Chapter 9
Structured
Query
Language

New syllabus 2022-23

Computer Science

Class XII (As per CBSE Board)

SQL is an acronym of Structured Query Language.It is a standard language developed and used for accessing and modifying relational databases.

The SQL language was originally developed at the IBM research laboratory in San José, in connection with a project developing a prototype for a relational database management system called System R in the early 70s.

SQL is being used by many database management systems. Some of them are:

- MySQL
- PostgreSQL
- Oracle
- SQLite
- Microsoft SQL Server

Advantages of using SQL

❖ Interactive Language-This language can be used for communicating with the databases and receive answers to the complex questions in seconds.

❖ Multiple data views-The users can make different views of database structure and databases for the different users.

❖ Portability-SQL can be used in the program in PCs, servers, laptops, and even some of the mobile phones and even on different dbms softwares

❖ No coding needed-It is very easy to manage the database systems without any need to write the substantial amount of code by using the standard SQL.

❖ Well defined standards-Long established are used by the SQL databases that is being used by ISO and ANSI. There are no standards adhered by the non-SQL databases.

MySQL is currently the most popular open source database software. It is a multi-user, multithreaded database management system. MySQL is especially popular on the web. It is one of the parts of the very popular LAMP platform. Linux, Apache, MySQL and PHP or WIMP platform Windows,Apache,MySQL and PHP.

MySQL AB was founded by Michael Widenius (Monty), David ~~~~~~~~~~~ Larsson in Sweden in year 1995.

MySQL Features

Open Source & Free of Cost:

It is Open Source and available at free of cost.

❑ Portability:

Small enough in size to instal and run it on any types of Hardware and OS like Linux,MS Windows or Mac etc.

❑ Security :

Its Databases are secured & protected with password.

❑ Connectivity

Various APIs are developed to connect it with many programming languages.

❑ Query Language

It supports SQL (Structured Query Language) for handling database

Types of SQL Commands

❑ DDL (Data Definition Language)
   To create database and table structure-commands like CREATE , ALTER , DROP etc.

❑ DML (Data Manipulation Language)
   Record/rows related operations.commands like SELECT...., INSERT..., DELETE..., UPDATE.... etc.

❑ DCL (Data Control Language)
   used to manipulate permissions or access rights to the tables. commands like GRANT , REVOKE etc.

❑ Transactional control Language.
   Used to control the transactions.commands like COMMIT, ROLLBACK, SAVEPOINT etc.

Data type in MySQL

☐ **Numeric Data Types:**

- **INTEGER or INT** – up to 11 digit number without decimal.
- **SMALLINT** – up to 5 digit number without decimal.
- **FLOAT (M,D) or DECIMAL(M,D) or NUMERIC(M,D)**
  Stores Real numbers upto **M** digit length (including .) with **D** decimal places.
  e.g. Float (10,2) can store 1234567.89

☐ **Date & Time Data Types:**

- **DATE** - Stores date in YYYY-MM-DD format.
- **TIME** - Stores time in HH:MM:SS format.

☐ **String or Text Data Type:**

- **CHAR(Size)**
  A fixed length string up to 255 characters. (default is 1)
- **VARCHAR(Size)**
  A variable length string up to 255 characters.

Char, **Varchar**, **Date** and **Time** values should be enclosed with single (` `) or double ("") quotes in MySQL. varchar is used in MySQL and varchar2 is used in Oracle.

