**MySQL-SQL**

## WHAT IS SQL?

1. SQL stands for Structured Query Language.
2. Used for managing and manipulating relational databases.
3. SQL lets you access and manipulate databases.
4. SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987.

## WHAT CAN SQL DO?

1. SQL can execute queries against a database.
2. SQL can retrieve data from a database.
3. SQL can insert records in a database.
4. SQL can update records in a database.
5. SQL can delete records from a database.
6. SQL can create new databases.
7. SQL can create new tables in a database.
8. SQL can create stored procedures in a database.
9. SQL can create views in a database.
10. SQL can set permissions on tables, procedures, and views.

## LIST OF WELL KNOWN RELATIONAL DATABASE MANAGEMENT SYSTEMS

1. MySQL
2. PostgreSQL
3. Oracle Database
4. Microsoft SQL Server
5. SQLite
6. IBM Db2
7. MariaDB

## CASE SENSITIVE OR NOT?

* KEYWORDS AND IDENTIERS ARE CASE INSENSITIVE LITERALS ARE CASE SENSITIVE.

**WHAT DO YOU MEAN BY DBMS? WHAT ARE ITS DIFFERENT TYPES?**

Database is a structured collection of data or information

Database consists of tables, rows(records), columns, indexes, views, and other database objects that organize and store data in a structured format.

A DBMS is a software system that provides an interface for users to interact with databases.

A Database Management System (DBMS) is a software application that interacts with the user, applications and the database itself to capture and analyse data.

A DBMS allows a user to interact with the database using query language such as SQL. The data stored in the database can be modified, retrieved and deleted and can be of any type like strings, numbers, images etc.

**THERE ARE TWO TYPES OF DBMS:**

1. **Relational Database Management System:** The data is stored in relations (tables). Example – MySQL, Oracle SQL.
2. **Non-Relational Database Management System:** There is no concept of relations, tuples and attributes. Example – Mongo

## WHAT ARE THE DIFFERENT SUBSETS OF SQL? WHAT ARE ALL THE DIFFERENT SQL COMMANDS CATAGORY?

The standard SQL commands to interact with relational databases are CREATE, SELECT, INSERT, UPDATE, DELETE and DROP. These commands can be classified into the following groups based on their nature.

### **DDL - Data Definition Language**

|  |  |
| --- | --- |
|  | **Command & Description** |
| 1 | **CREATE**  Creates a new table, a view of a table, or other object in the database. |
| 2 | **ALTER**  Modifies an existing database object, such as a table. |
| 3 | **DROP**  Deletes an entire table, a view of a table or other objects in the database. |

### **DML - Data Manipulation Language**

|  |  |
| --- | --- |
|  | **Command & Description** |
| 1 | **SELECT**  Retrieves certain records from one or more tables. |
| 2 | **INSERT**  Creates a record. |
| 3 | **UPDATE**  Modifies records. |
| 4 | **DELETE**  Deletes records. |

### **DCL - Data Control Language**

|  |  |
| --- | --- |
|  | **Command & Description** |
| 1 | **GRANT**  Gives a privilege to user. |
| 2 | **REVOKE**  Takes back privileges granted from user. |

### **DQL - Data Query Language**

|  |  |
| --- | --- |
|  | **Command & Description** |
| 1 | **SELECT**  The SELECT statement is used to retrieve data from one or more tables. |
| 2 | **DISTINCT**  The DISTINCT keyword is used with SELECT to retrieve unique values from a specified column or a combination of columns. |
| 3 | **FROM**  The FROM clause specifies the table or tables from which you want to retrieve data. |
| 4 | **WHERE**  **The WHERE clause is used to filter rows based on a specified condition. It allows you to retrieve only the rows that meet the criteria you specify.** |
| 5 | **ORDER BY**  **The ORDER BY clause is used to sort the result set in ascending (ASC) or descending (DESC) order based on one or more columns.** |
| 6 | **GROUP BY**  **The GROUP BY clause is used to group rows with the same values in one or more columns into summary rows.** |
| 7 | **HAVING**  **The HAVING clause is used to filter the results of a GROUP BY query based on a condition applied to the aggregated values.** |

**WHAT DO YOU MEAN BY TABLE AND FIELD IN SQL?**

A table refers to a collection of data in an organised manner in form of rows and columns. A field refers to the number of columns in a table. For example:

**Table:** StudentInformation

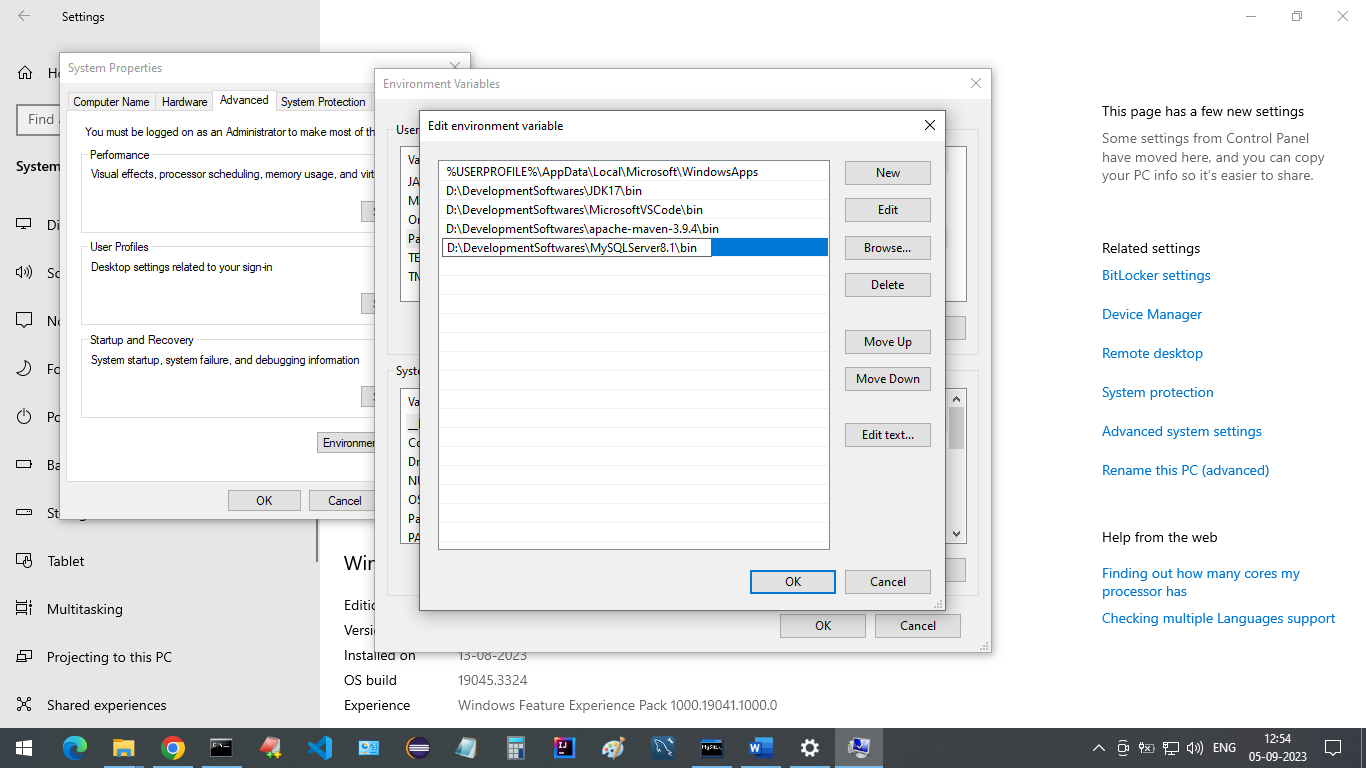
**Field:** StudentId, StudentName, StudentMarks

**Please follow the link to Learn how download and install MySQL Database and MySQL Workbench**

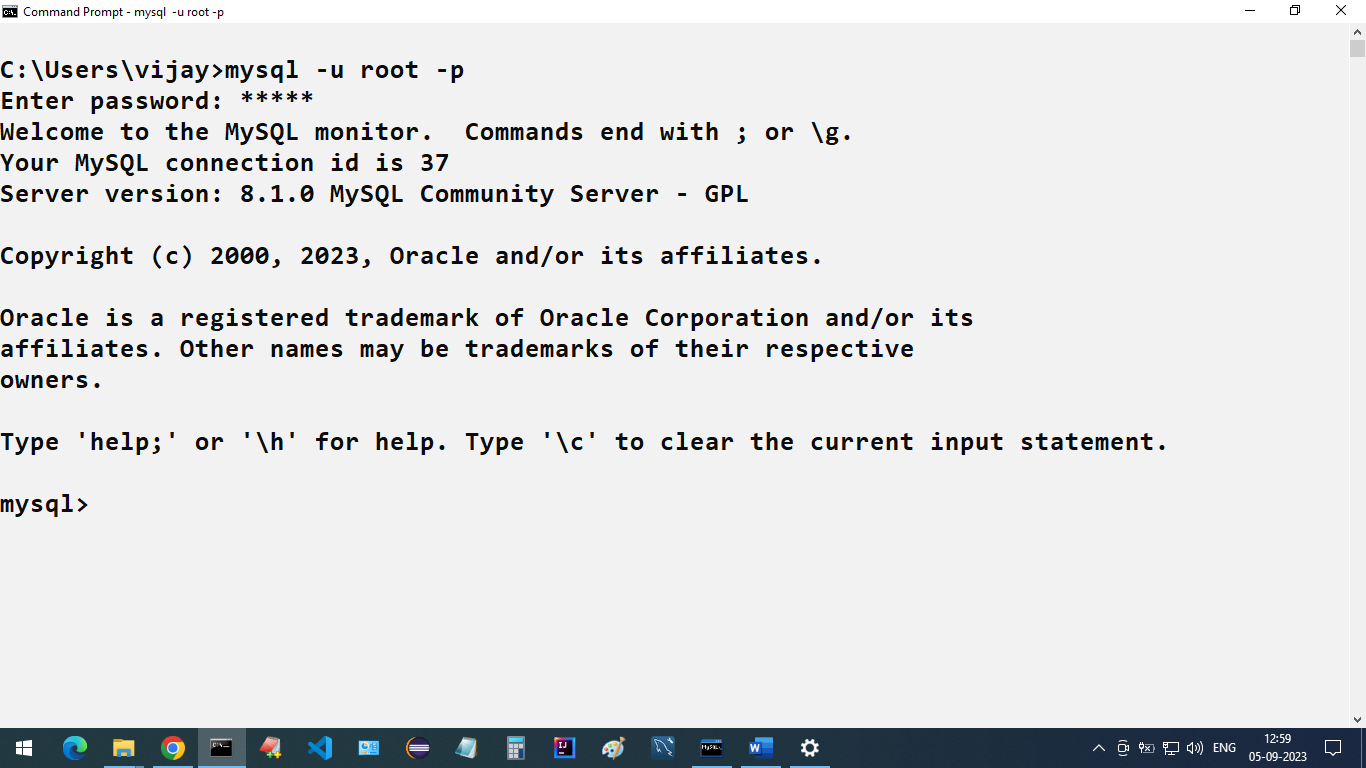
[**https://rb.gy/3hcwf**](https://rb.gy/3hcwf)

**Note:** If you are getting error while installing MySQL Server and MySQL Workbench like ‘MySql Workbench installer requires Visual C++ 2015’ then follow <https://aka.ms/vs/17/release/vc_redist.x64.exe> this link and download and install this piece of software.

**To Access the SQL Prompt from the windows command Line client set the path**

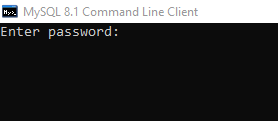


**HOW TO LOGIN WITH A PARTICULAR USER FROM THE WINDOWS CMD PROMPT?**



**HOW TO CHANGE USER IN THE MYSQL COMMAND LINE CLIENT?**

**Note:** By default when you launch, you will be asked to enter the password for the root user.



Later you can chage the user with the following command.

**SYSTEM mysql -u vijay -p;**

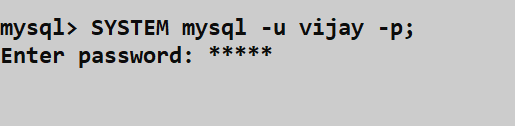
**Enter password: \*\*\*\*\***

**Alternatively you can use**

**\! mysql -u vijay -p**

**Enter password: \*\*\*\*\***

**Note:** In the Windows Command Prompt, you can log in with a specific user at the beginning. However, if you want to change the user, you can use the same command.



**HOW TO CLOSE MYSQL COMMAND LINE CLIENT AS WELL AS TO EXIT FROM THE MYSQL PROMPT FROM WINDOWS COMMAND PROMPT?**

**EXIT**

**HOW TO DISPLAY THE CURRENT USER?**

You can use the USER() function to retrieve the current user. The USER() function returns the current user name and host name combination that the server used to authenticate the current client.

**SELECT USER();**

**+----------------+**

**| USER() |**

**+----------------+**

**| root@localhost |**

**+----------------+**

**1 row in set (0.00 sec)**

**HOW TO DISPLAY ALL THE DATABASES?**

**SHOW DATABASES** command to get list of databases. Run the following query to show list of databases.

**SHOW DATABASES;**

**+--------------------+**

**| Database |**

**+--------------------+**

**| information\_schema |**

**| mysql |**

**| mysql\_notes |**

**| performance\_schema |**

**| student\_tracker |**

**| sys |**

**+--------------------+**

**HOW TO CREATE A NEW DATABASE?**

**CREATE DATABASE MYSQL\_NOTES;**

**Query OK, 1 row affected (0.01 sec)**

**SHOW DATABASES;**

**+--------------------+**

**| Database |**

**+--------------------+**

**| information\_schema |**

**| mysql |**

**| mysql\_notes |**

**| performance\_schema |**

**| student\_tracker |**

**| sys |**

**+--------------------+**

**HOW TO DELETE A DATABASE?**

**DROP DATABASE MYSQL\_NOTES;**

**Query OK, 0 rows affected (0.01 sec)**

**SHOW DATABASES;**

**+--------------------+**

**| Database |**

**+--------------------+**

**| information\_schema |**

**| mysql |**

**| performance\_schema |**

**| student\_tracker |**

**| sys |**

**+--------------------+**

**HOW TO SET OR SELECT A DATABASE?**

* Before doing anything first we need to connect to a database.

**USE MYSQL\_NOTES;**

**Database changed**

**HOW TO CHECK CURRENTLY WHICH DATABASE YOU ARE IN?**

**SELECT DATABASE();**

**+-------------+**

**| DATABASE() |**

**+-------------+**

**| mysql\_notes |**

**+-------------+**

**1 row in set (0.00 sec)**

**HOW TO CREATE A NEW USER?**

**CREATE USER 'new\_user'@'localhost' IDENTIFIED BY 'password';**

new\_user is the name we’ve given to our new user account and the IDENTIFIED BY ‘password’ section sets a passcode for this user. You can replace these values with your own, inside the quotation marks.

In order to grant all privileges of the database for a newly created user, execute the following command:

**GRANT ALL PRIVILEGES ON \* . \* TO 'new\_user'@'localhost';**

For changes to take effect immediately flush these privileges by typing in the command:

**FLUSH PRIVILEGES;**

**HOW TO DISPLAY ALL THE USERS FROM A DATABASE?**

**SELECT user, host FROM mysql.user;**

**In the above query mysql is the database.**

## IS IT MANDATORY TO KEEP SEMICOLON AFTER SQL STATEMENTS?

1. Some database systems require a semicolon at the end of each SQL statement.
2. Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.

**HOW TO CREATE A NEW USER WITH PASSWORD?**

**CREATE USER 'manager'@'localhost' IDENTIFIED BY 'admin';**

**CREATE USER 'vijay'@'localhost' IDENTIFIED BY 'admin';**

**HOW TO DROP EXISTING USER?**

**DROP USER 'manager'@'localhost';**

**DROP USER 'vijay'@'localhost';**

**HOW TO GRANT ALL PRIVILIAGES TO THE USER?**

**GRANT ALL PRIVILEGES ON \*.\* TO 'vijay'@'localhost';**

**HOW TO CHECK CURRENT USER PRIVILIAGES?**

**SHOW GRANTS FOR 'root'@'localhost';**

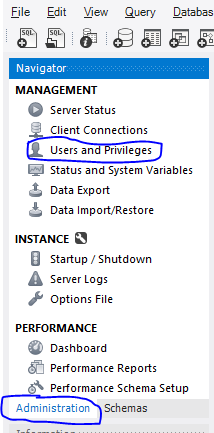
**HOW TO FOR CHANGES TO TAKE EFFECT IMMEDIATELY?**

**FLUSH PRIVILEGES;**

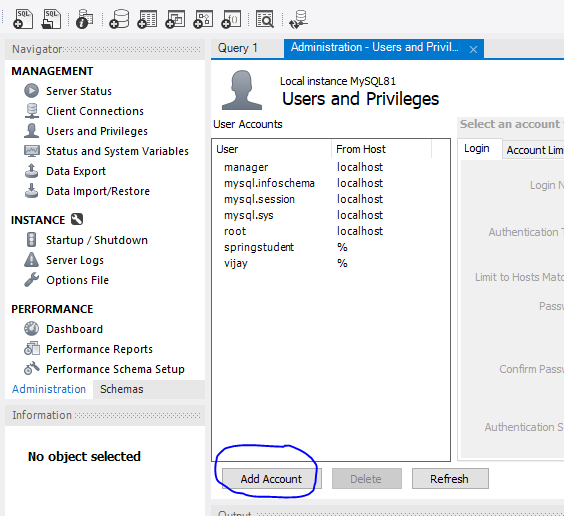
**Note:** Starting from MySQL 5.7.3, the FLUSH PRIVILEGES; statement is no longer strictly required after executing GRANT or REVOKE statements. The server automatically reloads the grant tables in these cases.

**HOW TO CREATE A NEW USER IN THE MYSQL WORKBENCH?**

1. Log in to any connection
2. Click on Administration on the left hand side

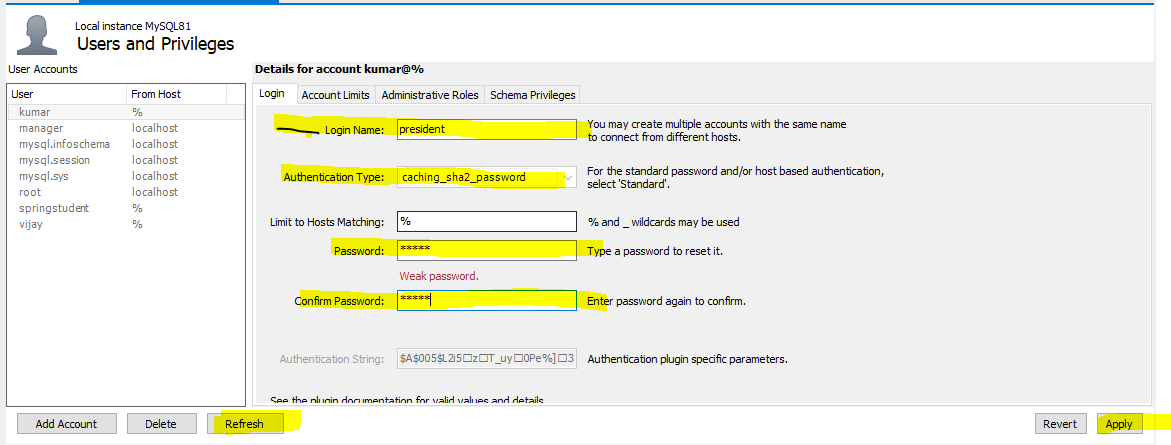


1. Click on Users and Privileges
2. Click on Add account to create a new account



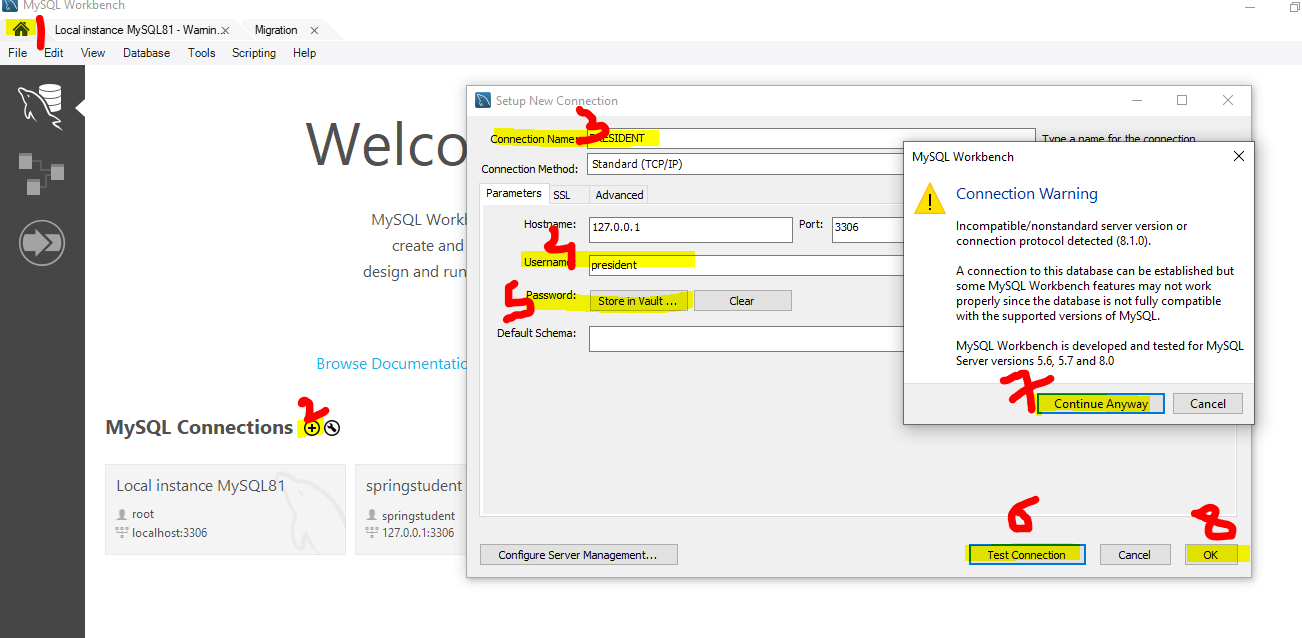
1. Fill in the deatails

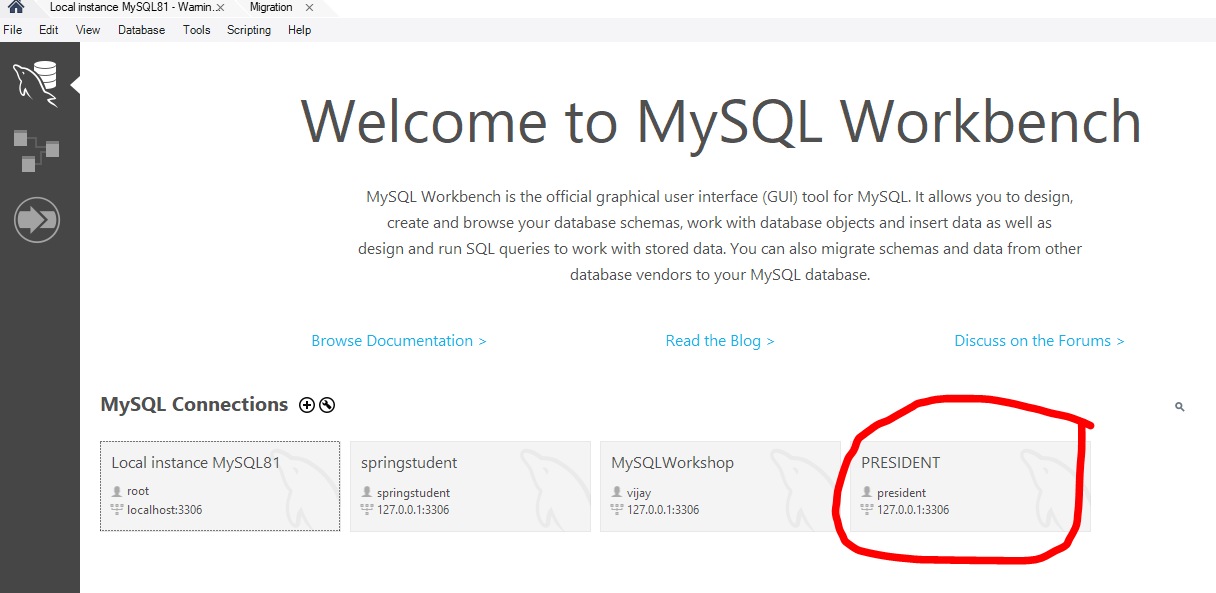
**Note:** Authentication type should be same as other accounts(check for root)



1. Click on Apply and Refresh

**HOW TO ADD A NEW CONNECTION TO THE MYSQL WORKBENCH HOME?**





**Note:** While creating the connection the user must be available(created already). Password is the user password that you have given at the time of creating a user.

