

CSIT Department

# **Bhaktapur Multiple Campus**

Doodhpati, Bhaktapur



**Lab sheet -8**

## **A LAB REPORT OF Database Management System (CSC260)**

**Submitted By:**

Bharati Chaudhary

Roll No:68

**Submitted To**

Madan Nath

## Table Of content

<b>SQL(Structured Query Language)</b> .....	<b>2</b>
<b>SQL DATA TYPES</b> .....	<b>2</b>
• String .....	2
• Integer .....	3
• Date .....	3
• Timestamp .....	3
<b>Constraints in Sql</b> .....	<b>3</b>
• Primary key Constraint .....	3
• Foreign key Constraint .....	3
• Not Null constraints .....	4
<b>SQL Commands</b> .....	<b>4</b>
• DDL(Data Definition Language).....	4
• DML(Data Manipulation Language) .....	5
• DCL(Data Control Language) .....	5
• TCL(Transaction Control Language) .....	6
• DQL(Data Query Language) .....	6
<b>Basic SQL Statements</b> .....	<b>7</b>
Connecting MySQL to terminal .....	7
Showing Databases .....	7
Creating a Database .....	8
Create a table with constraint .....	8
Using a Database .....	8
Inserting data in table.....	8
Selecting data from table .....	9
Selecting columns from table .....	9
Selecting records based on conditions .....	10
Ordering the result set .....	11
Updating table data .....	12
Updating multiple columns .....	13
Deleting Data from tables .....	13
Removing a table from database .....	14
Add Column.....	14

Dropping Columns .....	15
<b>Min and Max</b> .....	<b>16</b>
<b>Like Operations</b> .....	<b>16</b>
<b>AND OR and NOT operators in SQL</b> .....	<b>18</b>
<b>Union</b> .....	<b>18</b>
<b>Parent Child Relationship.....</b>	<b>19</b>
• Primary Key .....	19
• Foreign Key .....	20
<b>SQL Joins:</b> .....	<b>21</b>
• Inner Join .....	21
• Outer Joins .....	22
• Left Join .....	22
• Right Join .....	22
• Full Join .....	22

## SQL (Structured Query Language)

SQL is a standard programming language specifically designed for storing, retrieving, managing or manipulating the data inside a relational database management system (RDBMS). SQL is the most widely-implemented database language and supported by the popular relational database systems, like MySQL, SQL Server, and Oracle.

- Structured Query Language(SQL) is a programming language designed for working with databases.
- It helps to create, manipulate and share the data (specifically data from relational database management systems).
- In other words , SQL is the programming language that helps to execute commands which are used to create and manipulate relational databases.
- SQL is the declarative language mainly concerned with what result you want to obtain.

### SQL DATA TYPES

It specifies the data types that will be inserted in each column of the table.

#### String

- The text format in SQL and the variable in alphanumeric data type.
- Digit , symbol or blank spaces can be also used in string format.
- No mathematical operation can be done using the string.
- Only convey text information.
- eg: address: 'Huge Street 45'. String data type

String data type	Symbol	Storage	Example
Character	char	Fixed	CHAR(5)-> 5 represent that maximum number of symbol allowed to use writing value in this format.
Variable	varchar	variable	VARCHAR(5)-> 5 represent maximum number of the symbol. Required shortest storage if value inserted is shorter than maximum length.
Enumerate	enum		ENUM('M', 'F')->Mysql will show error if we attempt to insert any value different from 'M' or 'F'

## **Integer**

- It is Numeric data type.
- Denoted by the symbol 'int'.
- It is a whole number with no decimal points.
- Example: 5,10,15,1000 etc

## **Date**

- It is used to represent dates in format YYYY-MM-DD.
- Example:'1997-11-24'.

## **Timestamp**

- used or well defined, exact point of time.
- Example: 24th of November 1970 UTC- 1st of July 2020,03:14:07 UTC

## **Constraints in Sql**

- specific rules, or limits, that we define in our tables.
- The role of constraints is to outline the existing relationships between different tables in our database.

### **Primary key Constraint**

- Unique.
- A column(or a set of columns) whose value exists and is unique For every record of the table is called primary key.
- Each table can have one and only one primary key.
- May be composed of a set of columns.
- It can't contain null values.
- It is also called a unique identifier.

