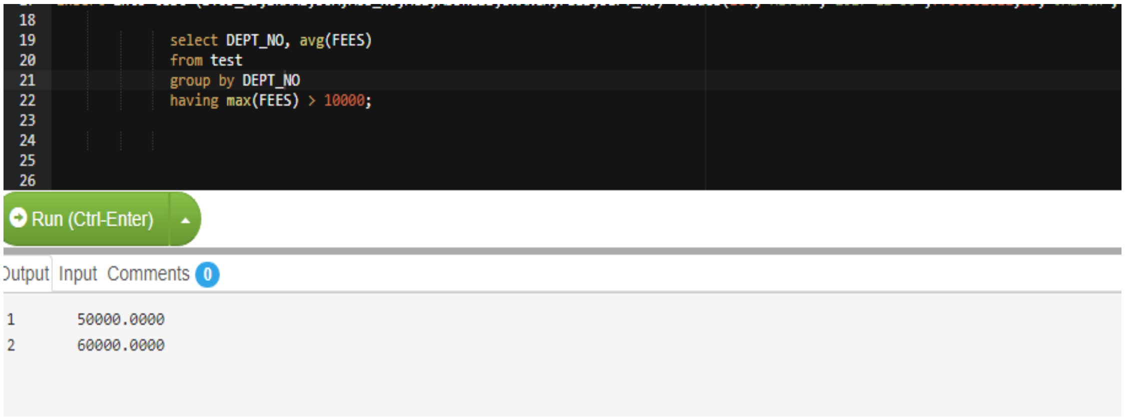
Group condition- Restricts the groups of rows returned to those groups for which the specified condition is TRUE.

We use the HAVING clause to specify which groups are displayed.

e.g.

```
select DEPTNO, avg(FEES)

from STUDENT_INFO

group by DEPTNO

having max(FEES) > 10000;
```

OUTPUT :



```
select DEPT_NO, avg(FEES)
from test
group by DEPT_NO
having max(FEES) > 10000;
```

```
1       50000.0000
2       60000.0000
```

CONCLUSION

Thus we have studied and performed the above mysql commands.

Viva voce questions

1. Define the terms i) DDL ii) DML

2. What is a candidate key?

3. What is a SELECT operation?

4. What is a primary key**?**

5. Differentiate between having and where clause.

Practical 06

AIM

SQL *PLUS FUNCTIONS

OBJECTIVE

- Learn about selected MySQL date and time functions
- Be able to perform Concat manipulations
- Perform conversions between data types

THEORY

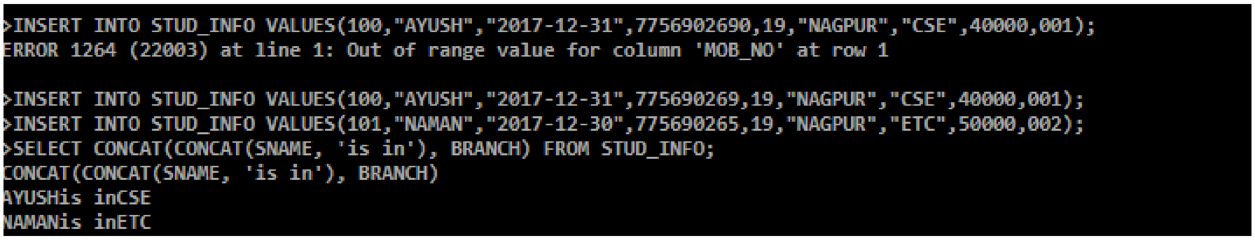
Functions are predefined sets of commands that may operate on one row or a group of rows. They are typically used in SELECT, WHERE, GROUP BY and ORDER BY Clauses.

CONCAT(char1, char2)

Returns char 1 concatenated with char 2.
Example :

Select CONCAT(CONCAT(SNAME, ‘is in’),BRANCH)
From STUD_INFO;

OUTPUT :



```
>INSERT INTO STUD_INFO VALUES(100,"AYUSH","2017-12-31",7756902690,19,"NAGPUR","CSE",40000,001);
ERROR 1264 (22003) at line 1: Out of range value for column 'MOB_NO' at row 1

>INSERT INTO STUD_INFO VALUES(100,"AYUSH","2017-12-31",775690269,19,"NAGPUR","CSE",40000,001);
>INSERT INTO STUD_INFO VALUES(101,"NAMAN","2017-12-30",775690265,19,"NAGPUR","ETC",50000,002);
>SELECT CONCAT(CONCAT(SNAME, 'is in'), BRANCH) FROM STUD_INFO;
CONCAT(CONCAT(SNAME, 'is in'), BRANCH)
AYUSHis inCSE
NAMANis inETC
```

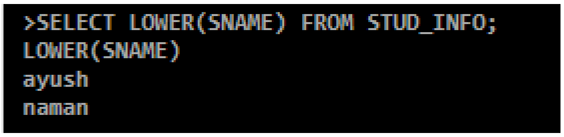
LOWER(char)

Converts strings to capitalised lowercase

Example :

Select LOWER(SNAME)
From STUD_INFO;

OUTPUT :



```
>SELECT LOWER(SNAME) FROM STUD_INFO;
LOWER(SNAME)
ayush
naman
```

LTRIM(char,set)

Removes characters from the left of char, with all the leftmost characters that appear in set removed.

Example :

```
Select LTRIM(xyxXxy LAST WORD’,’xy’)
From dual;
```

RTRIM(char,set)

Returns char, with all the rightmost characters that appear in set removed.

<u>Date and time functions</u>

This can store the time of day as well as the date within a date field, there are functions that refer to both date and time.

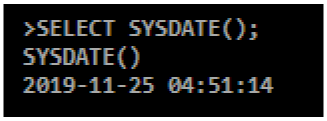
SYSDATE

Returns the current date and time. Requires no arguments.

Example :

```
Select SYSDATE();
```

OUTPUT :

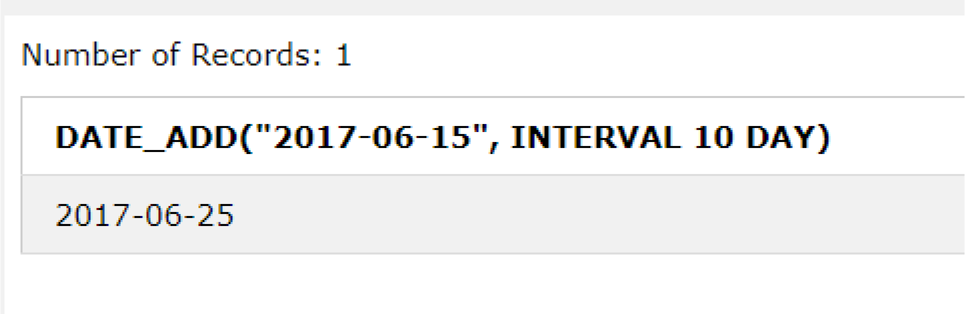


```
>SELECT SYSDATE();
SYSDATE()
2019-11-25 04:51:14
```

DATE_ADD(‘d’,n)

Add 10 days to a date and return the date:

SELECT DATE_ADD("2017-06-15", INTERVAL 10 DAY);



Number of Records: 1

DATE_ADD("2017-06-15", INTERVAL 10 DAY)
2017-06-25

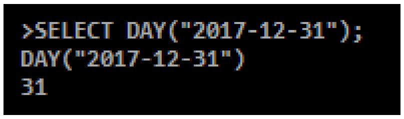
DAY(d)

Return the day of the month for a date:.

Example:

```
SELECT DAY("2017-12-31");
```

OUTPUT :



```
>SELECT DAY("2017-12-31");
DAY("2017-12-31")
31
```

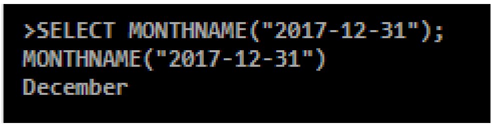
MONTHSNAME (d1)

Return the name of the month for a date:.

Example :

```
Select MONTHNAME("2017-12-31");
```

OUTPUT :



```
>SELECT MONTHNAME("2017-12-31");
MONTHNAME("2017-12-31")
December
```

CONCLUSION

Thus we have studied and performed the above mysql commands.

Viva Voce questions

1. What does SQL stand for?
2. Which SQL statement is used to extract data from a database?
3. Which SQL statement is used to update data in a database?

Practical 07

AIM

To use comparison operators in SQL .

OBJECTIVE

- Use arithmetic operators in SQL statements
- Select rows from a table with conditional restrictions
- Apply logical operators to have multiple conditions

THEORY

Comprasion operators:-

•IN ,NOT IN

•BETWEEN, NOT BETWEEN

•LIKE, NOT LIKE

•IS NULL, IS NOT NULL

•ANY, ALL

IN and NOT IN predicates

In case a value needs to be compare to a list of values, then the IN predicate is used. We can check a single value against multiple values by using the IN predicate.

IN

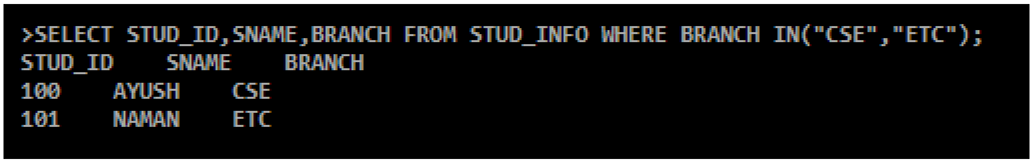
Syntax:-

```
select col_name 1, col_name2,……..,col_name n
from<table_name>
where col_name in(‘ value1’, ‘value2’,…….,’valuen’);
```

e.g.

```
select STUD_ID,SNAME,BRANCH
from STUD_INFO
where BRANCH in(‘CSE’, ‘ETC’);
```

OUTPUT :



```
>SELECT STUD_ID,SNAME,BRANCH FROM STUD_INFO WHERE BRANCH IN("CSE","ETC");
STUD_ID    SNAME    BRANCH
100    AYUSH    CSE
101    NAMAN    ETC
```

NOT IN

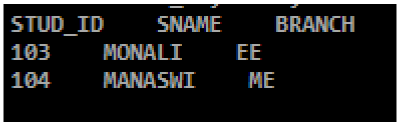
Syntax:-

select col_name 1, col_name2,……..,col_name n
from<table_name>
where col_name not in(‘ value1’, ‘value2’,…….,’valuen’);

e.g.

select STUD_ID,SNAME,BRANCH
from STUD_INFO
where BRANCH NOT IN(‘CSE’, ‘ETC’);

OUTPUT :



```
STUD_ID    SNAME    BRANCH
103    MONALI    EE
104    MANASWI    ME
```

Guidelines:

•The IN & NOT IN operators can be used with any datatype.