Database Commands in MySql

Getting listings of available databases

mysql> SHOW DATABASES;

Creating a database-

mysql> CREATE database myschool;

Deleting a database mysql> DROP database <databasename>;

to remove table mysql> drop table <tablename>;

After database creation we can open the database using USE command

mysql> USE myschool;

To show list of tables in opened database

mysql> SHOW TABLES;

Creating a table in the database is achieved with CREATE table statement.

mysql> CREATE TABLE student (lastname varchar(15),firstname varchar(15), city varchar(20), class char(2));

The command DESCRIBE is used to view the structure of a table.

mysql> DESCRIBE student;

Database Commands in MySql

To insert new rows into an existing table use the INSERT command

mysql>INSERT INTO student values('dwivedi','freya','Udaipur','4');

We can insert record with specific column only

mysql>INSERT INTO student(lastname,firstname,city) values('dwivedi','Mohak','Udaipur',);

With the SELECT command we can retrieve previously inserted rows:

A general form of SELECT is:

SELECT what to select(field name) FROM table(s)

WHERE condition that the data must satisfy;

• Comparison operators are: < ; <= ; = ; != or <> ; >= ; >
• Logical operators are: AND ; OR ; NOT
• Comparison operator for special value NULL: IS

mysql> SELECT * FROM student;

Database Commands in MySql

Selecting rows by using the WHERE clause in the SELECT command

mysql> SELECT * FROM student WHERE class="4";

Selecting specific columns(Projection) by listing their names

mysql> SELECT first_name, class FROM student;

Selecting rows with null values in specific column

mysql> SELECT * FROM Student WHERE City IS NULL ;

❖ BETWEEN- to access data in specified range

mysql> SELECT * FROM Student WHERE class between 4 and 6;

❖ IN- operator allows us to easily test if the expression in the list of values.

mysql> SELECT * FROM Student WHERE class in (4,5,6);

Database Commands in MySql

❖ Pattern Matching – LIKE Operator

A string pattern can be used in SQL using the following wild card

❑ % Represents a substring in any length

❑ _ Represents a single character

Example:

'A%' represents any string starting with 'A' character.

'_ _A' represents any 3 character string ending with 'A'.

'_B%' represents any string having second character 'B'

'_ _ _' represents any 3 letter string.

A pattern is case sensitive and can be used with LIKE operator.

mysql> SELECT * FROM Student WHERE Name LIKE 'A%';

mysql> SELECT * FROM Student WHERE Name LIKE '%Singh%';

mysql> SELECT Name, City FROM Student WHERE Class>=8 AND Name LIKE '%Kumar%' ;

Database Commands in MySql

mysql> SELECT * FROM Student ORDER BY class;

To get descending order use DESC key word.

mysql> SELECT * FROM Student ORDER BY class DESC;

To display data after removal of duplicate values from specific column.

mysql> select distinct class from student;

Deleting selected rows from a table using the DELETE command

mysql> DELETE FROM student WHERE firstname="amar";

To modify or update entries in the table use the UPDATE command

mysql> UPDATE student SET class="V" WHERE firstname="freya";

Database Commands in MySql

Creating Table with Constraints

The following constraints are commonly used in SQL:

NOT NULL -It Ensures that a column cannot have a NULL value

UNIQUE - It Ensures that all values in a column are different

PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table

FOREIGN KEY - It Uniquely identifies a row/record in another table

CHECK - It Ensures that all values in a column satisfies a specific condition

DEFAULT - It Sets a default value for a column when no value is specified

INDEX - It is Used to create and retrieve data from the database very quickly

Database Commands in MySql

Creating Table with Constraints

mysql> CREATE TABLE Persons (
    ID int NOT NULL PRIMARY KEY,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255) DEFAULT 'Jaipur',
    CONSTRAINT CHK_Person CHECK (Age>=18)
);
mysql> CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    FOREIGN KEY (PersonID) REFERENCES Persons(ID)   );

Database Commands in MySql

Altering Table

The SQL ALTER TABLE command is used to add, delete or modify columns in an existing table. You should also use the ALTER TABLE command to add and drop various constraints on an existing table.

Syntax

The basic syntax of an ALTER TABLE command to add a New Column in an existing table is as follows.

ALTER TABLE table_name ADD column_name datatype;

The basic syntax of an ALTER TABLE command to DROP COLUMN in an existing table is as follows.