### **Foreign key Constraint**

- Foreign key points to a column of another table and, thus, links the two tables.
- Foreign keys identify relationships between tables but not themselves.
- The table containing foreign key is called is called child table which is also called referencing table.
- The table containing primary key is called parent table which is also called referenced table.

## Not Null constraints

- The “not null” restriction is applied through the NOT NULL Constraint
- When you insert values in the table, you cannot leave the respective field empty
- If we leave it empty, MySQL will signal an error

## SQL Commands

- SQL commands are instructions. It is used to communicate with the database. It is also used to perform specific tasks, functions, and queries of data.
- SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permission for users.

## Types of SQL Commands

There are five types of SQL commands: DDL, DML, DCL, TCL, and DQL.

### 1. Data Definition Language (DDL)

- DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.
- All the command of DDL are auto-committed that means it permanently save all the changes in the database.
- Some commands that come under DDL:
  - CREATE: It is used to create a new table in the database.  
Syntax: CREATE TABLE TABLE\_NAME (COLUMN\_NAME DATATYPE[,...]);
  - DROP: It is used to delete both the structure and record stored in the table.  
Syntax: DROP TABLE table\_name;
  - ALTER: It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.  
Syntax: ALTER TABLE table\_name ADD column\_name COLUMN-definition;
  - TRUNCATE: It is used to delete all the rows from the table and free the space containing the table.
  - Syntax: TRUNCATE TABLE table\_name;

## 2. Data Manipulation Language

- DML commands are used to modify the database. It is responsible for all form of changes in the database.
- The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.
- Here are some commands that come under DML:
  - INSERT: The INSERT statement is a SQL query. It is used to insert data into the row of a table.  
Syntax: INSERT INTO TABLE\_NAME(col1, col2, col3,.... col N)  
VALUES (value1, value2, value3, .... valueN);
  - UPDATE: This command is used to update or modify the value of a column in the table.  
Syntax: UPDATE table\_name SET [column\_name1= value1,...  
column\_nameN = valueN] [WHERE CONDITION]
  - DELETE: It is used to remove one or more row from a table.  
Syntax: DELETE FROM table\_name [WHERE condition];

## 3. Data Control Language

- DCL commands are used to grant and take back authority from any database user.
- Here are some commands that come under DCL:
  - Grant: It is used to give user access privileges to a database.  
Example: GRANT SELECT, UPDATE ON MY\_TABLE TO SOME\_USER,  
ANOTHER\_USER;
  - Revoke: It is used to take back permissions from the user.  
Example:  
REVOKE SELECT, UPDATE ON MY\_TABLE FROM USER1, USER2;

#### 4. Transaction Control Language

- TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.
- These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.
- Some commands that come under TCL:

➤ COMMIT: Commit command is used to save all the transactions to the database.

Syntax: COMMIT;

➤ ROLLBACK: Rollback command is used to undo transactions that have not already been saved to the database.

Syntax: ROLLBACK;

➤ SAVEPOINT: It is used to roll the transaction back to a certain point without rolling back the entire transaction.

Syntax: SAVEPOINT SAVEPOINT\_NAME;

#### 5. Data Query Language

- DQL is used to fetch the data from the database.
- It uses only one command:
  - SELECT: This is the same as the projection operation of relational algebra. It is used to select the attribute based on the condition described by WHERE clause.

Syntax: SELECT expressions FROM TABLES WHERE conditions;



# Basic SQL Statements

## Connecting MySQL to terminal:

### Syntax:

- `Mysql -u root -p -h 127.0.0.1`

```
C:\> Command Prompt - mysql -u root -p -h 127.0.0.1
Microsoft Windows [Version 10.0.19043.1766]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>cd/

C:\>cd xampp

C:\xampp>cd mysql

C:\xampp\mysql>cd bin

C:\xampp\mysql\bin>mysql -u root -p -h 127.0.0.1
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.4.21-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]>
```

## Showing Databases:

### Syntax:

- `SHOW DATABASES`

```
MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| college |
| csit    |
| information_schema |
| mysql   |
| office  |
| performance_schema |
| phpmyadmin |
| student |
| test    |
+-----+
9 rows in set (0.033 sec)
```

## **Creating a Database**

Before doing anything with the data we must need to create a database first. The SQL CREATE DATABASE statement is used to create a database.