•If characters or dates are used in the list, they must be enclosed in single quotation marks(‘ ‘).

LIKE using %(percentage)

Syntax:-

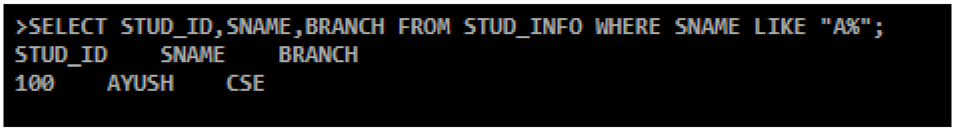
select col_name 1, column_name2,……..,col_name n
from<table_name>
where col_name like ‘value%’;

e.g.

select STUD_ID,SNAME,BRANCH
from STUD_INFO
where SNAME like ‘A%’;

The above example will display all rows from table SNAME where student name begins with ‘A’.

OUTPUT :



```
>SELECT STUD_ID,SNAME,BRANCH FROM STUD_INFO WHERE SNAME LIKE "A%";
STUD_ID    SNAME    BRANCH
100    AYUSH    CSE
```

NOT LIKE using %(percentage)

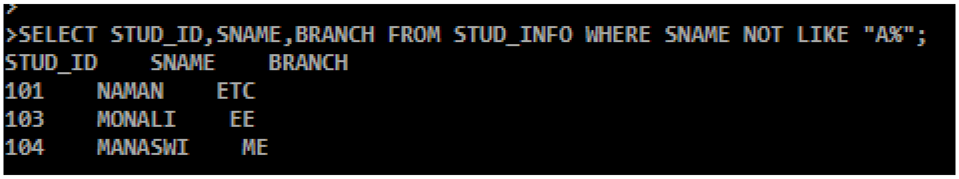
Syntax:-

select col_name 1, col_name2,……..,col_name n
from<table_name>
where col_name not like ‘value%’;

e.g.

select STUD_ID,SNAME,BRANCH
from STUD_INFO
where SNAME NOT like ‘A%’;

OUTPUT :



```
>
>SELECT STUD_ID,SNAME,BRANCH FROM STUD_INFO WHERE SNAME NOT LIKE "A%";
STUD_ID    SNAME    BRANCH
101    NAMAN    ETC
103    MONALI    EE
104    MANASWI    ME
```

The above example will display all rows from table STUD_INFO where student name not begins with ‘A’.

LIKE using ‘_’(Underscore)

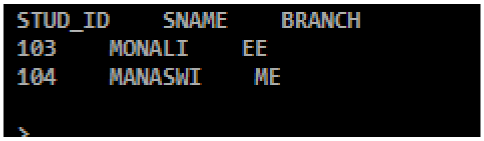
Syntax:-

select col_name 1, col_name2,……..,col_name n
from<table_name>
where col_name like‘_value%’;

e.g.

select STUD_ID,SNAME,BRANCH
from STUD_INFO
where SNAME like ‘%I’;

OUTPUT :



```
STUD_ID    SNAME    BRANCH
103    MONALI    EE
104    MANASWI    ME

>
```

The above example will display all rows where the address starts with any character but ends with ‘I’.

BETWEEN

Syntax:-

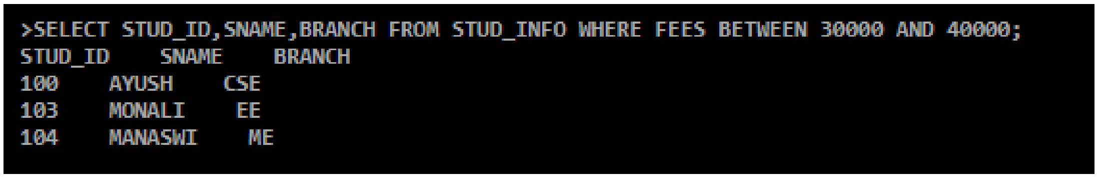
select col_name 1, col_name2,……..,col_name n
from<table_name>
where col_name between lower bound and upper bound;

You can display rows based on a range of values using BETWEEN operator. The range that you specify contains a lower bound and upper bound.

e.g.

select STUD_ID,SNAME,BRANCH
from STUD_INFO
where FEES between 1000 and 3000;

OUTPUT :



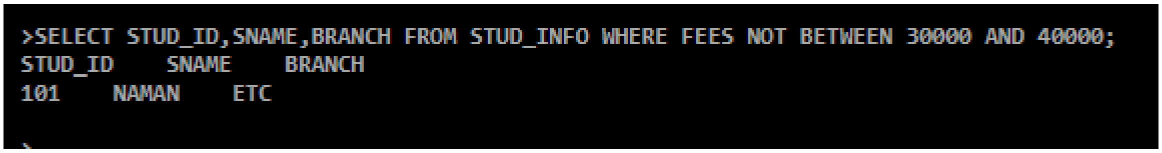
```
>SELECT STUD_ID,SNAME,BRANCH FROM STUD_INFO WHERE FEES BETWEEN 30000 AND 40000;
STUD_ID    SNAME    BRANCH
100    AYUSH    CSE
103    MONALI    EE
104    MANASWI    ME
```

NOT BETWEEN

Syntax:

select col_name 1, col_name2,……..,col_name n
from<table_name>
where col_name not between lower bound and upper bound;

OUTPUT :



```
>SELECT STUD_ID,SNAME,BRANCH FROM STUD_INFO WHERE FEES NOT BETWEEN 30000 AND 40000;
STUD_ID    SNAME    BRANCH
101    NAMAN    ETC

>
```

You can display rows based on a range of values using BETWEEN operator. The range tha you specify contains a lower bound and upper bound.

e.g.

```
select STUD_ID,SNAME,BRANCH
from STUD_INFO
where FEES NOT between 1000 and 3000;
```

Guidelines:-

•Values specified with the BETWEEN operator inclusive.

•You must specify the lower limit first.

IS NULL:-

Syntax:-

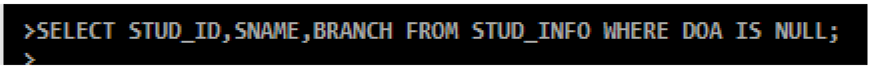
```
select col_name1,……..col_name n
from <table_name>
where col_name is null;
```

The IS NULL operator tests for values that are null. A Null value means the value is unavailable, unsigned, unknown, unequal to any value or zero length string.

e.g Display all the records from STUD_INFO table where mgr is null.

```
select STUD_ID,SNAME,BRANCH
from STUD_INFO
where DOA is null;
```

OUTPUT :



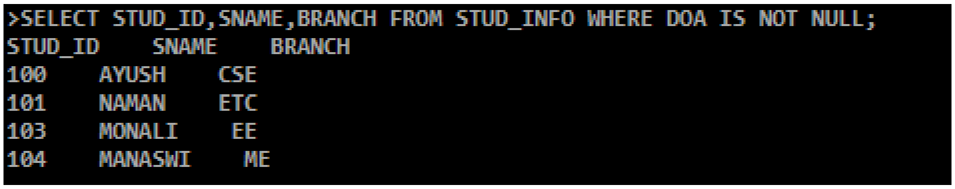
```
>SELECT STUD_ID,SNAME,BRANCH FROM STUD_INFO WHERE DOA IS NULL;
>
```

IS NOT NULL:-

Syntax:-

```
select col_name1,……..col_name n
from <table_name>
where col_name is not null;
```

OUTPUT :



```
>SELECT STUD_ID,SNAME,BRANCH FROM STUD_INFO WHERE DOA IS NOT NULL;
STUD_ID    SNAME    BRANCH
100    AYUSH    CSE
101    NAMAN    ETC
103    MONALI    EE
104    MANASWI    ME
```

The IS NOT NULL operator tests for values that are null. A Not Null value means the value is available, signed, known, equal to any value.

e.g Display all the records from STUD_INFO table where mgr is not null.

```
select STUD_ID,SNAME,BRANCH
from STUD_INFO
 where DOA is NOT null;
```

ANY operator:-

The ANY operator compares a value returned by a subquery.
< ANY means less than the maximum.
< ANY means more than the minimum.
= ANY is equivalent to IN.

ALL operator:-

The all operator compares a value to every value returned by a subquery.
>ALL means more than the maximum.
<ALL means less than the minimum.

CONCLUSION

Thus we have studied and performed the above mysql commands

Viva Voce questions

Write syntax for the following -

IN ( )
NOT
BETWEEN
IS NULL
IS NOT NULL
LIKE
EXISTS

Practical 08

AIM

To use Group function in a database.

OBJECTIVE

- Apply SQL aggregate functions

THEORY

A group function returns a result based on a group of rows.

The group functions are listed below :

- AVG
- SUM
- MIN
- MAX
- COUNT

1. AVG -

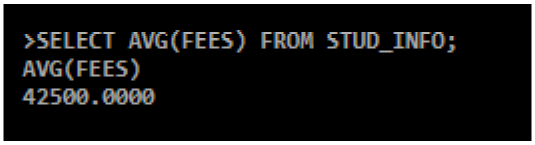
This command returns the average value of the specified column of number data type .

Format:-

avg (col_name)

e.g. select avg(FEES) from STUDENT_INFO;

OUTPUT :



```
>SELECT AVG(FEES) FROM STUD_INFO;
AVG(FEES)
42500.0000
```

2. SUM -

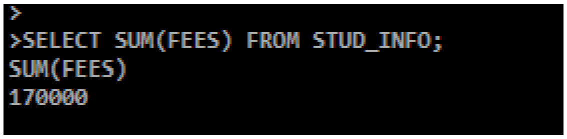
This command returns the summation of the specified column of number data type .

