**MYSQL**

**Database**: it is an application which stores the collection of data.

Each database has one or more distinct API’s (Application programming interface) for creation of data, managing, searching the data.

**File:** unchanged and can’t update and can’t organise the data.

**Excel:** for storage purpose and for some security reasons and when we go on adding the data we speed will be reduced.

**DBMS (Data base management system)**

We store the data in the form of tables

**Challenges of DBMS:**

Relations is not possible in dbms

**RDBMS(Relational database management system** )

We can store the data in the form of tables and can also map them from locations.

Advantages

It will retrieve the data very Fastly and operations will be very effective

**Database is 2 types**

1. **RDBMS**
2. **NON-RDBMS**

Stores the data in the form of key-values in the Json format

**Perquisites to create a project**

1. **Frontend** to view the data
2. **Backend** it provides the interaction between the data and programming
3. **Database** to store the data

**Database components**

1. **Client**
2. **Server**

**MySQL uses two types of commands:**

1. **DDL (data definition language)**
2. **DML (data manipulation language)**

**DDL commands:**

**1.create –** to create a database and tables

**2.Alter** – add a row/column

**3.DROP –** delete the records from the database

**4.Turnicate** – remove the records from the data

**5. Rename** – to rename the table or records

**DML Commands**

1. **Insert –** insert the data into table
2. **Update –** to update the existing data in the table
3. **Delete –** deletes the records from the database of a table

**Datatypes**

**1.CHAR(size)** - a fixed length of string characters are allowed

**2.VARCHAR** – a variable length \

3.binary – equal to char, default it will be

**4.text** – it holds a string with a max length of 65 or 535 bytes only

**5.TINTEXT** – it holds the string of MAX length 255

**Task:**

create database gopinadh;

use gopinadh;

create table student(name varchar(20), id int, place varchar(20), phno varchar(10), age int);

insert into student (name,id,place,phno,age)

values ('gopi', 62, 'khammam', '9347528265', 23),

('ravi', 65, 'medhipatnam', '9347528275', 19),

('jathin', 77, 'knr', '970503875', 18),

('roja', 60, 'hyd', '1234567890', 24),

('akhil', 49, 'khmm', '630257598', 27),

('navya', 61, 'nalagonda', '970502344', 25),

('akhila', 55, 'wrangal', '987456321', 24),

('sai', 19, 'hyd', '951357462', 23),

('raju', 60, 'nijamabad', '4562851793', 20),

('rani', 59, 'vizag', '8526541397', 23);

**use gopinadh;**

CREATE table bikeDetails (brand VARCHAR (20), no INT, priceVARCHAR(20),modeVARCHAR(30),varient VARCHAR(10));

INSERT INTO bikeDetails (brand, no, price, model, varient)

VALUES

('hero', 1234, '25000', 'BS6', 'EV'),

('honda', 5678, '22000', 'BS6', 'Petrol'),

('shine', 9101, '28000', 'BS6', 'EV'),

('bmw', 1121, '50000', 'BS4', 'PETROL'),

('bajaj', 1234, '10000000', 'BS6', 'EV'),

('FZ', 4567, '70000000', 'BS4', 'PETROL'),

('ktm', 9876, '4000000', 'BS6', 'EV'),

('duke', 2345, '200000', 'BS4', 'EV'),

('herohonda', 3425, '3000000', 'BS4', 'PETROL'),

('pulsar', 6547, '700000', 'BS6', 'EV');

**Clauses and operators**

**1.where:** it is used to filter records in a query and filter records those meet the specific conditions.

**Query:**

select \* from bikedetails

where price > 50000;

**2.AND:** and operator will combine multiple conditions in the where clause if all the conditions are true then row will be added in the result.

**Query:**

select \* from bikedetails

where price > 50000 and varient = 'EV';

**3.OR:** or operator will combine multiple conditions in the where operator if any of the condition is true then row will be added in the result

**Query:**

select \* from bikedetails

where varient = 'EV' or price > 50000;

**3.NOT:** display the records other than the given condition

**Query:**

select brand,varient

from bikedetails

where not brand = 'FZ';

**4.Orderby:** mainly used of sorting the records ascending order or descending order . default(ascending order)

**Query:**

select \* from bikedetails

order by price;

**5.insert into:** used to insert new records in the existing table.

**Query:**

insert into test(name, rollno, bloodgroup)

values('p.gopinadh',56 , 'o+');

**6.select clause:** to display to obtain the data from particular tables.

**Query :**

Select\* from students;

**7.delete:** delete the existing records from the table.

**Query:**

delete from bikedetails

where brand = 'FZ';

