Database and SQL

- **Data** is the raw material for information. It is unprocessed and unstructured.

- **A proper collection of data is known as information.** It is processed data that has been given meaning.

- **What is a database?**
  - ⇨ A database is a structured and organised collection of data that can be efficiently stored, managed, accessed, and updated using any database management software product like MySQL, Oracle Database, SQL Server, IBM Db2, and PostgreSQL.

    A database has many advantages, such as accessibility, reliability, flexibility, consistency, integration, reusability, security, backup, and recovery.

- **What is DBMS?**
  - ⇨ A database management system (DBMS) is a software application that is used to store, retrieve, manage, and access data in a database.

- **What is a relational database?**
  - ⇨ A relational database is a database that stores data in table format. Each table has rows and columns. Each row represents a single record, and each column represents a specific attribute of that record.
  - ⇨ **The basic structures of a relational database include:**
    Tables: Structures for organizing data in rows and columns.
    Columns: Define the types of data stored in tables.
    Rows: Contain individual data entries within tables.
    Keys: Ensure data integrity and establish relationships between tables, including primary, foreign, unique, and composite keys.

- **What is RDBMS (Relational Database Management System)?**
  - ⇨ A relational database management system is a software application that is used to store, retrieve, manage, and access data in relational databases.

    RDBMS offers many advantages, such as accessibility, reliability, flexibility, consistency, integration, reusability, security, backup, and recovery.

- **Difference between DBMS and RDBMS.**

  - ⇨ **Data structure:** DBMS can store data in a variety of data structures, such as hierarchical, network, and relational. RDBMS stores data in a relational data structure, which is a table of rows and columns.
  - ⇨ **Data integrity:** DBMS does not enforce data integrity constraints. RDBMS enforces data integrity constraints, such as referential integrity and entity integrity.

- ⇨ **Performance:** DBMS can be slower than RDBMS, especially for complex queries. RDBMS is designed to be efficient for complex queries.

- **What are the different types of RDBMS and what are the essential features of an RDBMS?**

  ⇨ **There are four main types of RDBMS:**

  1. **Hierarchical RDBMS:** This type of RDBMS stores data in a tree-like structure, with each node having one or more child nodes.
  2. **Network RDBMS:** This type of RDBMS stores data in a network-like structure, with each node being able to have multiple parent and child nodes.
  3. **Relational RDBMS:** This is the most common type of RDBMS. It stores data in tables, with each table having rows and columns.
  4. **Object-oriented RDBMS:** This type of RDBMS stores data in objects, which can have attributes and methods.

  ⇨ **The essential features of an RDBMS are:**

  Data abstraction: The RDBMS hides the physical storage details from users, providing a logical representation of data. This makes it easier for users to work with data, as they do not need to know how the data is actually stored.

  Data integrity: The RDBMS ensures that the data is accurate and consistent. This is done through a variety of mechanisms, such as constraints and triggers.

  Data independence: The RDBMS separates the data from the application. This makes it easier to change the data schema without affecting the application.

  Security: The RDBMS provides security features to protect data from unauthorized access.

  Querying: The RDBMS provides a powerful query language, such as SQL, that allows users to retrieve, manipulate and manage data.

  Backup and Recovery: RDBMS systems offer mechanisms for data backup and recovery to prevent data loss in case of system failures.

- **What is a SQL statement and what are the different types of SQL statements?**

  ⇨ A SQL statement is a command that is used to interact with a relational database management system. These statements can be used to create, read, update, and delete data in a database.

There are five main types of SQL statements (commands):

- ⇨ **DDL (Data Definition Language):** DDL is used to change or define the structure of the database, such as creating, modifying, and deleting tables. They do not affect the data in the database.

  CREATE: Creates a new database objects like tables, indexes, or views.

  ALTER: Modifies an existing database objects, such as adding, modifying, or deleting columns in a table.

  RENAME: Changes the name of an existing database object.

  TRUNCATE: Used to remove all records (rows) from a table while retaining the table structure.

  DROP: Used to delete database objects like tables, indexes, or views along with all their data.

- ⇨ **DQL (Data Query Language):** DQL are used to retrieve data from a database.

  SELECT: The SELECT command is used to view or retrieve data from a database.

- ⇨ **DML (Data Manipulation Language):** DML commands are used to manipulate the data in the database, such as inserting, updating, and deleting rows.

  INSERT: Inserts new rows into a table.

  UPDATE: Updates existing rows in a table.

  DELETE: Deletes rows from a table.

