**SQL**

1. **Defination :** SQL stands for Structured Query Language. It is a domain-specific language used in programming and designed for managing and manipulating data held in a relational database management system (RDBMS).
2. **SQL vs NoSQL :**
3. **SQL or Relational Databases :** In sql database data are stored in the form of rows and columns.

**MySQL,PostgreSQL**

1. **noSQL or non-Relational Databases :** In this database data are stored in the form of key-value paire or dictionary or document format**,**

**MongoDB**

1. **DBMS :** Database management system It is a software application , we cant directly access the database but DBMS help to interact with the databases.
2. **DATABASE Create & Drop :**
3. **Create Database :** The CREATE DATABASE statement is used to create a new SQL database.

CREATE DATABASE databasename;

create database databasename;

|  |
| --- |
| create database college; |

**Check database exist or not :**

CREATE DATABASE IF NOT EXISTS databaseName

|  |
| --- |
| create database if not exists college; |

1. **Delete Database :** The DROP DATABASE statement is used to drop an existing SQL database.

DROP DATABASE databasename;

drop database databasename;

**Drop or remove a database if it exists :**

DROP DATABASE IF EXISTS databaseName

|  |
| --- |
| drop database if exists college; |

1. **USE databasename :** It is use to select and do task with the specific database.

use databasename

1. **Show databases :** It will return all the existing databases in your system.

show databases;

1. **Show tables :** It will return all the existing tables ;

show tables;

1. **Types of SQL Commands :**

* **DDL (Data definition language)** **🡪** Create , Drop , Alter , Rename , Truncate.
* **DQL (Data Query Language)** **🡪** Select.
* **DML (Data Manupulation Language) 🡪** Insert,Update,Delete.
* **DCL (Data Control Language) 🡪** grant & revoke permission to users.
* **TCL (Transaction Control Language) 🡪** start transaction , commit , rollback.

1. **Tables Related Query :**
2. **Create Table :** The CREATE TABLE statement is used to create a new table in a database.

|  |  |
| --- | --- |
| CREATE TABLE table\_name (     column1 datatype constraints,     column2 datatype constraints,     column3 datatype constraints,    .... ); | CREATE TABLE student(  StudentID INT PRIMARY KEY,  FirstName VARCHAR(50),  LastName VARCHAR(50),  DateOfBirth date,  Gender VARCHAR(10),  Email VARCHAR(10)  ); |

1. **Insert :** insert into databasename values(data);

|  |
| --- |
| insert into student values(1,"Piyush","Thaware","7 Jan 2001","male","Piyush@gmail.com"); |
| **Insert Data Into specific columns**  **insert into tableName (column1,column2) values(data);**  insert into record (fname,lname) values("Shreyash","Thaware"); |

1. **Read :** select \* from databaseName

|  |
| --- |
| select \* from student; |

1. **Drop :** It will remove the table from the database **🡪** drop table tableName

|  |
| --- |
| drop table student; |

1. **Select** : select & view columns : **READ**

|  |
| --- |
| select \* from student;  **Select Or Read a specifc column**  select columnName-1,columnName-2 from tableName;  SELECT name FROM student;  SELECT name,city FROM student;  **DISTINCT : Show Unique values of a specific column as below**  SELECT DISTINCT columnName FROM tablesName;  SELECT DISTINCT city FROM student; |

1. **Keys :**

* **Primary Key :** A PRIMARY KEY in a database is a column or a set of columns that uniquely identifies each row (record) in a table.It should be not null and uniquely identify.
* **Foreign Key :** A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the [PRIMARY KEY](https://www.w3schools.com/sql/sql_primarykey.asp) in another table.It can have duplicate & null values.It can have multiple values.

1. **Constraints :** It is like adding some condition while inserting data into database like example : age or student should be greater then 18.