ALTER TABLE table_name DROP COLUMN column_name;

The basic syntax of an ALTER TABLE command to change the DATA TYPE of a column in a table is as follows.

ALTER TABLE table_name MODIFY COLUMN column_name

Database Commands in MySql
Altering Table

The basic syntax of an ALTER TABLE command to add a NOT NULL constraint to a column in a table is as follows.

ALTER TABLE table_name MODIFY column_name datatype NOT NULL;

The basic syntax of ALTER TABLE to ADD UNIQUE CONSTRAINT to a table is as follows.

ALTER TABLE table_name
ADD CONSTRAINT MyUniqueConstraint UNIQUE(column1, column2...);

The basic syntax of an ALTER TABLE command to ADD CHECK CONSTRAINT to a table is as follows.

ALTER TABLE table_name
ADD CONSTRAINT MyUniqueConstraint CHECK (CONDITION);

The basic syntax of an ALTER TABLE command to ADD PRIMARY KEY constraint to a table is as follows.

ALTER TABLE table_name
ADD CONSTRAINT MyPrimaryKey PRIMARY KEY (column1, column2...);

The basic syntax of an ALTER TABLE command to DROP CONSTRAINT from a table is as follows.

ALTER TABLE table_name
DROP CONSTRAINT MyUniqueConstraint;

Database Commands in MySql
Altering Table

ALTER TABLE table_name
DROP INDEX MyUniqueConstraint;

The basic syntax of an ALTER TABLE command to DROP PRIMARY KEY constraint from a table is as follows.

ALTER TABLE table_name
DROP CONSTRAINT MyPrimaryKey;

If we are using MySQL, the code is as follows –

ALTER TABLE table_name
DROP PRIMARY KEY;

MySQL Order By clause is used to sort the table data in either Ascending order or Descending order. By default, data is not inserted into Tables in any order unless we have an index.

So, If we want to retrieve the data in any particular order, we have to sort it by using MySQL Order By statement.

Syntax:-SELECT Column_Names

FROM Table_Name

ORDER BY {Column1}[ASC | DESC] {Column2}[ASC | DESC]

MySQL Order by– e.g.

Suppose we are having student table with following data.

```
mysql> select * from student;
+--------+--------+-------+-------+
| rollno | name   | class | marks |
+--------+--------+-------+-------+
|      1 | freya  |    10 |    88 |
|      2 | mohak  |     1 |    99 |
|      3 | vishal |    10 |    84 |
|      4 | vimal  |    10 |    82 |
|      5 | anil   |     2 |    82 |
+--------+--------+-------+-------+
```

Now we write the query – select * from student order by class;

```
mysql> select * from student order by class;
+--------+--------+-------+-------+
| rollno | name   | class | marks |
+--------+--------+-------+-------+
|      2 | mohak  |     1 |    99 |
|      5 | anil   |     2 |    82 |
|      1 | freya  |    10 |    88 |
|      3 | vishal |    10 |    84 |
|      4 | vimal  |    10 |    82 |
+--------+--------+-------+-------+
```

Query result will be in ascending order of class.If we not specify asc/desc in query then ascending clause is applied by default

MySQL Order by– e.g.

Suppose we are having student table with following data.

```
mysql> select * from student;
+--------+--------+-------+-------+
| rollno | name   | class | marks |
+--------+--------+-------+-------+
|      1 | freya  |    10 |    88 |
|      2 | mohak  |     1 |    99 |
|      3 | vishal |    10 |    84 |
|      4 | vimal  |    10 |    82 |
|      5 | anil   |     2 |    82 |
```

Now we write the query – select * from student order by class desc;

```
mysql> select * from student order by class desc;
+--------+--------+-------+-------+
| rollno | name   | class | marks |
+--------+--------+-------+-------+
|      1 | freya  |    10 |    88 |
|      3 | vishal |    10 |    84 |
|      4 | vimal  |    10 |    82 |
|      5 | anil   |     2 |    82 |
|      2 | mohak  |     1 |    99 |
```

Query result will be in descending order of class

MySQL Order by – e.g.

