SQL Interview question and answer

1. What is the difference between inner join and outer join ?

INNER JOIN

* Definition: Returns only the matching rows from both tables based on the join condition.
* Unmatched rows: Ignored (not included in the result).
* Usage: Common when you only want data that exists in both tables.

**OUTER JOIN**

* **Definition**: Returns **all rows** from one or both tables, even if there’s **no match**.
* There are 3 types:
  1. **LEFT OUTER JOIN (LEFT JOIN)** – All rows from the **left** table + matching rows from the right. Unmatched right side will be NULL.
  2. **RIGHT OUTER JOIN (RIGHT JOIN)** – All rows from the **right** table + matching rows from the left. Unmatched left side will be NULL.
  3. **FULL OUTER JOIN** – All rows from **both** tables. Unmatched rows from either side will be filled with NULL.

1. What is aggregate function in SQL ?

An **aggregate function** in SQL **performs a calculation on a set of values** and returns a **single summary value** (such as sum, average, count, etc.).

**🔸 Common Aggregate Functions:**

| **Function** | **Description** | **Example** |
| --- | --- | --- |
| SUM() | Returns the total sum of a numeric column | SELECT SUM(salary) FROM employees; |
| AVG() | Returns the average value | SELECT AVG(age) FROM students; |
| COUNT() | Returns the number of rows | SELECT COUNT(\*) FROM orders; |
| MAX() | Returns the maximum value | SELECT MAX(score) FROM results; |
| MIN() | Returns the minimum value | SELECT MIN(price) FROM products; |

**🔹 Usage with GROUP BY:**

Aggregate functions are often used with the GROUP BY clause to group rows before performing the calculation.

SELECT department, AVG(salary)

FROM employees

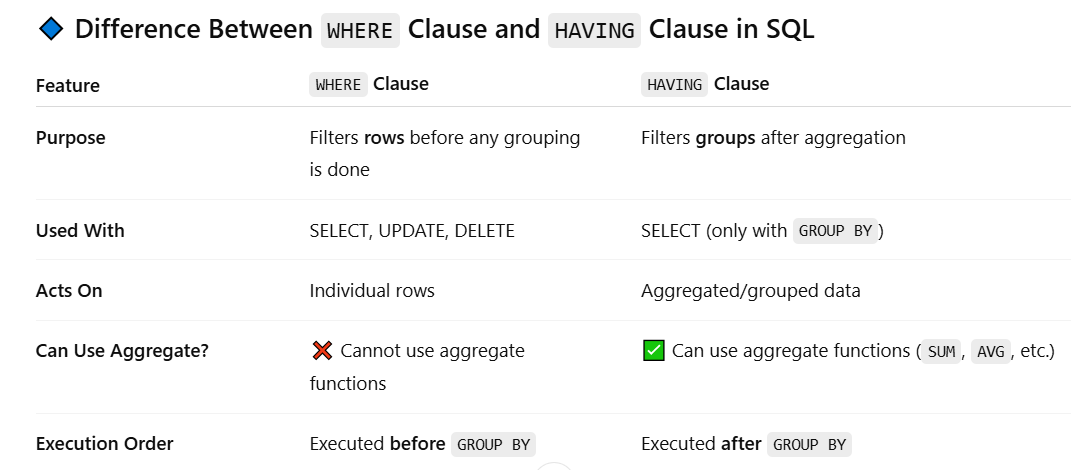
GROUP BY department;

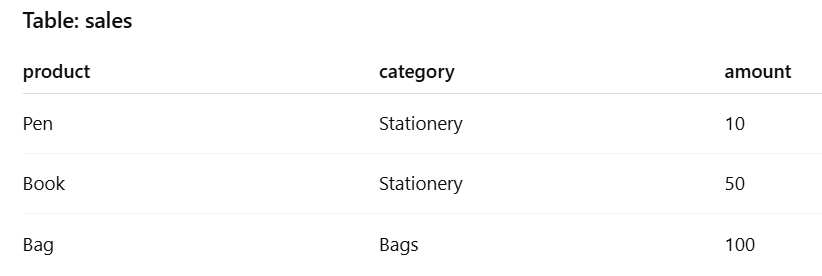
This query gives the **average salary per department**.