**HOW TO CLEAR THE SCREEN IN MYSQL?**

\! cls

**HOW TO CREATE A SIMPLE TABLE?**

**CREATE TABLE STUDENT (ID INTEGER, FIRST\_NAME VARCHAR(90), AGE INTEGER, COURSE VARCHAR(10));**

**Query OK, 0 rows affected (0.03 sec)**

* INTEGER is a data type synonym for INT.
* You can use both INT and INTEGER datatype to specify number types.

**HOW TO INSERT RECORDS TO THE TABLE?**

**INSERT INTO STUDENT VALUES (101, 'ARUN', 20, 'CSE');**

**INSERT INTO STUDENT VALUES (102, 'BHAVESH', 21, 'ISE');**

**INSERT INTO STUDENT VALUES (103, 'CHAITANYA', 22, 'ECE');**

**INSERT INTO STUDENT VALUES (104, 'DEEPIKA', 23, 'MECH');**

**INSERT INTO STUDENT VALUES (105, 'DHANUSH', 24, 'DS');**

**INSERT INTO STUDENT VALUES (106, 'EKTA', 25, 'AI');**

**INSERT INTO STUDENT VALUES (107, 'GAURAV', 26, 'ARCH');**

**INSERT INTO STUDENT VALUES (108, 'HARSHITA', 27, 'CHEMICAL');**

**INSERT INTO STUDENT VALUES (109, 'ISHAAN', 28, 'CIVIL');**

**INSERT INTO STUDENT VALUES (110, 'JANU', 29, 'EEE');**

**HOW TO DISPLAY ALL THE RECORDS WITH ALL THE COLUMNS?**

**SELECT \* FROM STUDENT;**

**+------+------------+------+----------+**

**| ID | FIRST\_NAME | AGE | COURSE |**

**+------+------------+------+----------+**

**| 101 | ARUN | 20 | CSE |**

**| 102 | BHAVESH | 21 | ISE |**

**| 103 | CHAITANYA | 22 | ECE |**

**| 104 | DEEPIKA | 23 | MECH |**

**| 105 | DHANUSH | 24 | DS |**

**| 106 | EKTA | 25 | AI |**

**| 107 | GAURAV | 26 | ARCH |**

**| 108 | HARSHITA | 27 | CHEMICAL |**

**| 109 | ISHAAN | 28 | CIVIL |**

**| 110 | JANU | 29 | EEE |**

**+------+------------+------+----------+**

**10 rows in set (0.00 sec)**

**CAN WE INSERT NULL VALUES TO THE COLUMNS?**

* By default, columns will be allowing **duplicate** values.

**INSERT INTO STUDENT VALUES (110, 'JANU', 29, 'EEE');**

**Query OK, 1 row affected (0.00 sec)**

**SELECT \* FROM STUDENT;**

**+------+------------+------+----------+**

**| ID | FIRST\_NAME | AGE | COURSE |**

**+------+------------+------+----------+**

**| 101 | ARUN | 20 | CSE |**

**| 102 | BHAVESH | 21 | ISE |**

**| 103 | CHAITANYA | 22 | ECE |**

**| 104 | DEEPIKA | 23 | MECH |**

**| 105 | DHANUSH | 24 | DS |**

**| 106 | EKTA | 25 | AI |**

**| 107 | GAURAV | 26 | ARCH |**

**| 108 | HARSHITA | 27 | CHEMICAL |**

**| 109 | ISHAAN | 28 | CIVIL |**

**| 110 | JANU | 29 | EEE |**

**| 110 | JANU | 29 | EEE |**

**+------+------------+------+----------+**

**11 rows in set (0.00 sec)**

* By default, columns will be allowing ‘null’ values.
* In MySQL, NULL represents an unknown or missing value in a database table.

**INSERT INTO STUDENT(ID, FIRST\_NAME) VALUES(111, 'PRANAV');**

**Query OK, 1 row affected (0.01 sec)**

**SELECT \* FROM STUDENT;**

**+------+------------+------+----------+**

**| ID | FIRST\_NAME | AGE | COURSE |**

**+------+------------+------+----------+**

**| 101 | ARUN | 20 | CSE |**

**| 102 | BHAVESH | 21 | ISE |**

**| 103 | CHAITANYA | 22 | ECE |**

**| 104 | DEEPIKA | 23 | MECH |**

**| 105 | DHANUSH | 24 | DS |**

**| 106 | EKTA | 25 | AI |**

**| 107 | GAURAV | 26 | ARCH |**

**| 108 | HARSHITA | 27 | CHEMICAL |**

**| 109 | ISHAAN | 28 | CIVIL |**

**| 110 | JANU | 29 | EEE |**

**| 110 | JANU | 29 | EEE |**

**| 111 | PRANAV | NULL | NULL |**

**+------+------------+------+----------+**

**12 rows in set (0.00 sec)**

**HOW TO UPDATE SINGLE COLUMN IN THE RECORD?**

**UPDATE STUDENT SET FIRST\_NAME = 'RISHI' WHERE ID = 108;**

**Query OK, 1 row affected (0.01 sec)**

**Rows matched: 1 Changed: 1 Warnings: 0**

**SELECT \* FROM STUDENT;**

**+------+------------+------+----------+**

**| ID | FIRST\_NAME | AGE | COURSE |**

**+------+------------+------+----------+**

**| 101 | ARUN | 20 | CSE |**

**| 102 | BHAVESH | 21 | ISE |**

**| 103 | CHAITANYA | 22 | ECE |**

**| 104 | DEEPIKA | 23 | MECH |**

**| 105 | DHANUSH | 24 | DS |**

**| 106 | EKTA | 25 | AI |**

**| 107 | GAURAV | 26 | ARCH |**

**| 108 | RISHI | 27 | CHEMICAL |**

**| 109 | ISHAAN | 28 | CIVIL |**

**| 110 | JANU | 29 | EEE |**

**| 110 | JANU | 29 | EEE |**

**| 111 | PRANAV | NULL | NULL |**

**+------+------------+------+----------+**

**12 rows in set (0.00 sec)**

**HOW TO UPDATE MULTIPLE COLUMNS IN THE RECORD?**

**UPDATE STUDENT SET ID = 112, FIRST\_NAME = 'RAJAT', AGE = 29, COURSE = 'AUTOMOBILE' WHERE ID = 105;**

**Query OK, 1 row affected (0.01 sec)**

**Rows matched: 1 Changed: 1 Warnings: 0**

**SELECT \* FROM STUDENT;**

**+------+------------+------+------------+**

**| ID | FIRST\_NAME | AGE | COURSE |**

**+------+------------+------+------------+**

**| 101 | ARUN | 20 | CSE |**

**| 102 | BHAVESH | 21 | ISE |**

**| 103 | CHAITANYA | 22 | ECE |**

**| 104 | DEEPIKA | 23 | MECH |**

**| 112 | RAJAT | 29 | AUTOMOBILE |**

**| 106 | EKTA | 25 | AI |**

**| 107 | GAURAV | 26 | ARCH |**

**| 108 | RISHI | 27 | CHEMICAL |**

**| 109 | ISHAAN | 28 | CIVIL |**

**| 110 | JANU | 29 | EEE |**

**| 110 | JANU | 29 | EEE |**

**| 111 | PRANAV | NULL | NULL |**

**+------+------------+------+------------+**

**12 rows in set (0.00 sec)**

**WHAT IS `NULL` IN SQL?**

NULL is a special marker in SQL that represents the absence of a value or a undefined value in a database.

Note: `NULL` is case insensitive

**HOW TO USE `IS NULL`?**

IS NULL is a condition used to check if a particular column in a database table has a NULL value.

**UPDATE STUDENT SET AGE = 30 WHERE AGE IS NULL;**

**Query OK, 1 row affected (0.01 sec)**

**Rows matched: 1 Changed: 1 Warnings: 0**

**SELECT \* FROM STUDENT;**

**+------+------------+------+------------+**

**| ID | FIRST\_NAME | AGE | COURSE |**

**+------+------------+------+------------+**

**| 101 | ARUN | 20 | CSE |**

**| 102 | BHAVESH | 21 | ISE |**

**| 103 | CHAITANYA | 22 | ECE |**

**| 104 | DEEPIKA | 23 | MECH |**

**| 112 | RAJAT | 29 | AUTOMOBILE |**

**| 106 | EKTA | 25 | AI |**

**| 107 | GAURAV | 26 | ARCH |**

**| 108 | RISHI | 27 | CHEMICAL |**

**| 109 | ISHAAN | 28 | CIVIL |**

**| 110 | JANU | 29 | EEE |**

**| 110 | JANU | 29 | EEE |**

**| 111 | PRANAV | 30 | NULL |**

**+------+------------+------+------------+**

**12 rows in set (0.00 sec)**

**HOW TO USE `IS NOT NULL`?**

The IS NOT NULL condition is used to filter rows where a particular column does not contain a NULL value. It is the opposite of the IS NULL condition.

**UPDATE STUDENT SET AGE = 20 WHERE FIRST\_NAME IS NOT NULL;**

**Query OK, 11 rows affected (0.01 sec)**

**Rows matched: 12 Changed: 11 Warnings: 0**

**SELECT \* FROM STUDENT;**

**+------+------------+------+------------+**

**| ID | FIRST\_NAME | AGE | COURSE |**

**+------+------------+------+------------+**

**| 101 | ARUN | 20 | CSE |**

**| 102 | BHAVESH | 20 | ISE |**

**| 103 | CHAITANYA | 20 | ECE |**

**| 104 | DEEPIKA | 20 | MECH |**

**| 112 | RAJAT | 20 | AUTOMOBILE |**

**| 106 | EKTA | 20 | AI |**

**| 107 | GAURAV | 20 | ARCH |**

**| 108 | RISHI | 20 | CHEMICAL |**

**| 109 | ISHAAN | 20 | CIVIL |**

**| 110 | JANU | 20 | EEE |**

**| 110 | JANU | 20 | EEE |**

**| 111 | PRANAV | 20 | NULL |**

**+------+------------+------+------------+**

**12 rows in set (0.00 sec)**

**HOW TO DELETE ALL THE RECORDS FROM A TABLE?**

**DELETE FROM STUDENT;**

**Query OK, 12 rows affected (0.01 sec)**

**HOW TO INSERT RECORDS USING A SINGLE STATMENT?**

**INSERT INTO STUDENT VALUES**

**(101, 'ARUN', 20, 'CSE'),**

**(102, 'BHAVESH', 21, 'ISE'),**

**(103, 'CHAITANYA', 22, 'ECE'),**

**(104, 'DEEPIKA', 23, 'MECH'),**

**(105, 'DHANUSH', 24, 'DS'),**

**(106, 'EKTA', 25, 'AI'),**

**(107, 'GAURAV', 26, 'ARCH'),**

**(108, 'HARSHITA', 27, 'CHEMICAL'),**

**(109, 'ISHAAN', 28, 'CIVIL'),**

**(110, 'JANU', 29, 'EEE');**

**SELECT \* FROM STUDENT;**

**+------+------------+------+----------+**

**| ID | FIRST\_NAME | AGE | COURSE |**

**+------+------------+------+----------+**

**| 101 | ARUN | 20 | CSE |**

**| 102 | BHAVESH | 21 | ISE |**

**| 103 | CHAITANYA | 22 | ECE |**

**| 104 | DEEPIKA | 23 | MECH |**

**| 105 | DHANUSH | 24 | DS |**

**| 106 | EKTA | 25 | AI |**

**| 107 | GAURAV | 26 | ARCH |**

**| 108 | HARSHITA | 27 | CHEMICAL |**

**| 109 | ISHAAN | 28 | CIVIL |**

**| 110 | JANU | 29 | EEE |**

**+------+------------+------+----------+**

**10 rows in set (0.00 sec)**

**HOW WOULD YOU UPDATE THE FIRST\_NAME COLUMN IN THE STUDENT TABLE FOR ALL RECORDS WHERE THE ID IS GREATER THAN 104, SETTING THE FIRST\_NAME TO 'ANANYA'?**

**UPDATE STUDENT SET FIRST\_NAME = 'ANANYA' WHERE ID > 104;**

**Query OK, 6 rows affected (0.01 sec)**

**Rows matched: 6 Changed: 6 Warnings: 0**

**SELECT \* FROM STUDENT;**

**+------+------------+------+----------+**

**| ID | FIRST\_NAME | AGE | COURSE |**

**+------+------------+------+----------+**

**| 101 | ARUN | 20 | CSE |**

**| 102 | BHAVESH | 21 | ISE |**

**| 103 | CHAITANYA | 22 | ECE |**

**| 104 | DEEPIKA | 23 | MECH |**

**| 105 | ANANYA | 24 | DS |**

**| 106 | ANANYA | 25 | AI |**

**| 107 | ANANYA | 26 | ARCH |**

**| 108 | ANANYA | 27 | CHEMICAL |**

**| 109 | ANANYA | 28 | CIVIL |**

**| 110 | ANANYA | 29 | EEE |**

**+------+------------+------+----------+**

**10 rows in set (0.00 sec)**

**HOW WOULD YOU UPDATE MULTILE COLUMNS?**

**UPDATE STUDENT SET AGE = 22, ID = 10 WHERE ID <= 107;**

**Query OK, 7 rows affected (0.00 sec)**

**Rows matched: 7 Changed: 7 Warnings: 0**

**SELECT \* FROM STUDENT;**

**+------+------------+------+----------+**

**| ID | FIRST\_NAME | AGE | COURSE |**

**+------+------------+------+----------+**

**| 10 | ARUN | 22 | CSE |**

**| 10 | BHAVESH | 22 | ISE |**

**| 10 | CHAITANYA | 22 | ECE |**

**| 10 | DEEPIKA | 22 | MECH |**

**| 10 | ANANYA | 22 | DS |**

**| 10 | ANANYA | 22 | AI |**

**| 10 | ANANYA | 22 | ARCH |**

**| 108 | ANANYA | 27 | CHEMICAL |**

**| 109 | ANANYA | 28 | CIVIL |**

**| 110 | ANANYA | 29 | EEE |**

**+------+------------+------+----------+**

**10 rows in set (0.00 sec)**

**HOW WOULD YOU UPDATE ALL THE COLUMNS?**

**UPDATE STUDENT SET AGE = 42, ID = 15;**

**Query OK, 10 rows affected (0.00 sec)**

**Rows matched: 10 Changed: 10 Warnings: 0**

**SELECT \* FROM STUDENT;**

**+------+------------+------+----------+**

**| ID | FIRST\_NAME | AGE | COURSE |**

**+------+------------+------+----------+**

**| 15 | ARUN | 42 | CSE |**

**| 15 | BHAVESH | 42 | ISE |**

**| 15 | CHAITANYA | 42 | ECE |**

**| 15 | DEEPIKA | 42 | MECH |**

**| 15 | ANANYA | 42 | DS |**

**| 15 | ANANYA | 42 | AI |**

**| 15 | ANANYA | 42 | ARCH |**

**| 15 | ANANYA | 42 | CHEMICAL |**

**| 15 | ANANYA | 42 | CIVIL |**

**| 15 | ANANYA | 42 | EEE |**

**+------+------------+------+----------+**

**10 rows in set (0.00 sec)**

**DELETE FROM STUDENT;**

**Query OK, 10 rows affected (0.01 sec)**

**HOW WOULD YOU EXECUTE MULTIPLE STATMENTS IN THE SQL WORKBENCH?**

1. Write Your SQL Statements:

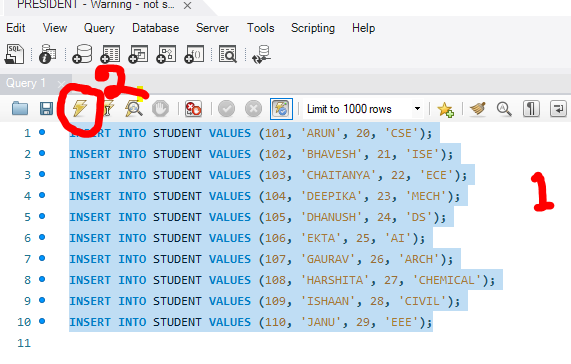
Open SQL Workbench and write the SQL statements you want to execute. Separate each statement with a semicolon (;).

1. Highlight the Statements:

Highlight all the SQL statements you want to execute.

1. Execute the Statements:

Execute the highlighted statements by either clicking on the "Execute" button(flash symbol), or pressing the appropriate shortcut (e.g., F5), or selecting the "Execute SQL" option from the menu.



