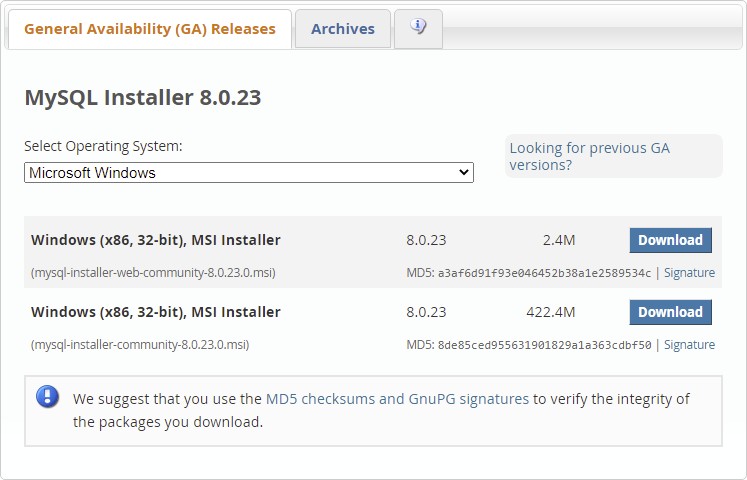
Database Management System Lab Manual



# Experiment - 1

**Objective:** Installing MYSQL

**MySQL** server is an open-source relational database management system which is a major support for web based applications. Databases and related tables are the main component of many websites and applications as the data is stored and exchanged over the web.

Select mysql-installer-web-community-8.0.23.msi if you have good internet connection, otherwise choose mysql-installer-community-8.0.23.msi.

Install MySQL

After downloading, unzip it, and double click the MSI installer .exe file. Then follow the steps below:

1. **"Choosing a Setup Type"** screen: Choose "Full" setup type. This installs all MySQL products and features. Then click the "Next" button to continue.

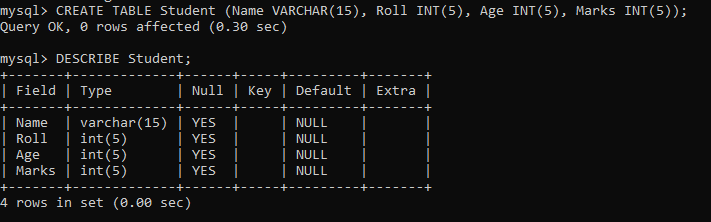
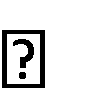
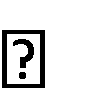
Prashant Gupta 1 2201331540141

Database Management System Lab Manual

1. **"Check Requirements"** screen: The installer checks if your pc has the requirements needed. If there is some failing requirements, click on each item to try to resolve them by clicking on the Execute button that will install all requirements automatically. Click "Next".
2. **"Installation"** screen: See what products that will be installed. Click "Execute" to download and install the Products. After finishing the installation, click "Next".
3. **"Product Configuration"** screen: See what products that will be configured. Click the "MySQL Server 8.0.23" option to configure the MySQL Server. Click the "Next" button. Choose the "Standalone MySQL Server/Classic MySQL Replication" option and click on the "Next" button. In page "Type and Networking" set Config Type to "Development Computer" and "Connectivity" to "TCP/IP" and "Port" to "3006". Then, click the "Next" button.
4. **"Authentication Method"** screen: Choose "Use Strong Password Encryption for Authentication". Click "Next".
5. **"Accounts and Roles"** screen: Set a password for the root account. Click "Next".
6. **"Windows Service"** screen: Here, you configure the Windows Service to start the server. Keep the default setup, then click "Next".
7. **"Apply Configuration"** screen: Click the "Execute" button to apply the Server configuration. After finishing, click the "Finish" button.
8. **"Product Configuration"** screen: See that the Product Configuration is completed. Keep the default setting and click on the "Next" and "Finish" button to complete the MySQL package installation.
9. In the next screen, you can choose to configure the Router. Click on "Next", "Finish" and then click the "Next" button.
10. **"Connect To Server"** screen: Type in the root password (from step 6). Click the "Check" button to check if the connection is successful or not. Click on the "Next" button.
11. **"Apply Configuration"** screen: Select the options and click the "Execute" button. After finishing, click the "Finish" button.
12. **"Installation Complete"** screen: The installation is complete. Click the "Finish" button.