### **Syntax:**

The basic syntax for creating a database can be given with  
CREATE DATABASE database\_name;

```
MariaDB [(none)]> create database user;
Query OK, 1 row affected (0.002 sec)

MariaDB [(none)]> use user;
Database changed
MariaDB [user]>
```

## **Using a Database**

Before creating tables inside the database, we must first use the database. The SQL USE statement is used to select a particular database.

### **Syntax :**

use database\_name;

## **Create a table with constraint:**

### **Syntax:**

CREATE table table\_name(column1 datatype,column2 datatype,column3 datatype)

```
MariaDB [student]> create table student(sid int,name varchar(20),dob date,address varchar(20));
Query OK, 0 rows affected (0.038 sec)

MariaDB [student]> show tables;
+-----+
| Tables_in_student |
+-----+
| student            |
+-----+
1 row in set (0.001 sec)
```

## **Inserting data in table**

The INSERT INTO statement is used to insert new rows in a database table.

### **Syntax:**

The basic syntax for inserting data into a table can be given with:

INSERT INTO table\_name (column1,column2,...) VALUES (value1,value2,...);

Here the column1, column2,..., etc. represents the name of the table columns, whereas the value1, value2,..., and so on represent the corresponding values for these columns.

```

MariaDB [student]> insert into exam(examno,subject,stuid)
-> values
-> (1,"TOC",1),
-> (2,"CN",2),
-> (3,"DBMS",3),
-> (4,"OS",4),
-> (5,"AI",5);
Query OK, 5 rows affected (0.006 sec)
Records: 5  Duplicates: 0  Warnings: 0

MariaDB [student]> select *from exam;
+-----+-----+-----+
| examno | subject | stuid |
+-----+-----+-----+
|      1 | TOC    |      1 |
|      2 | CN     |      2 |
|      3 | DBMS   |      3 |
|      4 | OS     |      4 |
|      5 | AI     |      5 |
+-----+-----+-----+
5 rows in set (0.003 sec)

```

### **Selecting data from table**

The SELECT statement is used to select or retrieve the data from one or more tables. You can use this statement to retrieve all the rows from a table in one go, as well as to retrieve only those rows that satisfy a certain condition or a combination of conditions.

#### **Syntax:**

The basic syntax for selecting the data from a table can be given with:

SELECT column1\_name, column2\_name, columnN\_name FROM table\_name;

```

MariaDB [csit]> select * from department;
+-----+-----+
| dept_id | subject |
+-----+-----+
|      1 | BCA    |
|      2 | BBA    |
+-----+-----+
2 rows in set (0.072 sec)

```

### Selecting columns from table:

If you don't require all the data, you can select specific columns.

```
MariaDB [student]> select *from student;
```

sid	name	dob	address
1	Lasta	2000-09-28	Sunakothi
2	Anjali	2000-01-30	Balkumari
3	Aseem	2000-01-01	Lokanthali
4	Sameer	2001-06-28	Patan
5	Bharati	2000-01-01	Gwarko

```
5 rows in set (0.034 sec)
```

```
MariaDB [student]> select sid,name,address from student;
```

sid	name	address
1	Lasta	Sunakothi
2	Anjali	Balkumari
3	Aseem	Lokanthali
4	Sameer	Patan
5	Bharati	Gwarko

```
5 rows in set (0.000 sec)
```

### Selecting records based on conditions:

Previously we've learnt how to fetch all the records from a table or table columns. But, in real world scenarios we generally need to select, update or delete only those records which fulfill certain conditions like users who belong to a certain age group, or country, etc.

The WHERE clause is used with the SELECT, UPDATE, and DELETE.

#### **Syntax:**

The WHERE clause is used with the SELECT statement to extract only those records that fulfill specified conditions. The basic syntax can be given with:

```
SELECT column_list FROM table_name WHERE condition;
```

Here, column\_list are the names of columns/fields like name, age, country etc. of a database table whose values you want to fetch. However, if you want to fetch the values of all the columns available in a table, you can use the following

#### **syntax:**

```
SELECT * FROM table_name WHERE condition;
```

```

MariaDB [student]> select *from student
-> where
-> address="Sunakothi";
+-----+-----+-----+-----+
| sid | name | dob | address |
+-----+-----+-----+-----+
| 1 | Lasta | 2000-09-28 | Sunakothi |
+-----+-----+-----+-----+
1 row in set (0.013 sec)

MariaDB [student]> select *from student
-> where
-> sid=4;
+-----+-----+-----+-----+
| sid | name | dob | address |
+-----+-----+-----+-----+
| 4 | Sameer | 2001-06-28 | Patan |
+-----+-----+-----+-----+
1 row in set (0.006 sec)

```

### **Ordering the result set:**

Generally when you use the SELECT statement to fetch data from a table, the rows in the result set are not in any particular order. If you want your result set in a particular order, you can specify the ORDER BY clause at the end of the statement which tells the server how to sort the data returned by the query. The default sorting order is ascending.