**🔸 Key Points:**

* Aggregate functions ignore NULL values (except COUNT(\*)).
* They are used to summarize data.
* Cannot be used in WHERE clause directly — use HAVING instead.

1. Differnce between where clause and having clause ?





SELECT \* FROM sales

WHERE category = 'Stationery';

Filters rows where category is **'Stationery'** before grouping or aggregating.

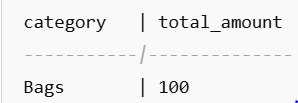
SELECT category, SUM(amount) as total\_amount

FROM sales

GROUP BY category

HAVING SUM(amount) > 60;

Output:



**🔸 Summary in Simple Terms:**

* **WHERE** = Filter **rows**
* **HAVING** = Filter **groups (after aggregation)**

**4) What Are the SQL Languages?**

SQL (Structured Query Language) is divided into **five main categories of sub-languages**, each serving a specific purpose in managing and interacting with a relational database.

**✅ 1. DDL – Data Definition Language**

Used to **define or modify the structure** of database objects like tables, schemas, indexes, etc.

**Commands:**

* CREATE – Create a new table or database
* ALTER – Modify an existing object
* DROP – Delete an object
* TRUNCATE – Remove all records from a table quickly

**✅ 2. DML – Data Manipulation Language**

Used to **manipulate data** (insert, update, delete) in existing tables.

**Commands:**

* INSERT – Add new data
* UPDATE – Modify existing data
* DELETE – Remove data
* MERGE – Combine insert/update in one operation (used in some DBs)

**✅ 3. DQL – Data Query Language**

Used to **query or retrieve data** from a database.

**Command:**

* SELECT – Fetch data from tables

This is the most commonly used SQL command in applications.

**✅ 4. DCL – Data Control Language**

Used to **control access** to data and manage permissions.

**Commands:**

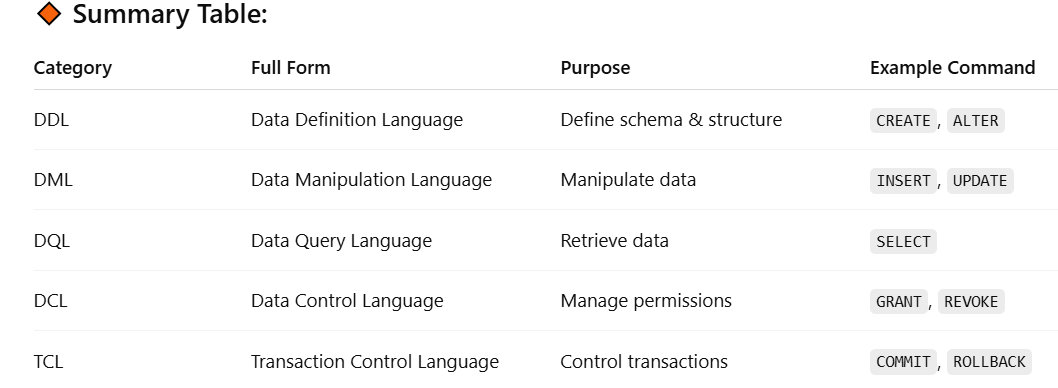
* GRANT – Give access/permissions
* REVOKE – Take back access

**✅ 5. TCL – Transaction Control Language**

Used to **manage transactions** in a database to ensure data integrity.