­­­­Prashant Gupta 2 2201331540141

Database Management System Lab Manual



# Experiment – 2

**Objective:** Implementation of DDL commands of SQL with suitable examples

Create table

Alter table

Drop Table

**DATA DEFINITION LANGUAGE (DDL):** The Data Definition Language (DDL) is used to create and destroy databases and database objects. These commands will primarily be used by database administrators during the setup and removal phases of a database project. Let's take a look at the structure and usage of four basic DDL commands:

1. CREATE 2. ALTER 3. DROP

**CREATE:**

CREATE TABLE Student (Name VARCHAR(15), Roll INT(5), Age INT(5), Marks INT(5));

**ALTER:**

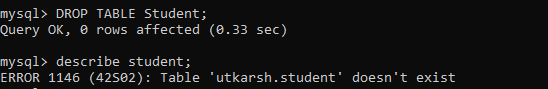
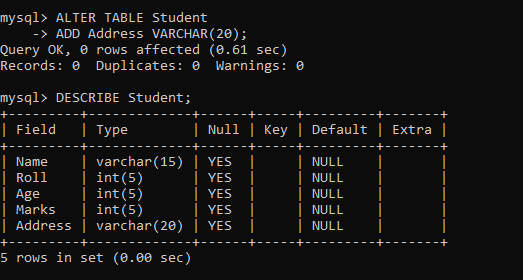
ALTER TABLE table\_name ADD column\_name datatype;

ALTER TABLE Student

ADD Address VARCHAR(20);

Prashant Gupta 3 2201331540141

Database Management System Lab Manual

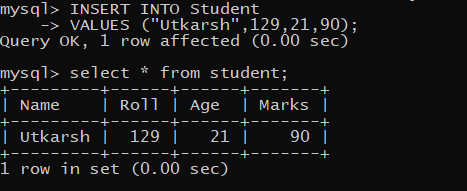
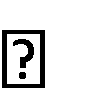
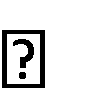


**DROP:**

DROP TABLE Student;

Prashant Gupta 4 2201331540141

Database Management System Lab Manual



# Experiment – 3

**Objective:** Implementation of DML commands of SQL with suitable examples

Insert table

Update table

Delete Table

**DATA MANIPULATION LANGUAGE (DML):** The Data Manipulation Language (DML) is used to retrieve, insert and modify database information. These commands will be used by all database users during the routine operation of the database. Let's take a brief look at the basic DML commands:

## 1. INSERT 2. UPDATE 3. DELETE

**INSERT:**

INSERT INTO *table\_name*

VALUES (*value1*, *value2*, *value3*, ...);

INSERT INTO Student

VALUES (“Utkarsh”,129,21,90);

**UPDATE:** The syntax for update command is-

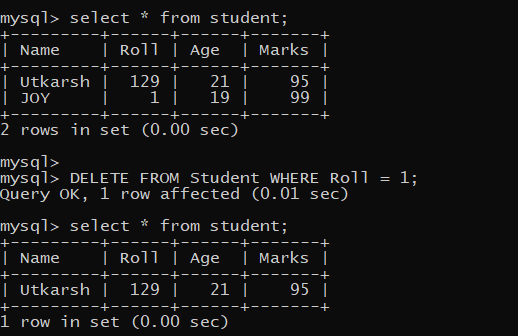
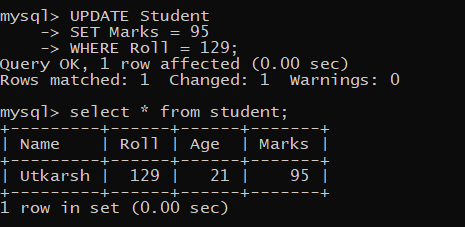
**UPDATE** table\_name