Suppose we are having student table with following data.

```
mysql> select * from student;
+--------+--------+--------+--------+
| rollno | name   | class  | marks  |
+--------+--------+--------+--------+
|      1 | freya  |     10 |     88 |
|      2 | mohak  |      1 |     99 |
|      3 | vishal |     10 |     84 |
|      4 | vimal  |     10 |     82 |
|      5 | anil   |      2 |     82 |
```

Now we write query–select * from student order by class asc, marks asc;

```
mysql> select * from student order by class asc,marks asc;
+--------+--------+-------+-------+
| rollno | name   | class | marks |
+--------+--------+-------+-------+
|      2 | mohak  |     1 |    99 |
|      5 | anil   |     2 |    82 |
|      4 | vimal  |    10 |    82 |
|      3 | vishal |    10 |    84 |
|      1 | freya  |    10 |    88 |
+--------+--------+-------+-------+
```

Query result will be ascending order of class and if same class exists then ordering will done on marks column(ascending order)

MySQL Order by– e.g.

Suppose we are having student table with following data.

```
mysql> select * from student;
+--------+--------+-------+-------+
| rollno | name   | class | marks |
+--------+--------+-------+-------+
|      1 | freya  |    10 |    88 |
|      2 | mohak  |     1 |    99 |
|      3 | vishal |    10 |    84 |
|      4 | vimal  |    10 |    82 |
|      5 | anil   |     2 |    82 |
```

Now we write query–select * from student order by class asc, marks desc;

```
mysql> select * from student order by class asc,marks desc;
+--------+--------+-------+-------+
| rollno | name   | class | marks |
+--------+--------+-------+-------+
|      2 | mohak  |     1 |    99 |
|      5 | anil   |     2 |    82 |
|      1 | freya  |    10 |    88 |
|      3 | vishal |    10 |    84 |
|      4 | vimal  |    10 |    82 |
+--------+--------+-------+-------+
```

Query result will be ascending order of class and if same class exists then ordering will done on marks column(descending order)

An **aggregate function** performs a calculation on multiple values and returns a single value. For example, you can use the AVG() aggregate function that takes multiple numbers and returns the average value of the numbers.Following is the list of aggregate functions supported by mysql.

Name	Purpose
SUM()	Returns the sum of given column.
MIN()	Returns the minimum value in the given column.
MAX()	Returns the maximum value in the given column.
AVG()	Returns the Average value of the given column.
COUNT()	Returns the total number of values/ records as per given column.

Aggregate Functions & NULL

Consider a table Emp having following records as-
Null values are excluded while (avg)aggregate function is used

Emp		
Code	Name	Sal
E1	Mohak	NULL
E2	Anuj	4500
E3	Vijay	NULL
E4	Vishal	3500
E5	Anil	4000

SQL Queries

mysql> Select Sum(Sal) from EMP;

mysql> Select Min(Sal) from EMP;

mysql> Select Max(Sal) from EMP;

mysql> Select Count(Sal) from EMP;

mysql> Select Avg(Sal) from EMP;

mysql> Select Count(*) from EMP;

Result of query

12000

3500

4500

3

4000

5

The GROUP BY clause groups a set of rows/records into a set of summary rows/records by values of columns or expressions. It returns one row for each group.

We often use the GROUP BY clause with aggregate functions such as SUM, AVG, MAX, MIN, and COUNT. The aggregate function that appears in the SELECT clause provides information about each group.

The GROUP BY clause is an optional clause of the SELECT statement.

Syntax –

SELECT 1, c2,…, cn, aggregate_function(ci)

FROM table WHERE where_conditions GROUP BY c1 , c2,…,cn;

Here c1,c2,ci,cn are column name

MySQL group by – e.g.