**Commands:**

* COMMIT – Save all changes
* ROLLBACK – Undo changes
* SAVEPOINT – Create a point to roll back to
* SET TRANSACTION – Define characteristics of a transaction



**🔹 What is Normalization in SQL?**

Normalization is the process of organizing data in a database to minimize

redundancy and dependency. It involves breaking down a table into

smaller tables and establishing relationships between them to eliminate

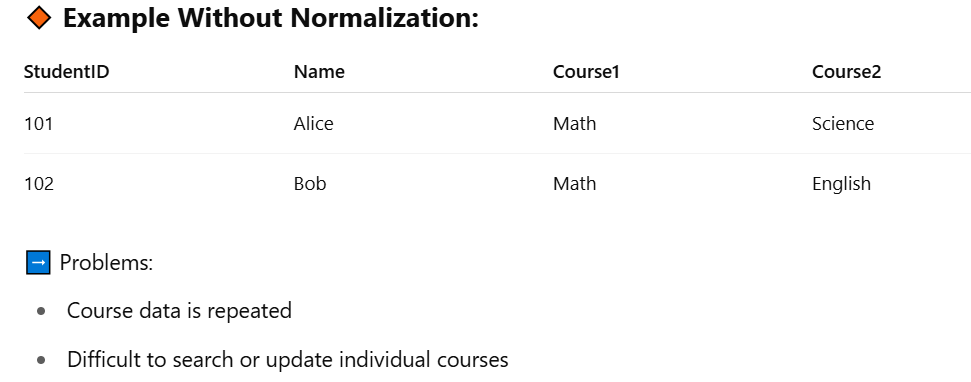
data anomalies.

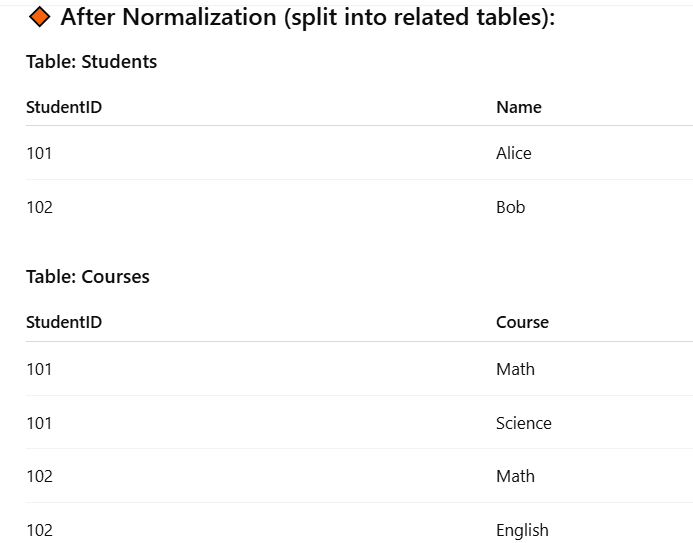
**Normalization** is the process of **organizing data in a database** to:

* Reduce data **redundancy** (duplicate data)
* Improve **data integrity**
* Make the database **more efficient and consistent**