**INSERT INTO STUDENT VALUES (101, 'ARUN', 20, 'CSE');**

**INSERT INTO STUDENT VALUES (102, 'BHAVESH', 21, 'ISE');**

**INSERT INTO STUDENT VALUES (103, 'CHAITANYA', 22, 'ECE');**

**INSERT INTO STUDENT VALUES (104, 'DEEPIKA', 23, 'MECH');**

**INSERT INTO STUDENT VALUES (105, 'DHANUSH', 24, 'DS');**

**INSERT INTO STUDENT VALUES (106, 'EKTA', 25, 'AI');**

**INSERT INTO STUDENT VALUES (107, 'GAURAV', 26, 'ARCH');**

**INSERT INTO STUDENT VALUES (108, 'HARSHITA', 27, 'CHEMICAL');**

**INSERT INTO STUDENT VALUES (109, 'ISHAAN', 28, 'CIVIL');**

**INSERT INTO STUDENT VALUES (110, 'JANU', 29, 'EEE');**

**SELECT \* FROM STUDENT;**

**+------+------------+------+----------+**

**| ID | FIRST\_NAME | AGE | COURSE |**

**+------+------------+------+----------+**

**| 101 | ARUN | 20 | CSE |**

**| 102 | BHAVESH | 21 | ISE |**

**| 103 | CHAITANYA | 22 | ECE |**

**| 104 | DEEPIKA | 23 | MECH |**

**| 105 | DHANUSH | 24 | DS |**

**| 106 | EKTA | 25 | AI |**

**| 107 | GAURAV | 26 | ARCH |**

**| 108 | HARSHITA | 27 | CHEMICAL |**

**| 109 | ISHAAN | 28 | CIVIL |**

**| 110 | JANU | 29 | EEE |**

**+------+------------+------+----------+**

**10 rows in set (0.00 sec)**

**HOW WOULD YOU HOW YOU WOULD USE AN SQL DELETE STATEMENT TO REMOVE A SPECIFIC STUDENT RECORD WITH THE ID OF 6 FROM THE STUDENT TABLE?**

**DELETE FROM STUDENT WHERE ID = 6;**

**Query OK, 0 rows affected (0.00 sec)**

**DELETE FROM STUDENT WHERE FIRST\_NAME = 'ISHAAN';**

**Query OK, 1 row affected (0.00 sec)**

**SELECT \* FROM STUDENT;**

**+------+------------+------+----------+**

**| ID | FIRST\_NAME | AGE | COURSE |**

**+------+------------+------+----------+**

**| 101 | ARUN | 20 | CSE |**

**| 102 | BHAVESH | 21 | ISE |**

**| 103 | CHAITANYA | 22 | ECE |**

**| 104 | DEEPIKA | 23 | MECH |**

**| 105 | DHANUSH | 24 | DS |**

**| 106 | EKTA | 25 | AI |**

**| 107 | GAURAV | 26 | ARCH |**

**| 108 | HARSHITA | 27 | CHEMICAL |**

**| 110 | JANU | 29 | EEE |**

**+------+------------+------+----------+**

**9 rows in set (0.00 sec)**

**DELETE FROM STUDENT;**

**Query OK, 9 rows affected (0.01 sec)**

**SELECT \* FROM STUDENT;**

**Empty set (0.00 sec)**

**HOW WOULD YOU REMOVE A TABLE FROM THE DATABASE?**

**DROP TABLE STUDENT;**

**Query OK, 0 rows affected (0.02 sec)**

**CREATE TABLE EMPLOYEE (ID INTEGER, FIRST\_NAME VARCHAR(90), LAST\_NAME VARCHAR(90), AGE INTEGER, SALARY INTEGER, EMAIL VARCHAR(90));**

**Query OK, 0 rows affected (0.03 sec)**

**INSERT INTO EMPLOYEE VALUES(1, 'ARUN', 'PATEL', 22, 40000, 'ARUN@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(2, 'BHAVESH', 'SHARMA', 24, 30000, 'BHAVESH@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(3, 'CHAITANYA', 'SINGH', 23, 50000, 'CHAITANYA@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(4, 'DEEPIKA', 'GUPTA', 26, 55000, 'DEEPIKA@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(5, 'DHANUSH', 'KUMAR', 25, 20000, 'DHANUSH@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(6, 'EKTA', 'YADAV', 28, 35000, 'YADAV@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(7, 'GAURAV', 'RAO', 21, 60000, 'GAURAV@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(8, 'HARSHITA', 'REDDY', 29, 56000, 'HARSHITA@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(9, 'ISHAAN', 'REDDY', 32, 70000, 'ISHAAN@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(10, 'JANU', 'MUKHERJEE', 30, 53000, 'JANU@GCOMPANY.IN');**

**SELECT \* FROM EMPLOYEE;**

**+------+------------+------------+------+--------+-----------------------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |**

**+------+------------+------------+------+--------+-----------------------+**

**| 1 | ARUN | PATEL | 22 | 40000 | ARUN@GCOMPANY.IN |**

**| 2 | BHAVESH | SHARMA | 24 | 30000 | BHAVESH@GCOMPANY.IN |**

**| 3 | CHAITANYA | SINGH | 23 | 50000 | CHAITANYA@GCOMPANY.IN |**

**| 4 | DEEPIKA | GUPTA | 26 | 55000 | DEEPIKA@GCOMPANY.IN |**

**| 5 | DHANUSH | KUMAR | 25 | 20000 | DHANUSH@GCOMPANY.IN |**

**| 6 | EKTA | YADAV | 28 | 35000 | YADAV@GCOMPANY.IN |**

**| 7 | GAURAV | RAO | 21 | 60000 | GAURAV@GCOMPANY.IN |**

**| 8 | HARSHITA | REDDY | 29 | 56000 | HARSHITA@GCOMPANY.IN |**

**| 9 | ISHAAN | REDDY | 32 | 70000 | ISHAAN@GCOMPANY.IN |**

**| 10 | JANU | MUKHERJEE | 30 | 53000 | JANU@GCOMPANY.IN |**

**+------+------------+------------+------+--------+-----------------------+**

**10 rows in set (0.00 sec)**

**WRITE AN SQL QUERY THAT RETRIEVES RECORDS OF THE EMPLOYEE TABLE WITH THE ID OF 5 FROM THE EMPLOYEE TABLE?**

**SELECT \* FROM EMPLOYEE WHERE ID = 5;**

**+------+------------+-----------+------+--------+---------------------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |**

**+------+------------+-----------+------+--------+---------------------+**

**| 5 | DHANUSH | KUMAR | 25 | 20000 | DHANUSH@GCOMPANY.IN |**

**+------+------------+-----------+------+--------+---------------------+**

**1 row in set (0.00 sec)**

**WRITE AN SQL QUERY THAT RETRIEVES RECORDS OF THE EMPLOYEE TABLE WITH THE ID GREATER THAN 5 FROM THE EMPLOYEE TABLE?**

**SELECT \* FROM EMPLOYEE WHERE ID > 5;**

**+------+------------+------------+------+--------+----------------------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |**

**+------+------------+------------+------+--------+----------------------+**

**| 6 | EKTA | YADAV | 28 | 35000 | YADAV@GCOMPANY.IN |**

**| 7 | GAURAV | RAO | 21 | 60000 | GAURAV@GCOMPANY.IN |**

**| 8 | HARSHITA | REDDY | 29 | 56000 | HARSHITA@GCOMPANY.IN |**

**| 9 | ISHAAN | REDDY | 32 | 70000 | ISHAAN@GCOMPANY.IN |**

**| 10 | JANU | MUKHERJEE | 30 | 53000 | JANU@GCOMPANY.IN |**

**+------+------------+------------+------+--------+----------------------+**

**5 rows in set (0.00 sec)**

**WRITE AN SQL QUERY THAT RETRIEVES RECORDS OF THE EMPLOYEE TABLE WHERE AGE RANGE OF 22 TO 28 FROM THE EMPLOYEE TABLE?**

**SELECT \* FROM EMPLOYEE WHERE AGE BETWEEN 22 AND 28;**

**+------+------------+-----------+------+--------+-----------------------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |**

**+------+------------+-----------+------+--------+-----------------------+**

**| 1 | ARUN | PATEL | 22 | 40000 | ARUN@GCOMPANY.IN |**

**| 2 | BHAVESH | SHARMA | 24 | 30000 | BHAVESH@GCOMPANY.IN |**

**| 3 | CHAITANYA | SINGH | 23 | 50000 | CHAITANYA@GCOMPANY.IN |**

**| 4 | DEEPIKA | GUPTA | 26 | 55000 | DEEPIKA@GCOMPANY.IN |**

**| 5 | DHANUSH | KUMAR | 25 | 20000 | DHANUSH@GCOMPANY.IN |**

**| 6 | EKTA | YADAV | 28 | 35000 | YADAV@GCOMPANY.IN |**

**+------+------------+-----------+------+--------+-----------------------+**

**6 rows in set (0.00 sec)**

**WRITE AN SQL QUERY THAT RETRIEVES RECORDS OF THE EMPLOYEE TABLE WHERE AGE NOT IN THE RANGE OF 22 TO 28 FROM THE EMPLOYEE TABLE?**

**SELECT \* FROM EMPLOYEE WHERE AGE NOT BETWEEN 22 AND 28;**

**+------+------------+------------+------+--------+----------------------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |**

**+------+------------+------------+------+--------+----------------------+**

**| 7 | GAURAV | RAO | 21 | 60000 | GAURAV@GCOMPANY.IN |**

**| 8 | HARSHITA | REDDY | 29 | 56000 | HARSHITA@GCOMPANY.IN |**

**| 9 | ISHAAN | REDDY | 32 | 70000 | ISHAAN@GCOMPANY.IN |**

**| 10 | JANU | MUKHERJEE | 30 | 53000 | JANU@GCOMPANY.IN |**

**+------+------------+------------+------+--------+----------------------+**

**4 rows in set (0.00 sec)**

**WRITE AN SQL QUERY THAT RETRIEVES RECORDS OF THE EMPLOYEE TABLE RETRIEVE DETAILS FOR EMPLOYEES WHOSE SALARIES MATCH SPECIFIC VALUES FROM THE EMPLOYEE TABLE?**

**SELECT \* FROM EMPLOYEE WHERE SALARY IN (40000, 55000, 70000);**

**+------+------------+-----------+------+--------+---------------------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |**

**+------+------------+-----------+------+--------+---------------------+**

**| 1 | ARUN | PATEL | 22 | 40000 | ARUN@GCOMPANY.IN |**

**| 4 | DEEPIKA | GUPTA | 26 | 55000 | DEEPIKA@GCOMPANY.IN |**

**| 9 | ISHAAN | REDDY | 32 | 70000 | ISHAAN@GCOMPANY.IN |**

**+------+------------+-----------+------+--------+---------------------+**

**3 rows in set (0.00 sec)**

**WRITE AN SQL QUERY THAT RETRIEVES RECORDS OF THE EMPLOYEE TABLE RETRIEVE DETAILS FOR EMPLOYEES WHOSE SALARIES DOESNT MATCH SPECIFIC VALUES FROM THE EMPLOYEE TABLE?**

**SELECT \* FROM EMPLOYEE WHERE SALARY NOT IN (40000, 55000, 70000);**

**+------+------------+------------+------+--------+-----------------------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |**

**+------+------------+------------+------+--------+-----------------------+**

**| 2 | BHAVESH | SHARMA | 24 | 30000 | BHAVESH@GCOMPANY.IN |**

**| 3 | CHAITANYA | SINGH | 23 | 50000 | CHAITANYA@GCOMPANY.IN |**

**| 5 | DHANUSH | KUMAR | 25 | 20000 | DHANUSH@GCOMPANY.IN |**

**| 6 | EKTA | YADAV | 28 | 35000 | YADAV@GCOMPANY.IN |**

**| 7 | GAURAV | RAO | 21 | 60000 | GAURAV@GCOMPANY.IN |**

**| 8 | HARSHITA | REDDY | 29 | 56000 | HARSHITA@GCOMPANY.IN |**

**| 10 | JANU | MUKHERJEE | 30 | 53000 | JANU@GCOMPANY.IN |**

**+------+------------+------------+------+--------+-----------------------+**

**7 rows in set (0.00 sec)**

**WRITE AN SQL QUERY THAT RETRIEVES RECORDS OF THE EMPLOYEE TABLE WHERE NAMES OF ALL EMPLOYEES WHOSE FIRST NAME INCLUDES THE LETTER "R" FROM THE EMPLOYEE TABLE?**

**SELECT \* FROM EMPLOYEE WHERE FIRST\_NAME LIKE '%R%';**

**+------+------------+-----------+------+--------+----------------------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |**

**+------+------------+-----------+------+--------+----------------------+**

**| 1 | ARUN | PATEL | 22 | 40000 | ARUN@GCOMPANY.IN |**

**| 7 | GAURAV | RAO | 21 | 60000 | GAURAV@GCOMPANY.IN |**

**| 8 | HARSHITA | REDDY | 29 | 56000 | HARSHITA@GCOMPANY.IN |**

**+------+------------+-----------+------+--------+----------------------+**

**3 rows in set (0.00 sec)**

**WRITE AN SQL QUERY THAT RETRIEVES RECORDS OF THE EMPLOYEE TABLE WHERE NAMES OF ALL EMPLOYEES WHOSE FIRST NAME ENDING WITH THE LETTER "R" FROM THE EMPLOYEE TABLE?**

**SELECT \* FROM EMPLOYEE WHERE FIRST\_NAME LIKE '%A';**

**+------+------------+-----------+------+--------+-----------------------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |**

**+------+------------+-----------+------+--------+-----------------------+**

**| 3 | CHAITANYA | SINGH | 23 | 50000 | CHAITANYA@GCOMPANY.IN |**

**| 4 | DEEPIKA | GUPTA | 26 | 55000 | DEEPIKA@GCOMPANY.IN |**

**| 6 | EKTA | YADAV | 28 | 35000 | YADAV@GCOMPANY.IN |**

**| 8 | HARSHITA | REDDY | 29 | 56000 | HARSHITA@GCOMPANY.IN |**

**+------+------------+-----------+------+--------+-----------------------+**

**4 rows in set (0.00 sec)**

**WRITE AN SQL QUERY THAT RETRIEVES RECORDS OF THE EMPLOYEE TABLE WHERE NAMES OF ALL EMPLOYEES WHOSE FIRST NAME STARTING WITH THE LETTER "R" FROM THE EMPLOYEE TABLE?**

**SELECT \* FROM EMPLOYEE WHERE FIRST\_NAME LIKE 'A%';**

**+------+------------+-----------+------+--------+------------------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |**

**+------+------------+-----------+------+--------+------------------+**

**| 1 | ARUN | PATEL | 22 | 40000 | ARUN@GCOMPANY.IN |**

**+------+------------+-----------+------+--------+------------------+**

**1 row in set (0.00 sec)**

**WRITE AN SQL QUERY THAT RETRIEVES RECORDS WHERE FIRST NAMES AND AGES OF ALL EMPLOYEES FROM THE EMPLOYEE TABLE?**

**SELECT FIRST\_NAME, AGE FROM EMPLOYEE;**

**+------------+------+**

**| FIRST\_NAME | AGE |**

**+------------+------+**

**| ARUN | 22 |**

**| BHAVESH | 24 |**

**| CHAITANYA | 23 |**

**| DEEPIKA | 26 |**

**| DHANUSH | 25 |**

**| EKTA | 28 |**

**| GAURAV | 21 |**

**| HARSHITA | 29 |**

**| ISHAAN | 32 |**

**| JANU | 30 |**

**+------------+------+**

**10 rows in set (0.00 sec)**

**WRITE AN SQL QUERY THAT RETRIEVES RECORDS WHERE FIRST NAMES, AGE AND EMAIL OF ALL EMPLOYEES FROM THE EMPLOYEE TABLE?**

**SELECT FIRST\_NAME, AGE, EMAIL FROM EMPLOYEE;**

**+------------+------+-----------------------+**

**| FIRST\_NAME | AGE | EMAIL |**

**+------------+------+-----------------------+**

**| ARUN | 22 | ARUN@GCOMPANY.IN |**

**| BHAVESH | 24 | BHAVESH@GCOMPANY.IN |**

**| CHAITANYA | 23 | CHAITANYA@GCOMPANY.IN |**

**| DEEPIKA | 26 | DEEPIKA@GCOMPANY.IN |**

**| DHANUSH | 25 | DHANUSH@GCOMPANY.IN |**

**| EKTA | 28 | YADAV@GCOMPANY.IN |**

**| GAURAV | 21 | GAURAV@GCOMPANY.IN |**

**| HARSHITA | 29 | HARSHITA@GCOMPANY.IN |**

**| ISHAAN | 32 | ISHAAN@GCOMPANY.IN |**

**| JANU | 30 | JANU@GCOMPANY.IN |**

**+------------+------+-----------------------+**

**10 rows in set (0.00 sec)**

**WRITE AN SQL QUERY THAT RETRIEVES RECORDS USING ALIAS NAMES FOR THE COLUMNS USING `AS` KEYWORD FROM THE EMPLOYEE TABLE?**

**SELECT FIRST\_NAME AS NAME, AGE AS EMPLOYEE\_AGE, LAST\_NAME FROM EMPLOYEE;**

**+------------+--------------+------------+**

**| NAME | EMPLOYEE\_AGE | LAST\_NAME |**

**+------------+--------------+------------+**

**| ARUN | 22 | PATEL |**

**| BHAVESH | 24 | SHARMA |**

**| CHAITANYA | 23 | SINGH |**

**| DEEPIKA | 26 | GUPTA |**

**| DHANUSH | 25 | KUMAR |**

**| EKTA | 28 | YADAV |**

**| GAURAV | 21 | RAO |**

**| HARSHITA | 29 | REDDY |**

**| ISHAAN | 32 | REDDY |**

**| JANU | 30 | MUKHERJEE |**

**+------------+--------------+------------+**

**10 rows in set (0.00 sec)**

**WRITE AN SQL QUERY THAT RETRIEVES RECORDS USING ALIAS NAMES FOR THE COLUMNS WITHOUT USING `AS` KEYWORD FROM THE EMPLOYEE TABLE?**

**SELECT FIRST\_NAME NAME, AGE EMPLOYEE\_AGE, LAST\_NAME FROM EMPLOYEE;**

**+------------+--------------+------------+**

**| NAME | EMPLOYEE\_AGE | LAST\_NAME |**

**+------------+--------------+------------+**

**| ARUN | 22 | PATEL |**

**| BHAVESH | 24 | SHARMA |**

**| CHAITANYA | 23 | SINGH |**

**| DEEPIKA | 26 | GUPTA |**

**| DHANUSH | 25 | KUMAR |**

**| EKTA | 28 | YADAV |**

**| GAURAV | 21 | RAO |**

**| HARSHITA | 29 | REDDY |**

**| ISHAAN | 32 | REDDY |**

**| JANU | 30 | MUKHERJEE |**

**+------------+--------------+------------+**

**10 rows in set (0.00 sec)**

**WRITE AN SQL QUERY THAT RETRIEVES THE TOTAL NUMBER OF EMPLOYEES IN THE COMPANY?**

The COUNT(\*) function in SQL is used to count the number of rows in a table or the result set of a query. It can be used in various ways to analyze and retrieve information from your data.

Here's a breakdown of what COUNT(\*) does:

Counts all rows: The asterisk (\*) indicates that all columns in every row should be counted, regardless of their value (including null values).

Returns an integer: The function returns a single integer value representing the total number of rows counted.

Used in SELECT statements: COUNT(\*) is typically used within a SELECT statement, often in conjunction with other functions like WHERE clauses to filter the data before counting.

**EXAMPLE: 1**

**SELECT COUNT(\*) FROM EMPLOYEE;**

**+----------+**

**| COUNT(\*) |**

**+----------+**

**| 10 |**

**+----------+**

**1 row in set (0.01 sec)**

**EXAMPLE: 2**

**SELECT COUNT(\*) AS "RECORDS COUNT" FROM EMPLOYEE;**

**+---------------+**

**| RECORDS COUNT |**

**+---------------+**

**| 10 |**

**+---------------+**

**1 row in set (0.00 sec)**

**EXAMPLE: 3**

**SELECT COUNT(\*) "RECORDS COUNT" FROM EMPLOYEE;**

**+---------------+**

**| RECORDS COUNT |**

**+---------------+**

**| 10 |**

**+---------------+**

**1 row in set (0.00 sec)**

**RETRIEVE THE COUNT OF EMPLOYEE RECORDS WHOSE AGE COLUMN HAVING A VALUE EXCEPT `NULL`?**

**SELECT COUNT(AGE) "AGE COLUMN COUNT" FROM EMPLOYEE;**

**+---------------+**

**| AGE COLUMN COUNT |**

**+---------------+**

**| 10 |**

**+---------------+**

**1 row in set (0.00 sec)**

**RETRIEVE THE COUNT OF EMPLOYEE RECORDS WHOSE LAST\_NAME COLUMN HAVING A VALUE EXCEPT `NULL`?**

**SELECT COUNT(LAST\_NAME) "RECORDS COUNT" FROM EMPLOYEE;**

**+---------------+**

**| RECORDS COUNT |**

**+---------------+**

**| 10 |**

**+---------------+**

**1 row in set (0.01 sec)**

**RETRIEVE MAXIMUM AGE FROM THE EMPLOYEE TABLE?**

**EXAMPLE: 1**

**SELECT MAX(AGE) FROM EMPLOYEE;**

**+----------+**

**| MAX(AGE) |**

**+----------+**

**| 32 |**

**+----------+**

**1 row in set (0.00 sec)**

**EXAMPLE: 2**

**SELECT MAX(AGE) AS "MAX AGE" FROM EMPLOYEE;**

**+---------+**

**| MAX AGE |**

**+---------+**

**| 32 |**

**+---------+**

**1 row in set (0.00 sec)**

**RETRIEVE MINIMUM SALARY FROM THE EMPLOYEE TABLE?**

**EXAMPLE: 1**

**SELECT MIN(SALARY) FROM EMPLOYEE;**

**+-------------+**

**| MIN(SALARY) |**

**+-------------+**

**| 20000 |**

**+-------------+**

**1 row in set (0.00 sec)**

**EXAMPLE: 2**

**SELECT MIN(SALARY) MIN\_SAL FROM EMPLOYEE;**

**+---------+**

**| MIN\_SAL |**

**+---------+**

**| 20000 |**

**+---------+**

**1 row in set (0.00 sec)**

**EXAMPLE: 3**

**SELECT MIN(SALARY) "MIN SAL" FROM EMPLOYEE;**

**+---------+**

**| MIN SAL |**

**+---------+**

**| 20000 |**

**+---------+**

**1 row in set (0.00 sec)**

**RETRIEVE AVERAGE SALARY FROM THE EMPLOYEE TABLE?**

**EXAMPLE: 1**

**SELECT AVG(SALARY) FROM EMPLOYEE;**

**+-------------+**

**| AVG(SALARY) |**

**+-------------+**

**| 46900.0000 |**

**+-------------+**

**1 row in set (0.00 sec)**

**EXAMPLE: 2**

**SELECT AVG(SALARY) "AVG SALARY" FROM EMPLOYEE;**

**+------------+**

**| AVG SALARY |**

**+------------+**

**| 46900.0000 |**

**+------------+**

**1 row in set (0.00 sec)**

**EXAMPLE: 3**

**SELECT AVG(AGE) "AVG AGE" FROM EMPLOYEE;**

**+---------+**

**| AVG AGE |**

**+---------+**

**| 26.0000 |**

**+---------+**

**1 row in set (0.00 sec)**

**RETRIEVE THE MINIMUM ASCII VALUE AMONG THE VALUES OF FIRST\_NAME COLUMN FROM THE EMPLOYEE TABLE?**

**SELECT MIN(FIRST\_NAME) FROM EMPLOYEE;**

**+-----------------+**

**| MIN(FIRST\_NAME) |**

**+-----------------+**

**| ARUN |**

**+-----------------+**

**1 row in set (0.00 sec)**

**RETRIEVE THE MAX ASCII VALUE AMONG THE VALUES OF FIRST\_NAME COLUMN FROM THE EMPLOYEE TABLE?**

**SELECT MAX(FIRST\_NAME) FROM EMPLOYEE;**

**+-----------------+**

**| MAX(FIRST\_NAME) |**

**+-----------------+**

**| JANU |**

**+-----------------+**

**1 row in set (0.00 sec)**

**DEMONSTRATE ORDER BY**

**SELECT \* FROM EMPLOYEE ORDER BY FIRST\_NAME;**

- ORDER BY in MySQL is like telling the database how you want your results to be arranged or sorted when you retrieve them from a table.