- ⇨ **DCL (Data Control Language):** DCL commands are used to control access to the database, such as granting and revoking permissions to users.

  GRANT: The GRANT command is used to give specific permissions to users or roles in a database. These permissions can include the ability to perform actions like SELECT, INSERT, UPDATE, DELETE, and more on database objects

  REVOKE: The Revoke is used to remove or take back previously granted permissions from users or roles in a database.

- ⇨ **TCL (Transaction Control Language):** TCL commands are used to manage transactions, which are a sequence of SQL statements that are executed together as a unit.

  START TRANSACTION: Starts a new transaction in a relational database.

  COMMIT: Commits a transaction, making all the changes made during that transaction permanent and visible to other users or processes.

ROLLBACK: Rolls back a transaction, undoing all the changes made during that transaction and returning the database to its previous state.

- **What are the key components of an RDBMS?**
  - ⇨ A Relational Database Management System (RDBMS) consists of several key components that work together to store, manage, and provide access to data. Such as database, tables, schema, columns, rows, primary key, foreign key, index, constraints, SQL, transaction manager, concurrency control, storage manager, and security and access control.

- **Explain the difference between SQL and NoSQL.**
  - SQL databases are relational databases with structured data, while NoSQL databases are non-relational databases with semi-structured or unstructured data.
  - SQL scales vertically, while NoSQL scales horizontally to handle high data volumes.
  - SQL has a fixed schema, while NoSQL has a dynamic schema or is schema-less.
  - SQL performance is good for complex queries, while NoSQL performance is good for high volumes of data.
  - Examples of SQL databases are MySQL and PostgreSQL, while examples of NoSQL databases are MongoDB and Redis.

- **Primary key:** A primary key is a column or group of columns in a table that uniquely identifies each row in the table.
  There can be only one primary key in a table; it must be unique and cannot be null.

- **Foreign key:** A foreign key is a column or group of columns in a table that refers to the primary key of another table. A foreign key can have duplicate values. It establishes a relationship between two or more tables.

- **Composite key:** A composite key is a combination of two or more columns that can be used to uniquely identify each row in a table. It is a type of primary key that is useful when a single column cannot uniquely identify each row in a table. A composite key ensures unique identification and data accuracy in tables using multiple columns.

  Example:
  ```
  CREATE TABLE Employee ( EmployeeID INT, DepartmentID INT, PRIMARY KEY (EmployeeID, DepartmentID) );
  ```

- **What is schema?**
  - ⇨ A schema is a logical container that holds database objects such as tables, views, indexes, procedures, and more. It provides a way to organize and manage these objects within a database and makes it easier to maintain.

- **What are the different types of relationships in a relational database?**
  - ⇨ Any association between two or more entities (tables) is known as a relationship.

    One-to-one: Each row in one table is associated with exactly one row in another table, and vice versa. Example: A person having one passport.

    One-to-Many: Each row in one table can be associated with multiple rows in another table. Example: A customer can open multiple accounts.

    Many-to-One: Many rows in one table can be associated with a single row in another table. For example, many students can take admission to a single university.

    Many-to-Many: Multiple rows in one table can be associated with multiple rows in another table. For example, many students can take multiple courses, and each course can be taken by multiple students.

- **What are the different types of constraints in SQL?**
  - ⇨ Constraints are the rules or conditions that are applied to the data in a table. They are used to ensure the accuracy, integrity, and reliability of the data in a relational database. Constraints can be applied to individual columns or to the entire table.

  There are many different types of constraints.

  - ➞ **NOT NULL:** This constraint ensures that a column cannot have a NULL value.
  - ➞ **UNIQUE:** This constraint ensures that all values in a column are unique.
  - ➞ **Primary Key:** This constraint is a combination of NOT NULL and UNIQUE.
  - ➞ **Foreign Key:** This constraint ensures that the values in one column are related to the values in another column.
  - ➞ **Check:** This constraint ensures that the values in a column satisfy a specific condition.
  - ➞ **DEFAULT:** This constraint sets a default value for a column if no value is specified.

- **What is CRUD operations in SQL?**
  - ⇨ CRUD stands for Create, Read, Update, and Delete. These are the four basic operations that can be performed on data in a database.

- **What are the different types of clauses in SQL?**
  - ⇨ A clause is a part of a SQL statement that specifies a particular action to be taken or a condition to be applied to the data.

    Here are some of the most common SQL clauses:

    - **SELECT**: The SELECT clause is used to select data from a table.
    - **FROM**: The FROM clause specifies the table or tables from which to retrieve data in the SELECT statement. It defines the data source for the query.