#### **Syntax:**

The ORDER BY clause is used to sort the data returned by a query in ascending or descending order. The basic syntax of this clause can be given with:

SELECT column\_list FROM table\_name ORDER BY column\_name ASC|DESC;

Here, column\_list are the names of columns/fields like name, age, country etc. of a database table whose values you want to fetch, whereas the column\_name is the name of the column you want to sort.

```

MariaDB [student]> select *from student order by name;
+-----+-----+-----+-----+
| sid | name | dob | address |
+-----+-----+-----+-----+
| 2 | Anjali | 2000-01-30 | Balkumari |
| 3 | Aseem | 2000-01-01 | Lokanthali |
| 5 | Bharati | 2000-01-01 | Gwarko |
| 1 | Lasta | 2000-09-28 | Sunakothi |
| 4 | Sameer | 2001-06-28 | Patan |
+-----+-----+-----+-----+
5 rows in set (0.001 sec)

```

### **Updating table data:**

The UPDATE statement is used to update existing data in a table.

#### **Syntax:**

```
UPDATE table_name SET column1_name = value1, column2_name = value2,...  
WHERE condition;
```

Here, column1\_name, column2\_name,... are the names of the columns or fields of a database table whose values you want to update. You can also combine multiple conditions using the AND or OR operators.

```
MariaDB [student]> update report_card  
-> set score=44  
-> where reportno=104;  
Query OK, 1 row affected (0.004 sec)  
Rows matched: 1  Changed: 1  Warnings: 0  
  
MariaDB [student]> select *from report_card;  
+-----+-----+-----+-----+-----+  
| reportno | subject | name   | score | exam_no |  
+-----+-----+-----+-----+-----+  
|      101 | TOC     | Lasta  | 57    | 1       |  
|      102 | CN      | Anjali | 55    | 2       |  
|      103 | DBMS    | Aseem  | 40    | 3       |  
|      104 | OS      | Sameer | 44    | 4       |  
|      105 | AI      | Bharati | 50    | 5       |  
+-----+-----+-----+-----+-----+  
5 rows in set (0.000 sec)
```

### **Updating multiple columns:**

Similarly, you can update multiple columns using a list of comma-separated column names and value pairs.



```

MariaDB [student]> select *from student order by name;
+-----+-----+-----+-----+
| sid | name   | dob       | address |
+-----+-----+-----+-----+
| 2   | Anjali | 2000-01-30 | Balkumari |
| 3   | Aseem  | 2000-01-01 | Lokanthali |
| 5   | Bharati | 2000-01-01 | Gwarko |
| 1   | Lasta  | 2000-09-28 | Sunakothi |
| 4   | Sameer | 2001-06-28 | Patan |
+-----+-----+-----+-----+
5 rows in set (0.001 sec)

MariaDB [student]> update student
-> set address="Bajrabarahi"
-> ,
-> name="Samir"
-> where sid=4;
Query OK, 1 row affected (0.009 sec)
Rows matched: 1 Changed: 1 Warnings: 0

MariaDB [student]> select *from student;
+-----+-----+-----+-----+
| sid | name   | dob       | address |
+-----+-----+-----+-----+
| 1   | Lasta  | 2000-09-28 | Sunakothi |
| 2   | Anjali | 2000-01-30 | Balkumari |
| 3   | Aseem  | 2000-01-01 | Lokanthali |
| 4   | Samir  | 2001-06-28 | Bajrabarahi |
| 5   | Bharati | 2000-01-01 | Gwarko |
+-----+-----+-----+-----+
5 rows in set (0.000 sec)

```

### **Deleting Data from tables:**

Just as you insert records into a table with the INSERT statement, you can delete records from a table as well with the DELETE statement.