|  |  |  |
| --- | --- | --- |
| * **NOT NULL** | Columns cannot have a null value | columnName int NOT NULL |
| * **UNIQUE** | All the values in the column are different | columnName int UNIQUE |
| * **PRIMARY KEY** | Make a column unique & not null but used for only one | ColumnName int primary key  EXAMPLE : create table temp2(  id int,  name varchar(50),  email varchar(100),  **primary key(id,name)** -- now it may be possible id can be dublicate or name but combination of both is not possible  );  insert into temp2 values(1,"Piyush","Piyush@gmail.com");  -- insert into temp2 values(1,"Piyush","Piyush@gmail.com"); ERROR  insert into temp2 values(2,"Piyush","Piyush@gmail.com"); |
| * **FOREIGN KEY** | A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the [PRIMARY KEY](https://www.w3schools.com/sql/sql_primarykey.asp) in another table. | create table tableName(  my\_id int,  foreign key (my\_id) references AnotherTableName(primarykey)  ); |
| * **DEFAULT** | Sets default value of a column. | fieldname datatype default value  Salary int default 25000  Example :  create table employee (  emp\_id int auto\_increment primary key,  emp\_name varchar(100),  emp\_salary int default 50000  );  insert into employee (emp\_name) values("Piyush");  insert into employee (emp\_name,emp\_salary) values("Shareyash",60000);  select \* from employee; |
| * **CHECK** | It can limit the values allowed in a column. | create table city(  id int primary key,  city varchar(50),  age int,  constraint age\_check CHECK (age>=18 and city="nagpur")  );  insert into city values(1,"nagpur",23);  insert into city values(2,"nagpur",13); -- Error Code: 3819. Check constraint 'age\_check' is violated. |

1. **Clauses :**
2. **Where clause : clauses are use to define some conditions.**

|  |
| --- |
| **SELECT \* FROM tableName WHERE condition;**  SELECT \* FROM student WHERE marks>80;  **CHECK MULTIPLE CONSITION WITH OPERATORS :**  SELECT \* FROM student WHERE marks<90 AND city=”mumbai”;  SELECT \* FROM student WHERE marks+10>100;  SELECT \* FROM student WHERE marks=88;  SELECT \* FROM student WHERE marks!=88; |

**Operators :**

1. **Arithmetic Operators 🡺 + , - , \* , / , %**
2. **Comparison Operators 🡺 = (**equal to) , != (not equal to) , > , < , >= , <=
3. **Logical Operator 🡺** AND , OR , NOT , IN , BETWEEN , ALL , LIKE , ANY
4. **Bitwise Operator 🡺** & (Bitwise AND) , | (Bitwise OR)

**BETWEEN :** selects for the given range

|  |
| --- |
| **SELECT \* FROM tableName WHERE columnName BETWEEN minimumValue AND maximumValue;**  SELECT \* FROM student WHERE marks BETWEEN 70 AND 90; |

**IN :** matches ay values in list

|  |
| --- |
| **SELECT \* FROM tableName WHERE columnName IN ("delhi","mumbai",….so on);**  SELECT \* FROM student WHERE city IN ("delhi","mumbai"); |

**NOT :** To negate the given condition

|  |
| --- |
| **SELECT \* FROM databaseName WHERE columnName NOT IN ("delhi","mumbai");**  SELECT \* FROM student WHERE city NOT IN ("delhi","mumbai"); |

1. **Limit Clause :** The LIMIT clause is used in SQL queries to restrict the number of rows returned by a query.

|  |
| --- |
| **SELECT \* FROM tableName LIMIT RowCount;**  **SELECT column1,column2 FROM student LIMIT 3;**  SELECT \* FROM student LIMIT 3;  SELECT column1,column2 FROM student LIMIT 3;  SELECT \* FROM student WHERE marks>78 LIMIT 5; |

1. **OrderBy Clause :** To sort inascending (ASC) or Descending order(DESC).

|  |
| --- |
| **SELECT \* FROM tableName ORDER BY columnName ASC;**  **SELECT \* FROM tableName ORDER BY columnName DESC;**  SELECT \* FROM student ORDER BY marks ASC;  SELECT \* FROM student ORDER BY marks DESC;  SELECT \* FROM student ORDER BY city ASC; |

1. **Aggregate Function :** Aggregate function perform a calculation on a set of values and return a single value.

|  |
| --- |
| **1. MAX()**  SELECT MAX(marks) FROM student;  **2. MIN()**  SELECT MIN(marks) FROM student;  **3. AVG()**  SELECT AVG(marks) FROM student;  **4. COUNT()**  SELECT COUNT(marks) FROM student;  **5. SUM()**  SELECT SUM(marks) FROM student; |

1. **Group By Clause :** groups the rows that have same values into summary or result rows, it collect data from multiple records and group the result by one or more column.generally aggregate functions are used to group.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SELECT city,count(name) FROM student GROUP BY city;  City count(name)   |  |  | | --- | --- | | Nagpur | 6 | | Rajastan | 3 | | Pune | 3 | | Delhi | 5 | | Mumbai | 3 | | EXAMPLE : | FIND TOTAL NO OF STUDENTS FROM NAGPUR AND OTHER STATES;  SELECT city,name,count(name),avg(marks) FROM student GROUP BY city,name; | |