Format:-

sum (col_name)

e.g. select sum(FEES) from STUDENT_INFO;

OUTPUT :



```
>
>SELECT SUM(FEES) FROM STUD_INFO;
SUM(FEES)
170000
```

3. MIN -

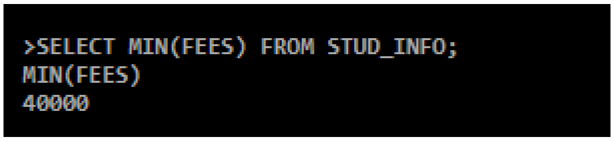
This command returns the lowest value from the specified column of number data type .

Format:-

min (col_name)

e.g. select min(FEES) from STUDENT_INFO;

OUTPUT :



```
>SELECT MIN(FEES) FROM STUD_INFO;
MIN(FEES)
40000
```

4. MAX-

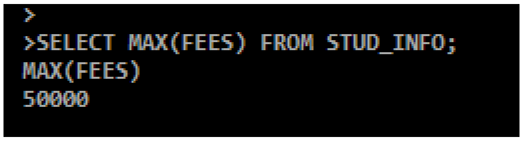
This command returns the highest value from the specified column of number data type .

Format:-

max (col_name)

e.g. select max (FEES) from STUDENT_INFO;

OUTPUT :



```
>
>SELECT MAX(FEES) FROM STUD_INFO;
MAX(FEES)
50000
```

5. COUNT -

It is used to count the number of rows .

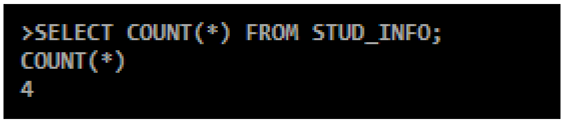
COUNT (*) – It counts all rows , inclusive of duplicate and nulls.

Format:-

count (*)

e.g. select count (*) from STUDENT_INFO;

OUTPUT :



```
>SELECT COUNT(*) FROM STUD_INFO;
COUNT(*)
4
```

COUNT (col_name)

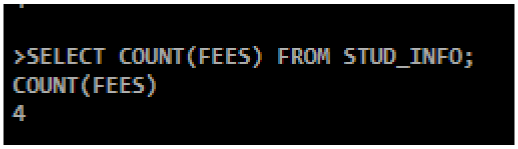
It is used to count the number of values present in the specified column without including nulls.

Format:-

count (col_name)

e.g. select count (comm.) from emp;

OUTPUT :



```
>SELECT COUNT(FEES) FROM STUD_INFO;
COUNT(FEES)
4
```

COUNT (distinct col_name)

It is used to eliminate the duplicate and null values in the specified column.

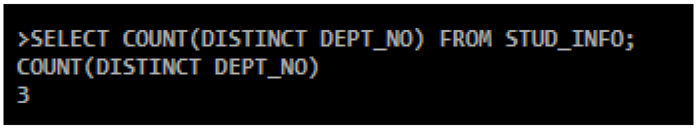
Format:-

count(distinct col_name)

e.g.

select count (distinct DEPT_NO) from STUD_INFO;

OUTPUT :



```
>SELECT COUNT(DISTINCT DEPT_NO) FROM STUD_INFO;
COUNT(DISTINCT DEPT_NO)
3
```

CONCLUSION

Thus we have studied and performed the above mysql commands

Viva Voce questions

Write and perform above sql commands for Employee table.

Practical 09

AIM

To use Transaction control language(TCL) commands .

OBJECTIVE

- Used savepoint instruction.

THEORY

MySQL provides us with the following important statement to control transactions:

To start a transaction, you use the START TRANSACTION statement. The BEGIN or BEGIN WORK are the aliases of the START TRANSACTION.

To commit the current transaction and make its changes permanent, you use the COMMIT statement.

To roll back the current transaction and cancel its changes, you use the ROLLBACK statement.

To disable or enable the auto-commit mode for the current transaction, you use the SET autocommit statement.

```
START TRANSACTION;

INSERT INTO CUSTOMER VALUES(5,"MEENA",35,"RAIPUR",3300);

INSERT INTO CUSTOMER VALUES(6,"MEGHNA",65,"SONPUR",3300);

 SAVEPOINT SP1;

 INSERT INTO CUSTOMER VALUES(7,"MEDHA",66,"DELHI",3340);

 COMMIT;

 UPDATE CUSTOMER SET NAME="ABHIJIT" WHERE ROLL_NO=7;
```

SAVEPOINT A;

INSERT INTO CUSTOMER VALUES(9,"MARIYA",68,"DEHRADUN",3340);

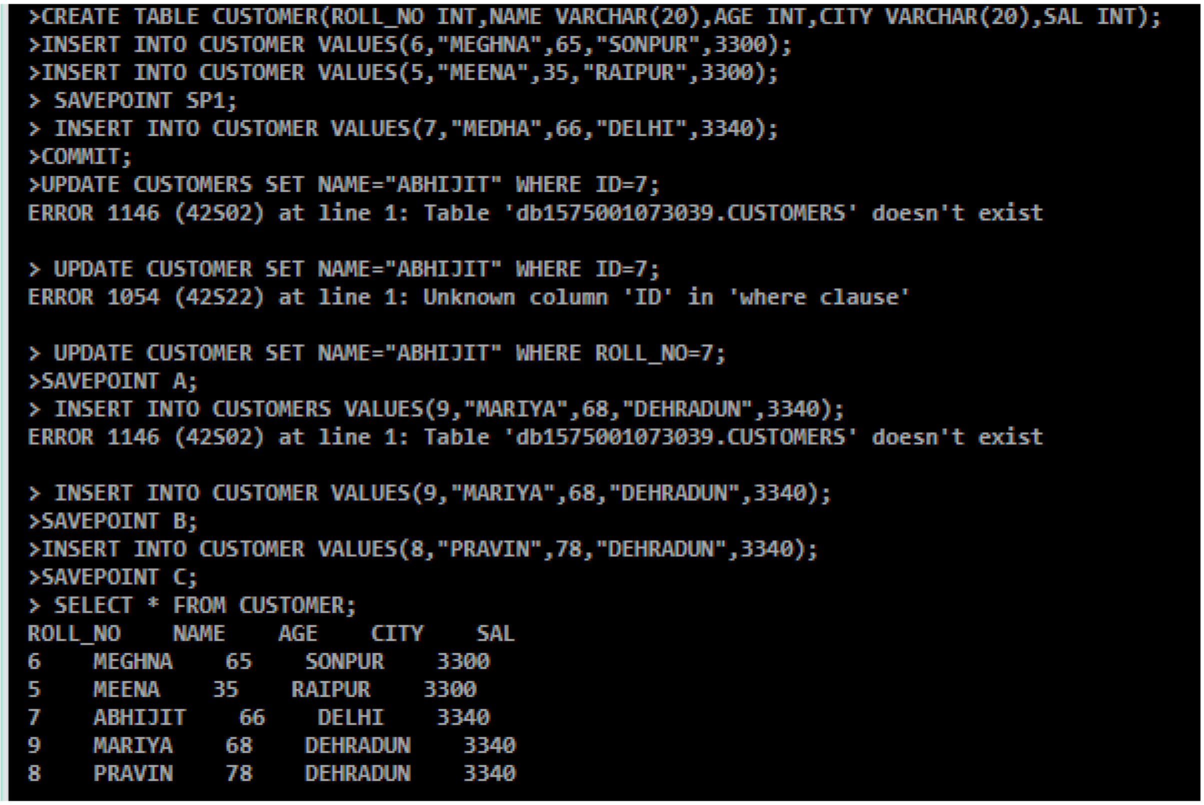
SAVEPOINT B;

INSERT INTO CUSTOMER VALUES(8,"PRAVIN",78,"DEHRADUN",3340);

SAVEPOINT C;

SELECT * FROM CUSTOMER;

OUTPUT :



```
>CREATE TABLE CUSTOMER(ROLL_NO INT,NAME VARCHAR(20),AGE INT,CITY VARCHAR(20),SAL INT);
>INSERT INTO CUSTOMER VALUES(6,"MEGHNA",65,"SONPUR",3300);
>INSERT INTO CUSTOMER VALUES(5,"MEENA",35,"RAIPUR",3300);
> SAVEPOINT SP1;
> INSERT INTO CUSTOMER VALUES(7,"MEDHA",66,"DELHI",3340);
>COMMIT;
>UPDATE CUSTOMERS SET NAME="ABHIJIT" WHERE ID=7;
ERROR 1146 (42S02) at line 1: Table 'db1575001073039.CUSTOMERS' doesn't exist

> UPDATE CUSTOMER SET NAME="ABHIJIT" WHERE ID=7;
ERROR 1054 (42S22) at line 1: Unknown column 'ID' in 'where clause'

> UPDATE CUSTOMER SET NAME="ABHIJIT" WHERE ROLL_NO=7;
>SAVEPOINT A;
> INSERT INTO CUSTOMERS VALUES(9,"MARIYA",68,"DEHRADUN",3340);
ERROR 1146 (42S02) at line 1: Table 'db1575001073039.CUSTOMERS' doesn't exist

> INSERT INTO CUSTOMER VALUES(9,"MARIYA",68,"DEHRADUN",3340);
>SAVEPOINT B;
>INSERT INTO CUSTOMER VALUES(8,"PRAVIN",78,"DEHRADUN",3340);
>SAVEPOINT C;
> SELECT * FROM CUSTOMER;
ROLL_NO    NAME    AGE    CITY    SAL
6    MEGHNA    65    SONPUR    3300
5    MEENA    35    RAIPUR    3300
7    ABHIJIT    66    DELHI    3340
9    MARIYA    68    DEHRADUN    3340
8    PRAVIN    78    DEHRADUN    3340
```