#### **Syntax:**

The DELETE statement is used to remove one or more rows from a table. DELETE FROM table\_name WHERE condition;

```

MariaDB [user]> select *from customer;
+-----+-----+-----+
| id | name | address |
+-----+-----+-----+
| 1 | Ram | Lalitpur |
| 2 | Sita | Bhaktapur |
| 3 | Hari | Kathmandu |
+-----+-----+-----+
3 rows in set (0.000 sec)

MariaDB [user]> delete from customer
-> where name="Hari";
Query OK, 1 row affected (0.005 sec)

MariaDB [user]> select *from customer;
+-----+-----+-----+
| id | name | address |
+-----+-----+-----+
| 1 | Ram | Lalitpur |
| 2 | Sita | Bhaktapur |
+-----+-----+-----+
2 rows in set (0.000 sec)

```

## **Removing a table from database:**

You can use the DROP TABLE statement to easily delete the database tables that you no longer need. The DROP TABLE statement permanently erases all data from the table, as well as the metadata that defines the table in the data dictionary.

### **Syntax:**

The DROP TABLE removes one or more tables. The syntax can be given with:

DROP TABLE table1\_name, table2\_name, ...;

Here, table1\_name, table2\_name, ... are the names of the tables that you want to delete.

```
MariaDB [user]> select *from customer;
+-----+-----+-----+
| id  | name | address |
+-----+-----+-----+
| 1   | Ram  | Lalitpur |
| 2   | Sita | Bhaktapur |
+-----+-----+-----+
2 rows in set (0.000 sec)

MariaDB [user]> drop table customer;
Query OK, 0 rows affected (0.013 sec)

MariaDB [user]> show tables;
Empty set (0.001 sec)
```

## **Add Column**

- Statement used: ALTER TABLE ADD
- Syntax: ALTER TABLE table\_name ADD column\_name datatype;



```

MariaDB [student]> select *from student;
+-----+-----+-----+-----+
| sid | name  | dob      | address |
+-----+-----+-----+-----+
| 1   | Lasta | 2000-09-28 | Sunakothi |
| 2   | Anjali | 2000-01-30 | Balkumari |
| 3   | Aseem | 2000-01-01 | Lokanthali |
| 4   | Samir | 2001-06-28 | Bajrabarahi |
| 5   | Bharati | 2000-01-01 | Gwarko |
+-----+-----+-----+-----+
5 rows in set (0.000 sec)

MariaDB [student]> alter table student add contact int;
Query OK, 0 rows affected (0.017 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [student]> select *from student;
+-----+-----+-----+-----+-----+
| sid | name  | dob      | address | contact |
+-----+-----+-----+-----+-----+
| 1   | Lasta | 2000-09-28 | Sunakothi | NULL |
| 2   | Anjali | 2000-01-30 | Balkumari | NULL |
| 3   | Aseem | 2000-01-01 | Lokanthali | NULL |
| 4   | Samir | 2001-06-28 | Bajrabarahi | NULL |
| 5   | Bharati | 2000-01-01 | Gwarko | NULL |
+-----+-----+-----+-----+-----+
5 rows in set (0.000 sec)

```

### **Dropping Columns:**

- Statement used : ALTER TABLE DROP
- Syntax: ALTER TABLE TABLE\_NAME DROP COLUMN COLUMN\_NAME

```

MariaDB [student]> alter table student drop column contact;
Query OK, 0 rows affected (0.014 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [student]> select *from student;
+-----+-----+-----+-----+
| sid | name  | dob      | address |
+-----+-----+-----+-----+
| 1   | Lasta | 2000-09-28 | Sunakothi |
| 2   | Anjali | 2000-01-30 | Balkumari |
| 3   | Aseem | 2000-01-01 | Lokanthali |
| 4   | Samir | 2001-06-28 | Bajrabarahi |
| 5   | Bharati | 2000-01-01 | Gwarko |
+-----+-----+-----+-----+
5 rows in set (0.000 sec)

```

## Min and Max:

The MIN() function returns the smallest value of the selected column.

The MAX() function returns the largest value of the selected column.