**8.limit**: to specify the number of records to return.

**Query**:

select \* from employee limit 5;

**9.min&max:**

**Min :**To display min value from the records

**Query:**

select brand,min(price) from vehicledetails

group by brand;

**Max:** To display max value from the records

**Query:**

select brand,max(price) from vehicledetails

group by brand;

**10.like:** we used to obtain a specific pattern in a column

**Query:**

select \* from employee

where role like 'software developer';

%a – finds names ending with ‘a’;

**Query:**

select \* from employee

where name like '%a';

a% - finds the names starting with ‘a’;

**Query:**

select \* from employee

where name like '%a';

\_a% - find the values in the second letter position is a

**Query:**

select \* from employee

where name like '\_a%';

**11.In:** allow us to specify multiple values in the where clause

**Query:**

select name,email,exprience

from employee

where name in ('gopinadh', 'ravi', 'jathin');

**12.Between:** selects the middle values from a range of values

**Query:**

select \* from vehicledetails

where price between 2500 and 50000;

**14.group by** : in order to perform aggregate functions we use this group by.

**Query:**

select brand,sum(price) as price from vehicledetails

group by brand;

**15.Avg ():** it returns the average value of the particular column.

**Query:**

select brand,avg(price) as price from vehicledetails

group by brand;

**16.sum ()**: it returns the total numbers present in a columns.

**Query:**

select brand,sum(price) as price from vehicledetails

group by brand;

**17.count :** returns the number of records which satisfies our condition.

**Query:**

Select count(\*) from student;

**Joins:**

Joins are used to combine two are more tables in relational database based on similar columns. We use joins with select statement only we can retrieve the data from multiple tables and to perform join operations data should be in a single database only using this join operations we can fetch the records from the different database.

**Types of SQL Joins:**

1. **Inner join**
2. **Outer join**
3. **Self join**
4. **Cross join**

**About inner join**

Inner join is the most commonly used in mysql if we give table\_name join and table\_name by default it will perform inner join operation it will combine two or more tables based on the similar columns.it will return all the rows from the tables where the join condition satisfies.

**Syntax:**

**Select columns from table1 inner join table2**

**On table1.col = table2.col;**

**Example:**

**Write a query to fetch employee names and their respective department names from `Employees` and `Departments` tables.**

create table departments1(

departments\_id int primary key auto\_increment,

departments\_name varchar(20)

);

insert into departments1(departments\_name)

values('IT'),

('NON IT'),

('HR');

create table employee2(

employee\_id int primary key auto\_increment,

name varchar(20),

email varchar(20),

phno long,

departments\_id int,

foreign key (departments\_id) references departments1(departments\_id)

);

insert into employee2(name, email, phno, departments\_id)

values ('gopi', 'gopi99@gmail.com', 9347528265, 1),

('jeevan', 'jeevan99@gmail.com', 987654123, 2),

('hari', 'hari99@gamil.com', 963258147, 3),

('santoosh', 'santoosh99@gmail.com', 789654123, 2),

('manibabu', 'manibabu99@gmail.com', 789654123, 1),

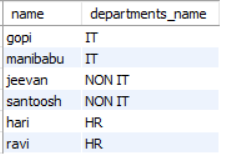
('ravi', 'ravi99@gmail.com', 963258714, 3);

select employee2.name ,departments1.departments\_name

from departments1 inner join employee2

on employee2.departments\_id = departments1.departments\_id;

**output:**

****

**2.Outer join:**

In outer join we have 3 types of joins

1.left join

2.right join

3. full join

**About left join:**

It is used to combine rows from tables based on condition.it will return all the rows from the left hand table and on right table it will return rows which satisfies the condition and remaining rows it will return as null values.

**Syntax:**

Select columns

from

table1 left join table2

on

table1.col = table2.col;

**example:**

**Write a SQL query to display each customer's name along with their order amounts. Include all customers, even if they don't have any orders. If a customer has no orders, display NULL for the order amount.**

create table customers1(

customer\_id int primary key auto\_increment,

name varchar(20),

email varchar(20)

);

insert into customers1(name,email)

values ('gopi', 'gopi99@gmail.com'),

('jathin', 'jathin99@gmail.com'),

('sai', 'sai99@gmail.com'),

('mani','mani99@gmail.com'),

('raju','raju99@gmail.com');

create table orders1(

order\_id int primary key auto\_increment,

order\_name varchar(20),

amount int,

customer\_id int,

foreign key (customer\_id) references customers1(customer\_id)

);

insert into orders1(order\_name, customer\_id, amount)

values ('Smartphone', 1, 500),

('T-shirt', 3, 1000),

('shirt', 4, 400),

('earpods', 1, 200),

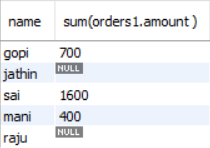
('paint', 3, 600);

select customers1.name , sum(orders1.amount )

from customers1 left join orders1

on orders1.customer\_id = customers1.customer\_id group by customers1.name;

**output:**

****

**About right join:**

It is used to combine rows from tables based on condition.it will return all the rows from the right hand table and on left table it will return rows which satisfies the condition and remaining rows it will return as null values.