- It is commonly used in conjunction with the SELECT statement.

- default sorting is ascending order.

**ORDER EMPLOYEE RECORDS USING ORDER BY IN THE ASCENDING ORDER BY CONSIDERING FIRST\_NAME COLUMN?**

**SELECT \* FROM EMPLOYEE ORDER BY FIRST\_NAME;**

**+------+------------+------------+------+--------+-----------------------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |**

**+------+------------+------------+------+--------+-----------------------+**

**| 1 | ARUN | PATEL | 22 | 40000 | ARUN@GCOMPANY.IN |**

**| 2 | BHAVESH | SHARMA | 24 | 30000 | BHAVESH@GCOMPANY.IN |**

**| 3 | CHAITANYA | SINGH | 23 | 50000 | CHAITANYA@GCOMPANY.IN |**

**| 4 | DEEPIKA | GUPTA | 26 | 55000 | DEEPIKA@GCOMPANY.IN |**

**| 5 | DHANUSH | KUMAR | 25 | 20000 | DHANUSH@GCOMPANY.IN |**

**| 6 | EKTA | YADAV | 28 | 35000 | YADAV@GCOMPANY.IN |**

**| 7 | GAURAV | RAO | 21 | 60000 | GAURAV@GCOMPANY.IN |**

**| 8 | HARSHITA | REDDY | 29 | 56000 | HARSHITA@GCOMPANY.IN |**

**| 9 | ISHAAN | REDDY | 32 | 70000 | ISHAAN@GCOMPANY.IN |**

**| 10 | JANU | MUKHERJEE | 30 | 53000 | JANU@GCOMPANY.IN |**

**+------+------------+------------+------+--------+-----------------------+**

**10 rows in set (0.00 sec)**

**ORDER EMPLOYEE RECORDS USING ORDER BY IN THE ASCENDING ORDER BY CONSIDERING FIRST\_NAME COLUMN USING AS KEYWORD?**

**SELECT \* FROM EMPLOYEE ORDER BY FIRST\_NAME ASC;**

**+------+------------+------------+------+--------+-----------------------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |**

**+------+------------+------------+------+--------+-----------------------+**

**| 1 | ARUN | PATEL | 22 | 40000 | ARUN@GCOMPANY.IN |**

**| 2 | BHAVESH | SHARMA | 24 | 30000 | BHAVESH@GCOMPANY.IN |**

**| 3 | CHAITANYA | SINGH | 23 | 50000 | CHAITANYA@GCOMPANY.IN |**

**| 4 | DEEPIKA | GUPTA | 26 | 55000 | DEEPIKA@GCOMPANY.IN |**

**| 5 | DHANUSH | KUMAR | 25 | 20000 | DHANUSH@GCOMPANY.IN |**

**| 6 | EKTA | YADAV | 28 | 35000 | YADAV@GCOMPANY.IN |**

**| 7 | GAURAV | RAO | 21 | 60000 | GAURAV@GCOMPANY.IN |**

**| 8 | HARSHITA | REDDY | 29 | 56000 | HARSHITA@GCOMPANY.IN |**

**| 9 | ISHAAN | REDDY | 32 | 70000 | ISHAAN@GCOMPANY.IN |**

**| 10 | JANU | MUKHERJEE | 30 | 53000 | JANU@GCOMPANY.IN |**

**+------+------------+------------+------+--------+-----------------------+**

**10 rows in set (0.00 sec)**

**ORDER EMPLOYEE RECORDS USING ORDER BY IN THE DESCENDING ORDER BY CONSIDERING FIRST\_NAME COLUMN?**

**SELECT \* FROM EMPLOYEE ORDER BY FIRST\_NAME DESC;**

**+------+------------+------------+------+--------+-----------------------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |**

**+------+------------+------------+------+--------+-----------------------+**

**| 10 | JANU | MUKHERJEE | 30 | 53000 | JANU@GCOMPANY.IN |**

**| 9 | ISHAAN | REDDY | 32 | 70000 | ISHAAN@GCOMPANY.IN |**

**| 8 | HARSHITA | REDDY | 29 | 56000 | HARSHITA@GCOMPANY.IN |**

**| 7 | GAURAV | RAO | 21 | 60000 | GAURAV@GCOMPANY.IN |**

**| 6 | EKTA | YADAV | 28 | 35000 | YADAV@GCOMPANY.IN |**

**| 5 | DHANUSH | KUMAR | 25 | 20000 | DHANUSH@GCOMPANY.IN |**

**| 4 | DEEPIKA | GUPTA | 26 | 55000 | DEEPIKA@GCOMPANY.IN |**

**| 3 | CHAITANYA | SINGH | 23 | 50000 | CHAITANYA@GCOMPANY.IN |**

**| 2 | BHAVESH | SHARMA | 24 | 30000 | BHAVESH@GCOMPANY.IN |**

**| 1 | ARUN | PATEL | 22 | 40000 | ARUN@GCOMPANY.IN |**

**+------+------------+------------+------+--------+-----------------------+**

**10 rows in set (0.00 sec)**

**ORDER EMPLOYEE RECORDS USING ORDER BY IN THE ASCENDING ORDER BY CONSIDERING AGE COLUMN?**

**SELECT \* FROM EMPLOYEE ORDER BY AGE;**

**+------+------------+------------+------+--------+-----------------------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |**

**+------+------------+------------+------+--------+-----------------------+**

**| 7 | GAURAV | RAO | 21 | 60000 | GAURAV@GCOMPANY.IN |**

**| 1 | ARUN | PATEL | 22 | 40000 | ARUN@GCOMPANY.IN |**

**| 3 | CHAITANYA | SINGH | 23 | 50000 | CHAITANYA@GCOMPANY.IN |**

**| 2 | BHAVESH | SHARMA | 24 | 30000 | BHAVESH@GCOMPANY.IN |**

**| 5 | DHANUSH | KUMAR | 25 | 20000 | DHANUSH@GCOMPANY.IN |**

**| 4 | DEEPIKA | GUPTA | 26 | 55000 | DEEPIKA@GCOMPANY.IN |**

**| 6 | EKTA | YADAV | 28 | 35000 | YADAV@GCOMPANY.IN |**

**| 8 | HARSHITA | REDDY | 29 | 56000 | HARSHITA@GCOMPANY.IN |**

**| 10 | JANU | MUKHERJEE | 30 | 53000 | JANU@GCOMPANY.IN |**

**| 9 | ISHAAN | REDDY | 32 | 70000 | ISHAAN@GCOMPANY.IN |**

**+------+------------+------------+------+--------+-----------------------+**

**10 rows in set (0.00 sec)**

**ORDER EMPLOYEE RECORDS USING ORDER BY IN THE ASCENDING ORDER BY CONSIDERING SALARY COLUMN?**

**SELECT \* FROM EMPLOYEE ORDER BY SALARY;**

**+------+------------+------------+------+--------+-----------------------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |**

**+------+------------+------------+------+--------+-----------------------+**

**| 5 | DHANUSH | KUMAR | 25 | 20000 | DHANUSH@GCOMPANY.IN |**

**| 2 | BHAVESH | SHARMA | 24 | 30000 | BHAVESH@GCOMPANY.IN |**

**| 6 | EKTA | YADAV | 28 | 35000 | YADAV@GCOMPANY.IN |**

**| 1 | ARUN | PATEL | 22 | 40000 | ARUN@GCOMPANY.IN |**

**| 3 | CHAITANYA | SINGH | 23 | 50000 | CHAITANYA@GCOMPANY.IN |**

**| 10 | JANU | MUKHERJEE | 30 | 53000 | JANU@GCOMPANY.IN |**

**| 4 | DEEPIKA | GUPTA | 26 | 55000 | DEEPIKA@GCOMPANY.IN |**

**| 8 | HARSHITA | REDDY | 29 | 56000 | HARSHITA@GCOMPANY.IN |**

**| 7 | GAURAV | RAO | 21 | 60000 | GAURAV@GCOMPANY.IN |**

**| 9 | ISHAAN | REDDY | 32 | 70000 | ISHAAN@GCOMPANY.IN |**

**+------+------------+------------+------+--------+-----------------------+**

**10 rows in set (0.00 sec)**

**ORDER EMPLOYEE RECORDS USING ORDER BY IN THE ASCENDING ORDER BY CONSIDERING AGE AND SALARY COLUMNS?**

**Note:** The first preference would be for the first column.If there are rows with the same value in column1, those rows will then be sorted by the values in column2 in ascending order. The default sorting is ascending order.

**SELECT \* FROM EMPLOYEE ORDER BY AGE, SALARY;**

**+------+------------+------------+------+--------+-----------------------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |**

**+------+------------+------------+------+--------+-----------------------+**

**| 7 | GAURAV | RAO | 21 | 60000 | GAURAV@GCOMPANY.IN |**

**| 1 | ARUN | PATEL | 22 | 40000 | ARUN@GCOMPANY.IN |**

**| 3 | CHAITANYA | SINGH | 23 | 50000 | CHAITANYA@GCOMPANY.IN |**

**| 2 | BHAVESH | SHARMA | 24 | 30000 | BHAVESH@GCOMPANY.IN |**

**| 5 | DHANUSH | KUMAR | 25 | 20000 | DHANUSH@GCOMPANY.IN |**

**| 4 | DEEPIKA | GUPTA | 26 | 55000 | DEEPIKA@GCOMPANY.IN |**

**| 6 | EKTA | YADAV | 28 | 35000 | YADAV@GCOMPANY.IN |**

**| 8 | HARSHITA | REDDY | 29 | 56000 | HARSHITA@GCOMPANY.IN |**

**| 10 | JANU | MUKHERJEE | 30 | 53000 | JANU@GCOMPANY.IN |**

**| 9 | ISHAAN | REDDY | 32 | 70000 | ISHAAN@GCOMPANY.IN |**

**| 11 | ARUL | PATEL | 35 | 40000 | ARUL@GCOMPANY.IN |**

**| 12 | ADITI | PATEL | 35 | 60000 | ADITI@GCOMPANY.IN |**

**+------+------------+------------+------+--------+-----------------------+**

**12 rows in set (0.00 sec)**

**INSERT INTO EMPLOYEE VALUES(11, 'ARUL', 'PATEL', 35, 40000, 'ARUL@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(12, 'ADITI', 'PATEL', 35, 60000, 'ADITI@GCOMPANY.IN');**

**ORDER EMPLOYEE RECORDS USING ORDER BY AND AGE IN ASCENDING ORDER AND SALARY IN DESC ORDER?**

**SELECT \* FROM EMPLOYEE ORDER BY AGE ASC, SALARY DESC;**

**Note: First considers the first column and sorts in the specified order, if two values of the first column are the same then it considers the second column and sorts in the specified order.**

**+------+------------+------------+------+--------+-----------------------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |**

**+------+------------+------------+------+--------+-----------------------+**

**| 7 | GAURAV | RAO | 21 | 60000 | GAURAV@GCOMPANY.IN |**

**| 1 | ARUN | PATEL | 22 | 40000 | ARUN@GCOMPANY.IN |**

**| 3 | CHAITANYA | SINGH | 23 | 50000 | CHAITANYA@GCOMPANY.IN |**

**| 2 | BHAVESH | SHARMA | 24 | 30000 | BHAVESH@GCOMPANY.IN |**

**| 5 | DHANUSH | KUMAR | 25 | 20000 | DHANUSH@GCOMPANY.IN |**

**| 4 | DEEPIKA | GUPTA | 26 | 55000 | DEEPIKA@GCOMPANY.IN |**

**| 6 | EKTA | YADAV | 28 | 35000 | YADAV@GCOMPANY.IN |**

**| 8 | HARSHITA | REDDY | 29 | 56000 | HARSHITA@GCOMPANY.IN |**

**| 10 | JANU | MUKHERJEE | 30 | 53000 | JANU@GCOMPANY.IN |**

**| 9 | ISHAAN | REDDY | 32 | 70000 | ISHAAN@GCOMPANY.IN |**

**| 12 | ADITI | PATEL | 35 | 60000 | ADITI@GCOMPANY.IN |**

**| 11 | ARUL | PATEL | 35 | 40000 | ARUL@GCOMPANY.IN |**

**+------+------------+------------+------+--------+-----------------------+**

**12 rows in set (0.00 sec)**

**WHAT IF TWO VALUES OF THE SAME COLUMN ARE SAME?**

**INSERT INTO EMPLOYEE VALUES(13, 'ARTI', ' PATEL', 35, 10000, ' ARTI@GCOMPANY.IN');**

**SELECT \* FROM EMPLOYEE ORDER BY AGE, SALARY;**

**Note: Then it considers the second colum minimum value**

**+------+------------+-----------+------+--------+-----------------------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |**

**+------+------------+-----------+------+--------+-----------------------+**

**| 7 | GAURAV | RAO | 21 | 60000 | GAURAV@GCOMPANY.IN |**

**| 1 | ARUN | PATEL | 22 | 40000 | ARUN@GCOMPANY.IN |**

**| 3 | CHAITANYA | SINGH | 23 | 50000 | CHAITANYA@GCOMPANY.IN |**

**| 2 | BHAVESH | SHARMA | 24 | 30000 | BHAVESH@GCOMPANY.IN |**

**| 5 | DHANUSH | KUMAR | 25 | 20000 | DHANUSH@GCOMPANY.IN |**

**| 4 | DEEPIKA | GUPTA | 26 | 55000 | DEEPIKA@GCOMPANY.IN |**

**| 6 | EKTA | YADAV | 28 | 35000 | YADAV@GCOMPANY.IN |**

**| 8 | HARSHITA | REDDY | 29 | 56000 | HARSHITA@GCOMPANY.IN |**

**| 10 | JANU | MUKHERJEE | 30 | 53000 | JANU@GCOMPANY.IN |**

**| 9 | ISHAAN | REDDY | 32 | 70000 | ISHAAN@GCOMPANY.IN |**

**| 13 | ARTI | PATEL | 35 | 10000 | ARTI@GCOMPANY.IN |**

**| 11 | ARUL | PATEL | 35 | 40000 | ARUL@GCOMPANY.IN |**

**| 12 | ADITI | PATEL | 35 | 60000 | ADITI@GCOMPANY.IN |**

**+------+------------+-----------+------+--------+-----------------------+**

**Note: We get error for the below query because MAX(SALARY) is not a column in the employee table.**

//SELECT FIRST\_NAME, MAX(SALAY) FROM EMPLOYEE;//ERROR

**WHICH EMPLOYEE HAS THE HIGHEST SALARY, AND WHAT IS THEIR FIRST NAME?**

**SELECT FIRST\_NAME FROM EMPLOYEE WHERE SALARY = (SELECT MAX(SALARY) FROM EMPLOYEE);**

**+------------+**

**| FIRST\_NAME |**

**+------------+**

**| ISHAAN |**

**+------------+**

**1 row in set (0.00 sec)**

**WHICH EMPLOYEE HAS THE HIGHEST AGE, AND WHAT IS THEIR FIRST NAME?**

**SELECT FIRST\_NAME FROM EMPLOYEE WHERE AGE = (SELECT MAX(AGE) FROM EMPLOYEE);**

**+------------+**

**| FIRST\_NAME |**

**+------------+**

**| ARUL |**

**| ADITI |**

**+------------+**

**WHICH EMPLOYEE HAS THE LOWEST AGE, AND WHAT IS THEIR FIRST NAME?**

**SELECT FIRST\_NAME FROM EMPLOYEE WHERE AGE = (SELECT MIN(AGE) FROM EMPLOYEE);**

**+------------+**

**| FIRST\_NAME |**

**+------------+**

**| GAURAV |**

**+------------+**

**1 row in set (0.00 sec)**

**WHICH EMPLOYEE HAS LESS THAN AVERAGE SALARY, AND WHAT IS THEIR FIRST NAME?**

**SELECT FIRST\_NAME FROM EMPLOYEE WHERE SALARY < (SELECT AVG(SALARY) FROM EMPLOYEE);**

**+------------+**

**| FIRST\_NAME |**

**+------------+**

**| ARUN |**

**| BHAVESH |**

**| DHANUSH |**

**| EKTA |**

**| ARUL |**

**+------------+**

**5 rows in set (0.00 sec)**

**WHAT IS THE MAXIMUM SALARY FROM THE EMPLOYEE TABLE?**

**SELECT MAX(SALARY) FROM EMPLOYEE;**

**+-------------+**

**| MAX(SALARY) |**

**+-------------+**

**| 70000 |**

**+-------------+**

**1 row in set (0.00 sec)**

**WHAT IS THE SECOND MAXIMUM SALARY IN THE EMPLOYEE TABLE?**

**SELECT MAX(SALARY) FROM EMPLOYEE WHERE SALARY < (SELECT MAX(SALARY) FROM EMPLOYEE);**

**+-------------+**

**| MAX(SALARY) |**

**+-------------+**

**| 60000 |**

**+-------------+**

**1 row in set (0.00 sec)**

**WHAT IS THE SECOND MAXIMUM SALARY IN THE EMPLOYEE TABLE?**

**WHAT IS THE SECOND MINIMUM SALARY IN THE EMPLOYEE TABLE?**

**SELECT MIN(SALARY) FROM EMPLOYEE WHERE SALARY > (SELECT MIN(SALARY) FROM EMPLOYEE);**

**+-------------+**

**| MIN(SALARY) |**

**+-------------+**

**| 30000 |**

**+-------------+**

**1 row in set (0.00 sec)**

**WHICH EMPLOYEE HAS SECOND MINIMUM SALARY, AND WHAT IS THEIR FIRST NAME?**

**SELECT FIRST\_NAME FROM EMPLOYEE WHERE SALARY = (SELECT MIN(SALARY) FROM EMPLOYEE WHERE SALARY > (SELECT MIN(SALARY) FROM EMPLOYEE));**

**+------------+**

**| FIRST\_NAME |**

**+------------+**

**| BHAVESH |**

**+------------+**

**1 row in set (0.00 sec)**

**WHICH EMPLOYEE HAS SECOND MAXIMUM SALARY, AND WHAT IS THEIR FIRST NAME?**

**SELECT FIRST\_NAME FROM EMPLOYEE WHERE SALARY = (SELECT MAX(SALARY) FROM EMPLOYEE WHERE SALARY < (SELECT MAX(SALARY) FROM EMPLOYEE));**

**+------------+**

**| FIRST\_NAME |**

**+------------+**

**| GAURAV |**

**| ADITI |**

**+------------+**

**2 rows in set (0.00 sec)**

**WHICH EMPLOYEE HAS SECOND MINIMUM SALARY, FETCH THEIR COMPLETE DETAILS?**

**SELECT \* FROM EMPLOYEE WHERE SALARY = (SELECT MIN(SALARY) FROM EMPLOYEE WHERE SALARY > (SELECT MIN(SALARY) FROM EMPLOYEE));**

**+------+------------+-----------+------+--------+---------------------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |**

**+------+------------+-----------+------+--------+---------------------+**

**| 2 | BHAVESH | SHARMA | 24 | 30000 | BHAVESH@GCOMPANY.IN |**

**+------+------------+-----------+------+--------+---------------------+**

**1 row in set (0.00 sec)**

**WHAT IS THE EMPLOYEE ID, FIRST NAME, LAST NAME, AGE, SALARY, EMAIL, AND RANK BASED ON SALARY IN DESCENDING ORDER FOR ALL EMPLOYEES?**

**Note:** the RANK() function is a window function that assigns a rank to each row within a partition of a result set. It is commonly used to assign a rank to rows based on the values in one or more columns.

* The RANK() function assigns a rank to each row based on its position in the ordered list within each partition.
* Ties are handled by assigning the same rank to all tied rows.