- **WHERE**: The WHERE clause is used to filter rows based on specified conditions. It allows us to retrieve only the rows that meet certain criteria.

- **GROUP BY**: GROUP BY is used to group rows of data based on the values in one or more columns, creating unique groups. It is often used in conjunction with aggregate functions to calculate summary statistics for each group.

- **HAVING**: The HAVING clause is used to filter the results of a query after grouping has been performed. It is used in combination with the GROUP BY clause and allows us to apply conditions with any aggregate function, such as COUNT(), SUM(), AVG(), and MAX().

- **ORDER BY**: The ORDER BY clause is used to sort the results in ascending or descending order.

- **LIMIT**: The LIMIT clause is used to limit the number of rows returned by a query.

- **JOIN**: The JOIN clause is used to combine data from two or more tables based on a related column between them.

- **UNION**: This clause is used to combine the results of two or more SELECT statements.

- **EXISTS:** The EXISTS clause checks for the existence of rows that meet specified criteria in a subquery.

- **WITH** clause defines a temporary result set that can be referenced in the main query.

▪ The order of SQL clauses is as follows:

1. **SELECT** column1, column2
2. **FROM** table_name
3. **JOIN**
4. **WHERE** condition
5. **GROUP BY** column1
6. **HAVING** aggregate_function(condition)
7. **ORDER BY** column1 ASC
8. **LIMIT** 12;

▪ **Operators:** An operator is a symbol that is used to perform operations on data. There are many different types of operators in SQL.

- **Arithmetic operators** are used to perform mathematical operations on data, such as + , - , * , /, %.
- **Comparison Operators**: Comparison operators are used to compare values and determine the relationship between them.
  Such as: equal to (=), not equal to (!= or <>), less than (<), greater than (>), less than or equal to (<=), and greater than or equal to (>=).

- **Logical Operators:** Logical operators are used to combine multiple conditions.
  Such as: AND, OR, and NOT

- **Concatenation Operator:** This operator is used to combine strings together. It is usually represented by Double pipe ||

- **IN:** Checks if a value is within a set of values or list.
- **BETWEEN:** Checks if a value is within a range of values.
- **IS NULL / IS NOT NULL**: These operators are used to check for the presence or absence of NULL values in columns.
- **LIKE**: The LIKE operator is used for pattern matching within strings. It allows us to search for specific patterns or substrings within text columns. We can use wildcard characters with LIKE:
  - ➞ % represents zero or more characters.
  - ➞ _ represents a single character.

  For example, LIKE 'abc%' would match strings that start with "abc," and LIKE '%xyz' would match strings that end with "xyz.
- **ILIKE** (Case-insensitive version of LIKE in some databases)
- **ANY** and **ALL**: The ANY and ALL operators can be used in the WHERE clause to filter the results of a query.
  - ➞ The **ANY** operator returns TRUE if the value is equal to any of the values in the set.
  - ➞ The **ALL** operator returns TRUE if the value is equal to all of the values in the set.

▪ **What are the different types of stored procedures in SQL?**

⇨ A stored procedure is a collection of pre-written SQL statements that are saved in a database. It can be executed as a single unit and reused over and over again, which saves time and effort and improves the performance of database applications.

There are four main types of stored procedures in SQL:

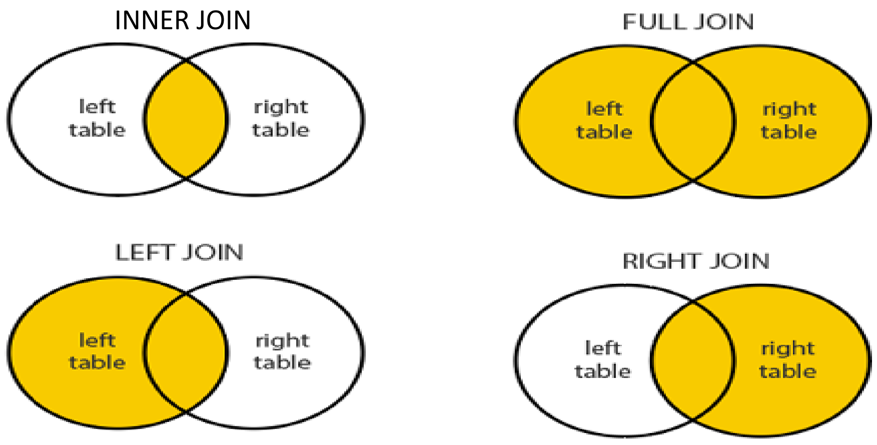
1. **System-stored procedures:** These procedures are pre-defined by the database server and are used to perform administrative tasks.
2. **User-defined stored procedures:** These procedures are created by database users and can be used to perform any SQL operation.
3. **CLR stored procedures:** These procedures are written in the.NET language and can perform tasks that are not possible with traditional SQL stored procedures.
4. **Temporary stored procedures:** These procedures are created in a temporary database and are available only to the current user session.