CONCLUSION

Thus we have studied and performed the above mysql commands

Practical 10

AIM

To execute the use of Join commands

OBJECTIVE

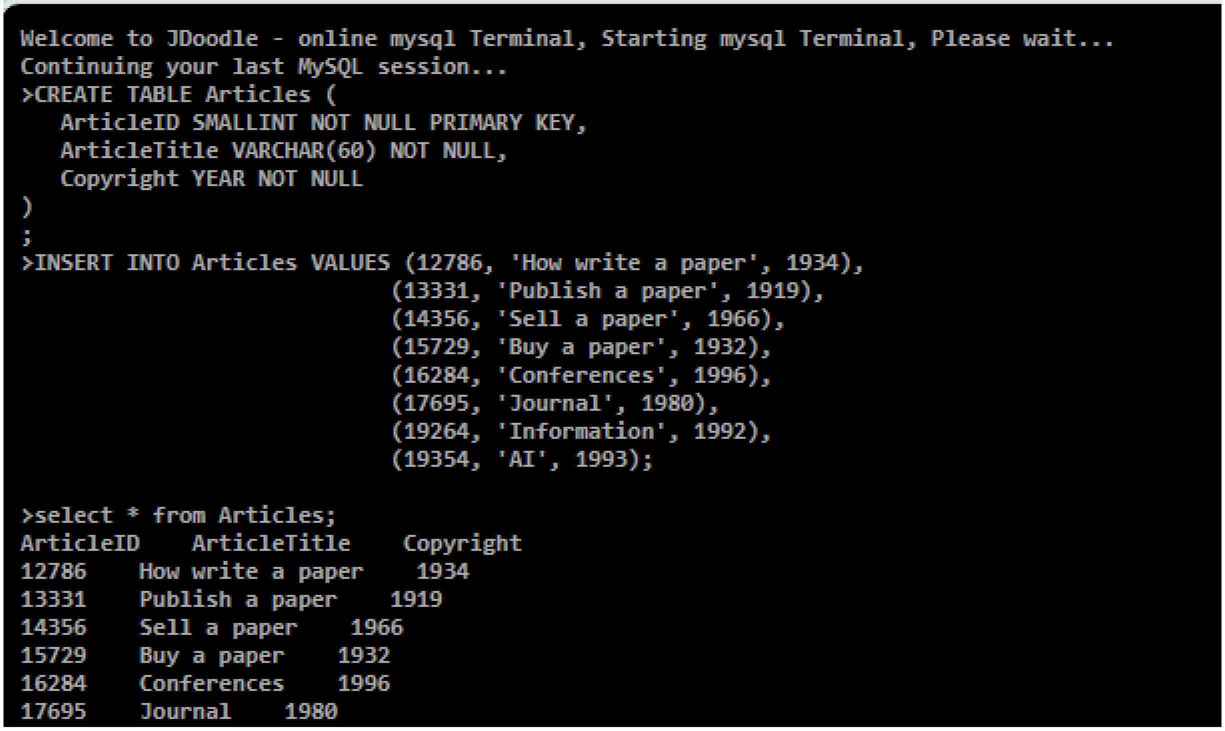
Learn how to perform the following types of database joins

- Cross Join
- Natural Join
- Outer Joins

THEORY

```
CREATE TABLE Articles (

  ArticleID SMALLINT NOT NULL PRIMARY KEY,

  ArticleTitle VARCHAR(60) NOT NULL,

  Copyright YEAR NOT NULL

)

INSERT INTO Articles VALUES (12786, 'How write a paper', 1934),

                   (13331, 'Publish a paper', 1919),

                   (14356, 'Sell a paper', 1966),

                   (15729, 'Buy a paper', 1932),

                   (16284, 'Conferences', 1996),

                   (17695, 'Journal', 1980),

                   (19264, 'Information', 1992),
                    (19354, 'AI', 1993);
```

OUTPUT :



```
Welcome to JDoodle - online mysql Terminal, Starting mysql Terminal, Please wait...
Continuing your last MySQL session...
>CREATE TABLE Articles (
   ArticleID SMALLINT NOT NULL PRIMARY KEY,
   ArticleTitle VARCHAR(60) NOT NULL,
   Copyright YEAR NOT NULL
)
;
>INSERT INTO Articles VALUES (12786, 'How write a paper', 1934),
                            (13331, 'Publish a paper', 1919),
                            (14356, 'Sell a paper', 1966),
                            (15729, 'Buy a paper', 1932),
                            (16284, 'Conferences', 1996),
                            (17695, 'Journal', 1980),
                            (19264, 'Information', 1992),
                            (19354, 'AI', 1993);

>select * from Articles;
ArticleID    ArticleTitle    Copyright
12786    How write a paper    1934
13331    Publish a paper    1919
14356    Sell a paper    1966
15729    Buy a paper    1932
16284    Conferences    1996
17695    Journal    1980
```

```
CREATE TABLE Authors (

  AuthID SMALLINT NOT NULL PRIMARY KEY,

  AuthorFirstName VARCHAR(20),

  AuthorMiddleName VARCHAR(20),

  AuthorLastName VARCHAR(20)

);

INSERT INTO Authors VALUES (1006, 'Henry', 'S.', 'Thompson'),
```

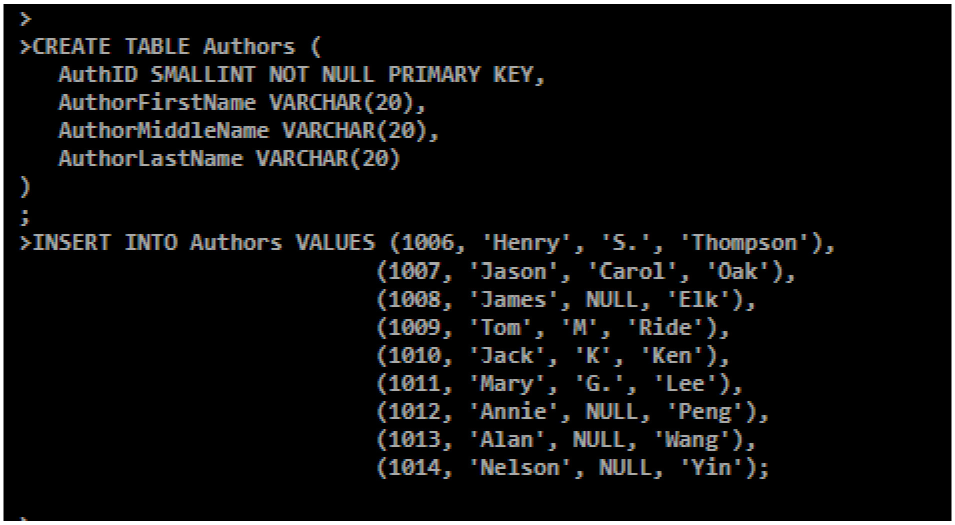
(1007, 'Jason', 'Carol', 'Oak'),   (1008, 'James', NULL, 'Elk'),

(1009, 'Tom', 'M', 'Ride'), (1010, 'Jack', 'K', 'Ken'),

(1011, 'Mary', 'G.', 'Lee'), (1012, 'Annie', NULL, 'Peng'),

(1013, 'Alan', NULL, 'Wang'), (1014, 'Nelson', NULL, 'Yin');

OUTPUT :



```
>
>CREATE TABLE Authors (
   AuthID SMALLINT NOT NULL PRIMARY KEY,
   AuthorFirstName VARCHAR(20),
   AuthorMiddleName VARCHAR(20),
   AuthorLastName VARCHAR(20)
)
;
>INSERT INTO Authors VALUES (1006, 'Henry', 'S.', 'Thompson'),
                           (1007, 'Jason', 'Carol', 'Oak'),
                           (1008, 'James', NULL, 'Elk'),
                           (1009, 'Tom', 'M', 'Ride'),
                           (1010, 'Jack', 'K', 'Ken'),
                           (1011, 'Mary', 'G.', 'Lee'),
                           (1012, 'Annie', NULL, 'Peng'),
                           (1013, 'Alan', NULL, 'Wang'),
                           (1014, 'Nelson', NULL, 'Yin');
```

CREATE TABLE AuthorArticle (

  AuthID SMALLINT NOT NULL,

  ArticleID SMALLINT NOT NULL,

  PRIMARY KEY (AuthID, ArticleID),

  FOREIGN KEY (AuthID) REFERENCES Authors (AuthID),

  FOREIGN KEY (ArticleID) REFERENCES Articles (ArticleID)

);

INSERT INTO AuthorArticle VALUES (1006, 14356), (1008, 15729), (1009, 12786), (1010, 17695),

(1011, 15729), (1012, 19264), (1012, 19354), (1014, 16284);

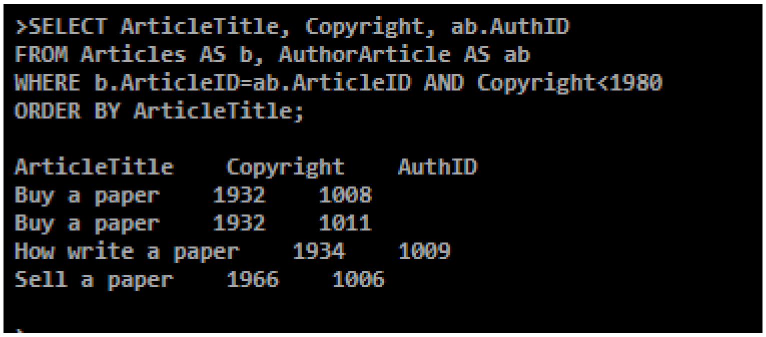
SELECT ArticleTitle, Copyright, ab.AuthID

FROM Articles AS b, AuthorArticle AS ab

WHERE b.ArticleID=ab.ArticleID AND Copyright<1980

ORDER BY ArticleTitle;