**SELECT ID, FIRST\_NAME, LAST\_NAME, AGE, SALARY, EMAIL, RANK() OVER (ORDER BY SALARY DESC) FROM EMPLOYEE;**

**+------+------------+------------+------+--------+-----------------------+------------------------------------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL | RANK() OVER (ORDER BY SALARY DESC) |**

**+------+------------+------------+------+--------+-----------------------+------------------------------------+**

**| 9 | ISHAAN | REDDY | 32 | 70000 | ISHAAN@GCOMPANY.IN | 1 |**

**| 7 | GAURAV | RAO | 21 | 60000 | GAURAV@GCOMPANY.IN | 2 |**

**| 12 | ADITI | PATEL | 35 | 60000 | ADITI@GCOMPANY.IN | 2 |**

**| 8 | HARSHITA | REDDY | 29 | 56000 | HARSHITA@GCOMPANY.IN | 4 |**

**| 4 | DEEPIKA | GUPTA | 26 | 55000 | DEEPIKA@GCOMPANY.IN | 5 |**

**| 10 | JANU | MUKHERJEE | 30 | 53000 | JANU@GCOMPANY.IN | 6 |**

**| 3 | CHAITANYA | SINGH | 23 | 50000 | CHAITANYA@GCOMPANY.IN | 7 |**

**| 1 | ARUN | PATEL | 22 | 40000 | ARUN@GCOMPANY.IN | 8 |**

**| 11 | ARUL | PATEL | 35 | 40000 | ARUL@GCOMPANY.IN | 8 |**

**| 6 | EKTA | YADAV | 28 | 35000 | YADAV@GCOMPANY.IN | 10 |**

**| 2 | BHAVESH | SHARMA | 24 | 30000 | BHAVESH@GCOMPANY.IN | 11 |**

**| 5 | DHANUSH | KUMAR | 25 | 20000 | DHANUSH@GCOMPANY.IN | 12 |**

**+------+------------+------------+------+--------+-----------------------+------------------------------------+**

**12 rows in set (0.01 sec)**

**RETRIEVE THE DETAILS OF THE EMPLOYEE WITH THE SECOND-HIGHEST SALARY, INCLUDING ID, FIRST NAME, LAST NAME, AGE, SALARY, AND EMAIL?**

**SELECT \* FROM (SELECT ID, FIRST\_NAME, LAST\_NAME, AGE, SALARY, EMAIL, RANK() OVER(ORDER BY SALARY DESC) RANKED\_EMPLOYEES FROM EMPLOYEE) AS RANKED\_EMPLOYEES WHERE RANKED\_EMPLOYEES = 2;**

**+------+------------+-----------+------+--------+--------------------+------------------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL | RANKED\_EMPLOYEES |**

**+------+------------+-----------+------+--------+--------------------+------------------+**

**| 7 | GAURAV | RAO | 21 | 60000 | GAURAV@GCOMPANY.IN | 2 |**

**| 12 | ADITI | PATEL | 35 | 60000 | ADITI@GCOMPANY.IN | 2 |**

**+------+------------+-----------+------+--------+--------------------+------------------+**

**2 rows in set (0.00 sec)**

**SELECT \* FROM EMPLOYEE;**

**+------+------------+------------+------+--------+-----------------------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |**

**+------+------------+------------+------+--------+-----------------------+**

**| 1 | ARUN | PATEL | 22 | 40000 | ARUN@GCOMPANY.IN |**

**| 2 | BHAVESH | SHARMA | 24 | 30000 | BHAVESH@GCOMPANY.IN |**

**| 3 | CHAITANYA | SINGH | 23 | 50000 | CHAITANYA@GCOMPANY.IN |**

**| 4 | DEEPIKA | GUPTA | 26 | 55000 | DEEPIKA@GCOMPANY.IN |**

**| 5 | DHANUSH | KUMAR | 25 | 20000 | DHANUSH@GCOMPANY.IN |**

**| 6 | EKTA | YADAV | 28 | 35000 | YADAV@GCOMPANY.IN |**

**| 7 | GAURAV | RAO | 21 | 60000 | GAURAV@GCOMPANY.IN |**

**| 8 | HARSHITA | REDDY | 29 | 56000 | HARSHITA@GCOMPANY.IN |**

**| 9 | ISHAAN | REDDY | 32 | 70000 | ISHAAN@GCOMPANY.IN |**

**| 10 | JANU | MUKHERJEE | 30 | 53000 | JANU@GCOMPANY.IN |**

**| 11 | ARUL | PATEL | 35 | 40000 | ARUL@GCOMPANY.IN |**

**| 12 | ADITI | PATEL | 35 | 60000 | ADITI@GCOMPANY.IN |**

**+------+------------+------------+------+--------+-----------------------+**

**12 rows in set (0.00 sec)**

**DELETE FROM EMPLOYEE;**

**Query OK, 12 rows affected (0.01 sec)**

**INSERT INTO EMPLOYEE VALUES(1, 'ARUN', 'PATEL', 22, 40000, ' ARUN@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(2, 'BHAVESH', 'SHARMA', 24, 30000, 'BHAVESH@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(3, 'CHAITANYA', 'SINGH', 23, 50000, 'CHAITANYA@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(4, 'DEEPIKA', 'GUPTA', 26, 55000, 'DEEPIKA@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(5, 'DHANUSH', 'KUMAR', 25, 20000, 'DHANUSH@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(6, 'EKTA', 'YADAV', 28, 35000, 'YADAV@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(7, 'GAURAV', 'RAO', 21, 60000, 'GAURAV@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(8, 'HARSHITA', 'REDDY', 29, 56000, 'HARSHITA@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(9, 'ISHAAN', 'REDDY', 32, 70000, 'ISHAAN@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(10, 'JANU', 'MUKHERJEE', 30, 53000, 'JANU@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(1, 'ARUN', 'PATEL', 22, 40000, ' ARUN@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(2, 'BHAVESH', 'SHARMA', 24, 30000, 'BHAVESH@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(3, 'CHAITANYA', 'SINGH', 23, 50000, 'CHAITANYA@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(4, 'DEEPIKA', 'GUPTA', 26, 55000, 'DEEPIKA@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(5, 'DHANUSH', 'KUMAR', 25, 20000, 'DHANUSH@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(6, 'EKTA', 'YADAV', 28, 35000, 'YADAV@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(7, 'GAURAV', 'RAO', 21, 60000, 'GAURAV@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(8, 'HARSHITA', 'REDDY', 29, 56000, 'HARSHITA@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(9, 'ISHAAN', 'REDDY', 32, 70000, 'ISHAAN@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(10, 'JANU', 'MUKHERJEE', 30, 53000, 'JANU@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(1, 'ARUN', 'PATEL', 22, 40000, 'ARUN@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(2, 'BHAVESH', 'SHARMA', 24, 30000, 'BHAVESH@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(3, 'CHAITANYA', 'SINGH', 23, 50000, 'CHAITANYA@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(4, 'DEEPIKA', 'GUPTA', 26, 55000, 'DEEPIKA@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(5, 'DHANUSH', 'KUMAR', 25, 20000, 'DHANUSH@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(6, 'EKTA', 'YADAV', 28, 35000, 'YADAV@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(7, 'GAURAV', 'RAO', 21, 60000, 'GAURAV@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(8, 'HARSHITA', 'REDDY', 29, 56000, 'HARSHITA@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(9, 'ISHAAN', 'REDDY', 32, 70000, 'ISHAAN@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(10, 'JANU', 'MUKHERJEE', 30, 53000, 'JANU@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(1, 'ARUN', 'PATEL', 22, 40000, 'ARUN@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(2, 'BHAVESH', ' SHARMA', 24, 30000, 'BHAVESH@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(3, 'CHAITANYA', 'SINGH', 23, 50000, 'CHAITANYA@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(4, 'DEEPIKA', 'GUPTA', 26, 55000, 'DEEPIKA@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(5, 'DHANUSH', 'KUMAR', 25, 20000, 'DHANUSH@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(6, 'EKTA', 'YADAV', 28, 35000, 'YADAV@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(7, 'GAURAV', 'RAO', 21, 60000, 'GAURAV@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(8, 'HARSHITA', 'REDDY', 29, 56000, 'HARSHITA@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(9, 'ISHAAN', 'REDDY', 32, 70000, 'ISHAAN@GCOMPANY.IN');**

**INSERT INTO EMPLOYEE VALUES(10, 'JANU', 'MUKHERJEE', 30, 53000, 'JANU@GCOMPANY.IN');**

**SELECT COUNT(\*) FROM EMPLOYEE;**

**+----------+**

**| COUNT(\*) |**

**+----------+**

**| 40 |**

**+----------+**

**1 row in set (0.01 sec)**

**SELECT \* FROM EMPLOYEE;**

**SET @row\_number = 0;**

**SELECT ID, FIRST\_NAME, LAST\_NAME, AGE, EMAIL, SALARY,**

**(@row\_number:=@row\_number + 1) AS ROWNUM**

**FROM EMPLOYEE;**

**SET @row\_number = 0; initializes a user-defined variable @row\_number and sets it to 0.**

**(@row\_number:=@row\_number + 1) AS ROWNUM increments the @row\_number variable for each row, effectively assigning a row number to each result.**

**SELECT ID, FIRST\_NAME, LAST\_NAME, AGE, EMAIL, SALARY,(@row\_number:=@row\_number + 1) AS ROWNUM FROM EMPLOYEE;**

**+------+------------+------------+------+-----------------------+--------+--------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | EMAIL | SALARY | ROWNUM |**

**+------+------------+------------+------+-----------------------+--------+--------+**

**| 1 | ARUN | PATEL | 22 | ARUN@GCOMPANY.IN | 40000 | 1 |**

**| 2 | BHAVESH | SHARMA | 24 | BHAVESH@GCOMPANY.IN | 30000 | 2 |**

**| 3 | CHAITANYA | SINGH | 23 | CHAITANYA@GCOMPANY.IN | 50000 | 3 |**

**| 4 | DEEPIKA | GUPTA | 26 | DEEPIKA@GCOMPANY.IN | 55000 | 4 |**

**| 5 | DHANUSH | KUMAR | 25 | DHANUSH@GCOMPANY.IN | 20000 | 5 |**

**| 6 | EKTA | YADAV | 28 | YADAV@GCOMPANY.IN | 35000 | 6 |**

**| 7 | GAURAV | RAO | 21 | GAURAV@GCOMPANY.IN | 60000 | 7 |**

**| 8 | HARSHITA | REDDY | 29 | HARSHITA@GCOMPANY.IN | 56000 | 8 |**

**| 9 | ISHAAN | REDDY | 32 | ISHAAN@GCOMPANY.IN | 70000 | 9 |**

**| 10 | JANU | MUKHERJEE | 30 | JANU@GCOMPANY.IN | 53000 | 10 |**

**| 1 | ARUN | PATEL | 22 | ARUN@GCOMPANY.IN | 40000 | 11 |**

**| 2 | BHAVESH | SHARMA | 24 | BHAVESH@GCOMPANY.IN | 30000 | 12 |**

**| 3 | CHAITANYA | SINGH | 23 | CHAITANYA@GCOMPANY.IN | 50000 | 13 |**

**| 4 | DEEPIKA | GUPTA | 26 | DEEPIKA@GCOMPANY.IN | 55000 | 14 |**

**| 5 | DHANUSH | KUMAR | 25 | DHANUSH@GCOMPANY.IN | 20000 | 15 |**

**| 6 | EKTA | YADAV | 28 | YADAV@GCOMPANY.IN | 35000 | 16 |**

**| 7 | GAURAV | RAO | 21 | GAURAV@GCOMPANY.IN | 60000 | 17 |**

**| 8 | HARSHITA | REDDY | 29 | HARSHITA@GCOMPANY.IN | 56000 | 18 |**

**| 9 | ISHAAN | REDDY | 32 | ISHAAN@GCOMPANY.IN | 70000 | 19 |**

**| 10 | JANU | MUKHERJEE | 30 | JANU@GCOMPANY.IN | 53000 | 20 |**

**| 1 | ARUN | PATEL | 22 | ARUN@GCOMPANY.IN | 40000 | 21 |**

**| 2 | BHAVESH | SHARMA | 24 | BHAVESH@GCOMPANY.IN | 30000 | 22 |**

**| 3 | CHAITANYA | SINGH | 23 | CHAITANYA@GCOMPANY.IN | 50000 | 23 |**

**| 4 | DEEPIKA | GUPTA | 26 | DEEPIKA@GCOMPANY.IN | 55000 | 24 |**

**| 5 | DHANUSH | KUMAR | 25 | DHANUSH@GCOMPANY.IN | 20000 | 25 |**

**| 6 | EKTA | YADAV | 28 | YADAV@GCOMPANY.IN | 35000 | 26 |**

**| 7 | GAURAV | RAO | 21 | GAURAV@GCOMPANY.IN | 60000 | 27 |**

**| 8 | HARSHITA | REDDY | 29 | HARSHITA@GCOMPANY.IN | 56000 | 28 |**

**| 9 | ISHAAN | REDDY | 32 | ISHAAN@GCOMPANY.IN | 70000 | 29 |**

**| 10 | JANU | MUKHERJEE | 30 | JANU@GCOMPANY.IN | 53000 | 30 |**

**| 1 | ARUN | PATEL | 22 | ARUN@GCOMPANY.IN | 40000 | 31 |**

**| 2 | BHAVESH | SHARMA | 24 | BHAVESH@GCOMPANY.IN | 30000 | 32 |**

**| 3 | CHAITANYA | SINGH | 23 | CHAITANYA@GCOMPANY.IN | 50000 | 33 |**

**| 4 | DEEPIKA | GUPTA | 26 | DEEPIKA@GCOMPANY.IN | 55000 | 34 |**

**| 5 | DHANUSH | KUMAR | 25 | DHANUSH@GCOMPANY.IN | 20000 | 35 |**

**| 6 | EKTA | YADAV | 28 | YADAV@GCOMPANY.IN | 35000 | 36 |**

**| 7 | GAURAV | RAO | 21 | GAURAV@GCOMPANY.IN | 60000 | 37 |**

**| 8 | HARSHITA | REDDY | 29 | HARSHITA@GCOMPANY.IN | 56000 | 38 |**

**| 9 | ISHAAN | REDDY | 32 | ISHAAN@GCOMPANY.IN | 70000 | 39 |**

**| 10 | JANU | MUKHERJEE | 30 | JANU@GCOMPANY.IN | 53000 | 40 |**

**+------+------------+------------+------+-----------------------+--------+--------+**

**40 rows in set, 1 warning (0.00 sec)**

**PAGINATION IN MYSQL**

Even if we have so many records in a table what if I want to display a particular number of records. In such case we can use pagination concept. In case of Oracle We have rownum but in mysql we don’t have that.

**SET @row\_number = 0;**

**Query OK, 0 rows affected (0.00 sec)**

**SELECT ID, FIRST\_NAME, LAST\_NAME, AGE, EMAIL, SALARY,(@row\_number:=@row\_number + 1) AS RN FROM EMPLOYEE WHERE (@row\_number:=@row\_number + 1) <= 10;**

**+------+------------+-----------+------+-----------------------+--------+------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | EMAIL | SALARY | RN |**

**+------+------------+-----------+------+-----------------------+--------+------+**

**| 1 | ARUN | PATEL | 22 | ARUN@GCOMPANY.IN | 40000 | 2 |**

**| 2 | BHAVESH | SHARMA | 24 | BHAVESH@GCOMPANY.IN | 30000 | 4 |**

**| 3 | CHAITANYA | SINGH | 23 | CHAITANYA@GCOMPANY.IN | 50000 | 6 |**

**| 4 | DEEPIKA | GUPTA | 26 | DEEPIKA@GCOMPANY.IN | 55000 | 8 |**

**| 5 | DHANUSH | KUMAR | 25 | DHANUSH@GCOMPANY.IN | 20000 | 10 |**

**+------+------------+-----------+------+-----------------------+--------+------+**

**5 rows in set, 2 warnings (0.00 sec)**

**LIMIT:**

LIMIT is used to restrict the number of rows returned by a query.

It takes one or two arguments: LIMIT x or LIMIT x, y.

x specifies the maximum number of rows to return.

y (optional) specifies the offset or the number of rows to skip before starting to return rows.

Retrieve the first 5 rows from a table.

SELECT \* FROM EMPLOYEE LIMIT 5;

+------+------------+-----------+------+--------+-----------------------+

| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |

+------+------------+-----------+------+--------+-----------------------+

| 1 | ARUN | PATEL | 22 | 40000 | ARUN@GCOMPANY.IN |

| 2 | BHAVESH | SHARMA | 24 | 30000 | BHAVESH@GCOMPANY.IN |

| 3 | CHAITANYA | SINGH | 23 | 50000 | CHAITANYA@GCOMPANY.IN |

| 4 | DEEPIKA | GUPTA | 26 | 55000 | DEEPIKA@GCOMPANY.IN |

| 5 | DHANUSH | KUMAR | 25 | 20000 | DHANUSH@GCOMPANY.IN |

+------+------------+-----------+------+--------+-----------------------+

5 rows in set (0.00 sec)

Retrieve rows 6 through 10 from a table.

SELECT \* FROM EMPLOYEE LIMIT 5, 5;

+------+------------+------------+------+--------+----------------------+

| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |

+------+------------+------------+------+--------+----------------------+

| 6 | EKTA | YADAV | 28 | 35000 | YADAV@GCOMPANY.IN |

| 7 | GAURAV | RAO | 21 | 60000 | GAURAV@GCOMPANY.IN |

| 8 | HARSHITA | REDDY | 29 | 56000 | HARSHITA@GCOMPANY.IN |

| 9 | ISHAAN | REDDY | 32 | 70000 | ISHAAN@GCOMPANY.IN |

| 10 | JANU | MUKHERJEE | 30 | 53000 | JANU@GCOMPANY.IN |

+------+------------+------------+------+--------+----------------------+

5 rows in set (0.00 sec)

**OFFSET:**

OFFSET is used to skip a specified number of rows before starting to return rows.

It's usually used in combination with LIMIT.

The OFFSET value starts from 0.

Skip the first 3 rows and return the next 5.

SELECT \* FROM EMPLOYEE LIMIT 5 OFFSET 3;

+------+------------+-----------+------+--------+----------------------+

| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |

+------+------------+-----------+------+--------+----------------------+

| 4 | DEEPIKA | GUPTA | 26 | 55000 | DEEPIKA@GCOMPANY.IN |

| 5 | DHANUSH | KUMAR | 25 | 20000 | DHANUSH@GCOMPANY.IN |

| 6 | EKTA | YADAV | 28 | 35000 | YADAV@GCOMPANY.IN |

| 7 | GAURAV | RAO | 21 | 60000 | GAURAV@GCOMPANY.IN |

| 8 | HARSHITA | REDDY | 29 | 56000 | HARSHITA@GCOMPANY.IN |

+------+------------+-----------+------+--------+----------------------+

5 rows in set (0.00 sec)

Alternatively, you can use the shorter form LIMIT x, y where x is the offset and y is the number of rows to return.

Skip the first 2 rows and return the next 8.

SELECT \* FROM EMPLOYEE LIMIT 2, 8;

+------+------------+------------+------+--------+-----------------------+

| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |

+------+------------+------------+------+--------+-----------------------+

| 3 | CHAITANYA | SINGH | 23 | 50000 | CHAITANYA@GCOMPANY.IN |

| 4 | DEEPIKA | GUPTA | 26 | 55000 | DEEPIKA@GCOMPANY.IN |

| 5 | DHANUSH | KUMAR | 25 | 20000 | DHANUSH@GCOMPANY.IN |

| 6 | EKTA | YADAV | 28 | 35000 | YADAV@GCOMPANY.IN |

| 7 | GAURAV | RAO | 21 | 60000 | GAURAV@GCOMPANY.IN |

| 8 | HARSHITA | REDDY | 29 | 56000 | HARSHITA@GCOMPANY.IN |

| 9 | ISHAAN | REDDY | 32 | 70000 | ISHAAN@GCOMPANY.IN |

| 10 | JANU | MUKHERJEE | 30 | 53000 | JANU@GCOMPANY.IN |

+------+------------+------------+------+--------+-----------------------+

8 rows in set (0.00 sec)

**TO FETCH ALTERNATIVE RECORDS FROM A TABLE**

SELECT \* FROM EMPLOYEE WHERE MOD(id, 2) = 0;

+------+------------+------------+------+--------+----------------------+

| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |

+------+------------+------------+------+--------+----------------------+

| 2 | BHAVESH | SHARMA | 24 | 30000 | BHAVESH@GCOMPANY.IN |

| 4 | DEEPIKA | GUPTA | 26 | 55000 | DEEPIKA@GCOMPANY.IN |

| 6 | EKTA | YADAV | 28 | 35000 | YADAV@GCOMPANY.IN |

| 8 | HARSHITA | REDDY | 29 | 56000 | HARSHITA@GCOMPANY.IN |

| 10 | JANU | MUKHERJEE | 30 | 53000 | JANU@GCOMPANY.IN |

| 2 | BHAVESH | SHARMA | 24 | 30000 | BHAVESH@GCOMPANY.IN |

| 4 | DEEPIKA | GUPTA | 26 | 55000 | DEEPIKA@GCOMPANY.IN |

| 6 | EKTA | YADAV | 28 | 35000 | YADAV@GCOMPANY.IN |

| 8 | HARSHITA | REDDY | 29 | 56000 | HARSHITA@GCOMPANY.IN |

| 10 | JANU | MUKHERJEE | 30 | 53000 | JANU@GCOMPANY.IN |

| 2 | BHAVESH | SHARMA | 24 | 30000 | BHAVESH@GCOMPANY.IN |

| 4 | DEEPIKA | GUPTA | 26 | 55000 | DEEPIKA@GCOMPANY.IN |

| 6 | EKTA | YADAV | 28 | 35000 | YADAV@GCOMPANY.IN |

| 8 | HARSHITA | REDDY | 29 | 56000 | HARSHITA@GCOMPANY.IN |

| 10 | JANU | MUKHERJEE | 30 | 53000 | JANU@GCOMPANY.IN |

| 2 | BHAVESH | SHARMA | 24 | 30000 | BHAVESH@GCOMPANY.IN |

| 4 | DEEPIKA | GUPTA | 26 | 55000 | DEEPIKA@GCOMPANY.IN |

| 6 | EKTA | YADAV | 28 | 35000 | YADAV@GCOMPANY.IN |

| 8 | HARSHITA | REDDY | 29 | 56000 | HARSHITA@GCOMPANY.IN |

| 10 | JANU | MUKHERJEE | 30 | 53000 | JANU@GCOMPANY.IN |

+------+------------+------------+------+--------+----------------------+

20 rows in set (0.00 sec)

MOD(id, 2) calculates the remainder when id is divided by 2. This will be 0 for even ids and 1 for odd ids.

WHERE MOD(id, 2) = 0 filters the rows to only include those where the id is even.

This query will retrieve all the rows with even id values from the EMPLOYEE.