Suppose we are having student table with following data.

```
mysql> select * from student;
+--------+--------+--------+--------+
| rollno | name   | class  | marks  |
+--------+--------+--------+--------+
|      1 | freya  |     10 |     88 |
|      2 | mohak  |      1 |     99 |
|      3 | vishal |     10 |     84 |
|      4 | vimal  |     10 |     82 |
|      5 | anil   |      2 |     82 |
```

Now we write query–select class from student group by class;

```
mysql> select class from student group by class;
+--------+
| class  |
+--------+
|      1 |
|      2 |
|     10 |
+--------+
```

Query result will be unique occurrences of class values,just similar to use distinct clause like(select distinct class from student).

MySQL GROUP BY with aggregate functions

The aggregate functions allow us to perform the calculation of a set of rows and return a single value. The GROUP BY clause is often used with an aggregate function to perform calculation and return a single value for each subgroup.

For example, if we want to know the number of student in each class, you can use the COUNT function with the GROUP BY clause as follows:Suppose we are having student table with following data.

```
mysql> select * from student;
+--------+--------+-------+-------+
| rollno | name   | class | marks |
+--------+--------+-------+-------+
|      1 | freya  |    10 |    88 |
|      2 | mohak  |     1 |    99 |
|      3 | vishal |    10 |    84 |
|      4 | vimal  |    10 |    82 |
|      5 | anil   |     2 |    82 |
```

Now we write query–select class,count(*) from student group by class;

```
mysql> select class,count(*) from student group by class;
+-------+----------+
| class | count(*) |
+-------+----------+
|     1 |        1 |
|     2 |        1 |
|    10 |        3 |
+-------+----------+
```

Query result will be unique occurrences of class values along with counting of students(records) of each class(sub group).

MySQL GROUP BY with aggregate functions

we are having student table with following data.

```
mysql> select * from student;
+--------+--------+--------+--------+
| rollno | name   | class  | marks  |
+--------+--------+--------+--------+
|      1 | freya  |     10 |     88 |
|      2 | mohak  |      1 |     99 |
|      3 | vishal |     10 |     84 |
|      4 | vimal  |     10 |     82 |
|      5 | anil   |      2 |     82 |
```

Now we write query–select class,avg(marks) from student group by class;

```
mysql> select class,avg(marks) from student group by class;
+-------+------------+
| class | avg(marks) |
+-------+------------+
|     1 |    99.0000 |
|     2 |    82.0000 |
|    10 |    84.6667 |
+-------+------------+
```

Query result will be unique occurrences of class values along with average marks of each class(sub group).

MySQL GROUP BY with aggregate functions (with where and order by clause)
we are having student table with following data.

```
mysql> select * from student;
+--------+--------+-------+-------+
| rollno | name   | class | marks |
+--------+--------+-------+-------+
|      1 | freya  |    10 |    88 |
|      2 | mohak  |     1 |    99 |
|      3 | vishal |    10 |    84 |
|      4 | vimal  |    10 |    82 |
|      5 | anil   |     2 |    82 |
```

Now we write query–select class,avg(marks) from student where class<10 group by class order by marks desc;

```
mysql> select class,avg(marks) from student where class<10 group by class order
by marks desc;
+-------+------------+
| class | avg(marks) |
+-------+------------+
|     1 |    99.0000 |
|     2 |    82.0000 |
+-------+------------+
```

Query result will be unique occurrences of class values where class<10 along with average marks of each class(sub group) and descending ofer of marks.

The HAVING clause is used in the SELECT statement to specify filter conditions for a group of rows or aggregates. The HAVING clause is often used with the GROUP BY clause to filter groups based on a specified condition. To filter the groups returned by GROUP BY clause, we use a HAVING clause.

WHERE is applied before GROUP BY, HAVING is applied after (and can filter on aggregates).