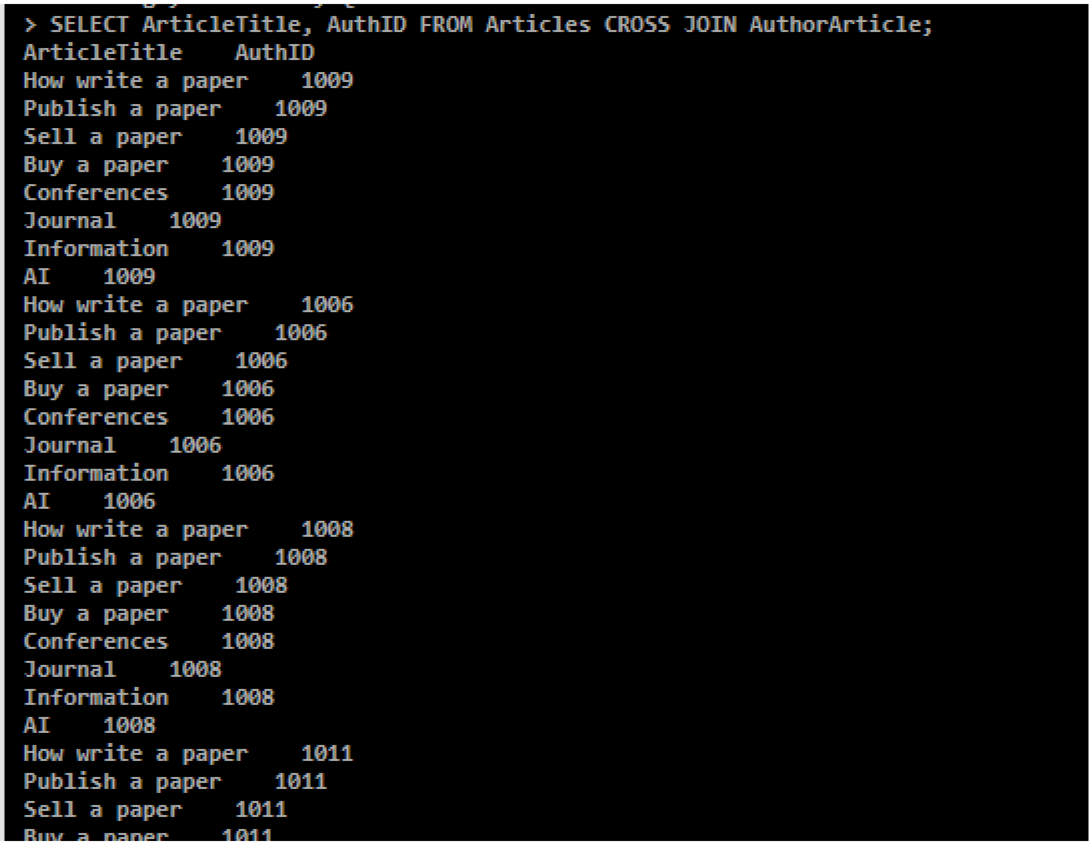
OUTPUT :



```
>SELECT ArticleTitle, Copyright, ab.AuthID
FROM Articles AS b, AuthorArticle AS ab
WHERE b.ArticleID=ab.ArticleID AND Copyright<1980
ORDER BY ArticleTitle;

ArticleTitle    Copyright    AuthID
Buy a paper    1932    1008
Buy a paper    1932    1011
How write a paper    1934    1009
Sell a paper    1966    1006

>
```

SELECT ArticleTitle, AuthID FROM Articles CROSS JOIN AuthorArticle;

OUTPUT :

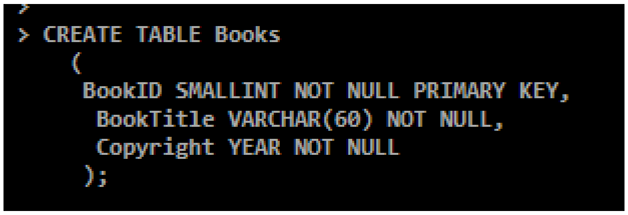


```
> SELECT ArticleTitle, AuthID FROM Articles CROSS JOIN AuthorArticle;
ArticleTitle    AuthID
How write a paper    1009
Publish a paper    1009
Sell a paper    1009
Buy a paper    1009
Conferences    1009
Journal    1009
Information    1009
AI    1009
How write a paper    1006
Publish a paper    1006
Sell a paper    1006
Buy a paper    1006
Conferences    1006
Journal    1006
Information    1006
AI    1006
How write a paper    1008
Publish a paper    1008
Sell a paper    1008
Buy a paper    1008
Conferences    1008
Journal    1008
Information    1008
AI    1008
How write a paper    1011
Publish a paper    1011
Sell a paper    1011
Buy a paper    1011
```

FULL JOIN

```
CREATE TABLE Books
  (
   BookID SMALLINT NOT NULL PRIMARY KEY,
    BookTitle VARCHAR(60) NOT NULL,
    Copyright YEAR NOT NULL
   );
```

OUTPUT :



```
> CREATE TABLE Books
    (
     BookID SMALLINT NOT NULL PRIMARY KEY,
      BookTitle VARCHAR(60) NOT NULL,
      Copyright YEAR NOT NULL
     );
```

INSERT INTO Books VALUES

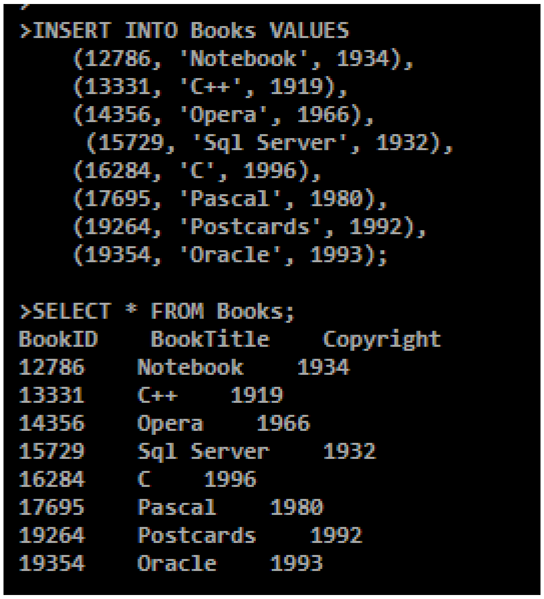
(12786, 'Notebook', 1934), (13331, 'C++', 1919),

(14356, 'Opera', 1966), (15729, 'Sql Server', 1932),

(16284, 'C', 1996), (17695, 'Pascal', 1980),

(19264, 'Postcards', 1992), (19354, 'Oracle', 1993);

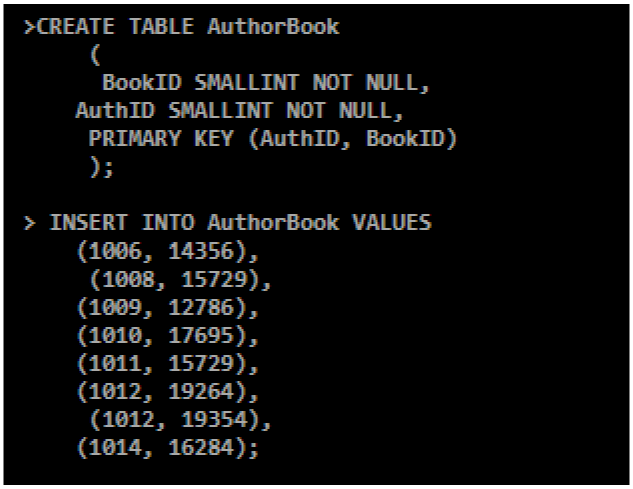
OUTPUT :



```
>INSERT INTO Books VALUES
    (12786, 'Notebook', 1934),
    (13331, 'C++', 1919),
    (14356, 'Opera', 1966),
     (15729, 'Sql Server', 1932),
    (16284, 'C', 1996),
    (17695, 'Pascal', 1980),
    (19264, 'Postcards', 1992),
    (19354, 'Oracle', 1993);

>SELECT * FROM Books;
BookID    BookTitle    Copyright
12786    Notebook    1934
13331    C++    1919
14356    Opera    1966
15729    Sql Server    1932
16284    C    1996
17695    Pascal    1980
19264    Postcards    1992
19354    Oracle    1993
```

```
CREATE TABLE Authors
    (
    AuthID SMALLINT NOT NULL PRIMARY KEY,
   AuthFN VARCHAR(20),
    AuthMN VARCHAR(20),
    AuthLN VARCHAR(20)
   );
INSERT INTO Authors VALUES
   (1006, 'Hunter', 'S.', 'Thompson'),
    (1007, 'Joyce', 'Carol', 'Oates'),
    (1008, 'Black', NULL, 'Elk'),
    (1009, 'Rainer', 'Maria', 'Rilke'),
    (1010, 'John', 'Kennedy', 'Toole'),
    (1011, 'John', 'G.', 'Neihardt'),
    (1012, 'Annie', NULL, 'Proulx'),
    (1013, 'Alan', NULL, 'Watts'),
    (1014, 'Nelson', NULL, 'Algren');


CREATE TABLE AuthorBook
    (
    BookID SMALLINT NOT NULL,
   AuthID SMALLINT NOT NULL,
    PRIMARY KEY (AuthID, BookID)
    );
  INSERT INTO AuthorBook VALUES
   (1006, 14356), (1008, 15729),
   (1009, 12786), (1010, 17695),
   (1011, 15729), (1012, 19264),
    (1012, 19354), (1014, 16284);
```

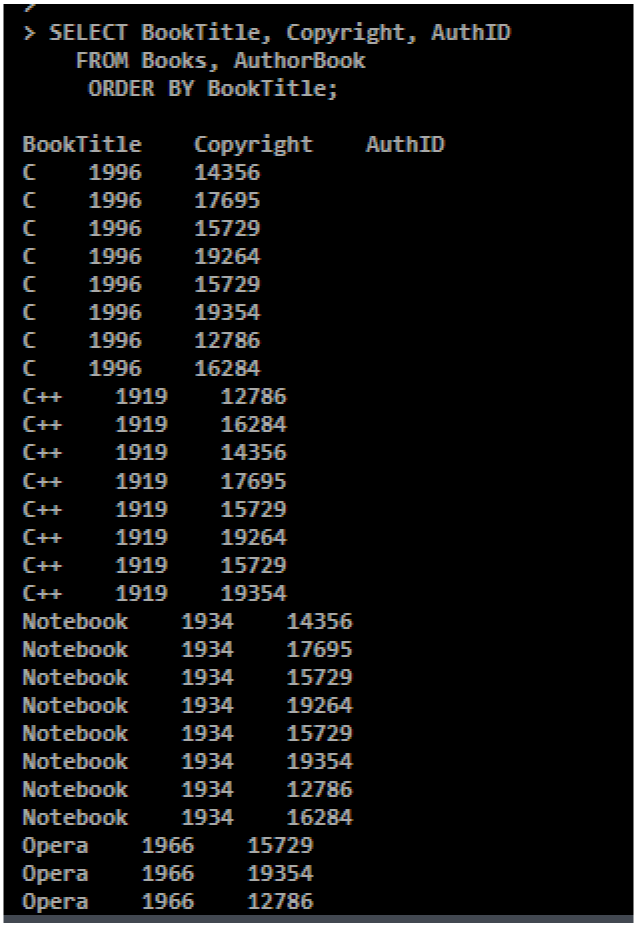
OUTPUT :