### **Syntax:**

MIN() Syntax SELECT MIN(column\_name) FROM table\_name WHERE condition;

MAX() syntax SELECT MAX(column\_name) FROM table\_name WHERE condition;

```
MariaDB [student]> select *from report_card;
+-----+-----+-----+-----+-----+
| reportno | subject | name   | score | exam_no |
+-----+-----+-----+-----+-----+
| 101      | TOC     | Lasta  | 57    | 1        |
| 102      | CN      | Anjali | 55    | 2        |
| 103      | DBMS    | Aseem  | 40    | 3        |
| 104      | OS      | Sameer | 44    | 4        |
| 105      | AI      | Bharati | 50    | 5        |
+-----+-----+-----+-----+-----+
5 rows in set (0.010 sec)

MariaDB [student]> select min(score) from report_card;
+-----+
| min(score) |
+-----+
| 40          |
+-----+
1 row in set (0.024 sec)

MariaDB [student]> select max(score) from report_card;
+-----+
| max(score) |
+-----+
| 57          |
+-----+
1 row in set (0.000 sec)
```

## Like Operations

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the LIKE operator:

1. The percent sign (%) represents zero, one, or multiple characters
2. The underscore sign (\_) represents one, single character

### **LIKE Operator**

### **Description**

WHERE CustomerName LIKE 'a%'

Finds any values that start with "a"

WHERE CustomerName LIKE '%a'

Finds any values that end with "a"

WHERE CustomerName LIKE '%or%'

Finds any values that have "or" in any position

WHERE CustomerName LIKE '\_r%'

Finds any values that have "r" in the second position

WHERE CustomerName LIKE 'a\_%'

Finds any values that start with "a" and are at least 2 characters in length

WHERE CustomerName LIKE 'a\_\_%'

Finds any values that start with "a" and are at least 3 characters in length

WHERE ContactName LIKE 'a%o'

Finds any values that start with "a" and ends with "o"

### Syntax:

SELECT column1, column2, ...

FROM table\_name

WHERE columnN LIKE pattern;

```
MariaDB [student]> select *from student
-> where name like "A_%";
+-----+
| sid | name   | dob       | address   |
+-----+
| 2   | Anjali | 2000-01-30 | Balkumari |
| 3   | Aseem  | 2000-01-01 | Lokanthali |
+-----+
2 rows in set (0.005 sec)
```

```
MariaDB [student]> select *from student
-> where name like "b%";
+-----+
| sid | name   | dob       | address   |
+-----+
| 5   | Bharati | 2000-01-01 | Gwarko    |
+-----+
1 row in set (0.001 sec)
```

```
MariaDB [student]> select *from student
-> where name like "%b";
Empty set (0.000 sec)
```

```
MariaDB [student]> select *from student
-> where name like "_a";
Empty set (0.000 sec)
```

```
MariaDB [student]> select *from student
-> where name like "_a%";
+-----+
| sid | name   | dob       | address   |
+-----+
| 1   | Lasta  | 2000-09-28 | Sunakothi |
| 4   | Samir  | 2001-06-28 | Bajrabarahi |
+-----+
2 rows in set (0.001 sec)
```

```
MariaDB [student]> select *from student
-> where name like "a%m";
+-----+
| sid | name   | dob       | address   |
+-----+
| 3   | Aseem  | 2000-01-01 | Lokanthali |
+-----+
```

## AND OR and NOT operators in SQL

The WHERE clause is combined with AND ,OR and NOT operators creating a condition.

### **Syntax:**

SELECT \*FROM table\_name WHERE condition1 AND condition2;

```
MariaDB [student]> select *from student
-> where name="Bharati" and sid=5;
+-----+-----+-----+-----+
| sid | name   | dob       | address |
+-----+-----+-----+-----+
| 5   | Bharati | 2000-01-01 | Gwarko  |
+-----+-----+-----+-----+
1 row in set (0.007 sec)

MariaDB [student]> select *from student
-> where not address="Lokanthali";
+-----+-----+-----+-----+
| sid | name   | dob       | address |
+-----+-----+-----+-----+
| 1   | Lasta  | 2000-09-28 | Sunakothi |
| 2   | Anjali | 2000-01-30 | Balkumari |
| 4   | Samir  | 2001-06-28 | Bajrabarahi |
| 5   | Bharati | 2000-01-01 | Gwarko  |
+-----+-----+-----+-----+
4 rows in set (0.000 sec)

MariaDB [student]> select *from student
-> where address="Sunakothi" or name="Anjali";
+-----+-----+-----+-----+
| sid | name   | dob       | address |
+-----+-----+-----+-----+
| 1   | Lasta  | 2000-09-28 | Sunakothi |
| 2   | Anjali | 2000-01-30 | Balkumari |
+-----+-----+-----+-----+
2 rows in set (0.000 sec)
```

## Union:

The UNION operator is used to combine the result-set of two or more SELECT statements.