**WHAT IF THE TABLE DOESN’T HAVE ID COLUMN**

SET @row\_number := -1;

Query OK, 0 rows affected (0.00 sec)

SELECT \* FROM (SELECT \*,@row\_number := @row\_number + 1 AS row\_num FROM EMPLOYEE) AS numbered\_table WHERE row\_num % 2 = 0;

+------+------------+-----------+------+--------+-----------------------+---------+

| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL | row\_num |

+------+------------+-----------+------+--------+-----------------------+---------+

| 1 | ARUN | PATEL | 22 | 40000 | ARUN@GCOMPANY.IN | 0 |

| 3 | CHAITANYA | SINGH | 23 | 50000 | CHAITANYA@GCOMPANY.IN | 2 |

| 5 | DHANUSH | KUMAR | 25 | 20000 | DHANUSH@GCOMPANY.IN | 4 |

| 7 | GAURAV | RAO | 21 | 60000 | GAURAV@GCOMPANY.IN | 6 |

| 9 | ISHAAN | REDDY | 32 | 70000 | ISHAAN@GCOMPANY.IN | 8 |

| 1 | ARUN | PATEL | 22 | 40000 | ARUN@GCOMPANY.IN | 10 |

| 3 | CHAITANYA | SINGH | 23 | 50000 | CHAITANYA@GCOMPANY.IN | 12 |

| 5 | DHANUSH | KUMAR | 25 | 20000 | DHANUSH@GCOMPANY.IN | 14 |

| 7 | GAURAV | RAO | 21 | 60000 | GAURAV@GCOMPANY.IN | 16 |

| 9 | ISHAAN | REDDY | 32 | 70000 | ISHAAN@GCOMPANY.IN | 18 |

| 1 | ARUN | PATEL | 22 | 40000 | ARUN@GCOMPANY.IN | 20 |

| 3 | CHAITANYA | SINGH | 23 | 50000 | CHAITANYA@GCOMPANY.IN | 22 |

| 5 | DHANUSH | KUMAR | 25 | 20000 | DHANUSH@GCOMPANY.IN | 24 |

| 7 | GAURAV | RAO | 21 | 60000 | GAURAV@GCOMPANY.IN | 26 |

| 9 | ISHAAN | REDDY | 32 | 70000 | ISHAAN@GCOMPANY.IN | 28 |

| 1 | ARUN | PATEL | 22 | 40000 | ARUN@GCOMPANY.IN | 30 |

| 3 | CHAITANYA | SINGH | 23 | 50000 | CHAITANYA@GCOMPANY.IN | 32 |

| 5 | DHANUSH | KUMAR | 25 | 20000 | DHANUSH@GCOMPANY.IN | 34 |

| 7 | GAURAV | RAO | 21 | 60000 | GAURAV@GCOMPANY.IN | 36 |

| 9 | ISHAAN | REDDY | 32 | 70000 | ISHAAN@GCOMPANY.IN | 38 |

+------+------------+-----------+------+--------+-----------------------+---------+

20 rows in set, 1 warning (0.00 sec)

Explanation:

SET @row\_number := -1;: Initializes a user-defined variable @row\_number and sets it to -1. This variable will be used to assign a row number to each record.

SELECT \*, @row\_number := @row\_number + 1 AS row\_num FROM my\_table: This subquery assigns a row number to each record. The @row\_number variable is incremented by 1 for each row, effectively numbering the rows.

AS numbered\_table: This gives the subquery a name (numbered\_table) that we can reference in the outer query.

WHERE row\_num % 2 = 0;: This condition selects only the rows where the row number is even. This effectively gives you alternative records.

SELECT DISTINCT ID, FIRST\_NAME, LAST\_NAME, AGE, EMAIL, SALARY FROM EMPLOYEE;

+------+------------+------------+------+-----------------------+--------+

| ID | FIRST\_NAME | LAST\_NAME | AGE | EMAIL | SALARY |

+------+------------+------------+------+-----------------------+--------+

| 1 | ARUN | PATEL | 22 | ARUN@GCOMPANY.IN | 40000 |

| 2 | BHAVESH | SHARMA | 24 | BHAVESH@GCOMPANY.IN | 30000 |

| 3 | CHAITANYA | SINGH | 23 | CHAITANYA@GCOMPANY.IN | 50000 |

| 4 | DEEPIKA | GUPTA | 26 | DEEPIKA@GCOMPANY.IN | 55000 |

| 5 | DHANUSH | KUMAR | 25 | DHANUSH@GCOMPANY.IN | 20000 |

| 6 | EKTA | YADAV | 28 | YADAV@GCOMPANY.IN | 35000 |

| 7 | GAURAV | RAO | 21 | GAURAV@GCOMPANY.IN | 60000 |

| 8 | HARSHITA | REDDY | 29 | HARSHITA@GCOMPANY.IN | 56000 |

| 9 | ISHAAN | REDDY | 32 | ISHAAN@GCOMPANY.IN | 70000 |

| 10 | JANU | MUKHERJEE | 30 | JANU@GCOMPANY.IN | 53000 |

+------+------------+------------+------+-----------------------+--------+

10 rows in set (0.00 sec)

* If we want to get only unique records then we should use DISTINCT.
* If 2 records has same ID, FIRST\_NAME, LAST\_NAME, AGE, EMAIL, SALARY in the EMPLOYEE table then only one record will be selected.

SELECT \* FROM EMPLOYEE GROUP BY ID, FIRST\_NAME, LAST\_NAME, AGE, EMAIL, SALARY;

+------+------------+------------+------+--------+-----------------------+

| ID | FIRST\_NAME | LAST\_NAME | AGE | SALARY | EMAIL |

+------+------------+------------+------+--------+-----------------------+

| 1 | ARUN | PATEL | 22 | 40000 | ARUN@GCOMPANY.IN |

| 2 | BHAVESH | SHARMA | 24 | 30000 | BHAVESH@GCOMPANY.IN |

| 3 | CHAITANYA | SINGH | 23 | 50000 | CHAITANYA@GCOMPANY.IN |

| 4 | DEEPIKA | GUPTA | 26 | 55000 | DEEPIKA@GCOMPANY.IN |

| 5 | DHANUSH | KUMAR | 25 | 20000 | DHANUSH@GCOMPANY.IN |

| 6 | EKTA | YADAV | 28 | 35000 | YADAV@GCOMPANY.IN |

| 7 | GAURAV | RAO | 21 | 60000 | GAURAV@GCOMPANY.IN |

| 8 | HARSHITA | REDDY | 29 | 56000 | HARSHITA@GCOMPANY.IN |

| 9 | ISHAAN | REDDY | 32 | 70000 | ISHAAN@GCOMPANY.IN |

| 10 | JANU | MUKHERJEE | 30 | 53000 | JANU@GCOMPANY.IN |

+------+------------+------------+------+--------+-----------------------+

10 rows in set (0.00 sec)

* Whichever the records has same ID, FIRST\_NAME, LAST\_NAME, AGE, EMAIL, SALARY in the EMPLOYEE table are grouping into one.
* Totally 10 groups are creating, each group contains 4 records of same data.
* From every group only one record is displayed.
* DISTINCT and GROUP BY are similar.

- MySQL does not have a built-in concept of a "ROWID" like some other databases (e.g., Oracle) do. In MySQL, you typically rely on primary keys (which are unique) to uniquely identify rows in a table.

**Constraints**

DROP TABLE IF EXISTS WORKTAB1;

CREATE TABLE WORKTAB1(ID INTEGER, NAME VARCHAR2(90), AGE INTEGER);

SELECT \* FROM WORKTAB1;

+------+------+------+

| ID | NAME | AGE |

+------+------+------+

| 1 | NULL | NULL |

+------+------+------+

1 row in set (0.00 sec)

* By default column allows NULL values.

INSERT INTO WORKTAB1(ID, NAME) VALUES(2, 'ABC');

Query OK, 1 row affected (0.01 sec)

SELECT \* FROM WORKTAB1;

+------+------+------+

| ID | NAME | AGE |

+------+------+------+

| 1 | NULL | NULL |

| 2 | ABC | NULL |

+------+------+------+

2 rows in set (0.00 sec)

INSERT INTO WORKTAB1(ID, AGE) VALUES(3, 33);

Query OK, 1 row affected (0.01 sec)

SELECT \* FROM WORKTAB1;

+------+------+------+

| ID | NAME | AGE |

+------+------+------+

| 1 | NULL | NULL |

| 2 | ABC | NULL |

| 3 | NULL | 33 |

+------+------+------+

3 rows in set (0.00 sec)

INSERT INTO WORKTAB1(NAME, AGE) VALUES('AMAN', 23);

Query OK, 1 row affected (0.01 sec)

SELECT \* FROM WORKTAB1;

+------+------+------+

| ID | NAME | AGE |

+------+------+------+

| 1 | NULL | NULL |

| 2 | ABC | NULL |

| 3 | NULL | 33 |

| NULL | AMAN | 23 |

+------+------+------+

4 rows in set (0.00 sec)

INSERT INTO WORKTAB1(NAME) VALUES('MANOHAR');

Query OK, 1 row affected (0.01 sec)

SELECT \* FROM WORKTAB1;

+------+---------+------+

| ID | NAME | AGE |

+------+---------+------+

| 1 | NULL | NULL |

| 2 | ABC | NULL |

| 3 | NULL | 33 |

| NULL | AMAN | 23 |

| NULL | MANOHAR | NULL |

+------+---------+------+

5 rows in set (0.00 sec)

INSERT INTO WORKTAB1(AGE) VALUES(25);

Query OK, 1 row affected (0.00 sec)

SELECT \* FROM WORKTAB1;

+------+---------+------+

| ID | NAME | AGE |

+------+---------+------+

| 1 | NULL | NULL |

| 2 | ABC | NULL |

| 3 | NULL | 33 |

| NULL | AMAN | 23 |

| NULL | MANOHAR | NULL |

| NULL | NULL | 25 |

+------+---------+------+

6 rows in set (0.00 sec)

DROP TABLE IF EXISTS WORKTAB2;

Query OK, 0 rows affected, 1 warning (0.01 sec)

CREATE TABLE WORKTAB2(ID INTEGER, NAME VARCHAR(90) NOT NULL, AGE INTEGER);

Query OK, 0 rows affected (0.03 sec)

DESC WORKTAB2;

+-------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------+-------------+------+-----+---------+-------+

| ID | int | YES | | NULL | |

| NAME | varchar(90) | NO | | NULL | |

| AGE | int | YES | | NULL | |

+-------+-------------+------+-----+---------+-------+

3 rows in set (0.01 sec)

* By using NOT NULL we can make sure that column are not having null values.
* In one table any number of columns can be NOT NULL.

INSERT INTO WORKTAB2(ID) VALUES(1);

ERROR 1364 (HY000): Field 'NAME' doesn't have a default value

INSERT INTO WORKTAB2(ID, NAME) VALUES(2, 'AMAN');

Query OK, 1 row affected (0.01 sec)

SELECT \* FROM WORKTAB2;

+------+------+------+

| ID | NAME | AGE |

+------+------+------+

| 2 | AMAN | NULL |

+------+------+------+

1 row in set (0.00 sec)

INSERT INTO WORKTAB2(ID, AGE) VALUES(3, 33);

ERROR 1364 (HY000): Field 'NAME' doesn't have a default value

INSERT INTO WORKTAB2(NAME, AGE) VALUES('MADHU', 23);

Query OK, 1 row affected (0.01 sec)

SELECT \* FROM WORKTAB2;

+------+-------+------+

| ID | NAME | AGE |

+------+-------+------+

| 2 | AMAN | NULL |

| NULL | MADHU | 23 |

+------+-------+------+

2 rows in set (0.00 sec)

INSERT INTO WORKTAB2(NAME) VALUES('MANOHAR');

Query OK, 1 row affected (0.01 sec)

SELECT \* FROM WORKTAB2;

+------+---------+------+

| ID | NAME | AGE |

+------+---------+------+

| 2 | AMAN | NULL |

| NULL | MADHU | 23 |

| NULL | MANOHAR | NULL |

+------+---------+------+

3 rows in set (0.00 sec)

INSERT INTO WORKTAB2(AGE) VALUES(25);

ERROR 1364 (HY000): Field 'NAME' doesn't have a default value

SELECT \* FROM WORKTAB2;

+------+---------+------+

| ID | NAME | AGE |

+------+---------+------+

| 2 | AMAN | NULL |

| NULL | MADHU | 23 |

| NULL | MANOHAR | NULL |

+------+---------+------+

3 rows in set (0.00 sec)

DROP TABLE IF EXISTS WORKTAB3;

Query OK, 0 rows affected, 1 warning (0.01 sec)

CREATE TABLE WORKTAB3(ID INTEGER NOT NULL, NAME VARCHAR(90) NOT NULL, AGE INTEGER);

Query OK, 0 rows affected (0.02 sec)

DESC WORKTAB3;

+-------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------+-------------+------+-----+---------+-------+

| ID | int | NO | | NULL | |

| NAME | varchar(90) | NO | | NULL | |

| AGE | int | YES | | NULL | |

+-------+-------------+------+-----+---------+-------+

3 rows in set (0.01 sec)

INSERT INTO WORKTAB3(ID) VALUES(1);

ERROR 1364 (HY000): Field 'NAME' doesn't have a default value

INSERT INTO WORKTAB3(ID, NAME) VALUES(2, 'AMAN');

Query OK, 1 row affected (0.00 sec)

SELECT \* FROM WORKTAB3;

+----+------+------+

| ID | NAME | AGE |

+----+------+------+

| 2 | AMAN | NULL |

+----+------+------+

1 row in set (0.00 sec)

INSERT INTO WORKTAB3(ID, AGE) VALUES(3, 33);

ERROR 1364 (HY000): Field 'NAME' doesn't have a default value

INSERT INTO WORKTAB3(NAME, AGE) VALUES('MADHU', 23);

ERROR 1364 (HY000): Field 'ID' doesn't have a default value

INSERT INTO WORKTAB3(NAME) VALUES('MADHU');

ERROR 1364 (HY000): Field 'ID' doesn't have a default value

INSERT INTO WORKTAB3(AGE) VALUES(25);

ERROR 1364 (HY000): Field 'ID' doesn't have a default value

SELECT \* FROM WORKTAB3;

+----+------+------+

| ID | NAME | AGE |

+----+------+------+

| 2 | AMAN | NULL |

+----+------+------+

1 row in set (0.00 sec)

DROP TABLE WORKTAB4;

ERROR 1051 (42S02): Unknown table 'mysql\_notes.worktab4'

DROP TABLE IF EXISTS WORKTAB4;

Query OK, 0 rows affected, 1 warning (0.01 sec)

CREATE TABLE WORKTAB4(ID INTEGER, NAME VARCHAR(90), AGE INTEGER);

Query OK, 0 rows affected (0.03 sec)

SELECT \* FROM WORKTAB4;

Empty set (0.00 sec)

DESC WORKTAB4;

+-------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------+-------------+------+-----+---------+-------+

| ID | int | YES | | NULL | |

| NAME | varchar(90) | YES | | NULL | |

| AGE | int | YES | | NULL | |

+-------+-------------+------+-----+---------+-------+

3 rows in set (0.00 sec)

INSERT INTO WORKTAB4(ID, NAME) VALUES(1, 'MANOHAR');

Query OK, 1 row affected (0.01 sec)

INSERT INTO WORKTAB4(ID, NAME) VALUES(1, 'MANOHAR');

Query OK, 1 row affected (0.01 sec)

INSERT INTO WORKTAB4(ID, NAME, AGE) VALUES(2, 'AMAN', 22);

Query OK, 1 row affected (0.01 sec)

INSERT INTO WORKTAB4(ID, NAME, AGE) VALUES(2, 'AMAN', 22);

Query OK, 1 row affected (0.01 sec)

INSERT INTO WORKTAB4(NAME, AGE) VALUES('EHTESHAM', 22);

Query OK, 1 row affected (0.01 sec)

INSERT INTO WORKTAB4(NAME, AGE) VALUES('EHTESHAM', 22);

Query OK, 1 row affected (0.01 sec)

SELECT \* FROM WORKTAB4;

+------+----------+------+

| ID | NAME | AGE |

+------+----------+------+

| 1 | MANOHAR | NULL |

| 1 | MANOHAR | NULL |

| 2 | AMAN | 22 |

| 2 | AMAN | 22 |

| NULL | EHTESHAM | 22 |

| NULL | EHTESHAM | 22 |

+------+----------+------+

6 rows in set (0.00 sec)

DROP TABLE IF EXISTS WORKTAB5;

Query OK, 0 rows affected, 1 warning (0.01 sec)

CREATE TABLE WORKTAB5(ID INTEGER, NAME VARCHAR(90) UNIQUE, AGE INTEGER);

Query OK, 0 rows affected (0.03 sec)

DESC WORKTAB5

-> ;

+-------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------+-------------+------+-----+---------+-------+

| ID | int | YES | | NULL | |

| NAME | varchar(90) | YES | UNI | NULL | |

| AGE | int | YES | | NULL | |

+-------+-------------+------+-----+---------+-------+

3 rows in set (0.00 sec)

* By default columns allow duplicate values.
* In one table any number of columns can be UNIQUE
* By using UNIQUE we can avoid duplicate values in the same column in the table.
* UNIQUE column allows any number of NULL values but not duplicate values.
* Two NULL values are not same i.e. they are not duplicate.

INSERT INTO WORKTAB5(ID, NAME) VALUES(1, 'MANOHAR');

Query OK, 1 row affected (0.01 sec)

INSERT INTO WORKTAB5(ID, NAME) VALUES(1, 'MANOHAR');

ERROR 1062 (23000): Duplicate entry 'MANOHAR' for key 'worktab5.NAME'

INSERT INTO WORKTAB5(ID, NAME, AGE) VALUES(2, 'AMAN', 22);

Query OK, 1 row affected (0.00 sec)

INSERT INTO WORKTAB5(ID, NAME, AGE) VALUES(2, 'AMAN', 22);

ERROR 1062 (23000): Duplicate entry 'AMAN' for key 'worktab5.NAME'

INSERT INTO WORKTAB5(NAME, AGE) VALUES('AMAN', 22);

ERROR 1062 (23000): Duplicate entry 'AMAN' for key 'worktab5.NAME'

INSERT INTO WORKTAB5(NAME, AGE) VALUES('EHSTESHAM', 22);

Query OK, 1 row affected (0.00 sec)

INSERT INTO WORKTAB5(ID, AGE) VALUES(3, 22);

Query OK, 1 row affected (0.01 sec)

INSERT INTO WORKTAB5(ID, AGE) VALUES(3, 22);

Query OK, 1 row affected (0.00 sec)

SELECT \* FROM WORKTAB5;

+------+-----------+------+

| ID | NAME | AGE |

+------+-----------+------+

| 1 | MANOHAR | NULL |

| 2 | AMAN | 22 |

| NULL | EHSTESHAM | 22 |

| 3 | NULL | 22 |

| 3 | NULL | 22 |

+------+-----------+------+

5 rows in set (0.00 sec)

DROP TABLE IF EXISTS WORKTAB6;

Query OK, 0 rows affected, 1 warning (0.00 sec)

CREATE TABLE WORKTAB6(ID INTEGER, NAME VARCHAR(90) UNIQUE, AGE INTEGER UNIQUE);

Query OK, 0 rows affected (0.03 sec)

DESC WORKTAB6;

+-------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------+-------------+------+-----+---------+-------+

| ID | int | YES | | NULL | |

| NAME | varchar(90) | YES | UNI | NULL | |

| AGE | int | YES | UNI | NULL | |

+-------+-------------+------+-----+---------+-------+

3 rows in set (0.00 sec)

INSERT INTO WORKTAB6(ID, NAME) VALUES(1, 'MANOHAR');

Query OK, 1 row affected (0.01 sec)

INSERT INTO WORKTAB6(ID, NAME) VALUES(1, 'MANOHAR');

ERROR 1062 (23000): Duplicate entry 'MANOHAR' for key 'worktab6.NAME'

INSERT INTO WORKTAB6(ID, NAME, AGE) VALUES(2, 'AMAN', 22);

Query OK, 1 row affected (0.00 sec)

INSERT INTO WORKTAB6(ID, NAME, AGE) VALUES(2, 'AMAN', 22);

ERROR 1062 (23000): Duplicate entry 'AMAN' for key 'worktab6.NAME'

INSERT INTO WORKTAB6(NAME, AGE) VALUES('AMAN', 22);

ERROR 1062 (23000): Duplicate entry 'AMAN' for key 'worktab6.NAME'

INSERT INTO WORKTAB6(ID, AGE) VALUES(5, 22);

ERROR 1062 (23000): Duplicate entry '22' for key 'worktab6.AGE'

INSERT INTO WORKTAB6(ID) VALUES(5);

Query OK, 1 row affected (0.00 sec)

INSERT INTO WORKTAB6(ID) VALUES(5);

Query OK, 1 row affected (0.01 sec)

SELECT \* FROM WORKTAB6;

+------+---------+------+

| ID | NAME | AGE |

+------+---------+------+

| 1 | MANOHAR | NULL |

| 2 | AMAN | 22 |

| 5 | NULL | NULL |

| 5 | NULL | NULL |

+------+---------+------+

4 rows in set (0.00 sec)

DROP TABLE IF EXISTS WORKTAB7;

Query OK, 0 rows affected, 1 warning (0.01 sec)

CREATE TABLE WORKTAB7(ID INTEGER, NAME VARCHAR(90), AGE INTEGER, CONSTRAINT WORKTAB7\_UK1 UNIQUE(NAME), CONSTRAINT WORKTAB7\_UK2 UNIQUE(AGE));

Query OK, 0 rows affected (0.03 sec)

* Syntax CONSTRAINT(declaration) WORKTAB7\_UK1(IDENTIFIER) UNIQUE(NAME)(type of the constraint and column name)
* Every constraints should be having unique identifier names in across the tables.
* We can disable or permanently drop the constraints. It is the better approach than previous.

INSERT INTO WORKTAB7(ID, NAME) VALUES(1, 'MANOHAR');

Query OK, 1 row affected (0.01 sec)

INSERT INTO WORKTAB7(ID, NAME) VALUES(1, 'MANOHAR');

ERROR 1062 (23000): Duplicate entry 'MANOHAR' for key 'worktab7.WORKTAB7\_UK1'

INSERT INTO WORKTAB7(ID, NAME, AGE) VALUES(2, 'AMAN', 22);

Query OK, 1 row affected (0.00 sec)

INSERT INTO WORKTAB7(ID, NAME, AGE) VALUES(2, 'AMAN', 22);

ERROR 1062 (23000): Duplicate entry 'AMAN' for key 'worktab7.WORKTAB7\_UK1'

INSERT INTO WORKTAB7(NAME, AGE) VALUES('EHSTESHAM', 22);

ERROR 1062 (23000): Duplicate entry '22' for key 'worktab7.WORKTAB7\_UK2'

INSERT INTO WORKTAB7(ID, AGE) VALUES(5, 22);

ERROR 1062 (23000): Duplicate entry '22' for key 'worktab7.WORKTAB7\_UK2'

INSERT INTO WORKTAB7(ID) VALUES(5);

Query OK, 1 row affected (0.03 sec)

INSERT INTO WORKTAB7(ID) VALUES(6);

Query OK, 1 row affected (0.00 sec)

SELECT \* FROM WORKTAB7;

+------+---------+------+

| ID | NAME | AGE |

+------+---------+------+

| 1 | MANOHAR | NULL |

| 2 | AMAN | 22 |

| 5 | NULL | NULL |

| 6 | NULL | NULL |

+------+---------+------+

4 rows in set (0.00 sec)

DROP TABLE IF EXISTS WORKTAB8;

Query OK, 0 rows affected, 1 warning (0.01 sec)

CREATE TABLE WORKTAB8(ID INTEGER, NAME VARCHAR(90), AGE INTEGER, CONSTRAINT WORKTAB8\_UK1 UNIQUE(NAME, AGE));

Query OK, 0 rows affected (0.02 sec)

INSERT INTO WORKTAB8(ID, NAME) VALUES(1, 'AMAN');

Query OK, 1 row affected (0.01 sec)

INSERT INTO WORKTAB8(ID, NAME) VALUES(1, 'AMAN');

Query OK, 1 row affected (0.01 sec)

INSERT INTO WORKTAB8(ID, NAME) VALUES(1, 'AMAN');

Query OK, 1 row affected (0.00 sec)

DESC WORKTAB8;

+-------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------+-------------+------+-----+---------+-------+

| ID | int | YES | | NULL | |

| NAME | varchar(90) | YES | MUL | NULL | |

| AGE | int | YES | | NULL | |

+-------+-------------+------+-----+---------+-------+

3 rows in set (0.00 sec)

* In the above constraint two records cant be having same values for the NAME and AGE columns.
* We can refer to it as a composite unique key.