▪ **What are joins in SQL, and how are they used to combine data from multiple tables?**

➞ A JOIN is a clause that is used to combine rows from two or more tables based on a related column between them. We can use different types of joins to combine data from multiple tables based on the relationships between the tables. JOINS return data from multiple tables into a single result table.

MySQL

There are four main types of SQL JOINs:



INNER JOIN: An INNER JOIN returns only the rows that have matching values in both tables.

LEFT JOIN: A LEFT JOIN returns all the rows from the left table and the matching rows from the right table. If there are no matching rows in the right table, it returns NULL values for columns from the right table.

RIGHT JOIN: A RIGHT JOIN returns all rows from the right table and matching rows from the left table.

FULL JOIN: A FULL JOIN returns all rows from both tables and includes NULL values in columns regardless of whether there is a matching row in the other table.

➞ **INNER JOIN:**

```
SELECT * FROM student INNER JOIN course ON student.student_id=course.student_id;
SELECT * FROM student AS s INNER JOIN course AS c ON s.student_id=c.student_id;
```

→An alias is a temporary name or identifier assigned to a table or a column in a query result set like s and c;

➞ **LEFT JOIN:**

```
SELECT * FROM student AS s LEFT JOIN course AS c ON s.student_id=c.student_id;
```

➞ **RIGHT JOIN:**

```
SELECT * FROM student AS s RIGHT JOIN course AS c ON s.student_id=c.student_id;
```

➞ **FULL JOIN:** NOTE: In MySQL, there is no specific "FULL JOIN" keyword like we might find in some other database management systems, such as Oracle or PostgreSQL. However, we can achieve the same result as a full join in MySQL by using a combination of a left join and a right join and then combining the results using the UNION operator.

```
SELECT * FROM student AS s LEFT JOIN course AS c ON s.student_id=c.student_id
UNION
SELECT * FROM student AS s RIGHT JOIN course AS c ON s.student_id=c.student_id;
```

➞ **A right exclusive join** and **a left exclusive join** are two types of outer joins that can be used to identify rows in one table that do not have matching rows in another table.

- **A left exclusive join** returns all the rows from the left table that do not have matching rows in the right table.

```
SELECT * FROM left_table LEFT JOIN right_table ON left_table.id = right_table.id
WHERE right_table.id IS NULL;
```

- **A right exclusive join** returns all the rows from the right table that do not have matching rows in the left table.
  ```
  SELECT * FROM right_table RIGHT JOIN left_table ON right_table.id = left_table.id
  WHERE left_table.id IS NULL;
  ```

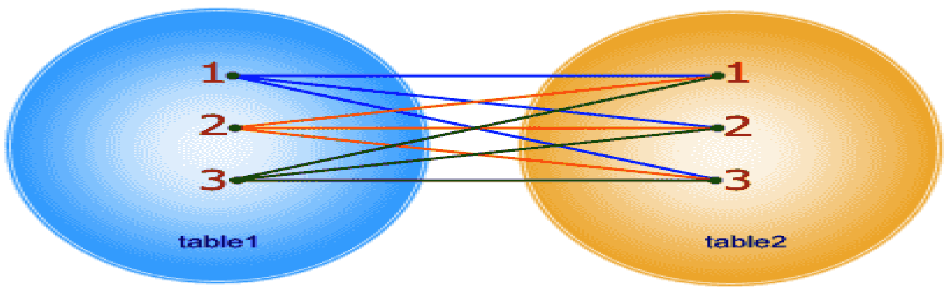
- **A full exclusive join** returns all rows from both tables, except for the rows that have matching rows in both tables.
  ➞ **Using LEFT JOIN and RIGHT JOIN:**
  ```
  SELECT * FROM TableA LEFT JOIN TableB ON TableA.key = TableB.key
  WHERE TableB.key IS NULL
  UNION
  SELECT * FROM TableA RIGHT JOIN TableB ON TableA.key = TableB.key
  WHERE TableA.key  IS NULL;
  ```
  Or
  ➞ **Using a FULL OUTER JOIN and NULL filtering:**
  ```
  SELECT * FROM TableA FULL OUTER JOIN TableB ON TableA.key = TableB.key
  WHERE TableA.key IS NULL OR TableB.key IS NULL;
  ```