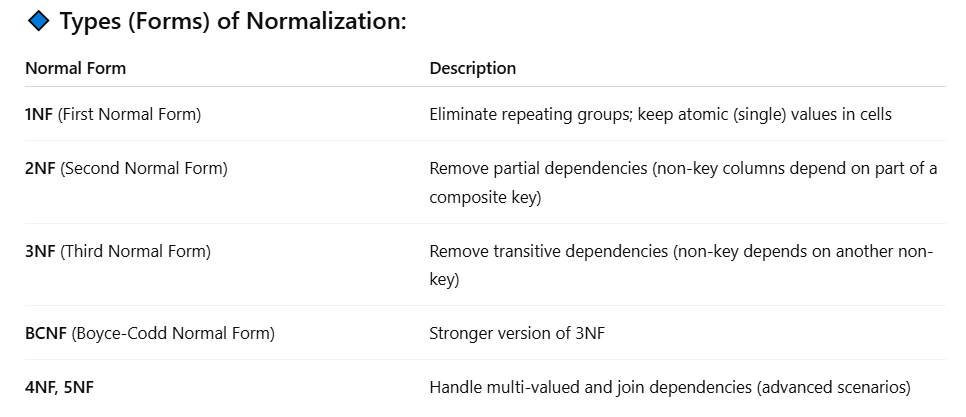
**✅ Goals of Normalization:**

1. Eliminate **duplicate data**
2. Ensure **logical data storage**
3. Simplify **updates, inserts, and deletes** (avoiding anomalies)
4. Create **smaller, related tables** and link them using **foreign keys**



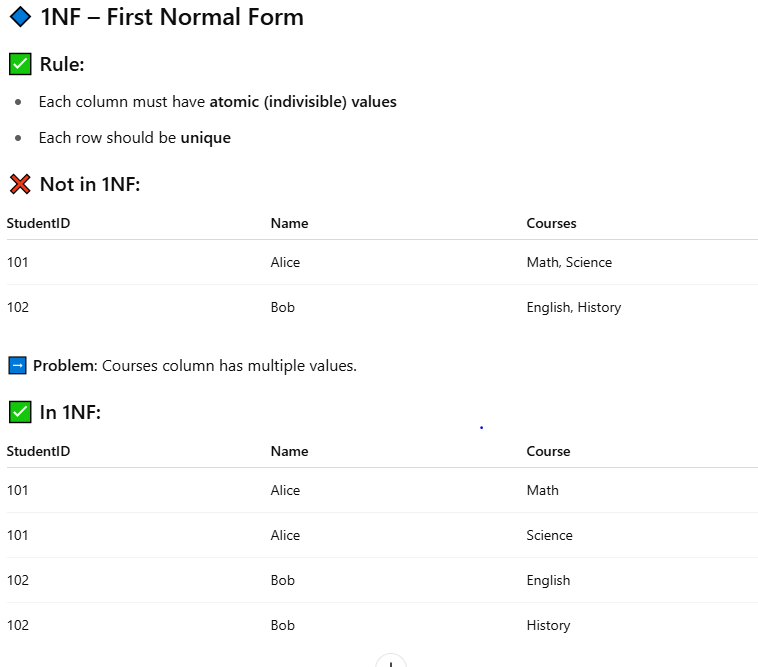


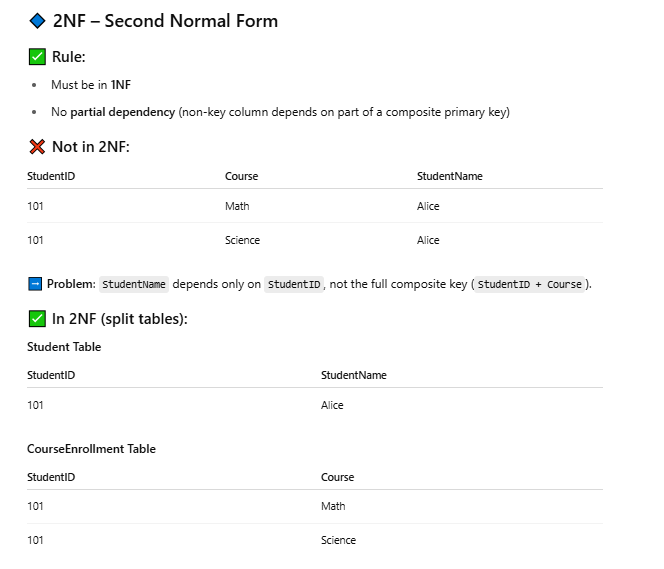
➡ Easier to maintain and query.