**Find the avg marks in each city in ascending order.**

|  |
| --- |
| SELECT city,AVG(marks) FROM student GROUP BY city ORDER BY city ASC; |

1. **HAVING CLAUSE :** Similar to Where Clause , it applies some condition on rows used when we want to apply any condition after grouping.

|  |
| --- |
| **Count number of students in each city where max marks cross 90.**  SELECT city,COUNT(rollno) FROM student  GROUP BY city  HAVING MAX(marks)>90; |

1. **General Order :**

|  |
| --- |
| SELECT columns  FROM tableName  WHERE condition  GROUP BY columns  HAVING condition  ORDER BY columns ASC |

Note : Where apply conditions on the row while

Having apply conditions on the groups

1. **TABLE RELATED QUERY :**
2. **UPDATE :** It is used to update the existing rows.

|  |  |
| --- | --- |
| UPDATE tableName  SET column1=value,column2=value  WHERE condition; | UPDATE student  SET name="Rohit Sharma",marks=34,grade="D",city="MIrzapur"  WHERE rollno=7;  **EXAMPLE 02 :**  UPDATE student  SET marks=100  WHERE name="Piyush Thaware" AND rollno=4;  **EXAMPLE 03** **: INCREASE EVERY STUDENT MARKS BY 1**  UPDATE student  SET marks=marks+1; |

|  |
| --- |
| **ENABLE / DISABLE SAFE MODE IN SQL TO UPDATE :** |
| SET SQL\_SAFE\_UPDATES = 0 (disable)  SET SQL\_SAFE\_UPDATES = 1 (enable) |

1. **DELETE :** To Delete an existing row

|  |  |
| --- | --- |
| DELETE FROM tableName  WHERE condition | **REMOVING DUBLICATE ROWS THAT ARE CREATED DUE TO MISTAKE**  DELETE FROM student  WHERE rollno BETWEEN 8 AND 27;  **TO REMOVE ALL THE ROWS FROM THE TABLE**  DELETE FROM student; |

1. **ALTER :** TO CHANGE THE SCHEMA

|  |
| --- |
| 1. **ADD NEW COLUMN** |
| |  |  | | --- | --- | | ALTER TABLE tableName  ADD COLUMN columnName datatype constraints | ALTER TABLE student  ADD COLUMN age INT; | |

|  |
| --- |
| 1. **DROP COLUMN** |
| |  |  | | --- | --- | | ALTER TABLE tableName  DROP COLUMN columnName | ALTER TABLE student  DROP COLUMN age; | |

|  |
| --- |
| 1. **RENAME TABLE** |
| |  |  | | --- | --- | | ALTER TABLE tableName  RENAME newTableName | ALTER TABLE student  RENAME TO students; | |

|  |
| --- |
| 1. **CHANGE COLUMN (Rename)** |
| |  |  | | --- | --- | | ALTER TABLE tableName  CHANGE COLUMN  **oldColumnName NewColumnName** datatype newConstraint; | ALTER TABLE student  CHANGE COLUMN name fullname VARCHAR(100); | |

|  |
| --- |
| 1. **MODIFY COLUMN (modify datatype or constraints)** |
| |  |  | | --- | --- | | ALTER TABLE tableName  MODIFY columnName datatype Constraints; | ALTER TABLE student  MODIFY rollno VARCHAR(50) UNIQUE; | |

|  |
| --- |
| 1. **TRUNCATE :** to delete table’s data , while with drop you can delete entire table |
| |  |  | | --- | --- | | ALTER TABLE tableName  MODIFY columnName datatype Constraints; | ALTER TABLE student  MODIFY rollno VARCHAR(50) UNIQUE; | |

1. **FOREIGN KEY :**

* **CASCADING FOR FOREIGN KEY :**

1. **On DELETE Cascade :** This is a cascading referential action that specifies when a row in the parent table is deleted, all matching rows in the child table(s) are automatically deleted as well.
2. **On UPDATE Cascade :** ON UPDATE CASCADE, it means that if a primary key value in the parent table is updated, the corresponding foreign key value(s) in the child table(s) will be automatically updated to match the new primary key value.