- **A self-join** is a specific type of join operation where a table is joined with itself. This can be useful for finding hierarchical relationships within a table or comparing rows within the same table.
  ```
  SELECT * FROM employee JOIN employee AS b ON employee.emp_id=b.manager_id;
  ```
  Or
  ```
  SELECT a.name as manager_name,b.name FROM employee AS a JOIN employee AS b
  ON a.emp_id=b.manager_id;
  ```

▪ **What is a Cartesian product or CROSS JOIN in SQL?**
A Cartesian product, also known as a CROSS JOIN in SQL, is a type of join operation that combines all rows from one table with all rows from another table and returns all possible combinations of rows from two or more tables, regardless of whether there is any relationship between the two tables.
This operation can produce very large result sets with all possible pairings without considering any specific conditions or keys.

Pictorial Presentation of SQL Cross Join syntax:

```
SELECT * FROM table1 CROSS JOIN  table2;
```



In CROSS JOIN, each row from 1st table joins with all the rows of another table.
If 1st table contain x rows and y rows in 2nd one the result set will be x * y rows.

Example:

SELECT * FROM Bharat as B CROSS JOIN Bharat; →CROSS JOIN with same table.

SELECT * FROM customers CROSS JOIN orders; →CROSS JOIN with different tables.

- **UNION:** The UNION is used to combine the results of two or more SELECT statements into a single result set. It removes any duplicate rows from the result set.
  - The number of columns in each SELECT statement must be the same.
  - The data types of the corresponding columns in each SELECT statement must be compatible (similar).
  - The columns in each SELECT statement must be in the same order.

  NOTE: We can use the UNION ALL operator to combine the results of two or more SELECT statements into a single result set, without removing any duplicate rows.

➞ **UNION:** SELECT name FROM employee UNION SELECT name FROM employee;

Subquery: A subquery is a query that is nested within another SQL query. It is also known as a nested query or an inner query. A subquery allows us to retrieve data from one or more tables based on the result of another query. Subqueries can be used in various parts of a SQL statement, including the SELECT, FROM, WHERE, and HAVING clauses. Example:

- **Find the name and marks of those students whose marks are higher than the average marks of the class.**
  SELECT name, marks FROM student WHERE marks>(SELECT avg(marks) FROM student);

- **Find the name and ID of those students whose ID is even.**
  SELECT name,id FROM student WHERE id IN (SELECT id FROM student WHERE id%2=0);

- **Find out all the details of the student who lives in Delhi and has the highest marks.**
  SELECT * FROM student WHERE marks=(SELECT MAX(marks) FROM student WHERE city="Delhi"); Or

  SELECT * FROM student WHERE city = 'Delhi' ORDER BY marks DESC LIMIT 1; Or

  SELECT MAX(marks) FROM (SELECT * FROM student WHERE city="Delhi") AS Delhi_stu;

- **VIEW:** A view is a virtual table based on a SQL query that allows us to present data from one or more tables in a different way. It can be used to simplify complex queries, enhance data security by restricting access to unneeded data, or create custom data representations.

  There are two main types of views in SQL:

  1. **Simple views** are based on a single table and are used for basic data presentation.

2. **Complex views** are based on multiple tables and can include joins, aggregations, and other complex operations. They are used for more advanced data manipulation and abstraction.

Example:
```
CREATE VIEW view1 AS SELECT id,marks FROM student;

CREATE VIEW view2 AS SELECT id,marks FROM student WHERE marks>90;

SELECT * FROM view2;

DROP VIEW view2;
```

- **TRIGGERS:** A trigger is a special type of database object that automatically runs SQL code when a specific event occurs on a table. Triggers are used to enforce data integrity, automate tasks, and audit changes to data, improving the efficiency and accuracy of SQL code.
  There are three main types of triggers in SQL:
  - **DML triggers:** These triggers are fired when DML statements (such as INSERT, UPDATE, and DELETE) are executed on a table.
  - **DDL triggers:** These triggers are fired when DDL statements (such as CREATE, ALTER, and DROP) are executed on a database.
  - **Logon triggers:** These triggers are fired when a user logs in to a database.