**SET** column\_name = value

**WHERE** (condition);

Prashant Gupta 5 2201331540141

Database Management System Lab Manual



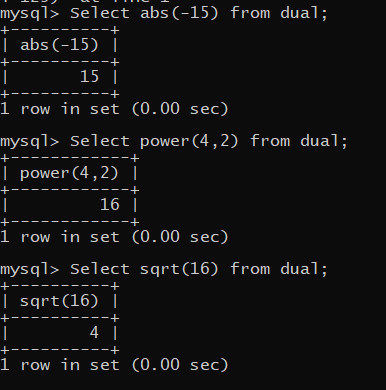
UPDATE Student SET Marks = 95 WHERE Roll = 129;

**DELETE:** Syntax - DELETE FROM table\_name WHERE condition;

DELETE FROM Student WHERE Roll = 1;

Prashant Gupta 6 2201331540141

Database Management System Lab Manual



# Experiment – 4

***Objective:*** Implementation of different types of functions with suitable examples.

Number Function

Aggregate Function

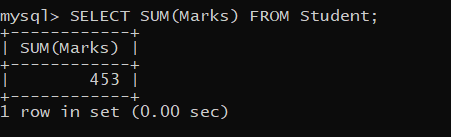
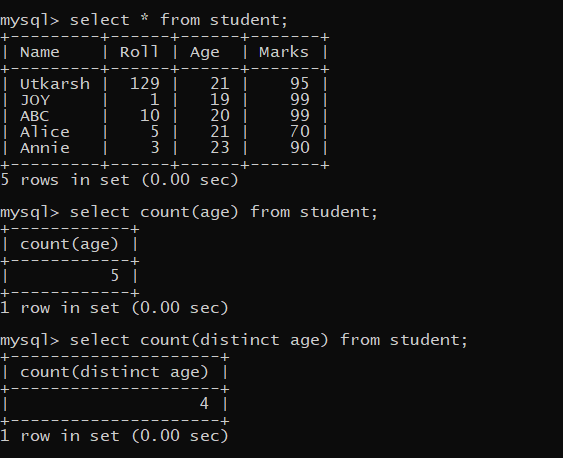
## NUMBER FUNCTIONS :

Abs(n): Select abs(-15) from dual; Exp(n): Select exp(4) from dual;

Power(m,n): Select power(4,2) from dual; Mod(m,n): Select mod (10,3) from dual; Round(m,n): Select round(100.256,2) from dual; Trunc(m,n): Select trunc(100.256,2) from dual; Sqrt(m,n): Select sqrt(16) from dual;

Prashant Gupta 7 2201331540141

Database Management System Lab Manual



## AGGREGATE FUNCTION :

**COUNT:**

SELECT COUNT(age) FROM student;

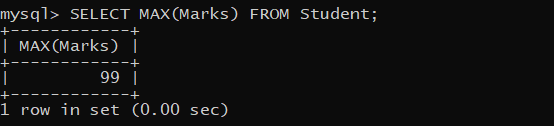
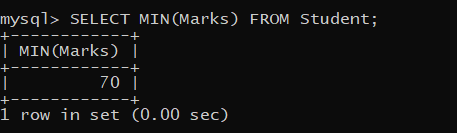
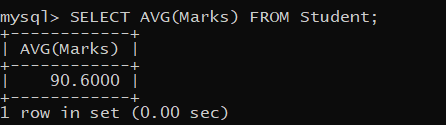
SELECT COUNT(DISTINCT age) FROM student;

## SUM:

SELECT SUM(Marks) FROM Student;

Prashant Gupta 8 2201331540141

Database Management System Lab Manual



## AVERAGE:

SELECT AVG(Marks) FROM Student;

## MIN:

SELECT MIN(Marks) FROM Student;

## MAX:

SELECT MAX(Marks) FROM Student;

Prashant Gupta 9 2201331540141

Database Management System Lab Manual



# Experiment – 5

***Objective:*** Implementation of different types of operators in SQL.

Arithmetic Operator

Logical Operator

Comparison Operator

## ARIHMETIC OPERATORS:

(+): Addition - Adds values on either side of the operator.

(-): Subtraction - Subtracts right hand operand from left hand operand. (\*): Multiplication - Multiplies values on either side of the operator. (/): Division - Divides left hand operand by right hand operand.