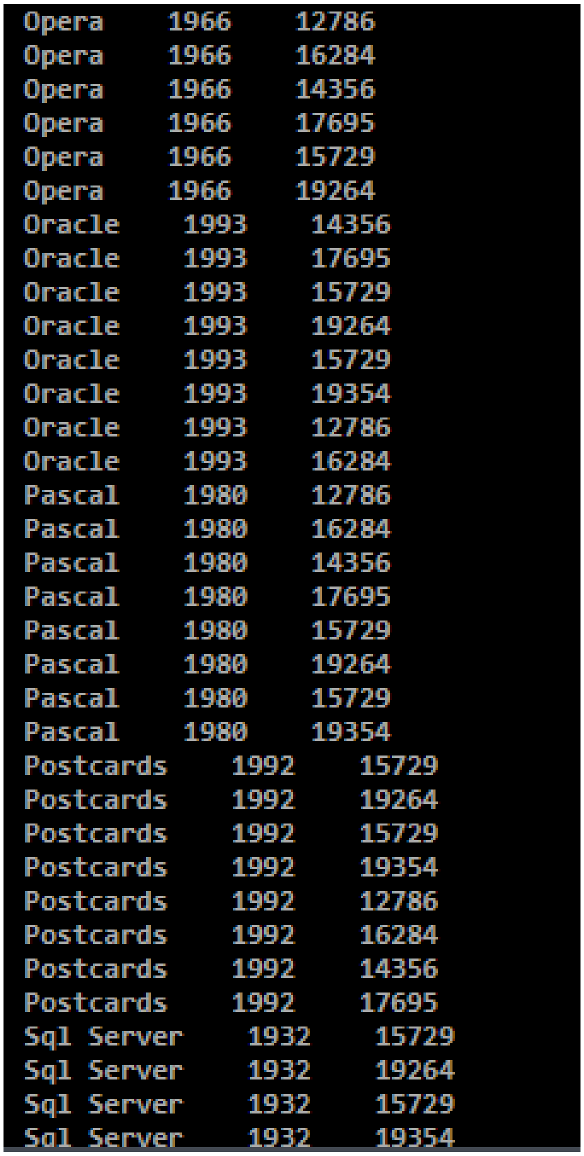
```
>CREATE TABLE AuthorBook
     (
      BookID SMALLINT NOT NULL,
    AuthID SMALLINT NOT NULL,
     PRIMARY KEY (AuthID, BookID)
     );

> INSERT INTO AuthorBook VALUES
    (1006, 14356),
     (1008, 15729),
    (1009, 12786),
    (1010, 17695),
    (1011, 15729),
    (1012, 19264),
     (1012, 19354),
    (1014, 16284);
```

```
SELECT BookTitle, Copyright, AuthID
 FROM Books, AuthorBook
```

ORDER BY BookTitle;



```
> SELECT BookTitle, Copyright, AuthID
    FROM Books, AuthorBook
     ORDER BY BookTitle;

BookTitle    Copyright    AuthID
C    1996    14356
C    1996    17695
C    1996    15729
C    1996    19264
C    1996    15729
C    1996    19354
C    1996    12786
C    1996    16284
C++    1919    12786
C++    1919    16284
C++    1919    14356
C++    1919    17695
C++    1919    15729
C++    1919    19264
C++    1919    15729
C++    1919    19354
Notebook    1934    14356
Notebook    1934    17695
Notebook    1934    15729
Notebook    1934    19264
Notebook    1934    15729
Notebook    1934    19354
Notebook    1934    12786
Notebook    1934    16284
Opera    1966    15729
Opera    1966    19354
Opera    1966    12786
```



```
Opera     1966    12786
Opera     1966    16284
Opera     1966    14356
Opera     1966    17695
Opera     1966    15729
Opera     1966    19264
Oracle     1993    14356
Oracle     1993    17695
Oracle     1993    15729
Oracle     1993    19264
Oracle     1993    15729
Oracle     1993    19354
Oracle     1993    12786
Oracle     1993    16284
Pascal     1980    12786
Pascal     1980    16284
Pascal     1980    14356
Pascal     1980    17695
Pascal     1980    15729
Pascal     1980    19264
Pascal     1980    15729
Pascal     1980    19354
Postcards     1992    15729
Postcards     1992    19264
Postcards     1992    15729
Postcards     1992    19354
Postcards     1992    12786
Postcards     1992    16284
Postcards     1992    14356
Postcards     1992    17695
Sql Server     1932    15729
Sql Server     1932    19264
Sql Server     1932    15729
Sql Server     1932    19354
```

drop table AuthorBook;

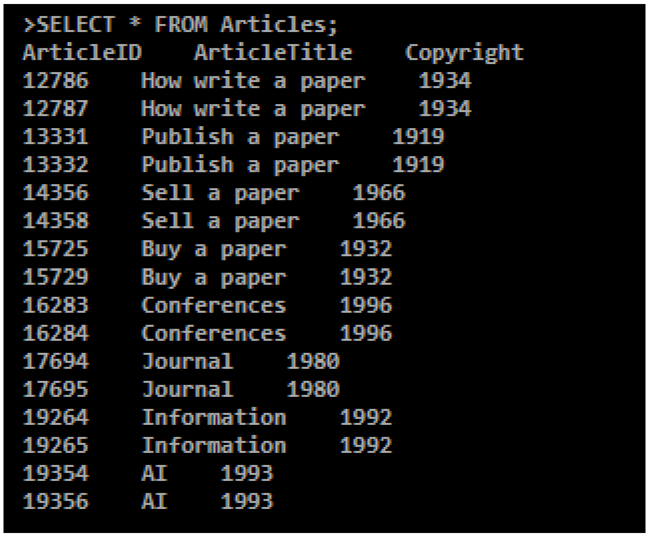
drop table Books;

drop table Authors;

INNER JOIN

```
CREATE TABLE Articles (

  ArticleID SMALLINT NOT NULL PRIMARY KEY,

  ArticleTitle VARCHAR(60) NOT NULL,

  Copyright YEAR NOT NULL

);

INSERT INTO Articles VALUES (12787, 'How write a paper', 1934),

                  (13332, 'Publish a paper', 1919),  (14358, 'Sell a paper', 1966),

                  (15725, 'Buy a paper', 1932),

                  (16283, 'Conferences', 1996),

                  (17694, 'Journal', 1980),

                  (19265, 'Information', 1992),(19356, 'AI', 1993);
```

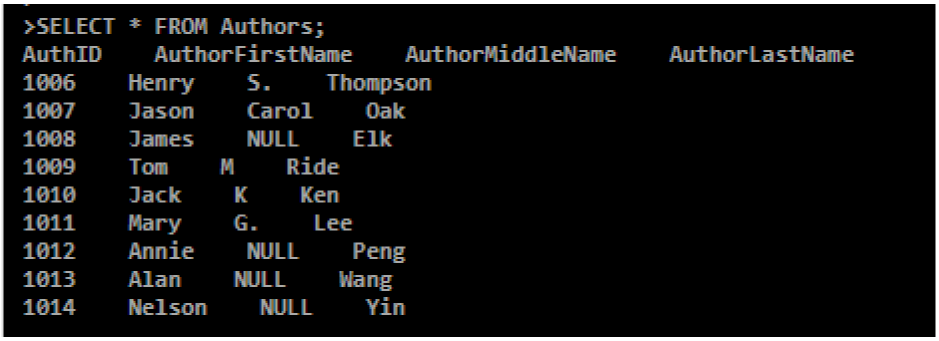
OUTPUT :



```
>SELECT * FROM Articles;
ArticleID    ArticleTitle    Copyright
12786    How write a paper    1934
12787    How write a paper    1934
13331    Publish a paper    1919
13332    Publish a paper    1919
14356    Sell a paper    1966
14358    Sell a paper    1966
15725    Buy a paper    1932
15729    Buy a paper    1932
16283    Conferences    1996
16284    Conferences    1996
17694    Journal    1980
17695    Journal    1980
19264    Information    1992
19265    Information    1992
19354    AI    1993
19356    AI    1993
```

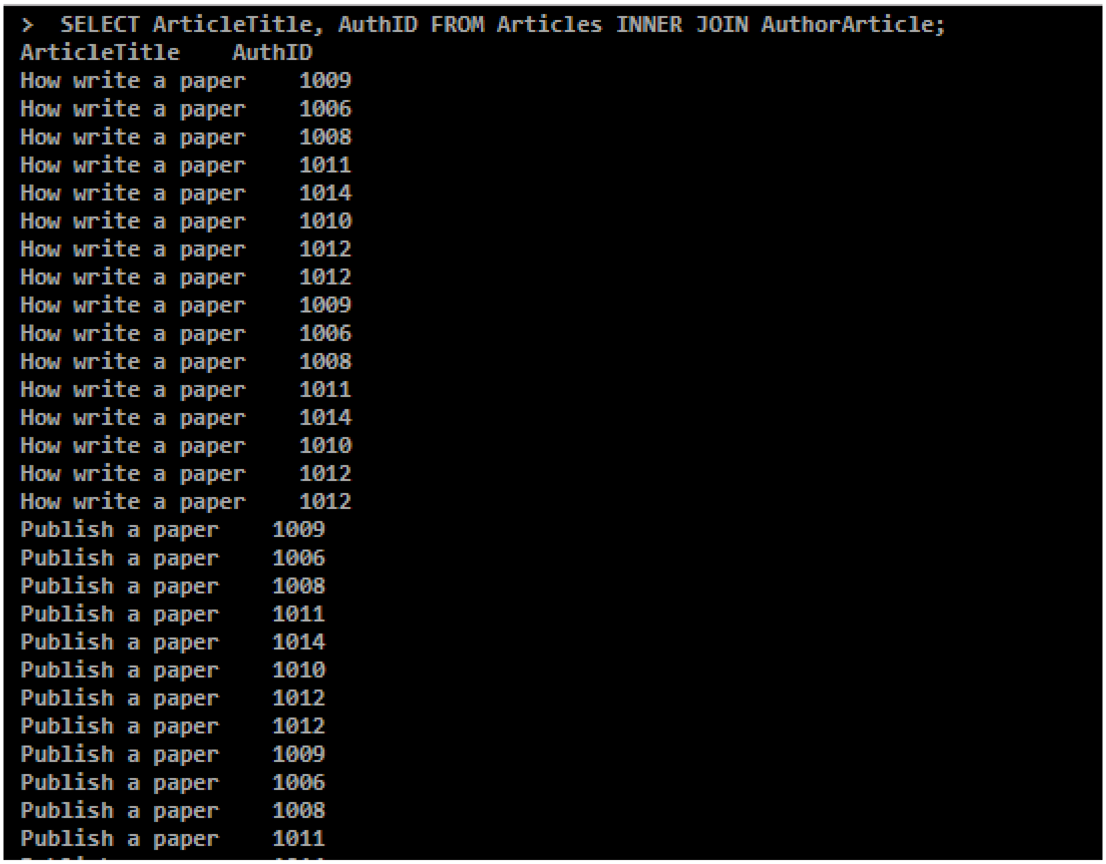
```
CREATE TABLE Authors (
  AuthID SMALLINT NOT NULL PRIMARY KEY,
  AuthorFirstName VARCHAR(20),
  AuthorMiddleName VARCHAR(20),
  AuthorLastName VARCHAR(20));
INSERT INTO Authors VALUES (1006, 'Henry', 'S.', 'Thompson'),
                  (1007, 'Jason', 'Carol', 'Oak'),
                  (1008, 'James', NULL, 'Elk'),
                  (1009, 'Tom', 'M', 'Ride'),
                  (1010, 'Jack', 'K', 'Ken'),
                  (1011, 'Mary', 'G.', 'Lee'),
                  (1012, 'Annie', NULL, 'Peng'),
                  (1013, 'Alan', NULL, 'Wang'),
                  (1014, 'Nelson', NULL, 'Yin');
```