MySQL GROUP BY with aggregate functions & having clause

we are having student table with following data.

```
mysql> select * from student;
+--------+--------+-------+-------+
| rollno | name   | class | marks |
+--------+--------+-------+-------+
|      1 | freya  |    10 |    88 |
|      2 | mohak  |     1 |    99 |
|      3 | vishal |    10 |    84 |
|      4 | vimal  |    10 |    82 |
|      5 | anil   |     2 |    82 |
```

Now we write query–select class,avg(marks) from student group by class having avg(marks)<90;

```
mysql> select class,avg(marks) from student group by class having avg(marks)<90;

+-------+------------+
| class | avg(marks) |
+-------+------------+
|     2 |    82.0000 |
|    10 |    84.6667 |
+-------+------------+
```

Query result will be unique occurrences of class values along with average marks of each class(sub group) and each class having average marks<90.

SQL

MySQL GROUP BY with aggregate functions & having clause

we are having student table with following data.

```
mysql> select * from student;
+--------+--------+--------+--------+
| rollno | name   | class  | marks  |
+--------+--------+--------+--------+
|      1 | freya  |     10 |     88 |
|      2 | mohak  |      1 |     99 |
|      3 | vishal |     10 |     84 |
|      4 | vimal  |     10 |     82 |
|      5 | anil   |      2 |     82 |
```

Now we write query–select class,avg(marks) from student group by class having count(*)<3;

```
mysql> select class,avg(marks) from student group by class having count(*)<3;
+--------+------------+
| class  | avg(marks) |
+--------+------------+
|      1 |    99.0000 |
|      2 |    82.0000 |
+--------+------------+
```

Query result will be unique occurrences of class values along with average marks of each class(sub group) and each class having less than 3 rows.

Visit : python.mykvs.in for regular updates

Cartesian product (X)/cross joint

Cartesian Product is denoted by X symbol. Lets say we have two relations R1 and R2 then the cartesian product of these two relations (R1 X R2) would combine each tuple of first relation R1 with the each tuple of second relation R2.

Cartesian product (X) example

Table a and Table b as shown

Mysql query –

Select * from a,b;

Select * from a cross join b;

```
mysql> select * from a;
+--------+------+
| Name   | val  |
+--------+------+
| vishal |   11 |
| ram    |   22 |
+--------+------+
2 rows in set (0.00 sec)

mysql> select * from b;
+---------+
| name    |
+---------+
| ram     |
| vikrant |
+---------+
2 rows in set (0.00 sec)
```

```
mysql> select * from a,b;
+--------+------+---------+
| Name   | val  | name    |
+--------+------+---------+
| vishal |   11 | ram     |
| ram    |   22 | ram     |
| vishal |   11 | vikrant |
| ram    |   22 | vikrant |
+--------+------+---------+
4 rows in set (0.00 sec)
```

Degree of cartesion product is 3 and cardinality is 4=(2 rows of a X 2 rows

Join – Join is used to fetch data from two or more tables, which is joined to appear as single set of data. It is used for combining column from two or more tables by using values common to both tables.
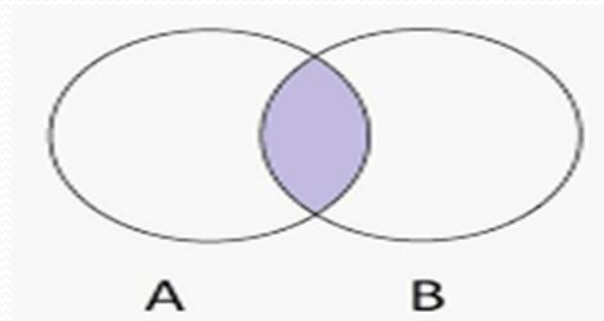
Types of JOIN
Following are the types of JOIN that we can use in SQL:
- Inner
- Outer
- Left
- Right

INNER Join or EQUI Join⋈

This is a simple JOIN in which the result is based on matched data as per the equality condition specified in the SQL query.