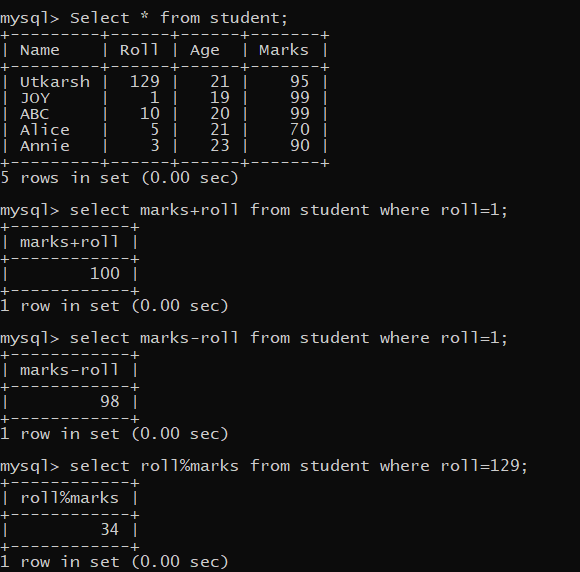
(^): Power- raise to power of.

(%): Modulus - Divides left hand operand by right hand operand and returns remainder.

SELECT marks+roll FROM student WHERE roll=1; SELECT marks-roll FROM student WHERE roll=1; SELECT roll%marks FROM student WHERE roll=129;

Prashant Gupta 10 2201331540141

Database Management System Lab Manual



## LOGICAL OPERATORS:

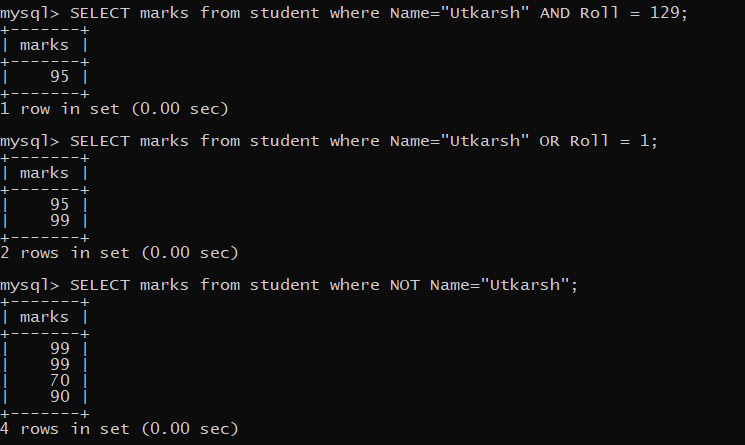
**AND :** The AND operator allows the existence of multiple conditions in an SQL statement's WHERE clause.

**OR :** The OR operator is used to combine multiple conditions in an SQL statement's WHERE clause.

**NOT :** The NOT operator reverses the meaning of the logical operator with which it is used.

Prashant Gupta 11 2201331540141

Database Management System Lab Manual



## COMPARISION OPERATORS:

**(=):** Checks if the values of two operands are equal or not, if yes then condition becomes true.

**(! =):** Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.

**(< >):** Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.

**(>):** Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.

**(<):** Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.

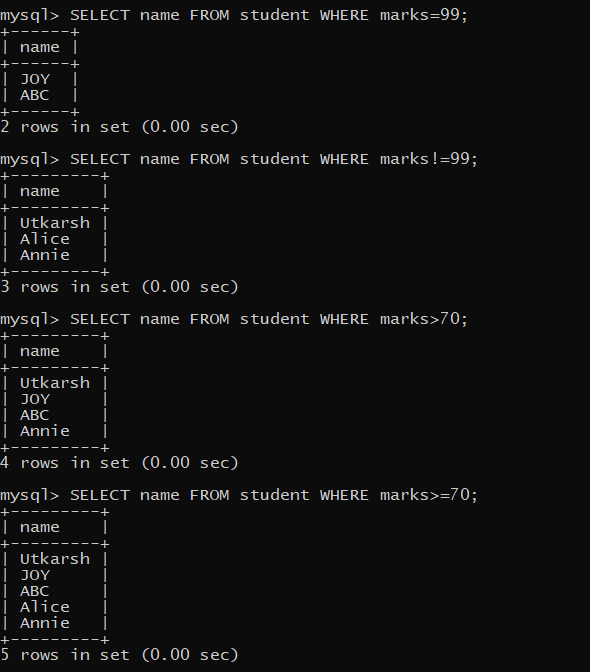
**(>=):** Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.