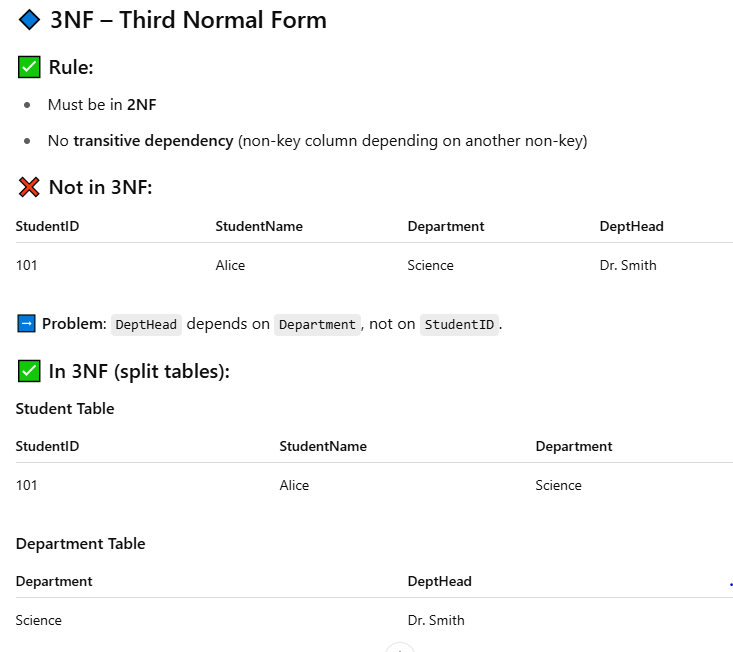


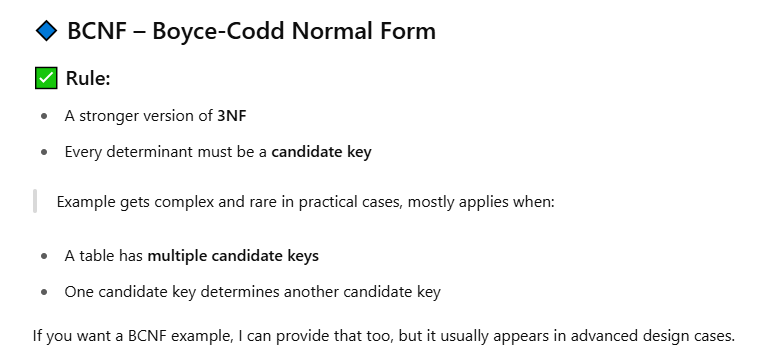
**🔸 In Simple Terms:**

**Normalization = Breaking big tables into smaller related ones** to avoid duplication and inconsistency.





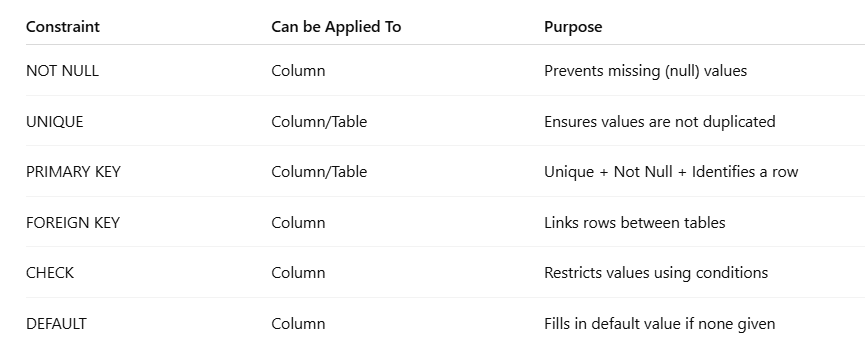




**🔹 What is a Constraint in SQL?**

A **constraint** is a rule applied to a column or table to **enforce data integrity, consistency, and accuracy** in the database.

Constraints ensure that **only valid data** is stored in a table — for example, preventing NULL values or duplicate entries.



**What is Indexing in SQL?**

An **index** in SQL is a **performance-tuning method** that allows faster retrieval of records from a database table.