INNER Join or EQUI Join example

Table a and Table b as shown

```
mysql> select * from a;
+--------+--------+
| Name   | val    |
+--------+--------+
| vishal |     11 |
| ram    |     22 |
+--------+--------+
2 rows in set (0.00 sec)

mysql> select * from b;
+----------+
| name     |
+----------+
| ram      |
| vikrant  |
+----------+
2 rows in set (0.00 sec)
```

Mysql query –

Select course.student_name from couse , student where course.student_name=student.student_name;

Select a.name from a inner join b where a.name=b.name;

```
mysql> select a.name from a inner join b where a.name=b.name;
+------+
| name |
+------+
| ram  |
+------+
```

Natural JOIN(⋈)

Natural Join is a type of Inner join which is based on column having same name and same datatype present in both the tables to be joined.E.g.

Select * from a natural join b;

```
mysql> select * from a natural join b;
+--------+--------+
| Name   | val    |
+--------+--------+
| ram    |     22 |
+--------+--------+
1 row in set (0.00 sec)
```

LEFT Outer Join ⋈

The left outer join returns a resultset table with the matched data from the two tables and then the remaining rows of the left table and null from the right table's columns. E.g.

Mysql query –

Select * from a left outer join b on (a.name=b.name);

```
mysql> select * from a;
+----------+--------+
| Name     | val    |
+----------+--------+
| vishal   |     11 |
| ram      |     22 |
+----------+--------+
2 rows in set (0.00 sec)

mysql> select * from b;
+----------+
| name     |
+----------+
| ram      |
| vikrant  |
+----------+
2 rows in set (0.00 sec)
```

```
mysql> select * from a left outer join b on (a.name=b.name);
+----------+--------+--------+
| Name     | val    | name   |
+----------+--------+--------+
| vishal   |     11 | NULL   |
| ram      |     22 | ram    |
+----------+--------+--------+
2 rows in set (0.02 sec)
```

RIGHT Outer Join

The right outer join returns a resultset table with the matched data from the two tables being joined, then the remaining rows of the right table and null for the remaining left table's columns.E.g.

```
mysql> select * from a;
+--------+------+
| Name   | val  |
+--------+------+
| vishal |   11 |
| ram    |   22 |
+--------+------+
2 rows in set (0.00 sec)

mysql> select * from b;
+---------+
| name    |
+---------+
| ram     |
| vikrant |
+---------+
2 rows in set (0.00 sec)
```

Mysql query –

Select * from a right outer join b on (a.name=b.name);

```
mysql> select * from a right outer join b on (a.name=b.name);
+------+------+---------+
| Name | val  | name    |
+------+------+---------+
| ram  |   22 | ram     |
| NULL | NULL | vikrant |
+------+------+---------+
2 rows in set (0.00 sec)
```

SQL

Full Outer Join

The full outer join returns a resultset table with the matched data of two table then remaining rows of both left table and then the right table.E.g.

```
mysql> select * from a;
+--------+-------+
| Name   | val   |
+--------+-------+
| vishal |    11 |
| ram    |    22 |
+--------+-------+
2 rows in set (0.00 sec)
mysql> select * from b;
+---------+
| name    |
+---------+
| ram     |
| vikrant |
+---------+
2 rows in set (0.00 sec)
```

Mysql query –

Select * from a left outer join b on (a.name=b.name) union Select * from a right outer join b on (a.name=b.name) ;

```
mysql> select * from a left outer join b on(a.name=b.name) union select * from a
 right outer join b on (a.name=b.name);
+--------+-------+---------+
| Name   | val   | name    |
+--------+-------+---------+
| vishal |    11 | NULL    |
| ram    |    22 | ram     |
| NULL   | NULL  | vikrant |
+--------+-------+---------+
3 rows in set (0.01 sec)
```