**(<=):** Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.

SELECT name FROM student WHERE marks=99; SELECT name FROM student WHERE marks!=99;

Prashant Gupta 12 2201331540141

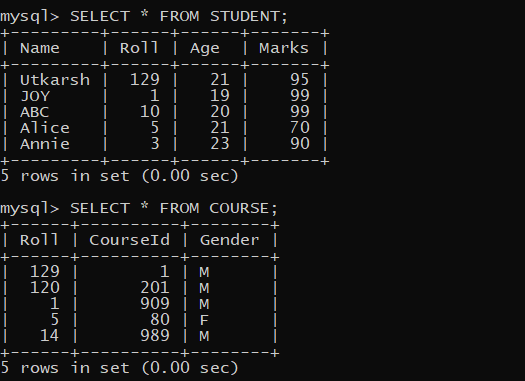
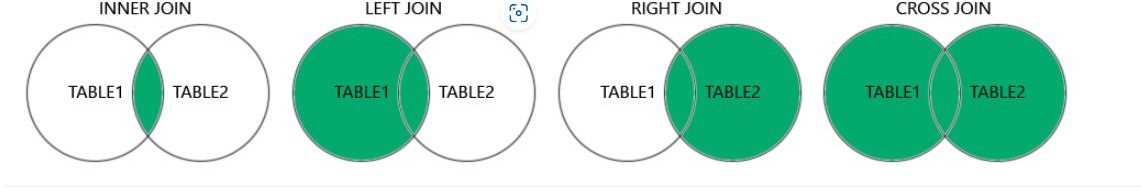
Database Management System Lab Manual



SELECT name FROM student WHERE marks>70; SELECT name FROM student WHERE marks>=70;

Prashant Gupta 13 2201331540141

Database Management System Lab Manual



# Experiment – 6

***Objective:*** Implementation of different types of Joins

Inner Join

Outer Join

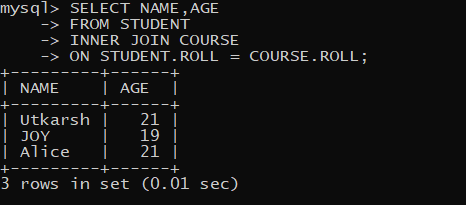
Natural Joined

## Types of Joins in MySQL

CONSIDER FOLLOWING 2 TABLES IN OUR DATABASE:

Prashant Gupta 14 2201331540141

Database Management System Lab Manual



**INNER JOIN**: Returns records that have matching values in both tables. SYNTAX :

SELECT column\_name(s) FROM table1

INNER JOIN table2

ON table1.column\_name = table2.column\_name;

SELECT NAME,AGE FROM STUDENT INNER JOIN COURSE

ON STUDENT.ROLL = COURSE.ROLL;

**LEFT JOIN**: Returns all records from the left table, and the matched records from the right table. SYNTAX –

SELECT column\_name(s) FROM table1

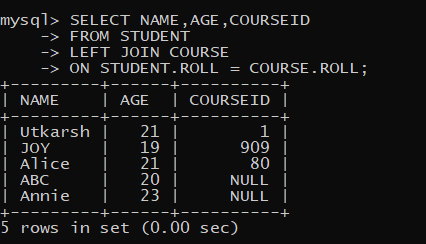
LEFT JOIN table2

ON table1.column\_name = table2.column\_name;

SELECT NAME,AGE,COURSEID

Prashant Gupta 15 2201331540141

Database Management System Lab Manual



FROM STUDENT LEFT JOIN COURSE

ON STUDENT.ROLL = COURSE.ROLL;

**RIGHT JOIN:** Returns all records from the right table, and the matched records from the left table. SYNTAX –

SELECT column\_name(s) FROM table1

RIGHT JOIN table2

ON table1.column\_name = table2.column\_name;