Think of it like an index in a book — instead of scanning every page (row), you can quickly jump to the exact location.

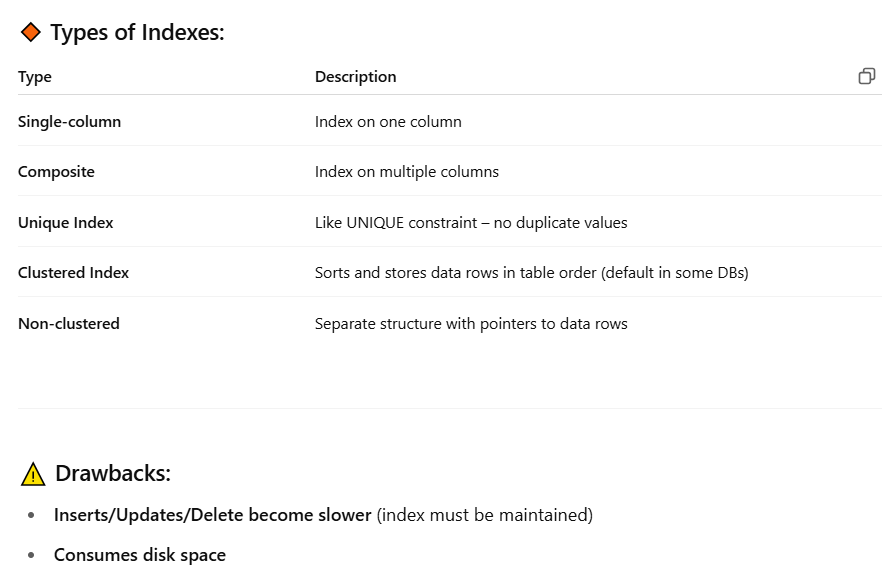
**✅ Benefits of Indexing:**

* **Faster SELECT queries**
* Efficient searching and sorting
* Improves performance for **WHERE**, **JOIN**, **ORDER BY** operations

Syntax

CREATE INDEX index\_name ON table\_name(column\_name);

CREATE INDEX idx\_customer\_name ON customers(customer\_name);



**🔹 What is a View in SQL?**

A **view** is a **virtual table** based on the result of a SELECT query. It **does not store data** itself, but shows data from one or more tables.

**✅ Benefits of Views:**

* Simplifies complex queries
* Enhances security (can expose only selected data)
* Makes reports and logic reusable

Syntax

======

CREATE VIEW view\_name AS

SELECT column1, column2

FROM table\_name

WHERE condition;

Example

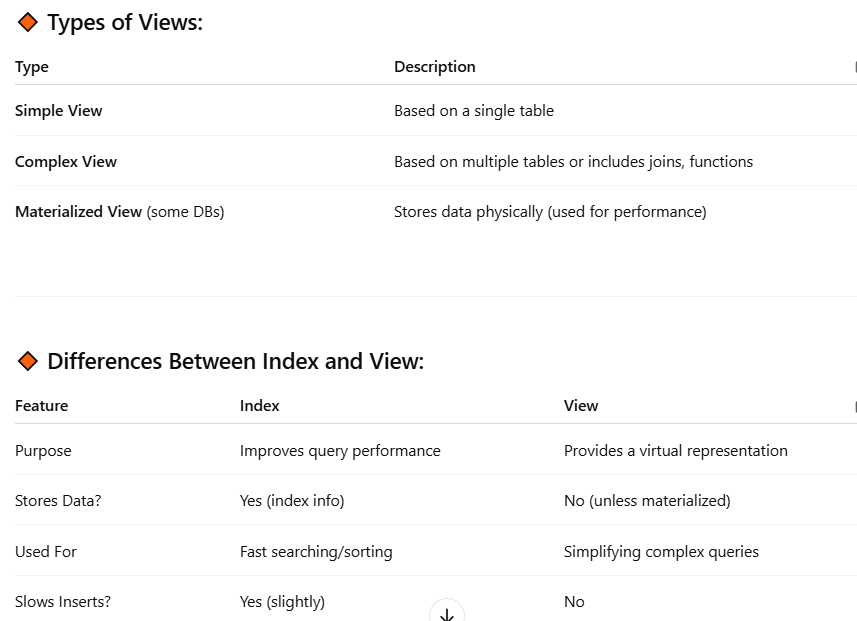
=======

CREATE VIEW active\_customers AS

SELECT customer\_id, name

FROM customers

WHERE status = 'Active';