- Every SELECT statement within UNION must have the same number of columns
- The columns must also have similar data types
- The columns in every SELECT statement must also be in the same order

### **Syntax:**

SELECT column\_name(s) FROM table1 UNION SELECT column\_name(s) FROM table2;

```

MariaDB [student]> select name from student union select subject from exam;
+-----+
| name |
+-----+
| Lasta |
| Anjali |
| Aseem |
| Samir |
| Bharati |
| TOC |
| CN |
| DBMS |
| OS |
| AI |
+-----+
10 rows in set (0.003 sec)

```

## Parent Child Relationship

A parent child relationship between two tables can be created only when there is primary key in one table and that key is used as foreign key in another table.

### Primary Key:

A primary key is a special relational database table column (or combination of columns) designated to uniquely identify each table record.

A primary key's main features are:

- It must contain a unique value for each row of data.
- It cannot contain null values.
- Every row must have a primary key value.

### **Syntax:**

Alter table table\_name add primary key(column\_name)

```

MariaDB [student]> alter table student
-> add primary key(sid);
Query OK, 0 rows affected (0.057 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [student]> alter table exam
-> add primary key(examno);
Query OK, 0 rows affected (0.051 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [student]> alter table report_card
-> add primary key(reportno);
Query OK, 0 rows affected (0.045 sec)
Records: 0 Duplicates: 0 Warnings: 0

```



## **Foreign Key:**

A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.

The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

### **Syntax:**

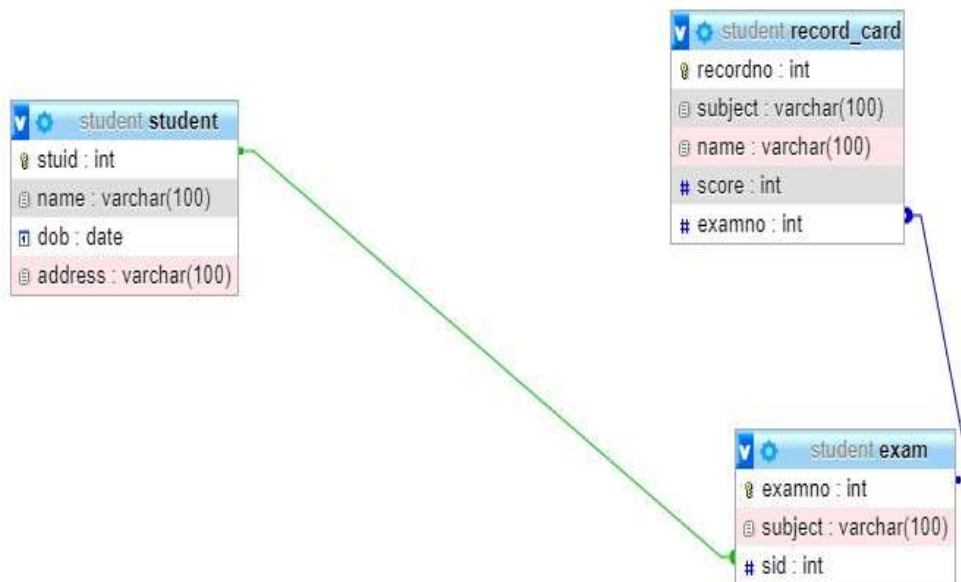
Alter table table\_name add foreign key(column\_name) references parent\_table(primarykey)

```
MariaDB [student]> alter table exam
-> add foreign key(stuid) references student(sid);
Query OK, 5 rows affected (0.061 sec)
Records: 5 Duplicates: 0 Warnings: 0

MariaDB [student]> alter table report_card
-> add foreign key(examno) references exam(examno);
ERROR 1072 (42000): Key column 'examno' doesn't exist in table
MariaDB [student]> alter table report_card
-> add foreign key(exam_no) references exam(examno);
Query OK, 5 rows affected (0.053 sec)
Records: 5 Duplicates: 0 Warnings: 0
```

Here student is parent table for exam and exam is parent table for report\_card.

## **Final design:**



## SQL Joins:

All the queries we've seen so far have been concentrated on a single table. But in real life situations you often need to query two or more tables at time and bring a combined result set. This is technically referred to as a join, since it involves joining different tables, based on a common field between them (the foreign key) to create new views of the data.