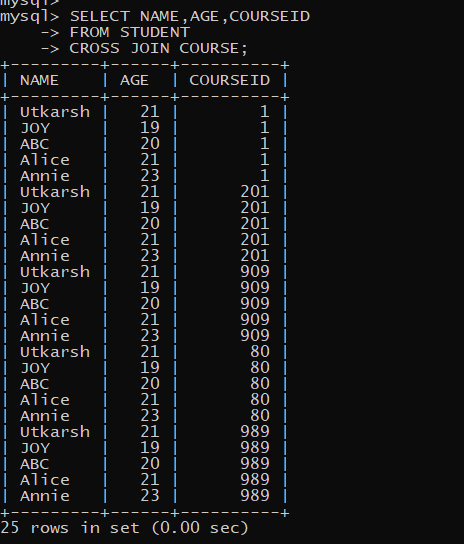
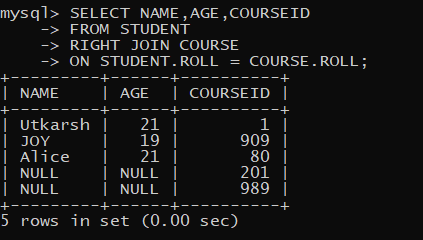
SELECT NAME,AGE,COURSEID FROM STUDENT

RIGHT JOIN COURSE

ON STUDENT.ROLL = COURSE.ROLL;

Prashant Gupta 16 2201331540141

Database Management System Lab Manual



**CROSS JOIN**: Returns all records from both tables. SYNTAX –

SELECT column\_name(s) FROM table1

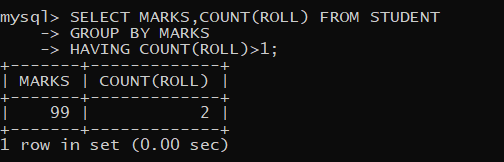
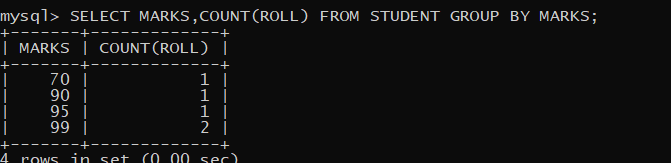
CROSS JOIN table2;

SELECT NAME,AGE,COURSEID FROM STUDENT

CROSS JOIN COURSE;

Prashant Gupta 17 2201331540141

Database Management System Lab Manual



# Experiment – 7

**Objective:** Study and Implementation of

* Group By & having clause
* Order by clause

**GROUP BY:** This query is used to group to all the records in a relation together for each and every value of a specific key(s) and then display them for a selected set of fields the relation.

***Syntax:*** SELECT <set of fields> FROM <relation\_name> GROUP BY <field\_name>;

***Example:*** SELECT MARKS,COUNT(ROLL) FROM STUDENT GROUP BY MARKS;

**GROUP BY-HAVING :** The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions. The HAVING clause must follow the GROUP BY clause in a query and must also precede the ORDER BY clause if used.

***Syntax:*** SELECT column\_name, aggregate\_function(column\_name) FROM table\_name WHERE column\_name operator value

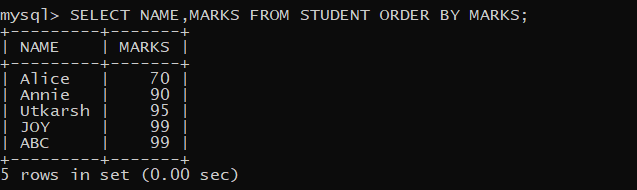
GROUP BY column\_name

HAVING aggregate\_function(column\_name) operator value; ***Example:*** SELECT MARKS,COUNT(ROLL) FROM STUDENT GROUP BY MARKS

HAVING COUNT(ROLL)>1;

Prashant Gupta 18 2201331540141

Database Management System Lab Manual



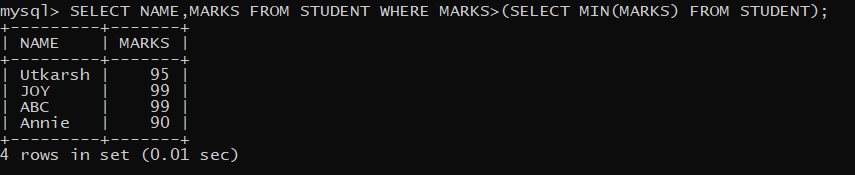
**ORDER BY:** This query is used to display a selected set of fields from a relation in an ordered manner base on some field.