OUTPUT :



```
>SELECT * FROM Authors;
AuthID    AuthorFirstName    AuthorMiddleName    AuthorLastName
1006    Henry    S.    Thompson
1007    Jason    Carol    Oak
1008    James    NULL    Elk
1009    Tom    M    Ride
1010    Jack    K    Ken
1011    Mary    G.    Lee
1012    Annie    NULL    Peng
1013    Alan    NULL    Wang
1014    Nelson    NULL    Yin
```

```
CREATE TABLE AuthorArticle (

   AuthID SMALLINT NOT NULL,

   ArticleID SMALLINT NOT NULL,

   PRIMARY KEY (AuthID, ArticleID),

   FOREIGN KEY (AuthID) REFERENCES Authors (AuthID),

   FOREIGN KEY (ArticleID) REFERENCES Articles (ArticleID));

INSERT INTO AuthorArticle VALUES (1006, 14356), (1008, 15729),
                              (1009, 12786), (1010, 17695), (1011, 15729), (1012, 19264),
                              (1012, 19354),   (1014, 16284);
  SELECT ArticleTitle, AuthID FROM Articles INNER JOIN
AuthorArticle;
```



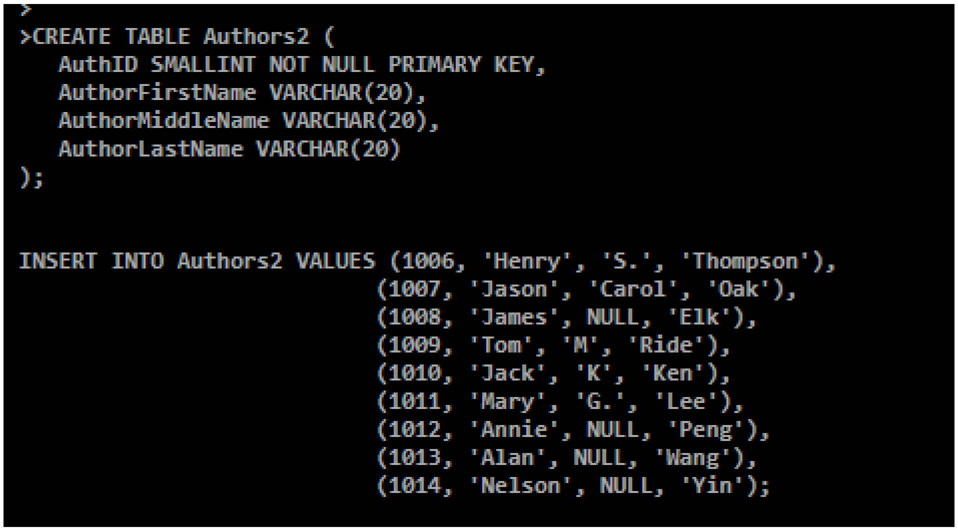
```
>  SELECT ArticleTitle, AuthID FROM Articles INNER JOIN AuthorArticle;
ArticleTitle    AuthID
How write a paper    1009
How write a paper    1006
How write a paper    1008
How write a paper    1011
How write a paper    1014
How write a paper    1010
How write a paper    1012
How write a paper    1012
How write a paper    1009
How write a paper    1006
How write a paper    1008
How write a paper    1011
How write a paper    1014
How write a paper    1010
How write a paper    1012
How write a paper    1012
Publish a paper    1009
Publish a paper    1006
Publish a paper    1008
Publish a paper    1011
Publish a paper    1014
Publish a paper    1010
Publish a paper    1012
Publish a paper    1012
Publish a paper    1009
Publish a paper    1006
Publish a paper    1008
Publish a paper    1011
```

DELETE JOIN

```
CREATE TABLE Articles (
  ArticleID SMALLINT NOT NULL PRIMARY KEY,
  ArticleTitle VARCHAR(60) NOT NULL,
  Copyright YEAR NOT NULL);
INSERT INTO Articles VALUES (12786, 'How write a paper', 1934),
                   (13331, 'Publish a paper', 1919),
                   (14356, 'Sell a paper', 1966),
                   (15729, 'Buy a paper', 1932),
                   (16284, 'Conferences', 1996),
                   (17695, 'Journal', 1980),
                   (19264, 'Information', 1992),
                   (19354, 'AI', 1993);
CREATE TABLE Authors2 (
  AuthID SMALLINT NOT NULL PRIMARY KEY,
  AuthorFirstName VARCHAR(20),
  AuthorMiddleName VARCHAR(20),
  AuthorLastName VARCHAR(20)
);
```

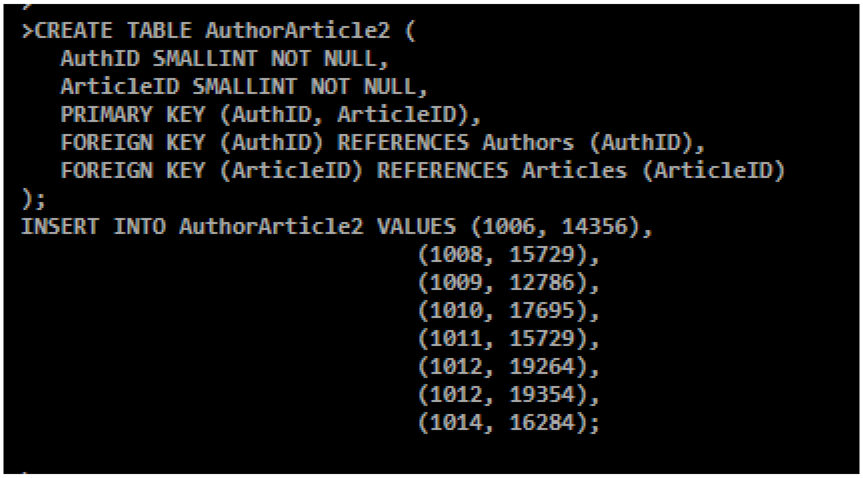
```
INSERT INTO Authors2 VALUES (1006, 'Henry', 'S.', 'Thompson'),
                (1007, 'Jason', 'Carol', 'Oak'),
                (1008, 'James', NULL, 'Elk'),
                (1009, 'Tom', 'M', 'Ride'),
                (1010, 'Jack', 'K', 'Ken'),
                (1011, 'Mary', 'G.', 'Lee'),
                (1012, 'Annie', NULL, 'Peng'),
                (1013, 'Alan', NULL, 'Wang'),
                (1014, 'Nelson', NULL, 'Yin');
```

OUTPUT :



```
>
>CREATE TABLE Authors2 (
   AuthID SMALLINT NOT NULL PRIMARY KEY,
   AuthorFirstName VARCHAR(20),
   AuthorMiddleName VARCHAR(20),
   AuthorLastName VARCHAR(20)
);


INSERT INTO Authors2 VALUES (1006, 'Henry', 'S.', 'Thompson'),
                            (1007, 'Jason', 'Carol', 'Oak'),
                            (1008, 'James', NULL, 'Elk'),
                            (1009, 'Tom', 'M', 'Ride'),
                            (1010, 'Jack', 'K', 'Ken'),
                            (1011, 'Mary', 'G.', 'Lee'),
                            (1012, 'Annie', NULL, 'Peng'),
                            (1013, 'Alan', NULL, 'Wang'),
                            (1014, 'Nelson', NULL, 'Yin');
```

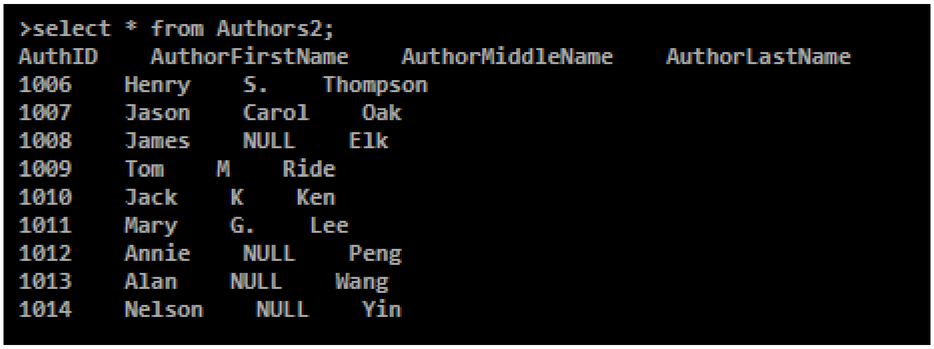
```
CREATE TABLE AuthorArticle2 (

   AuthID SMALLINT NOT NULL,

   ArticleID SMALLINT NOT NULL,

   PRIMARY KEY (AuthID, ArticleID),

   FOREIGN KEY (AuthID) REFERENCES Authors (AuthID),

   FOREIGN KEY (ArticleID) REFERENCES Articles (ArticleID)

);

INSERT INTO AuthorArticle2 VALUES (1006, 14356),

                              (1008, 15729), (1009, 12786),

                              (1010, 17695),(1011, 15729),

                              (1012, 19264),(1012, 19354),

                              (1014, 16284);
```



```
>CREATE TABLE AuthorArticle2 (
   AuthID SMALLINT NOT NULL,
   ArticleID SMALLINT NOT NULL,
   PRIMARY KEY (AuthID, ArticleID),
   FOREIGN KEY (AuthID) REFERENCES Authors (AuthID),
   FOREIGN KEY (ArticleID) REFERENCES Articles (ArticleID)
);
INSERT INTO AuthorArticle2 VALUES (1006, 14356),
                                  (1008, 15729),
                                  (1009, 12786),
                                  (1010, 17695),
                                  (1011, 15729),
                                  (1012, 19264),
                                  (1012, 19354),
                                  (1014, 16284);
```

select * from Authors2;