- **Indexes or Indexing:** An index is a database structure that enhances data retrieval by organizing data in a table. Indexes are created on one or more specific columns of a table, and they store the values of those columns in a sorted order. This enables the database to swiftly locate rows in the table that meet specific criteria. For example:
  ```
  CREATE INDEX index_name ON table_name (column_name1, column_name2, ...);
  ```

- **Normalization** is the process of organizing data in a database to reduce redundancy and improve data integrity. It is achieved by dividing a database into multiple tables and establishing relationships between them.

- **Denormalization** is a technique of combining data into a single table to make data retrieval faster. This is done by intentionally adding redundancy to the database.

SQL Data types

CHAR: Fixed-length character string. Range: (0-255)
VARCHAR: Variable-length character string. Range: (0-255)
BLOB: Binary large object.
INT: Integer within a specific range. Range: -2,147,483,648 to 2,147,483,647.
TINYINT: Small integer within a specific range. Range: -128 to 127.
BIGINT: Large integer within a specific range.
BIT: A specified number of bits (often used for boolean values).
FLOAT: Decimal number with precision to 23 digits.
DOUBLE: Decimal number with precision to 24 to 53 digits.
BOOLEAN: It representations are used for Boolean values: 0 for false and 1 for true.
DATE: Date in the format 'YYYY-MM-DD'.
YEAR: 4-digit year value (typically used for years).
A signed data type can store both positive and negative numbers. But a unsigned data type can only store positive numbers. Example: The range of TINYINT is -128 to 127; if I make it unsigned, then the range will be increased, like 128 + 127 = 255. TINYINT UNSIGNED (0 to 255)

- CREATE DATABASE Oracle; ⇨ Use for create database.
- DROP DATABASE world; ⇨ Use for delete database.
- USE Oracle; ⇨ Start using of a specific database.
- CREATE TABLE Student(ID VARCHAR(12) PRIMARY KEY, Name VARCHAR(24), Age INT NOT NULL, Mobile BIGINT); ⇨ Creating a table.
- INSERT INTO table_name (column1, column2, ...) VALUES (value1, value2, ...); ⇨ Inserting values.
    - INSERT INTO Students VALUES("S1", "Ravi",24 , 8728373483);
    - INSERT INTO employees (employee_id, employee_name, salary) VALUES (1, 'John Doe', 50000), (2, 'Jane Smith', 60000), (3, 'Bob Johnson', 55000);
- SELECT * FROM student; ⇨ View created table.
- CREATE DATABASE IF NOT EXISTS college;
- DROP DATABASE IF EXISTS college;
- SHOW DATABASES;
- SHOW TABLES;

MySQL

- DROP TABLE student; → It is used to delete or remove the table from the database.
- CREATE TABLE emp(ID int, salary FLOAT DEFAULT 25000, PRIMARY KEY(ID));

- CREATE TABLE stu(ID INT PRIMARY KEY, address VARCHAR(24), age INT, CONSTRAINT CHECK(age>5 AND address="Delhi"));

  INSERT INTO stu value(1,"Delhi",6); →1 row(s) affected
  INSERT INTO stu value(2,"Mumbai",6); →Error Code: Check constraint 'stu_chk_1' is violated.

- SELECT name, roll_no FROM student;

- SELECT DISTINCT city FROM student; → The DISTINCT keyword is used to fetch unique values from one or more columns in a table, removing duplicates from the result set of a SELECT statement.

- SELECT * FROM student WHERE marks > 85;

- SELECT * FROM student WHERE marks > 90 AND city='Chandigarh';

- SELECT * FROM student WHERE marks+10 > 100;

- SELECT * FROM student WHERE marks > 90 OR city='Delhi';

- SELECT * FROM student WHERE marks BETWEEN 80 AND 90;

- SELECT * FROM student WHERE city IN('Chandigarh','Delhi');

- SELECT * FROM student LIMIT 3;

- SELECT * FROM student ORDER BY city ASC;

- SELECT * FROM student ORDER BY marks DESC LIMIT 3;

- **Aggregate functions** are functions in SQL that operate on a set of values and return a single value.
  Here are some of the most common aggregate functions in SQL:
  COUNT - Returns the number of rows in a set.
  SUM - Returns the sum of all the values in a column.
  AVG - Returns the average of all the values in a column.
  MIN - Returns the minimum value in a column.
  MAX - Returns the maximum value in a column.
  COUNT DISTINCT: Counts the number of unique values in a column.
  STDDEV - Returns the standard deviation of all the values in a column.
  VARIANCE - Returns the variance of all the values in a column.