What is the difference between UNION and UNION ALL?

UNION removes duplicates.

UNION ALL keeps all records including duplicates.

SELECT city FROM customers

UNION

SELECT city FROM suppliers;

SELECT city FROM customers

UNION ALL

SELECT city FROM suppliers;

**How do you find the second highest salary from a table?**

SELECT MAX(salary)

FROM employees

WHERE salary < (SELECT MAX(salary) FROM employees);

SELECT salary FROM employees

ORDER BY salary DESC

LIMIT 1 OFFSET 1;

What is a subquery?

A subquery is a query inside another query.

SELECT name FROM employees

WHERE dept\_id IN (SELECT id FROM departments WHERE location = 'NY');

What is the difference between DELETE, TRUNCATE, and DROP?

* **DELETE: removes specific rows, can be rolled back, uses WHERE.**
* **TRUNCATE: removes all rows, cannot be rolled back, faster than DELETE.**
* **DROP: removes the entire table from the database.**

**What is the difference between RANK(), DENSE\_RANK(), and ROW\_NUMBER()?**

RANK(): Skips ranks if there are ties.

DENSE\_RANK(): No gaps in ranking.

ROW\_NUMBER(): Assigns a unique number to each row.

SELECT name, salary, RANK() OVER (ORDER BY salary DESC) AS rank

FROM employees;

How to fetch duplicate records in a table?

SELECT name, COUNT(\*)

FROM employees

GROUP BY name

HAVING COUNT(\*) > 1;

What is the difference between a **clustered** and **non-clustered index**?

**Clustered index**: Reorders the actual data to match the index (only one per table).

**Non-clustered index**: Contains pointers to the data (can be many per table).

Clustered index: It determines the physical order of data rows in a table.

Each table can have only one clustered index, and it affects the way data is

stored on disk.

Non-clustered index: It is a separate structure that contains a sorted copy

of selected columns from a table. A table can have multiple non-clustered

indexes, and they do not affect the physical order of data rows.

11.What is the difference between UNION and UNION ALL in SQL?

UNION: It is used to combine the result sets of two or more SELECT

statements into a single result set. It removes duplicate rows from the

result set.

UNION ALL: It also combines the result sets of two or more SELECT

statements into a single result set but does not remove duplicate rows. It

is faster than UNION as it does not perform duplicate removal

12. What is ACID in the context of database transactions?

ACID stands for Atomicity, Consistency, Isolation, and Durability. It is a set

of properties that guarantee reliable and consistent transactions in a

database.

Atomicity: Ensures that a transaction is treated as a single unit of work,

either fully completed or fully rolled back.

Consistency: Ensures that a transaction brings the database from one

valid state to another. It preserves the integrity of the data.

Isolation: Ensures that concurrent transactions do not interfere with each

other. Each transaction is isolated and operates as if it is the only

transaction running.

Durability: Ensures that once a transaction is committed, its changes are

permanent and will survive any subsequent failures.

16. What is the difference between a stored procedure and a function in

SQL?

Stored procedure: It is a named set of SQL statements that are stored in

the database. It can have input and output parameters, and it may or may

not return a value. Stored procedures are used for performing complex

database operations.

Function: It is a named program unit that returns a value. It always returns

a value and can be used in SQL statements wherever an expression can be

used. Functions are used for calculations and data manipulation.