INSERT INTO WORKTAB8(ID, NAME, AGE) VALUES(1, 'AMAN', 22);

Query OK, 1 row affected (0.01 sec)

INSERT INTO WORKTAB8(ID, NAME, AGE) VALUES(1, 'AMAN', 22);

ERROR 1062 (23000): Duplicate entry 'AMAN-22' for key 'worktab8.WORKTAB8\_UK1'

INSERT INTO WORKTAB8(ID, NAME, AGE) VALUES(1, 'AMAN', 23);

Query OK, 1 row affected (0.01 sec)

INSERT INTO WORKTAB8(ID, NAME, AGE) VALUES(1, 'AMAN', 23);

ERROR 1062 (23000): Duplicate entry 'AMAN-23' for key 'worktab8.WORKTAB8\_UK1'

INSERT INTO WORKTAB8(ID, NAME, AGE) VALUES(1, 'AMAN', 24);

Query OK, 1 row affected (0.01 sec)

INSERT INTO WORKTAB8(ID, NAME, AGE) VALUES(1, 'MANOHAR', 25);

Query OK, 1 row affected (0.01 sec)

INSERT INTO WORKTAB8(ID, NAME, AGE) VALUES(1, 'EHTESHAM', 25);

Query OK, 1 row affected (0.00 sec)

INSERT INTO WORKTAB8(ID) VALUES(6);

Query OK, 1 row affected (0.01 sec)

INSERT INTO WORKTAB8(ID) VALUES(6);

Query OK, 1 row affected (0.00 sec)

* We don’t get error because NULL, NULL is not considered as combination.

INSERT INTO WORKTAB8(ID, NAME) VALUES(1, 'AMAN');

Query OK, 1 row affected (0.01 sec)

INSERT INTO WORKTAB8(ID, NAME) VALUES(1, 'AMAN');

Query OK, 1 row affected (0.00 sec)

* We don’t get error because AMAN, NULL and again AMAN, NULL is not considered as combination.
* The reason you're not seeing an error when inserting a record with a value and NULL in the same columns is because NULL is considered a distinct value. Therefore, a combination of ('John', NULL) and ('John', NULL) is considered unique according to the rules of MySQL

INSERT INTO TAB8(ID, NAME, AGE) VALUES(2, 'RAMU', 22);

INSERT INTO TAB8(ID, NAME, AGE) VALUES(2, 'RAMU', 23);

INSERT INTO TAB8(ID, NAME, AGE) VALUES(2, 'AMU', 23);

INSERT INTO TAB8(ID, NAME, AGE) VALUES(2, 'RAMU', 22);//ERROR

INSERT INTO TAB8(NAME, AGE) VALUES('RAMU', 22);//ERROR

INSERT INTO TAB8(NAME, AGE) VALUES('RAMU', 22);//ERROR

INSERT INTO TAB8(ID, AGE) VALUES(5, 22); //ERROR

INSERT INTO TAB8(ID, AGE) VALUES(5, 22); //ERROR

INSERT INTO TAB8(ID) VALUES(5);

INSERT INTO TAB8(ID) VALUES(5);

INSERT INTO TAB8(ID) VALUES(5);

INSERT INTO TAB8(ID) VALUES(5);

* The above queries possible because under UNIQUE constraint any number of NULL values and that won’t be considered as a combination.

INSERT INTO TAB8(AGE) VALUES(25);

SELECT \* FROM TAB8;

DROP TABLE IF EXISTS WORKTAB9;

Query OK, 0 rows affected, 1 warning (0.05 sec)

CREATE TABLE WORKTAB9(ID INTEGER, NAME VARCHAR(90) PRIMARY KEY, AGE INTEGER);

Query OK, 0 rows affected (0.07 sec)

* PRIMARY KEY is a combination of NOT NULL and UNIQUE.
* If any column is declared as PRIMARY KEY then that column value should not be NULL value and should not contain duplicate value.

INSERT INTO WORKTAB9(ID, NAME) VALUES(1, 'MANOHAR');

Query OK, 1 row affected (0.01 sec)

INSERT INTO WORKTAB9(ID, NAME) VALUES(1, 'MANOHAR');

ERROR 1062 (23000): Duplicate entry 'MANOHAR' for key 'worktab9.PRIMARY'

INSERT INTO WORKTAB9(ID, NAME, AGE) VALUES(2, 'AMAN', 22);

Query OK, 1 row affected (0.01 sec)

INSERT INTO WORKTAB9(ID, NAME, AGE) VALUES(2, 'AMAN', 22);

ERROR 1062 (23000): Duplicate entry 'AMAN' for key 'worktab9.PRIMARY'

INSERT INTO WORKTAB9(ID, AGE) VALUES(5, 22);

ERROR 1364 (HY000): Field 'NAME' doesn't have a default value

INSERT INTO WORKTAB9(ID) VALUES(5);

ERROR 1364 (HY000): Field 'NAME' doesn't have a default value

INSERT INTO WORKTAB9(NAME) VALUES('VIJAY');

Query OK, 1 row affected (0.01 sec)

INSERT INTO WORKTAB9(NAME) VALUES('VIJAY');

ERROR 1062 (23000): Duplicate entry 'VIJAY' for key 'worktab9.PRIMARY'

SELECT \* FROM WORKTAB9;

+------+-----------+------+

| ID | NAME | AGE |

+------+-----------+------+

| 2 | AMAN | 22 |

| NULL | EHSTESHAM | 22 |

| 1 | MANOHAR | NULL |

| NULL | VIJAY | NULL |

+------+-----------+------+

4 rows in set (0.00 sec)

CREATE TABLE WORKTAB10(ID INTEGER, NAME VARCHAR(90) PRIMARY KEY, AGE INTEGER PRIMARY KEY);

ERROR 1068 (42000): Multiple primary key defined

* In a table there should be only one column declared as PRIMARY KEY not more than one column.

CREATE TABLE WORKTAB11(ID INTEGER, NAME VARCHAR(90), AGE INTEGER, CONSTRAINT WORKTAB11\_PK1 PRIMARY KEY(AGE));

Query OK, 0 rows affected (0.04 sec)

INSERT INTO WORKTAB11(ID, NAME) VALUES(1, 'MANOHAR');

ERROR 1364 (HY000): Field 'AGE' doesn't have a default value

INSERT INTO WORKTAB11(ID, NAME, AGE) VALUES(2, 'AMAN', 22);

Query OK, 1 row affected (0.00 sec)

INSERT INTO WORKTAB11(ID, NAME, AGE) VALUES(2, 'AMAN', 22);

ERROR 1062 (23000): Duplicate entry '22' for key 'worktab11.PRIMARY'

INSERT INTO WORKTAB11(NAME, AGE) VALUES('AMAN', 22);

ERROR 1062 (23000): Duplicate entry '22' for key 'worktab11.PRIMARY'

INSERT INTO WORKTAB11(NAME, AGE) VALUES('AMAN', 28);

Query OK, 1 row affected (0.00 sec)

INSERT INTO WORKTAB11(ID, AGE) VALUES(5, 28);

ERROR 1062 (23000): Duplicate entry '28' for key 'worktab11.PRIMARY'

INSERT INTO WORKTAB11(ID) VALUES(5);

ERROR 1364 (HY000): Field 'AGE' doesn't have a default value

INSERT INTO WORKTAB11(AGE) VALUES(25);

Query OK, 1 row affected (0.00 sec)

INSERT INTO WORKTAB11(NAME) VALUES('VIJAY');

ERROR 1364 (HY000): Field 'AGE' doesn't have a default value

SELECT \* FROM WORKTAB11;

+------+------+-----+

| ID | NAME | AGE |

+------+------+-----+

| 2 | AMAN | 22 |

| NULL | NULL | 25 |

| NULL | AMAN | 28 |

+------+------+-----+

CREATE TABLE WORKTAB12(ID INTEGER, NAME VARCHAR(90), AGE INTEGER, CONSTRAINT WORKTAB12\_PK1 PRIMARY KEY(AGE, NAME));

* Composite PRIMARY key is possible.

INSERT INTO WORKTAB12 VALUES(1, 'AMAN', 22);

Query OK, 1 row affected (0.01 sec)

INSERT INTO WORKTAB12 VALUES(2, 'AMAN', 22);

ERROR 1062 (23000): Duplicate entry '22-AMAN' for key 'worktab12.PRIMARY'

INSERT INTO WORKTAB12 VALUES(3, 'AMAN', 23);

Query OK, 1 row affected (0.01 sec)

INSERT INTO WORKTAB12 VALUES(4, 'MANOHAR', 23);

Query OK, 1 row affected (0.00 sec)

INSERT INTO WORKTAB12(ID, NAME) VALUES(5, 'JAGAN');

ERROR 1364 (HY000): Field 'AGE' doesn't have a default value

INSERT INTO WORKTAB12(ID, AGE) VALUES(6, 25);

ERROR 1364 (HY000): Field 'NAME' doesn't have a default value

**FOREIGN KEY**

CREATE TABLE FS\_STUDENT(ID INTEGER UNIQUE, FIRST\_NAME VARCHAR(90), LAST\_NAME VARCHAR(90), AGE INTEGER, EMAIL VARCHAR(90));

Query OK, 0 rows affected (0.04 sec)

CREATE TABLE FS\_STUDENT\_ADDRESS(HOUSE\_NO VARCHAR(90), STREET\_NAME VARCHAR(90), CITY VARCHAR(90), STATE VARCHAR(90), FS\_STUDENT\_ID INTEGER, CONSTRAINT FS\_STUDENT\_ADDRESS\_FK1 FOREIGN KEY(FS\_STUDENT\_ID) REFERENCES FS\_STUDENT(ID));

Query OK, 0 rows affected (0.03 sec)

* For FOREIGN KEY purpose REFERENCES table column either UNIQUE or PRIMARY

INSERT INTO FS\_STUDENT VALUES(1, 'AMAN', 'GUPTA', 22, 'AMAN@VP.COM');

Query OK, 1 row affected (0.01 sec)

INSERT INTO FS\_STUDENT\_ADDRESS VALUES('123/A', 'BTM', 'BLR', 'KAR', 1);

Query OK, 1 row affected (0.01 sec)

INSERT INTO FS\_STUDENT VALUES(2, 'MANOHAR', 'VERMA', 24, 'MANOHAR@VP.COM');

Query OK, 1 row affected (0.01 sec)

INSERT INTO FS\_STUDENT\_ADDRESS VALUES('256/B', 'BSK', 'BLR', 'KAR', 2);

Query OK, 1 row affected (0.00 sec)

INSERT INTO FS\_STUDENT VALUES(3, 'VIJAY', 'KUMAR', 26, 'VIJAY@VP.COM');

Query OK, 1 row affected (0.00 sec)

INSERT INTO FS\_STUDENT\_ADDRESS VALUES('126/C', 'JPN', 'BLR', 'KAR', 3);

Query OK, 1 row affected (0.01 sec)

INSERT INTO FS\_STUDENT\_ADDRESS VALUES('450/D', 'KKC', 'BLR', 'KAR', 5);

ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`mysql\_notes`.`fs\_student\_address`, CONSTRAINT `FS\_STUDENT\_ADDRESS\_FK1` FOREIGN KEY (`FS\_STUDENT\_ID`) REFERENCES `fs\_student` (`ID`))

* Trying to insert child record straight away without parent.
* FOREIGN KEY should have reference value of the column from the parent.

INSERT INTO FS\_STUDENT VALUES(4, 'JAGAN', 'REDDY', 27, 'JAGAN@VP.COM');

Query OK, 1 row affected (0.01 sec)

INSERT INTO FS\_STUDENT\_ADDRESS VALUES('450/D', 'KKC', 'BLR', 'KAR', 4);

Query OK, 1 row affected (0.01 sec)

DELETE FROM FS\_STUDENT WHERE ID = 4;

ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (`mysql\_notes`.`fs\_student\_address`, CONSTRAINT `FS\_STUDENT\_ADDRESS\_FK1` FOREIGN KEY (`FS\_STUDENT\_ID`) REFERENCES `fs\_student` (`ID`))

* FS\_STUDENT ID = 4 having child in the FS\_STUDENT\_ADDRESS.
* You can’t delete parent record without deleting a child record.

DELETE FROM FS\_STUDENT\_ADDRESS WHERE FS\_STUDENT\_ID = 4;

Query OK, 1 row affected (0.01 sec)

DELETE FROM FS\_STUDENT WHERE ID = 4;

Query OK, 1 row affected (0.01 sec)

DROP TABLE FS\_STUDENT;

ERROR 3730 (HY000): Cannot drop table 'fs\_student' referenced by a foreign key constraint 'FS\_STUDENT\_ADDRESS\_FK1' on table 'fs\_student\_address'.

DROP TABLE FS\_STUDENT\_ADDRESS;

Query OK, 0 rows affected (0.02 sec)

DROP TABLE FS\_STUDENT;

Query OK, 0 rows affected (0.01 sec)

* Straight away we cant delete STUDENT table.
* STUDENT table is a parent to ADDRESS table
* Without deleting the child we cant able to delete PARENT
* Even though ADDRESS table is empty we cant able to drop STUDENT table.
* First we need to drop ADDRESS table then only we can able to drop the STUDENT table.

CREATE TABLE FS\_STUDENT(ID INTEGER, FIRST\_NAME VARCHAR(90), LAST\_NAME VARCHAR(90), AGE INTEGER, EMAIL VARCHAR(90));

Query OK, 0 rows affected (0.03 sec)

CREATE TABLE FS\_STUDENT\_ADDRESS(HOUSE\_NO VARCHAR(90), STREET\_NAME VARCHAR(90), CITY VARCHAR(90), STATE VARCHAR(90), FS\_STUDENT\_ID INTEGER, CONSTRAINT FS\_STUDENT\_ADDRESS\_FK1 FOREIGN KEY(FS\_STUDENT\_ID) REFERENCES FS\_STUDENT(ID));

ERROR 1822 (HY000): Failed to add the foreign key constraint. Missing index for constraint 'FS\_STUDENT\_ADDRESS\_FK1' in the referenced table 'fs\_student'

DROP TABLE FS\_STUDENT;

CREATE TABLE FS\_STUDENT(ID INTEGER UNIQUE, FIRST\_NAME VARCHAR(90), LAST\_NAME VARCHAR(90), AGE INTEGER, EMAIL VARCHAR(90));

Query OK, 0 rows affected (0.04 sec)

CREATE TABLE FS\_STUDENT\_ADDRESS(HOUSE\_NO VARCHAR(90), STREET\_NAME VARCHAR(90), CITY VARCHAR(90), STATE VARCHAR(90), FS\_STUDENT\_ID INTEGER, CONSTRAINT FS\_STUDENT\_ADDRESS\_FK1 FOREIGN KEY(FS\_STUDENT\_ID) REFERENCES FS\_STUDENT(ID));

INSERT INTO FS\_STUDENT VALUES(1, 'AMAN', 'GUPTA', 22, 'AMAN@GMAIL.COM');

Query OK, 1 row affected (0.01 sec)

INSERT INTO FS\_STUDENT\_ADDRESS(HOUSE\_NO, STREET\_NAME, CITY, STATE) VALUES('140/F', 'RRN', 'BLR', 'KAR');

Query OK, 1 row affected (0.01 sec)

* we can have NULL values for FOREIGN KEY REFERENCE.
* If FS\_STUDENT\_ID is not PRIMARY KEY in the ADDRESS we can have NULL values.
* We are inserting an ADDRESS which doesn’t belong to any STUDENT.
* By default FOREIGN KEY allows NULL values.

UPDATE FS\_STUDENT\_ADDRESS SET FS\_STUDENT\_ID = 3 WHERE HOUSE\_NO = '140/F';

ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`mysql\_notes`.`fs\_student\_address`, CONSTRAINT `FS\_STUDENT\_ADDRESS\_FK1` FOREIGN KEY (`FS\_STUDENT\_ID`) REFERENCES `fs\_student` (`ID`))

* There is no corresponding record.

UPDATE FS\_STUDENT\_ADDRESS SET FS\_STUDENT\_ID = 1 WHERE HOUSE\_NO = '140/F';

Query OK, 1 row affected (0.01 sec)

Rows matched: 1 Changed: 1 Warnings: 0

INSERT INTO FS\_STUDENT(FIRST\_NAME, LAST\_NAME, AGE, EMAIL) VALUES('MANOHAR', 'VERMA', 22, 'MANOHAR@GMAIL.COM');

Query OK, 1 row affected (0.03 sec)

INSERT INTO FS\_STUDENT\_ADDRESS(HOUSE\_NO, STREET\_NAME, CITY, STATE) VALUES('224/Y', 'RGN', 'BLR', 'KAR');

Query OK, 1 row affected (0.01 sec)

* In the base table ID column is NULL and in the child table FS\_STUDENT\_ID is also NULL.
* In the parent table there is a record with no ID.
* In the child table there is record which doesn’t match to any of the parent table records.
* NULL cant be assigned to another NULL.
* NULL cant be mapped to another NULL.

UPDATE FS\_STUDENT SET ID = 2 WHERE FIRST\_NAME ='MANOHAR';

Query OK, 1 row affected (0.01 sec)

Rows matched: 1 Changed: 1 Warnings: 0

UPDATE FS\_STUDENT\_ADDRESS SET FS\_STUDENT\_ID = 2 WHERE HOUSE\_NO = '224/Y';

Query OK, 1 row affected (0.01 sec)

Rows matched: 1 Changed: 1 Warnings: 0

**ONE-TO-ONE**

DROP TABLE IF EXISTS PERSON;

CREATE TABLE PERSON (ID INTEGER PRIMARY KEY, FIRST\_NAME VARCHAR(90), LAST\_NAME VARCHAR(90), AGE INTEGER);

* Because of ID column is PRIMARY KEY PERSON table can become a parent to child table.

DROP TABLE IF EXISTS ADDRESS;

CREATE TABLE ADDRESS(HOUSE\_NO VARCHAR(90), STREET\_NAME VARCHAR(90), CITY VARCHAR(90), STATE VARCHAR(90), PERSON\_ID INTEGER UNIQUE, CONSTRAINT ADDRESS\_FK1 FOREIGN KEY(PERSON\_ID) REFERENCES PERSON(ID));

INSERT INTO PERSON VALUES(1, 'AMAN', 'GUPTA', 22);

INSERT INTO PERSON VALUES(2, 'MANOHAR', 'VERMA', 24);

INSERT INTO PERSON VALUES(3, 'SWETHA', 'SHARMA', 21);

INSERT INTO PERSON VALUES(4, 'VIJAY', 'VAISHNAV', 23);

INSERT INTO PERSON VALUES(5, 'KUMAR', 'SINHA', 23);

SELECT \* FROM PERSON;

+----+------------+-----------+------+

| ID | FIRST\_NAME | LAST\_NAME | AGE |

+----+------------+-----------+------+

| 1 | AMAN | GUPTA | 22 |

| 2 | MANOHAR | VERMA | 24 |

| 3 | SWETHA | SHARMA | 21 |

| 4 | VIJAY | VAISHNAV | 23 |

| 5 | KUMAR | SINHA | 23 |

+----+------------+-----------+------+

INSERT INTO ADDRESS VALUES('185/A', 'BSK', 'BLR', 'KAR', 1);

Query OK, 1 row affected (0.01 sec)

INSERT INTO ADDRESS VALUES('224/B', 'JPN', 'BLR', 'KAR', 1);

ERROR 1062 (23000): Duplicate entry '1' for key 'address.PERSON\_ID'

* FOREIGN KEY is a UNIQUE so we can’t insert duplicates.
* One record of PERSON mapping to only one record of ADDRESS. So we call it as one to one mapping.

INSERT INTO ADDRESS VALUES('185/A', 'BSK', 'BLR', 'KAR', 1);

Query OK, 1 row affected (0.01 sec)

INSERT INTO ADDRESS VALUES('224/B', 'JPN', 'BLR', 'KAR', 1);

ERROR 1062 (23000): Duplicate entry '1' for key 'address.PERSON\_ID'

INSERT INTO ADDRESS VALUES('228/C', 'RRN', 'BLR', 'KAR', 2);

Query OK, 1 row affected (0.01 sec)

INSERT INTO ADDRESS VALUES('356/D', 'KKC', 'BLR', 'KAR', 3);

Query OK, 1 row affected (0.00 sec)

INSERT INTO ADDRESS VALUES('521/F', 'RJN', 'BLR', 'KAR', 4);

Query OK, 1 row affected (0.01 sec)

INSERT INTO ADDRESS VALUES('652/G', 'KRL', 'BLR', 'KAR', 6);

ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`mysql\_notes`.`address`, CONSTRAINT `ADDRESS\_FK1` FOREIGN KEY (`PERSON\_ID`) REFERENCES `person` (`ID`))

* No PERSON with ID as 6

INSERT INTO ADDRESS(HOUSE\_NO, STREET\_NAME, CITY, STATE) VALUES('224/I', 'BTM', 'BLR', 'KAR');

* We can insert ADDRESS without choosing PERSON\_ID because this column is UNIQUE and allows NULL values.

