| **Aspect** | **Structured Database** | **Unstructured Database** |
| --- | --- | --- |
| **Definition** | Stores data in a **fixed, organized format** like tables | Stores data in **no predefined format** or structure |
| **Data Type** | Relational data (numbers, text in rows/columns) | Multimedia, images, videos, emails, documents, social media data |
| **Storage Model** | Uses **Relational Database Management Systems (RDBMS)** | Uses **NoSQL databases**, file systems, data lakes |
| **Schema** | Has a **predefined schema** (data model is fixed) | **Schema-less** or flexible schema |
| **Examples** | MySQL, Oracle, PostgreSQL, SQL Server | MongoDB, Cassandra, Hadoop, Elasticsearch, file systems |
| **Data Retrieval** | SQL queries, easy to filter and retrieve | Requires specialized tools (e.g., text mining, AI, search engines) |
| **Scalability** | Vertical scaling (add more power to a single server) | Horizontal scaling (add more servers to distribute data) |
| **Usage** | Banking, ERP systems, inventory, CRM | Big Data, social media, image repositories, IoT data |
| **Data Integrity & Accuracy** | High data integrity and constraints enforcement | Less structured, may need preprocessing for accuracy |
| **Speed for Querying** | Faster for structured queries with defined relationships | Slower or complex querying without structure |

**Summary:**

* **Structured Database** = Organized, relational, predictable, tabular data.
* **Unstructured Database** = Free-form, diverse formats, large volume data (Big Data).

Let me know if you want differences with examples or a focus on semi-structured data too!

Sql Questions-

### ****Display employees along with their experience in years****

SELECT name,

join\_date,

TIMESTAMPDIFF(YEAR, join\_date, CURDATE()) AS experience\_years

FROM employees;

### ****15. Retrieve the second highest salary****

SELECT MAX(salary) AS second\_highest\_salary

FROM employees

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

### ****16.Find employees who don’t belong to any department****

SELECT \*

FROM employees

WHERE department\_id IS NULL;

### ✅ ****17. Get the number of employees hired per year****

SELECT YEAR(join\_date) AS year\_hired, COUNT(\*) AS employee\_count

FROM employees

GROUP BY YEAR(join\_date);

### ****Write a query to swap department names between two departments (for example, dept\_id 1 and 2)****

UPDATE departments

SET department\_name = CASE

WHEN department\_id = 1 THEN (SELECT department\_name FROM departments WHERE department\_id = 2)

WHEN department\_id = 2 THEN (SELECT department\_name FROM departments WHERE department\_id = 1)

END

WHERE department\_id IN (1, 2);

### ****Get the employees with the same salary****

SELECT e1.name, e1.salary

FROM employees e1

JOIN employees e2 ON e1.salary = e2.salary AND e1.id != e2.id;

### ✅ ****25. Retrieve the employee with the longest name****

SELECT name

FROM employees

ORDER BY LENGTH(name) DESC

LIMIT 1;

### ****26. Write a query to list departments that have more than 5 employees****

SELECT department\_id, COUNT(\*) AS emp\_count

FROM employees

GROUP BY department\_id

HAVING COUNT(\*) > 5;

### ✅ ****27. Find employees who joined on the same day****

SELECT join\_date, COUNT(\*) AS emp\_count

FROM employees

GROUP BY join\_date

HAVING COUNT(\*) > 1;

### ****Get cumulative salary by department ordered by salary (Window Function)****

SELECT name, department\_id, salary,

SUM(salary) OVER(PARTITION BY department\_id ORDER BY salary) AS cumulative\_salary

FROM employees;

### ✅ ****29. Find employees with a salary greater than the average salary of their department****

SELECT e.name, e.salary, e.department\_id

FROM employees e

JOIN (

SELECT department\_id, AVG(salary) AS avg\_salary

FROM employees

GROUP BY department\_id

) d\_avg ON e.department\_id = d\_avg.department\_id

WHERE e.salary > d\_avg.avg\_salary;

### ✅ ****30. Display each department’s highest paid employee****

SELECT e.name, e.salary, e.department\_id

FROM employees e

WHERE salary = (

SELECT MAX(salary)

FROM employees

WHERE department\_id = e.department\_id

);

### ****JOIN with Aggregate****

**Q:** Find the total salary per department with the department name.

SELECT d.dept\_name, SUM(e.salary) AS total\_salary

FROM employees e

INNER JOIN departments d ON e.dept\_id = d.dept\_id

GROUP BY d.dept\_name;

### ****JOIN with Filtering****

**Q:** List employees who work in the IT department.

SELECT e.name AS Employee, d.dept\_name

FROM employees e

INNER JOIN departments d ON e.dept\_id = d.dept\_id

WHERE d.dept\_name = 'IT';

### ****Join Multiple Tables****

**Q:** Display employee name, department name, and location city.

SELECT e.name AS Employee, d.dept\_name, l.city

FROM employees e

JOIN departments d ON e.dept\_id = d.dept\_id

JOIN locations l ON d.location\_id = l.location\_id;

### ****10. Join with Date Condition****

**Q:** Fetch names of employees and their departments for employees who joined after 2015.

SELECT e.name AS Employee, d.dept\_name

FROM employees e

INNER JOIN departments d ON e.dept\_id = d.dept\_id

WHERE YEAR(e.join\_date) > 2015;

### ****Join and Order By****

**Q:** List employees with their department names, ordered by department and then salary descending.

SELECT e.name AS Employee, d.dept\_name AS Department, e.salary

FROM employees e

JOIN departments d ON e.dept\_id = d.dept\_id

ORDER BY d.dept\_name ASC, e.salary DESC;

### ✅ ****Combined Example****

✅ **Get departments where average salary > 50000 but only consider employees who joined after 2015**

SELECT department\_id, AVG(salary) AS avg\_salary

FROM employees

WHERE YEAR(join\_date) > 2015 -- Filters records first

GROUP BY department\_id

HAVING AVG(salary) > 50000; -- Filters aggregated groups