What is self join ?

A self-join is a type of join where a table is joined with itself. It is used

when data in a table relates to other data within the same table. By using

aliases for the table, different rows in the table can be joined together.

Write an SQL query to get the latest record per group.

SELECT \*

FROM orders o

WHERE o.order\_date = (

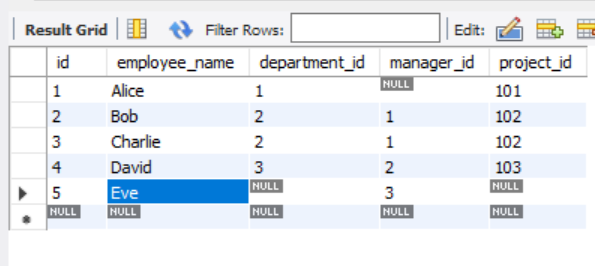
SELECT MAX(order\_date)

FROM orders

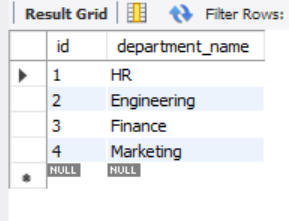
WHERE customer\_id = o.customer\_id

);

Employee table



Department



// Get all employees and their department names.

SELECT e.employee\_name, d.department\_name

FROM employees e

INNER JOIN departments d ON e.department\_id = d.id;

// Get all employees, including those who are not assigned to any department.

SELECT e.employee\_name, d.department\_name

FROM employees e

LEFT JOIN departments d ON e.department\_id = d.id;

// Get all departments and the employees assigned to them.

SELECT e.employee\_name, d.department\_name

FROM employees e

RIGHT JOIN departments d ON e.department\_id = d.id;

// Get all employees and all departments, even if they don’t match.

-- For databases that support FULL OUTER JOIN:

SELECT e.employee\_name, d.department\_name

FROM employees e

FULL OUTER JOIN departments d ON e.department\_id = d.id;

-- MySQL alternative using UNION:

SELECT e.employee\_name, d.department\_name

FROM employees e

LEFT JOIN departments d ON e.department\_id = d.id

UNION

SELECT e.employee\_name, d.department\_name

FROM employees e

RIGHT JOIN departments d ON e.department\_id = d.id;

//Find managers and their subordinates from an employee table.

SELECT e.employee\_name AS employee, m.employee\_name AS manager

FROM employees e

JOIN employees m ON e.manager\_id = m.id;

// Get departments and count of employees in each.

SELECT d.department\_name, COUNT(e.id) AS employee\_count

FROM departments d

LEFT JOIN employees e ON e.department\_id = d.id

GROUP BY d.department\_name;

// Get employee names, their department, and the project they are working on.

SELECT e.employee\_name, d.department\_name, p.project\_name

FROM employees e

JOIN departments d ON e.department\_id = d.id

JOIN projects p ON e.project\_id = p.id;

// Get employees who are **not assigned** to any department.

SELECT e.employee\_name

FROM employees e

LEFT JOIN departments d ON e.department\_id = d.id

WHERE d.id IS NULL;

// Find departments that have more than 1 employee.

SELECT d.department\_name, count(e.id)

FROM employees e

JOIN departments d ON e.department\_id = d.id group by d.department\_name having count(e.id) > 1;

// List employee names along with their manager names.

SELECT e.employee\_name, m.employee\_name

FROM employees e

JOIN employees m ON e.id = m.manager\_id;

// List all departments with the number of employees, sorted by highest number of employees.

select d.department\_name, count(e.employee\_name) FROM employees e

JOIN departments d ON e.department\_id = d.id group by d.department\_name order by count(e.employee\_name) desc;

//find the employee whose salary is greater than or equal to manager salary

Select e.\* from Employee e JOIN Employee m ON e.manager\_id = m.emp\_id

WHERE e.salary >= m.salary;