***Syntax:*** SELECT <set of fields> FROM <relation\_name> ORDER BY <field\_name>;

***Example:*** SELECT NAME,MARKS FROM STUDENT ORDER BY MARKS;

Prashant Gupta 19 2201331540141

Database Management System Lab Manual



# Experiment – 8

***Objective:*** Study & Implementation of

Sub queries

**SUBQUERIES:** The query within another is known as a sub query. A statement containing sub query is called parent statement.

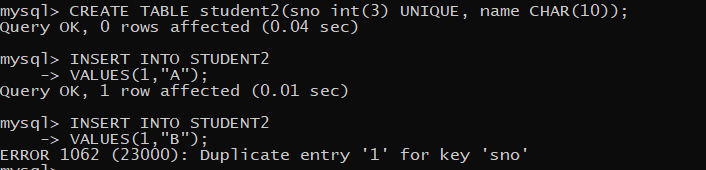
A subquery is a SELECT statement that is embedded in a clause of another SELECT statement.

## Example:

SELECT NAME,MARKS FROM STUDENT WHERE MARKS>(SELECT MIN(MARKS) FROM STUDENT);

Prashant Gupta 20 2201331540141

Database Management System Lab Manual



# Experiment – 9

***Objective:*** • Study & Implementation of different types of constraints

## CONSTRAINTS:

Constraints are used to specify rules for the data in a table. If there is any violation between the constraint and the data action, the action is aborted by the constraint.

1. **NOT NULL:** When a column is defined as NOTNULL, then that column becomes a mandatory column. It implies that a value must be entered into the column if the record is to be accepted for storage in the table.

***Syntax:* CREATE TABLE** Table Name (column\_name data\_type (*size*) **NOT NULL,** );

***Example:* CREATE TABLE** student (sno **INT(3) NOT NULL,** name **CHAR**(**10**));

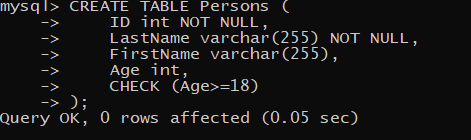
1. **UNIQUE:** The purpose of a unique key is to ensure that information in the column(s) is unique i.e. a value entered in column(s) defined in the unique constraint must not be repeated across the column(s). A table may have many unique keys.

***Syntax:* CREATE TABLE** Table\_Name(column\_name data\_type(*size*) **UNIQUE, ….**);

***Example:* CREATE TABLE** student2 (sno **INT(3) UNIQUE,** name **CHAR**(**10**));

Prashant Gupta 21 2201331540141

Database Management System Lab Manual



1. **CHECK:** Specifies a condition that each row in the table must satisfy. To satisfy the constraint, each row in the table must make the condition either TRUE or unknown (due to a null).

***Syntax:* CREATE TABLE** Table\_Name(column\_name data\_type(*size*) **CHECK(*logical expression*), ….**);

***Example:*** CREATE TABLE Persons ( ID int NOT NULL,

LastName varchar(255) NOT NULL, FirstName varchar(255),

Age int,

CHECK (Age>=18)

);

1. **PRIMARY KEY:** A field which is used to identify a record uniquely. A column or combination of columns can be created as primary key, which can be used as a reference from other tables. A table contains primary key is known as Master Table.