|  |  |
| --- | --- |
| **Course Table** | **Teacher Table** |
| CREATE TABLE IF NOT EXISTS course(  course\_id INT PRIMARY KEY,  course\_name VARCHAR(100)  );  INSERT INTO course values  (101,"Science"),  (102,"English"),  (103,"Maths"),  (104,"Hindi");  SELECT \* FROM course;  **-- NOW SUPPOSE I HAVE CHNAGE THE COURSE\_ID for "hindi" subject then it should be replicated for each teacher in teacher table**  UPDATE course  SET course\_id=105  WHERE course\_name="Hindi"; | CREATE TABLE IF NOT EXISTS teacher(  teacher\_id INT PRIMARY KEY,  teacher\_name VARCHAR(100),  teacher\_course\_id INT,  FOREIGN KEY(teacher\_course\_id) REFERENCES course(course\_id)  ON UPDATE CASCADE **-- Agar course table mein changes hoga toh vo changes yaha bhi hoga**  ON DELETE CASCADE  );  INSERT INTO teacher values  (001,"Piyush",103),  (002,"Shreyash",102),  (003,"Sahil",103),  (004,"Pawan",101),  (005,"Herik",104);  SELECT \* FROM teacher; |

1. **JOINS IN SQL :**

Join is used to combine rows from two or more tables based in a related column between them.

|  |  |
| --- | --- |
| 1. **INNER JOIN:**   Returns rows from both tables where the join condition is met. | SELECT columns  FROM table1  INNER JOIN table2 ON table1.column = table2.column;  **EXAMPLE** **:**  SELECT \* FROM teacher  INNER JOIN course  ON teacher.teacher\_course\_id = course.course\_id;  **EXAMPLE :**  SELECT teacher\_course\_id FROM teacher  INNER JOIN course  ON teacher.teacher\_course\_id = course.course\_id; |
| 1. **LEFT JOIN (or LEFT OUTER JOIN)**:   Returns all rows from the left table and matching rows from the right table based on the join condition.  If no matching rows are found in the right table, NULL values are returned for the right table columns.  Jo record apko pura dekhna hai kisi bhi ek table ka usko table1 consider karo or jiska reference kar rahe ho usko table 2 | SELECT columns  FROM table1  LEFT JOIN table2 ON table1.column = table2.column;  **EXAMPLE :**  SELECT \*  FROM course as c  LEFT JOIN teacher as t  ON c.course\_id = t.teacher\_course\_id; |
| 1. **RIGHT JOIN (or RIGHT OUTER JOIN):**   Returns all rows from the right table and matching rows from the left table based on the join condition.  If no matching rows are found in the left table, NULL values are returned for the left table columns. | SELECT columns  FROM table1  RIGHT JOIN table2 ON table1.column = table2.column;  SELECT \*  FROM course as c  RIGHT JOIN teacher as t  ON c.course\_id = t.teacher\_course\_id; |
| 1. **FULL JOIN (or FULL OUTER JOIN):**   Returns all rows from both tables and matches rows based on the join condition where available.  If no match is found, NULL values are used for missing columns from the other table.  **NOTE :** MySQL Don’t have **FULL JOIN** Method so use this example to implement full join | SELECT \* FROM course AS c  LEFT JOIN teacher AS t  ON c.course\_id = t.teacher\_course\_id  UNION  SELECT \* FROM course AS c  RIGHT JOIN teacher AS t  ON c.course\_id = t.teacher\_course\_id;  **SYNTAX :**  LEFT JOIN  UNION  RIGHT JOIN |
| 1. **LEFT EXCLUSIVE JOIN :**   A left excluding join would return all rows from the left table that do not have matching rows in the right table, similar to a left join but excluding matching rows. | SELECT \*  FROM table1 as t1  LEFT JOIN table2 as t2  ON t1.id = t2.id  WHERE t2.id IS NULL; |
| 1. **RIGHT EXCLUSING JOIN**   A right excluding join would return all rows from the right table that do not have matching rows in the left table, similar to a right join but excluding matching rows. | SELECT \*  FROM table1 as t1  RIGHT JOIN table2 as t2  ON t1.id = t2.id  WHERE t1.id IS NULL; |
| 1. **SELF JOIN**   Joins a table to itself, treating it as if it were two separate tables.  Useful for comparing rows within the same table. | SELECT \*  FROM table1 as t1  JOIN table2 as t2  ON t1.id = t2.id |

|  |  |
| --- | --- |
| 1. **UNION JOIN**   The **UNION** operator is used to combine the result sets of two or more SELECT statements into a single result set.  It removes duplicate rows by default (use **UNION ALL** to include duplicates).  Columns in each SELECT statement must be of the same data type and in the same order. | SELECT columns FROM table1  UNION  SELECT columns FROM table2;  **EXAMPLE :**  SELECT name FROM table1  UNION  SELECT salary FROM table2;  RESULT : names are combined with salary in a single column. |