TOP 30 SQL Important Functions:

for interview preparation as well as for company use.

1. SELECT

Example Table: `employees`

id	name	age	department	salary
1	John Doe	28	Sales	5000
2	Jane Smith	34	Marketing	6000
3	Sam Brown	22	Sales	4000
4	Mike Jones	40	IT	8000

Query:

```
SELECT * FROM employees;
```

Explanation: This query retrieves all columns and rows from the `employees` table.

Result:

id	name	age	department	salary
1	John Doe	28	Sales	5000
2	Jane Smith	34	Marketing	6000
3	Sam Brown	22	Sales	4000
4	Mike Jones	40	IT	8000

---

2. WHERE

Query:

```
SELECT * FROM employees WHERE age > 30;
```

Explanation: This query filters the `employees` table to return only those rows where the `age` is greater than 30.

Result:

id	name	age	department	salary
2	Jane Smith	34	Marketing	6000
4	Mike Jones	40	IT	8000

---

3. `JOIN` (INNER JOIN)

Example Tables: `employees` and `departments`

`employees` Table:

id	name	age	department_id	salary
1	John Doe	28	1	5000
2	Jane Smith	34	2	6000
3	Sam Brown	22	1	4000
4	Mike Jones	40	3	8000

`departments` Table:

id	department_name
1	Sales
2	Marketing
3	IT

Query:

```
SELECT e.name, d.department_name
FROM employees e
INNER JOIN departments d
ON e.department_id = d.id;
```

Explanation: This query performs an `INNER JOIN` to combine the `employees` and `departments` tables based on matching `department_id`.

Result:

name	department_name
John Doe	Sales
Jane Smith	Marketing
Sam Brown	Sales
Mike Jones	IT

---

4. GROUP BY

Query:

```
SELECT department_id, COUNT(*) AS num_employees
FROM employees
GROUP BY department_id;
```

Explanation: This query groups the rows in the `employees` table by `department_id` and counts the number of employees in each department.

Result:

department_id	num_employees
1	2
2	1
3	1

---

5. HAVING

Query:

```
SELECT department_id, COUNT(*) AS num_employees
FROM employees
GROUP BY department_id
HAVING COUNT(*) > 1;
```

Explanation: This query is similar to the previous one but uses `HAVING` to filter out groups with only one employee.

Result:

department_id	num_employees
1	2

---

6. `ORDER BY`

Query:

```
SELECT * FROM employees ORDER BY salary DESC;
```

Explanation: This query orders the rows in the `employees` table by `salary` in descending order.

Result:

id	name	age	department_id	salary
4	Mike Jones	40	3	8000
2	Jane Smith	34	2	6000
1	John Doe	28	1	5000
3	Sam Brown	22	1	4000

---

7. `LIMIT`/`OFFSET`

Query:

```
SELECT * FROM employees ORDER BY salary DESC LIMIT 2 OFFSET 1;
```

Explanation: This query returns two rows starting from the second-highest salary (offset by 1).

Result:

id	name	age	department_id	salary
2	Jane Smith	34	2	6000
1	John Doe	28	1	5000

---

8. DISTINCT

Query:

```
SELECT DISTINCT department_id FROM employees;
```

Explanation: This query returns unique `department_id` values from the `employees` table.

Result:

department_id
1
2
3

---

9. COUNT()

Query:

```
SELECT COUNT(*) AS total_employees FROM employees;
```

Explanation: This query counts the total number of rows in the `employees` table.

Result:

total_employees
4

---

10. `SUM()`

Query:

```
SELECT SUM(salary) AS total_salary FROM employees;
```

Explanation: This query calculates the total sum of all `salary` values in the `employees` table.

Result:

total_salary
23000

---

11. `AVG()`

Query:

```
SELECT AVG(salary) AS average_salary FROM employees;
```

Explanation: This query calculates the average `salary` in the `employees` table.

Result:

average_salary
5750

---

12. `MIN()`

Query:

```
SELECT MIN(salary) AS min_salary FROM employees;
```

Explanation: This query finds the minimum `salary` in the `employees` table.