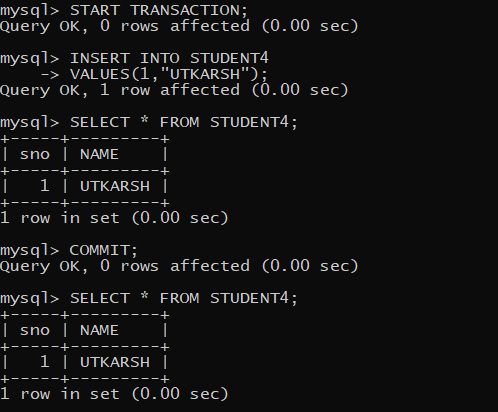
***Syntax:* CREATE TABLE** Table\_Name(column\_name data\_type(*size*) **PRIMARY KEY,**

**….**);

***Example:*** CREATE TABLE STUDENT4(sno INT(3) PRIMARY KEY,NAME VARCHAR(10));

Prashant Gupta 22 2201331540141

Database Management System Lab Manual



# Experiment – 10

***Objective:***

Study and Implementation of Database Backup & Recovery Commands.

Study and Implementation of Rollback, Commit, Save point.

1. **COMMIT:** This command is used to end a transaction only with the help of the commit command transaction changes can be made permanent to the database.

***Syntax:*** SQL> COMMIT;

***Example:*** SQL> COMMIT;

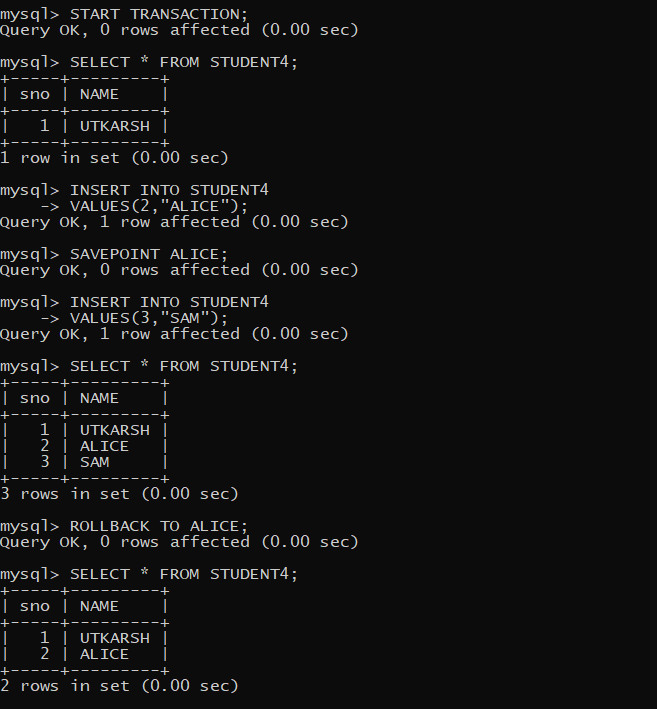
1. **SAVE POINT**: Save points are like marks to divide a very lengthy transaction to smaller once. They are used to identify a point in a transaction to which we can latter role back. Thus, save point is used in conjunction with rollback.

***Syntax:*** SQL> SAVE POINT ID;

***Example:*** SQL> SAVE POINT xyz;

Prashant Gupta 23 2201331540141

Database Management System Lab Manual



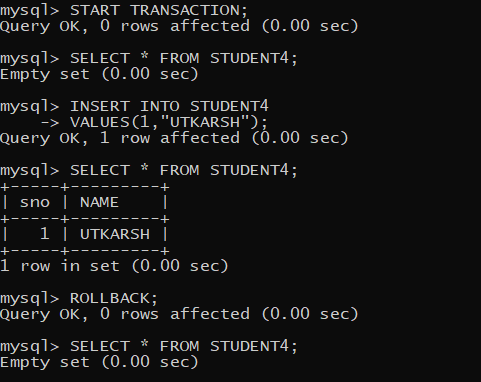
1. **ROLLBACK:** A role back command is used to undo the current transactions.We can roll back the entire transaction so that all changes made by SQL statements are undo (or) role back a transaction to a save point so that the SQL statements after the save point are rollback.

***Syntax:*** ROLLBACK (current transaction can be rollback) ROLLBACK to save point ID;

***Example:*** SQL> ROLLBACK TO SAVE POINT xyz;

Prashant Gupta 24 2201331540141

Database Management System Lab Manual



Prashant Gupta 25 2201331540141