OUTPUT :



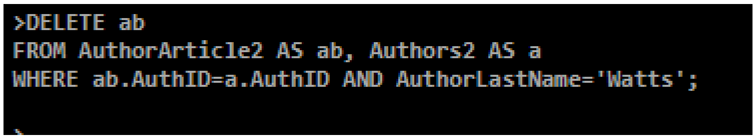
```
>select * from Authors2;
AuthID    AuthorFirstName    AuthorMiddleName    AuthorLastName
1006    Henry    S.    Thompson
1007    Jason    Carol    Oak
1008    James    NULL    Elk
1009    Tom    M    Ride
1010    Jack    K    Ken
1011    Mary    G.    Lee
1012    Annie    NULL    Peng
1013    Alan    NULL    Wang
1014    Nelson    NULL    Yin
```

DELETE ab

FROM AuthorArticle2 AS ab, Authors2 AS a

WHERE ab.AuthID=a.AuthID AND AuthorLastName='Watts';

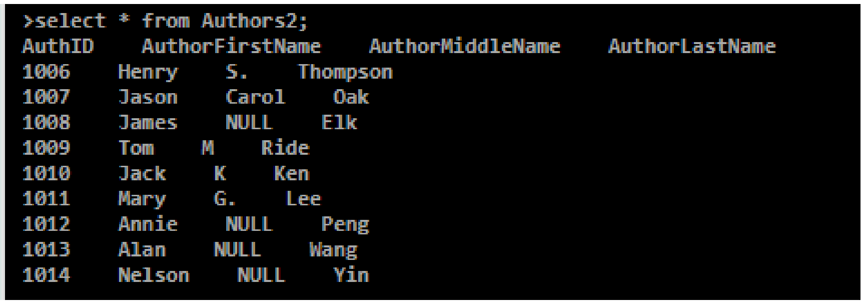
OUTPUT :



```
>DELETE ab
FROM AuthorArticle2 AS ab, Authors2 AS a
WHERE ab.AuthID=a.AuthID AND AuthorLastName='Watts';

>
```

select * from Authors2;

OUTPUT :



```
>select * from Authors2;
AuthID    AuthorFirstName    AuthorMiddleName    AuthorLastName
1006    Henry    S.    Thompson
1007    Jason    Carol    Oak
1008    James    NULL    Elk
1009    Tom    M    Ride
1010    Jack    K    Ken
1011    Mary    G.    Lee
1012    Annie    NULL    Peng
1013    Alan    NULL    Wang
1014    Nelson    NULL    Yin
```

CONCLUSION

Thus we have studied and performed the above mysql commands

Practical 11

AIM

To execute basic commands of Python.

OBJECTIVE

- Bridge the industry institute gap.

THEORY

```
#Press the Run Button!

print('Welcome to Python Programming.net!')
print('See how easy it is to program.')

# This is a 'for loop':
for each_number in range(4):
    print(each_number)

print('You just ran a Python program!')
print('Try playing with the editor values, like changing the range or print functions, or get started by clicking on the Start Learning button.')
```

```
>>> 2 + 2
4
>>> 50 - 5*6
20
>>> (50 - 5.0*6) / 4
5.0
>>> 8 / 5.0
1.6

17 / 3  # int / int -> int

5
>>> 17 / 3.0  # int / float -> float
```

```
5.666666666666667
>>> 17 // 3.0  # explicit floor division discards the fractional part
5.0
>>> 17 % 3  # the % operator returns the remainder of the division
2
>>> 5 * 3 + 2  # result * divisor + remainder
17

5 ** 2  # 5 squared

25
>>> 2 ** 7  # 2 to the power of 7
128

width = 20
>>> height = 5 * 9
>>> width * height

>>> tax = 12.5 / 100
>>> price = 100.50
>>> price * tax
12.5625
>>> price + _
113.0625
>>> round(_, 2)
113.06

>>> 'spam eggs'  # single quotes
'spam eggs'
>>> 'doesn\'t'  # use \' to escape the single quote...
"doesn't"
>>> "doesn't"  # ...or use double quotes instead
"doesn't"
>>> '"Yes," he said.'
'"Yes," he said.'
>>> "\"Yes,\" he said."
'"Yes," he said.'
>>> '"Isn\'t," she said.'
'"Isn\'t," she said.'
```

```
print '"Isn\'t," she said.'

s = 'First line.\nSecond line.'  # \n means newline

s
print s

>>> print 'C:\some\name'  # here \n means newline!
C:\some
ame
>>> print r'C:\some\name'  # note the r before the quote
C:\some\name


print """\
Usage: thingy [OPTIONS]
     -h                         Display this usage message
     -H hostname               Hostname to connect to
"""

>>> # 3 times 'un', followed by 'ium'
>>> 3 * 'un' + 'ium'
'unununium'

>>> 'Py' 'thon'
'Python'

prefix = 'Py'
prefix + 'thon'

>>> text = ('Put several strings within parentheses '
            'to have them joined together.')
>>> text
'Put several strings within parentheses to have them joined together.'

>>> word = 'Python'
>>> word[0]  # character in position 0
'P'
>>> word[5]  # character in position 5
'n'
```

```
>>> word[0:2]  # characters from position 0 (included) to 2 (excluded)
'Py'
>>> word[2:5]  # characters from position 2 (included) to 5 (excluded)
'tho'

>>> word[:2] + word[2:]
'Python'
>>> word[:4] + word[4:]
'Python'

>>> word[4:42]
'on'
>>> word[42:]
''

>>> 'J' + word[1:]
'Jython'
>>> word[:2] + 'py'
'Pypy'

>>> s = 'supercalifragilisticexpialidocious'
>>> len(s)
34
```

```
>>> squares = [1, 4, 9, 16, 25]
>>> squares
[1, 4, 9, 16, 25]

>>> squares[0]  # indexing returns the item
1
>>> squares[-1]
25
>>> squares[-3:]  # slicing returns a new list
[9, 16, 25]

>>> squares[:]
[1, 4, 9, 16, 25]
```

```
>>> squares + [36, 49, 64, 81, 100]
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
>>> letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> letters
['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> # replace some values
>>> letters[2:5] = ['C', 'D', 'E']
>>> letters
['a', 'b', 'C', 'D', 'E', 'f', 'g']
>>> # now remove them
>>> letters[2:5] = []
>>> letters
['a', 'b', 'f', 'g']
>>> # clear the list by replacing all the elements with an empty list
>>> letters[:] = []
>>> letters
[]
```

```
>>> a = ['a', 'b', 'c']
>>> n = [1, 2, 3]
>>> x = [a, n]
>>> x
[['a', 'b', 'c'], [1, 2, 3]]
>>> x[0]
['a', 'b', 'c']
>>> x[0][1]
'b'
```

```
>>> i = 256*256
>>> print 'The value of i is', i
The value of i is 65536
```

```
>>> # Fibonacci series:
... # the sum of two elements defines the next
... a, b = 0, 1
```

```
>>> while b < 10:
...     print b
...     a, b = b, a+b
...
```

```
>>> a, b = 0, 1
>>> while b < 1000:
...     print b,
...     a, b = b, a+b
```

```
words = ['cat', 'window', 'defenestrate']
>>> for w in words:
...     print w, len(w)
```

The range() Function

```
 range(10)
range(0, 10, 3)
range(-10, -100, -30)
```

```
a = ['Mary', 'had', 'a', 'little', 'lamb']
>>> for i in range(len(a)):
...     print i, a[i]
```

```
for num in range(2, 10):
    if num % 2 == 0:
      print "Found an even number", num
    continue
    print "Found a number", num
```

```
***the_world_is_flat = 1
if the_world_is_flat:
 print "Be careful not to fall off!"
```

```
type(1.25)
type('hello')
type([1,2,3])
```

```
len([2,4,6]
len(’abcd’)

list()
max(5,6,78)

str(23)
int(’125’)



def f(x,y):
    print 'You called f(x,y) with the value x = ' + str(x) + ' and y = ' + str(y)
    print 'x + y = ' + str(x+y)

f(3,2)
```

#FIBONACCI SERIES

```
a,b = 0,1

for n in range (1,900):
    if (a<900):
        a,b=b,a+b
        print a
```

# Python Program to find the L.C.M. of two input number

```
# define a function
def lcm(x, y):
   """This function takes two
   integers and returns the L.C.M."""


   # choose the greater number
   if x > y:
      greater = x
   else:
      greater = y

   while(True):
      if((greater % x == 0) and (greater % y == 0)):
          lcm = greater
          break
      greater += 1

   return lcm


# take input from the user
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))

print("The L.C.M. of", num1,"and", num2,"is", lcm(num1, num2))
```

# This program adds two numbers provided by the user

```
# Store input numbers
num1 = input('Enter first number: ')
num2 = input('Enter second number: ')

# Add two numbers
sum = float(num1) + float(num2)

# Display the sum
print('The sum of {0} and {1} is {2}'.format(num1, num2, sum))
```

CONCLUSION

Thus we have performed basic command of Python.

REFERENCES :

1. Lab manual of Distributed Operating System, Anjuman college of engineering and Technology, Nagpur.

2. https://www.jdoodle.com/online-mysql-terminal/

3. https://paiza.io/en/projects/new?language=mysql

4. https://www.w3schools.com/sql/

5. http://cws.cengage.co.uk/rcc_databases/students/mysqllabguide/mysqluide.pdf

6. http://www.math.umt.edu/olear/stat458/Rseminar_2.pdf

7. https://www.tutorialspoint.com/execute_python_online.php