- SELECT MAX(marks) FROM student;
- SELECT AVG(marks) AS avg_marks FROM student;
- SELECT city FROM student GROUP BY city;
- SELECT city, COUNT(name) FROM student GROUP BY city;
- SELECT city, AVG(marks) AS average_marks FROM student GROUP BY city;

MySQL

➞ SELECT city, AVG(marks) AS average_marks FROM student GROUP BY city ORDER BY average_marks DESC;

➞ SELECT grade, COUNT(roll_no) AS No_of_student FROM student GROUP BY grade ORDER BY grade ASC;

➞ SELECT city, count(roll_no) FROM student GROUP BY city HAVING max(marks) > 90;

➞ SELECT city FROM student WHERE grade='A' GROUP BY city HAVING max(marks)>91 ORDER BY city ASC;
➞ UPDATE student SET grade="O" WHERE grade='A'; → Off safe mode firstly (SET SQL_SAFE_UPDATES=0;)

➞ UPDATE student SET marks=83 WHERE roll_no=105;

➞ UPDATE student SET grade='A' WHERE marks BETWEEN 80 AND 90;

➞ UPDATE student SET marks=marks+1;

➞ DELETE FROM student WHERE marks <33;

➞ **FORIGN KEY:** CREATE TABLE dept(id INT PRIMARY KEY, name VARCHAR(45));
INSERT INTO dept VALUES(101, 'Science'), (102,'Math');

CREATE TABLE teacher(id INT PRIMARY KEY, name VARCHAR(34),
dept_id INT, FOREIGN KEY(dept_id) REFERENCES dept(id)
ON UPDATE CASCADE ON DELETE CASCADE);

INSERT INTO teacher VALUES(1, "Adaue",102), (2, "Ranfd", 101);

UPDATE dept SET id=105 WHERE id=102; Note: CASCADE is used to maintain referential integrity constraints, particularly with foreign keys, to ensure that child records are automatically updated or deleted when the corresponding parent record is updated or deleted. This helps to maintain data consistency and integrity in the database.

➞ ALTER TABLE student ADD COLUMN age INT; →UPDATE student SET age=22 WHERE roll_no=104;
➞ ALTER TABLE student RENAME COLUMN age TO stu_age;
➞ ALTER TABLE student DROP COLUMN stu_age;
➞ ALTER TABLE student RENAME TO stu;
➞ ALTER TABLE student CHANGE COLUMN age stu_age VARCHAR(3);
➞ ALTER TABLE student ADD COLUMN age INT NOT NULL DEFAULT 21;
➞ ALTER TABLE student MODIFY COLUMN age VARCHAR(3);

➞ TRUNCATE TABLE student; → It removes all rows from the table while retaining the table structure, including columns and constraints.

Basic SQL Queries:

- SQL query to select **all records** / **all rows** / **all columns** from a Student table in SQL?
  SELECT * FROM student;

- SQL query to select the first name, last name, and salary of all employees who work in the IT department.
  SELECT firstname, lastname, salary FROM employees WHERE department = 'IT';

- SQL query to find all customers who have placed an order in the last month.
  SELECT customer_id, customer_name FROM customers
  INNER JOIN orders ON customers.customer_id = orders.customer_id
  WHERE order_date >= CURRENT_DATE - INTERVAL 1 MONTH;

- SQL query to find the top 10 customers by order value.
  SELECT customer_id, customer_name, SUM(order_value) AS total_order_value
  FROM customers
  INNER JOIN orders ON customers.customer_id = orders.customer_id
  GROUP BY customer_id, customer_name
  ORDER BY total_order_value DESC LIMIT 10;

- SQL query to select the names of all employees who have been with the company for more than 5 years.
  SELECT firstname, lastname FROM employees
  WHERE hire_date < (CURRENT_DATE - INTERVAL 5 YEAR);

- SQL query to select the names of all employees who are managers.
  SELECT firstname, lastname FROM employees
  WHERE designation = 'manager'; →or

  SELECT firstname, lastname FROM employees
  WHERE employee_id IN (SELECT manager_id FROM employees);

- SQL query to join the employees table and the departments table to get the name of each employee and the name of their department.
  SELECT e.employee_name, d.department_name FROM employees e
  JOIN departments d ON e.department_id = d.department_id;

- SQL query to order the results of a query by the salary in descending order.
  SELECT employee_name, salary FROM employees ORDER BY salary DESC;

- SQL query to filter the results of a query to only include employees who make more than 10,000,00 per year.
  SELECT employee_name, salary FROM employees WHERE salary > 1000000;