### Types of Joins:

When you join tables, the type of join that you create in your query affects the rows that appear in the result set. You can create the following types of joins:

1. Inner Join
2. Outer Join

### Inner Join:

The INNER JOIN is the most common type of join. It returns only those rows that have a match in both joined tables.

Now, let's say you need to retrieve the id, name, hire date, and the department name of only those employees who are assigned to a particular department. Because, in real-life scenarios there may be some employees who are not yet assigned to a department, like the fifth employee "Martin Blank" in our employees table. But the question here is, how to retrieve the data from both the tables in the same SQL query? Well, let's find out.

If you see the employees table, you'll notice that it has a column named dept\_id which holds the ID of the department to which each employee is assigned i.e. in technical terms, the employees table's dept\_id column is the foreign key to the departments table, and therefore we will use this column as a bridge between these two tables.

```
mariaDB [student]> select *from student as s
-> inner join exam as e
-> on s.sid=e.stuid;
```

sid	name	dob	address	examno	subject	stuid
1	Lasta	2000-09-28	Sunakothi	1	TOC	1
2	Anjali	2000-01-30	Balkumari	2	CN	2
3	Aseem	2000-01-01	Lokanthali	3	DBMS	3
4	Sameer	2001-06-28	Patan	4	OS	4
5	Bharati	2000-01-01	Gwarko	5	AI	5

```
5 rows in set (0.001 sec)
```

## **Outer Joins:**

### **Left Join:**

A LEFT JOIN statement returns all rows from the left table along with the rows from the right table for which the join condition is met. Left join is a type of outer join that's why it is also referred to as left outer join.

```
MariaDB [student]> select name from student left join exam on student.name=exam.subject;
+-----+
| name |
+-----+
| Lasta |
| Anjali |
| Aseem |
| Samir |
| Bharati |
+-----+
5 rows in set (0.006 sec)
```

### **Right Join**

The RIGHT JOIN is the exact opposite of the LEFT JOIN. It returns all rows from the right table along with the rows from the left table for which the join condition is met.

Right join is a type of outer join that's why it is also referred to as right outer join.

```
MariaDB [student]> select examno from exam right join student on exam.examno=student.sid;
+-----+
| examno |
+-----+
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| NULL |
+-----+
5 rows in set (0.000 sec)
```

### **Full Join:**

The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.

### **Syntax:**

- `SELECT * FROM t1 LEFT JOIN t2 ON t1.id = t2.id UNION SELECT * FROM t1 RIGHT JOIN t2 ON t1.id = t2.id`


















```

MariaDB [student]> select *from student left join exam on student.sid=exam.examno
-> union select *from student right join exam on student.sid=exam.examno;
+-----+-----+-----+-----+-----+-----+
| sid | name | dob | address | examno | subject | stuid |
+-----+-----+-----+-----+-----+-----+
| 1 | Lasta | 2000-09-28 | Sunakothi | 1 | TOC | 1 |
| 2 | Anjali | 2000-01-30 | Balkumari | 2 | CN | 2 |
| 3 | Aseem | 2000-01-01 | Lokanthali | 3 | DBMS | 3 |
| 4 | Samir | 2001-06-28 | Bajrabarahi | 4 | OS | 4 |
| 5 | Bharati | 2000-01-01 | Gwarko | 5 | AI | 5 |
| 6 | Aliza | 2001-01-01 | Lele | NULL | NULL | NULL |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.004 sec)











```

## Final Databases:

### Student Table:

	sid	name	dob	address
  	1	Lasta	2000-09-28	Sunakothi
  	2	Anjali	2000-01-30	Balkumari
  	3	Aseem	2000-01-01	Lokanthali
  	4	Sameer	2001-06-28	Patan
  	5	Bharati	2000-01-01	Gwarko

### Exam Table:

	examno	subject	stuid
<input type="checkbox"/>   	1	TOC	1
<input type="checkbox"/>   	2	CN	2
<input type="checkbox"/>   	3	DBMS	3
<input type="checkbox"/>   	4	OS	4
<input type="checkbox"/>   	5	AI	5

### Final:

	reportno	subject	name	score	exam_no
  	101	TOC	Lasta	57	1
  	102	CN	Anjali	55	2
  	103	DBMS	Aseem	40	3
  	104	OS	Sameer	44	4
  	105	AI	Bharati	50	5