SELECT \* FROM PERSON;

SELECT \* FROM ADDRESS;

SELECT \* FROM PERSON;

+----+------------+-----------+------+

| ID | FIRST\_NAME | LAST\_NAME | AGE |

+----+------------+-----------+------+

| 1 | AMAN | GUPTA | 22 |

| 2 | MANOHAR | VERMA | 24 |

| 3 | SWETHA | SHARMA | 21 |

| 4 | VIJAY | VAISHNAV | 23 |

| 5 | KUMAR | SINHA | 23 |

+----+------------+-----------+------+

5 rows in set (0.00 sec)

SELECT \* FROM ADDRESS;

+----------+-------------+------+-------+-----------+

| HOUSE\_NO | STREET\_NAME | CITY | STATE | PERSON\_ID |

+----------+-------------+------+-------+-----------+

| 185/A | BSK | BLR | KAR | 1 |

| 228/C | RRN | BLR | KAR | 2 |

| 356/D | KKC | BLR | KAR | 3 |

| 521/F | RJN | BLR | KAR | 4 |

| 224/I | BTM | BLR | KAR | NULL |

+----------+-------------+------+-------+-----------+

SELECT \* FROM PERSON WHERE FIRST\_NAME = 'AMAN';

+----+------------+-----------+------+

| ID | FIRST\_NAME | LAST\_NAME | AGE |

+----+------------+-----------+------+

| 1 | AMAN | GUPTA | 22 |

+----+------------+-----------+------+

SELECT \* FROM ADDRESS WHERE HOUSE\_NO = '185/A';

+----------+-------------+------+-------+-----------+

| HOUSE\_NO | STREET\_NAME | CITY | STATE | PERSON\_ID |

+----------+-------------+------+-------+-----------+

| 185/A | BSK | BLR | KAR | 1 |

+----------+-------------+------+-------+-----------+

SELECT \* FROM ADDRESS WHERE PERSON\_ID = 3;

+----------+-------------+------+-------+-----------+

| HOUSE\_NO | STREET\_NAME | CITY | STATE | PERSON\_ID |

+----------+-------------+------+-------+-----------+

| 356/D | KKC | BLR | KAR | 3 |

+----------+-------------+------+-------+-----------+

SELECT \* FROM ADDRESS WHERE PERSON\_ID = (SELECT ID FROM PERSON WHERE FIRST\_NAME = 'AMAN');

+----------+-------------+------+-------+-----------+

| HOUSE\_NO | STREET\_NAME | CITY | STATE | PERSON\_ID |

+----------+-------------+------+-------+-----------+

| 185/A | BSK | BLR | KAR | 1 |

+----------+-------------+------+-------+-----------+

SELECT \* FROM PERSON WHERE ID = (SELECT PERSON\_ID FROM ADDRESS WHERE HOUSE\_NO = '185/A');

+----+------------+-----------+------+

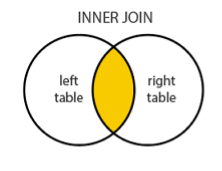
| ID | FIRST\_NAME | LAST\_NAME | AGE |

+----+------------+-----------+------+

| 1 | AMAN | GUPTA | 22 |

+----+------------+-----------+------+

* The default join is inner join, if you are not specifying any keywords.



SELECT \* FROM PERSON, ADDRESS WHERE PERSON.ID = ADDRESS.PERSON\_ID;

**+----+------------+-----------+------+----------+-------------+------+-------+-----------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | HOUSE\_NO | STREET\_NAME | CITY | STATE | PERSON\_ID |**

**+----+------------+-----------+------+----------+-------------+------+-------+-----------+**

**| 1 | AMAN | GUPTA | 22 | 185/A | BSK | BLR | KAR | 1 |**

**| 2 | MANOHAR | VERMA | 24 | 228/C | RRN | BLR | KAR | 2 |**

**| 3 | SWETHA | SHARMA | 21 | 356/D | KKC | BLR | KAR | 3 |**

**| 4 | VIJAY | VAISHNAV | 23 | 521/F | RJN | BLR | KAR | 4 |**

**+----+------------+-----------+------+----------+-------------+------+-------+-----------+**

SELECT \* FROM PERSON P, ADDRESS A WHERE P.ID = A.PERSON\_ID;

**SELECT \* FROM PERSON P INNER JOIN ADDRESS A ON P.ID = A.PERSON\_ID;**

**+----+------------+-----------+------+----------+-------------+------+-------+-----------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | HOUSE\_NO | STREET\_NAME | CITY | STATE | PERSON\_ID |**

**+----+------------+-----------+------+----------+-------------+------+-------+-----------+**

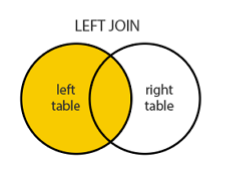
**| 1 | AMAN | GUPTA | 22 | 185/A | BSK | BLR | KAR | 1 |**

**| 2 | MANOHAR | VERMA | 24 | 228/C | RRN | BLR | KAR | 2 |**

**| 3 | SWETHA | SHARMA | 21 | 356/D | KKC | BLR | KAR | 3 |**

**| 4 | VIJAY | VAISHNAV | 23 | 521/F | RJN | BLR | KAR | 4 |**

**+----+------------+-----------+------+----------+-------------+------+-------+-----------+**



SELECT \* FROM PERSON P LEFT OUTER JOIN ADDRESS A ON P.ID = A.PERSON\_ID;

**+----+------------+-----------+------+----------+-------------+------+-------+-----------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | HOUSE\_NO | STREET\_NAME | CITY | STATE | PERSON\_ID |**

**+----+------------+-----------+------+----------+-------------+------+-------+-----------+**

**| 1 | AMAN | GUPTA | 22 | 185/A | BSK | BLR | KAR | 1 |**

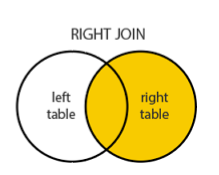
**| 2 | MANOHAR | VERMA | 24 | 228/C | RRN | BLR | KAR | 2 |**

**| 3 | SWETHA | SHARMA | 21 | 356/D | KKC | BLR | KAR | 3 |**

**| 4 | VIJAY | VAISHNAV | 23 | 521/F | RJN | BLR | KAR | 4 |**

**| 5 | KUMAR | SINHA | 23 | NULL | NULL | NULL | NULL | NULL |**

**+----+------------+-----------+------+----------+-------------+------+-------+-----------+**



SELECT \* FROM PERSON P RIGHT OUTER JOIN ADDRESS A ON P.ID = A.PERSON\_ID;

SELECT \* FROM PERSON P RIGHT OUTER JOIN ADDRESS A ON P.ID = A.PERSON\_ID;

**+------+------------+-----------+------+----------+-------------+------+-------+-----------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | HOUSE\_NO | STREET\_NAME | CITY | STATE | PERSON\_ID |**

**+------+------------+-----------+------+----------+-------------+------+-------+-----------+**

**| 1 | AMAN | GUPTA | 22 | 185/A | BSK | BLR | KAR | 1 |**

**| 2 | MANOHAR | VERMA | 24 | 228/C | RRN | BLR | KAR | 2 |**

**| 3 | SWETHA | SHARMA | 21 | 356/D | KKC | BLR | KAR | 3 |**

**| 4 | VIJAY | VAISHNAV | 23 | 521/F | RJN | BLR | KAR | 4 |**

**| NULL | NULL | NULL | NULL | 224/I | BTM | BLR | KAR | NULL |**

**+------+------------+-----------+------+----------+-------------+------+-------+-----------+**

* Unfortunately we don’t have full outer join in mysql database but this feature is available on oracle database.

SELECT \* FROM PERSON P FULL OUTER JOIN ADDRESS A ON P.ID = A.PERSON\_ID;

ERROR 1064 (42000): 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 'FULL OUTER JOIN ADDRESS A ON P.ID = A.PERSON\_ID' at line 1

**ONE-TO-MANY**

DROP TABLE IF EXISTS PERSON, ADDRESS;

CREATE TABLE PERSON (ID INTEGER PRIMARY KEY, FIRST\_NAME VARCHAR(90), LAST\_NAME VARCHAR(90), AGE INTEGER);

DROP IF EXISTS TABLE MAIL\_ACCOUNT;

CREATE TABLE EMAIL\_ID(USERNAME VARCHAR(90), PASSWORD VARCHAR(90), VENDOR VARCHAR(90), PERSON\_ID INTEGER, CONSTRAINT MA\_FK1 FOREIGN KEY(PERSON\_ID) REFERENCES PERSON(ID));

INSERT INTO PERSON VALUES(1, 'AMAN', 'GUPTA', 22);

INSERT INTO PERSON VALUES(2, 'MANOHAR', 'VERMA', 24);

INSERT INTO PERSON VALUES(3, 'SWETHA', 'SHARMA', 21);

INSERT INTO PERSON VALUES(4, 'VIJAY', 'VAISHNAV', 23);

INSERT INTO PERSON VALUES(5, 'KUMAR', 'SINHA', 23);

INSERT INTO EMAIL\_ID VALUES('AMAN', 'PASS', 'GMAIL', 1);

INSERT INTO EMAIL\_ID VALUES('AMAN', 'PASS', 'OUTLOOK', 1);

INSERT INTO EMAIL\_ID VALUES('AMAN', 'PASS', 'HOTMAIL', 1);

INSERT INTO EMAIL\_ID VALUES('MANOHAR', 'PASS', 'YAHOO', 2);

INSERT INTO EMAIL\_ID VALUES('MANOHAR', 'PASS', 'GMAIL', 2);

INSERT INTO EMAIL\_ID VALUES('SWETHA', 'PASS', 'GMAIL', 3);

INSERT INTO EMAIL\_ID VALUES('KUMAR', 'PASS', 'GMAIL', 4);

INSERT INTO EMAIL\_ID (USERNAME, PASSWORD, VENDOR) VALUES('VIJAY', 'PASS', 'GMAIL');

SELECT \* FROM PERSON;

+----+------------+-----------+------+

| ID | FIRST\_NAME | LAST\_NAME | AGE |

+----+------------+-----------+------+

| 1 | AMAN | GUPTA | 22 |

| 2 | MANOHAR | VERMA | 24 |

| 3 | SWETHA | SHARMA | 21 |

| 4 | VIJAY | VAISHNAV | 23 |

| 5 | KUMAR | SINHA | 23 |

+----+------------+-----------+------+

SELECT \* FROM EMAIL\_ID;

+----------+----------+---------+-----------+

| USERNAME | PASSWORD | VENDOR | PERSON\_ID |

+----------+----------+---------+-----------+

| AMAN | PASS | GMAIL | 1 |

| AMAN | PASS | OUTLOOK | 1 |

| AMAN | PASS | HOTMAIL | 1 |

| MANOHAR | PASS | YAHOO | 2 |

| MANOHAR | PASS | GMAIL | 2 |

| SWETHA | PASS | GMAIL | 3 |

| KUMAR | PASS | GMAIL | 4 |

| VIJAY | PASS | GMAIL | NULL |

+----------+----------+---------+-----------+

SELECT \* FROM PERSON WHERE FIRST\_NAME = 'AMAN';

+----+------------+-----------+------+

| ID | FIRST\_NAME | LAST\_NAME | AGE |

+----+------------+-----------+------+

| 1 | AMAN | GUPTA | 22 |

+----+------------+-----------+------+

SELECT \* FROM EMAIL\_ID WHERE USERNAME = 'AMAN';

+----------+----------+---------+-----------+

| USERNAME | PASSWORD | VENDOR | PERSON\_ID |

+----------+----------+---------+-----------+

| AMAN | PASS | GMAIL | 1 |

| AMAN | PASS | OUTLOOK | 1 |

| AMAN | PASS | HOTMAIL | 1 |

+----------+----------+---------+-----------+

SELECT \* FROM EMAIL\_ID WHERE PERSON\_ID = 3;

+----------+----------+--------+-----------+

| USERNAME | PASSWORD | VENDOR | PERSON\_ID |

+----------+----------+--------+-----------+

| SWETHA | PASS | GMAIL | 3 |

+----------+----------+--------+-----------+

SELECT \* FROM EMAIL\_ID WHERE PERSON\_ID =

-> (SELECT ID FROM PERSON WHERE FIRST\_NAME = 'AMAN');

+----------+----------+---------+-----------+

| USERNAME | PASSWORD | VENDOR | PERSON\_ID |

+----------+----------+---------+-----------+

| AMAN | PASS | GMAIL | 1 |

| AMAN | PASS | OUTLOOK | 1 |

| AMAN | PASS | HOTMAIL | 1 |

+----------+----------+---------+-----------+

SELECT \* FROM PERSON WHERE ID =

-> (SELECT PERSON\_ID FROM EMAIL\_ID WHERE USERNAME = 'AMAN');

ERROR 1242 (21000): Subquery returns more than 1 row

SELECT \* FROM PERSON, EMAIL\_ID WHERE PERSON.ID = EMAIL\_ID.PERSON\_ID;

**+----+------------+-----------+------+----------+----------+---------+-----------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | USERNAME | PASSWORD | VENDOR | PERSON\_ID |**

**+----+------------+-----------+------+----------+----------+---------+-----------+**

**| 1 | AMAN | GUPTA | 22 | AMAN | PASS | GMAIL | 1 |**

**| 1 | AMAN | GUPTA | 22 | AMAN | PASS | OUTLOOK | 1 |**

**| 1 | AMAN | GUPTA | 22 | AMAN | PASS | HOTMAIL | 1 |**

**| 2 | MANOHAR | VERMA | 24 | MANOHAR | PASS | YAHOO | 2 |**

**| 2 | MANOHAR | VERMA | 24 | MANOHAR | PASS | GMAIL | 2 |**

**| 3 | SWETHA | SHARMA | 21 | SWETHA | PASS | GMAIL | 3 |**

**| 4 | VIJAY | VAISHNAV | 23 | KUMAR | PASS | GMAIL | 4 |**

**+----+------------+-----------+------+----------+----------+---------+-----------+**

SELECT \* FROM PERSON P, EMAIL\_ID M WHERE P.ID = M.PERSON\_ID;

SELECT \* FROM PERSON P INNER JOIN EMAIL\_ID M ON P.ID = M.PERSON\_ID;

SELECT \* FROM PERSON P LEFT OUTER JOIN EMAIL\_ID M ON P.ID = M.PERSON\_ID;

**+----+------------+-----------+------+----------+----------+---------+-----------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | USERNAME | PASSWORD | VENDOR | PERSON\_ID |**

**+----+------------+-----------+------+----------+----------+---------+-----------+**

**| 1 | AMAN | GUPTA | 22 | AMAN | PASS | GMAIL | 1 |**

**| 1 | AMAN | GUPTA | 22 | AMAN | PASS | OUTLOOK | 1 |**

**| 1 | AMAN | GUPTA | 22 | AMAN | PASS | HOTMAIL | 1 |**

**| 2 | MANOHAR | VERMA | 24 | MANOHAR | PASS | YAHOO | 2 |**

**| 2 | MANOHAR | VERMA | 24 | MANOHAR | PASS | GMAIL | 2 |**

**| 3 | SWETHA | SHARMA | 21 | SWETHA | PASS | GMAIL | 3 |**

**| 4 | VIJAY | VAISHNAV | 23 | KUMAR | PASS | GMAIL | 4 |**

**| 5 | KUMAR | SINHA | 23 | NULL | NULL | NULL | NULL |**

**+----+------------+-----------+------+----------+----------+---------+-----------+**

SELECT \* FROM PERSON P RIGHT OUTER JOIN EMAIL\_ID M ON P.ID = M.PERSON\_ID;

**+------+------------+-----------+------+----------+----------+---------+-----------+**

**| ID | FIRST\_NAME | LAST\_NAME | AGE | USERNAME | PASSWORD | VENDOR | PERSON\_ID |**

**+------+------------+-----------+------+----------+----------+---------+-----------+**

**| 1 | AMAN | GUPTA | 22 | AMAN | PASS | GMAIL | 1 |**

**| 1 | AMAN | GUPTA | 22 | AMAN | PASS | OUTLOOK | 1 |**

**| 1 | AMAN | GUPTA | 22 | AMAN | PASS | HOTMAIL | 1 |**

**| 2 | MANOHAR | VERMA | 24 | MANOHAR | PASS | YAHOO | 2 |**

**| 2 | MANOHAR | VERMA | 24 | MANOHAR | PASS | GMAIL | 2 |**

**| 3 | SWETHA | SHARMA | 21 | SWETHA | PASS | GMAIL | 3 |**

**| 4 | VIJAY | VAISHNAV | 23 | KUMAR | PASS | GMAIL | 4 |**

**| NULL | NULL | NULL | NULL | VIJAY | PASS | GMAIL | NULL |**

**+------+------------+-----------+------+----------+----------+---------+-----------+**

**MANY-TO-MANY**

DROP TABLE IF EXISTS STUDENT;

CREATE TABLE STUDENT(ID INTEGER UNIQUE, FIRST\_NAME VARCHAR(90), LAST\_NAME VARCHAR(90));

DROP TABLE IF EXISTS SKILL;

CREATE TABLE SKILL(ID INTEGER UNIQUE, NAME VARCHAR(90));

INSERT INTO STUDENT VALUES(1, 'AMAN', 'GUPTA');

INSERT INTO STUDENT VALUES(2, 'MANOHAR', 'VERMA');

INSERT INTO STUDENT VALUES(3, 'JAGAN', 'REDDY');

INSERT INTO STUDENT VALUES(4, 'KUMAR', 'SINHA');

INSERT INTO SKILL VALUES(1, 'C');

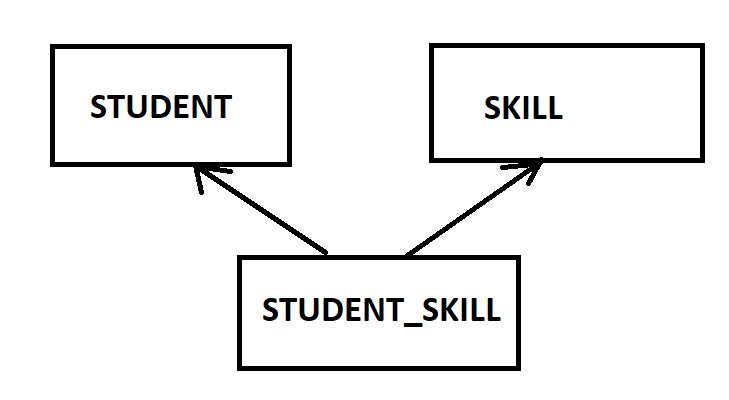
INSERT INTO SKILL VALUES(2, 'C++');

INSERT INTO SKILL VALUES(3, 'JAVA');

INSERT INTO SKILL VALUES(4, 'SQL');

DROP TABLE IF EXTISTS STUDENT\_SKILL;

CREATE TABLE STUDENT\_SKILL(STUDENT\_ID INTEGER, SKILL\_ID INTEGER, CONSTRAINT SS\_FK1 FOREIGN KEY(STUDENT\_ID) REFERENCES STUDENT(ID), CONSTRAINT SS\_FK2 FOREIGN KEY(SKILL\_ID) REFERENCES SKILL(ID));



INSERT INTO STUDENT\_SKILL VALUES(1, 1);

INSERT INTO STUDENT\_SKILL VALUES(1, 2);

INSERT INTO STUDENT\_SKILL VALUES(2, 2);

INSERT INTO STUDENT\_SKILL VALUES(3, 1);

INSERT INTO STUDENT\_SKILL VALUES(3, 4);

SELECT \* FROM STUDENT;

+------+------------+-----------+

| ID | FIRST\_NAME | LAST\_NAME |

+------+------------+-----------+

| 1 | AMAN | GUPTA |

| 2 | MANOHAR | VERMA |

| 3 | JAGAN | REDDY |

| 4 | KUMAR | SINHA |

+------+------------+-----------+

4 rows in set (0.00 sec)

SELECT \* FROM SKILL;

+------+------+

| ID | NAME |

+------+------+

| 1 | C |

| 2 | C++ |

| 3 | JAVA |

| 4 | SQL |

+------+------+

4 rows in set (0.00 sec)

SELECT \* FROM STUDENT\_SKILL;

+------------+----------+

| STUDENT\_ID | SKILL\_ID |

+------------+----------+

| 1 | 1 |

| 1 | 2 |

| 2 | 2 |

| 3 | 1 |

| 3 | 4 |

+------------+----------+

SELECT NAME FROM SKILL WHERE ID IN

(SELECT SKILL\_ID FROM STUDENT\_SKILL WHERE STUDENT\_ID =

(SELECT ID FROM STUDENT WHERE FIRST\_NAME = 'AMAN'));

SELECT FIRST\_NAME FROM STUDENT WHERE ID IN (SELECT STUDENT\_ID FROM STUDENT\_SKILL WHERE SKILL\_ID = (SELECT ID FROM SKILL WHERE NAME = 'C'));

+------------+

| FIRST\_NAME |

+------------+

| AMAN |

| JAGAN |

+------------+

SELECT \* FROM STUDENT, STUDENT\_SKILL, SKILL WHERE

STUDENT.ID = STUDENT\_SKILL.STUDENT\_ID AND

STUDENT\_SKILL.SKILL\_ID = SKILL.ID;

SELECT \* FROM STUDENT S INNER JOIN STUDENT\_SKILL SS

ON S.ID = SS.STUDENT\_ID INNER JOIN SKILL SK ON SS.SKILL\_ID = SK.ID;

*To generate database diagram in mysql(ER diagram)  
-------------------------  
  
Database -> Reverse Engineer -> choose your connection -> login -> hit continue -> select database schema -> hit continue -> continue again -> make sure items are checked -> execute -> again hit continue -> close*