MySQL

- SQL query to group the results of a query by the department and then calculate the average salary for each department. Or SQL query to find the average salary for each department.
  SELECT department_name, AVG(salary) AS average_salary FROM employees GROUP BY department_name;

- SQL query to find the total number of records in a table.
  SELECT COUNT(*) FROM table_name;

- Sort the result of an SQL query in ascending and descending order?
  SELECT column1, column2 FROM TableName ORDER BY column1 DESC, column2 ASC;

  SELECT * FROM customers ORDER BY customer_name ASC;

- SQL query to calculate the average value of a numeric column.
  SELECT AVG(numeric_column) AS AverageValue FROM TableName;

- How do you filter records to display only those where a specific column equals a certain value?
  SELECT * FROM TableName WHERE specific_column = 'certain_value';

  Example:— SELECT * FROM customers WHERE customer_name = 'John Doe';

- How do you update existing records in an SQL table?
  UPDATE TableName SET column1 = new_value1, column2 = new_value2, ... WHERE condition;

  UPDATE Employees SET Employee_name = 'Jane Doe', Employee_salary = 70000 WHERE Employee_id = 123;

- SQL query to retrieve the top 5 highest-paid employees from an "Employees" table.
  SELECT * FROM Employees ORDER BY salary DESC LIMIT 5;

- To find the second highest salary from an "employees" table.
  SELECT DISTINCT Salary FROM Employees ORDER BY Salary DESC
  LIMIT 1 OFFSET 1; → LIMIT 1 limits the result set to just one row.
  → OFFSET 1 skips the first row (highest salary) and retrieves the second highest salary.

- SQL query to delete all records from a table where a certain condition is met.
  DELETE FROM Employees WHERE salary < 5000;

- SQL query to find the total number of employees who have a salary greater than 100,000.
  SELECT COUNT(*) AS total_employees FROM employees WHERE salary > 100000;

- SQL query to find the total number of orders placed by each customer, and list the customer ID along with the order count.
  ```
  SELECT CustomerID, COUNT(OrderID) AS OrderCount FROM Orders
  GROUP BY CustomerID;
  ```

- SQL query to find departments that have a total salary greater than 1 million:
  ```
  SELECT department, SUM(salary) AS total_salary FROM employees
  GROUP BY department HAVING SUM(salary) > 1000000;
  ```

- Write a MySQL query to find the second highest salary from the "employees" table.
  ```
  SELECT MAX(salary) AS SecondHighestSalary
  FROM employees WHERE salary < (SELECT MAX(salary) FROM employees);
  ```

- **Create a new user 'ankit' with the password '1998'**
  ```
  CREATE USER 'ankit'@'localhost' IDENTIFIED BY '1998';
  ```

  Grant all privileges to the user 'ankit' on all databases and tables
  The 'WITH GRANT OPTION' allows 'ankit' to grant or revoke privileges of other users
  ```
   GRANT ALL PRIVILEGES ON *.* TO 'ankit'@'localhost' WITH GRANT OPTION;
  ```

  Apply the changes immediately
  ```
  FLUSH PRIVILEGES;
  ```

  Change the password for the user 'ankit' to 'new_password'
  This assumes that 'ankit' is an existing user and you have the necessary privileges to alter it
  ```
  ALTER USER 'ankit'@'localhost' IDENTIFIED BY 'new_password';
  ```

- **What is SQL injection and how to prevent it.**
  ⇨ SQL injection is a type of cyberattack where hackers inject malicious code into a web application's input fields to gain unauthorised access to a database or manipulate its data.
  To prevent SQL injection, we can use:
  – Use parameterized statements.
  – Limit database user permissions.
  – Sanitize user input.
  – Keep our software up to date.
  – Use a web application firewall.
  – Use strong passwords and security practices.
  – Monitor our database for suspicious activity.

- **Cursors** are pointers to data in a database, and PL/SQL controls the context through a cursor.

  There are two types of cursors:
  - **Implicit cursors** are automatically created by Oracle whenever an SQL statement is executed. They are typically used for DML statements such as INSERT, UPDATE, and DELETE.
  - **Explicit cursors** are defined by the programmer to gain more control over the context area. It can be used for any type of SQL statement, including SELECT statements.

॥ॐ नमः शिवाय॥

|| ॐ तत्पुरुषाय विद्महे, महादेवाय धीमहि, तन्नो रूद्र प्रचोदयात् ||

♨୭★ଓ♨