**Syntax:**

Select columns

from

table1 right join table2

on

table1.col = table2.col;

**example:**

**Write a SQL query to display each class name along with the names of students enrolled in those classes. Include all classes, even if no students are enrolled in them. If a class has no students, display NULL for the student name.**

**Query:**

create table classes(

class\_id int primary key auto\_increment,

class\_names varchar(20)

);

insert into classes(class\_names)

values ('telugu'),

('hindi'),

('english'),

('maths');

create table students(

students\_id int primary key auto\_increment,

students\_name varchar(20),

class\_id int,

foreign key (class\_id) references classes(class\_id)

);

insert into students(students\_name, class\_id)

values('gopi', 1),

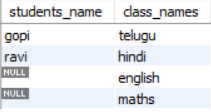
('ravi', 2);

select students.students\_name,classes.class\_names

from students right join classes

on classes.class\_id = students.class\_id;

Output:



**About full join:**

It is the combination of both left join and right join.

**Syntax:**

Select columns

From

Table1 full join table2

On table1.col = table2.col

**Query:**

create table authors(

author\_id int primary key auto\_increment,

author\_name varchar(20),

country varchar(5)

);

insert into authors(author\_name, country)

values('gopi', 'IND'),

('jathin', 'AUS'),

('ravi', 'US'),

('raju', 'UK'),

('navya', 'IND');

create table Books(

books\_id int primary key auto\_increment,

books\_name varchar(20),

author\_id int,

foreign key (author\_id) references authors(author\_id)

);

insert into Books(books\_name, author\_id)

values('Harry Potter', 1),

('Animal Farm', 2),

('Pride and Prejudice', 3),

('1984', 4);

SELECT authors.author\_name, authors.country, Books.books\_name

FROM authors

LEFT JOIN Books ON authors.author\_id = Books.author\_id

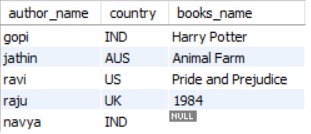
UNION

SELECT authors.author\_name, authors.country, Books.books\_name

FROM authors

RIGHT JOIN Books ON authors.author\_id = Books.author\_id;

OUTPUT:



**About Cross join:**

It combines each row of a table with all the rows in the another table.

Syntax:

Select column from table1 cross join table2

Question:

Write an SQL query to generate all possible combinations of employees and projects, so every employee is assigned to each project.

create table employee3(

employee\_id int primary key auto\_increment,

employee\_name varchar(20)

);

insert into employee3(employee\_name)

values('gopi'),

('mani');

create table projects(

project\_id int primary key auto\_increment,

project\_name varchar(20)

);

insert into projects(project\_name)

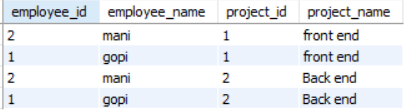
values('front end'),

('Back end');

select \* from employee3

cross join projects;

Output:



**About self join:**

The data or rows in the table are combined or joined with the same table. In self join we can use any type of join.

**Syntax:**

Select column from

Table as t1 inner join table as t2

Where condition

Question:

The **Products** table contains the following data:

| **ProductID** | **ProductName** | **CategoryID** |
| --- | --- | --- |
| 1 | Laptop | 101 |
| 2 | Mouse | 101 |
| 3 | Phone | 102 |
| 4 | Charger | 102 |
| 5 | Tablet | 103 |

Write an SQL query to find **pairs of products** that belong to the same category. Ensure each pair appears only once (e.g., "Laptop - Mouse" but not "Mouse - Laptop").

create table products2 (

products\_id int primary key auto\_increment,

product\_name varchar(20),

category\_id int

);

insert into products2 (product\_name, category\_id)

values ('Laptop', 101),

('Mouse', 101),

('Phone', 102),

('Charger', 102),

('Tablet', 103);

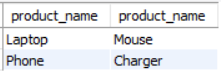
select tb1.product\_name, tb2.product\_name

from products2 as tb1 inner join products2 as tb2

on tb1.category\_id = tb2.category\_id

where tb1.products\_id < tb2.products\_id;

**output:**

****