Result:

min_salary
4000

---

13. `MAX()`

Query:

```
SELECT MAX(salary) AS max_salary FROM employees;
```

Explanation: This query finds the maximum `salary` in the `employees` table.

Result:

max_salary
8000

---

14. `CASE` / `WHEN`

Query:

```
SELECT name, salary,
       CASE
           WHEN salary > 5000 THEN 'High'
           ELSE 'Low'
       END AS salary_grade
FROM employees;
```

Explanation: This query uses `CASE` to create a new column `salary_grade` based on the value of `salary`.

Result:

name	salary	salary_grade
John Doe	5000	Low
Jane Smith	6000	High
Sam Brown	4000	Low
Mike Jones	8000	High

15. `COALESCE()`

Example Table: `employees_with_phone`

id	name	phone
1	John Doe	123456789
2	Jane Smith	NULL
3	Sam Brown	987654321

Query:

```
SELECT name, COALESCE(phone, 'No Phone') AS phone_number FROM
employees_with_phone;
```

Explanation: This query returns the phone number if available; otherwise, it returns 'No Phone'.

Result:

name	phone_number
John Doe	123456789
Jane Smith	No Phone
Sam Brown	987654321

16. `NULLIF()`

Query:

```
SELECT name, NULLIF(salary, 5000) AS adjusted_salary FROM employees;
```

Explanation: This query returns `NULL` for any salary equal to 5000.

Result:

name	adjusted_salary
John Doe	NULL

Jane Smith	6000
Sam Brown	4000
Mike Jones	8000

---

17. `CAST()`

Query:

```sql
SELECT name, CAST(salary AS VARCHAR) AS salary_string FROM employees;
```

Explanation: This query converts the `salary` column from an integer to a string.

Result:

name	salary_string
John Doe	5000
Jane Smith	6000
Sam Brown	4000
Mike Jones	8000

---

18. `CONCAT()`

Query:

```sql
SELECT CONCAT(name, ' (', department_id, ')') AS employee_info FROM employees;
```

Explanation: This query concatenates the `name` and `department_id` columns, with additional text to format the result.

Result:

employee_info
John Doe (1)

Jane Smith (2)
Sam Brown (1)
Mike Jones (3)

---

19. `SUBSTRING()`

Query:

```
SELECT name, SUBSTRING(name, 1, 3) AS short_name FROM employees;
```

Explanation: This query extracts the first three characters from the `name` column.

Result:

name	short_name
John Doe	Joh
Jane Smith	Jan
Sam Brown	Sam
Mike Jones	Mik

---

20. `TRIM()`

Example Table: `employees_with_spaces`

id	name
1	' John '
2	' Jane '
3	' Sam '

Query:

```
SELECT TRIM(name) AS trimmed_name FROM employees_with_spaces;
```

Explanation: This query removes leading and trailing spaces from the `name` column.

Result:

trimmed_name
John
Jane
Sam

---

21. `LENGTH()`

Query:

```
SELECT name, LENGTH(name) AS name_length FROM employees;
```

Explanation: This query returns the length of each `name`.

Result:

name	name_length
John Doe	8
Jane Smith	10
Sam Brown	9
Mike Jones	10

---

22. `REPLACE()`

Query:

```
SELECT name, REPLACE(name, 'o', '0') AS modified_name FROM employees;
```

Explanation: This query replaces the letter 'o' with '0' in the `name` column.

Result:

name	modified_name
John Doe	J0hn D0e
Jane Smith	Jane Smith
Sam Brown	Sam Br0wn
Mike Jones	Mike J0nes

---

23. `ROUND()`

Example Table: `salaries`

id	salary
1	5000.567
2	6000.123
3	4000.789
4	8000.345

Query:

```
SELECT salary, ROUND(salary, 1) AS rounded_salary FROM salaries;
```

Explanation: This query rounds the `salary` to one decimal place.

Result:

salary	rounded_salary
5000.567	5000.6
6000.123	6000.1
4000.789	4000.8
8000.345	8000.3

---

24. `DATEDIFF()`

Example Table: `employees_with_dates`

id	name	hire_date
1	John Doe	2020-01-01
2	Jane Smith	2019-05-10
3	Sam Brown	2018-11-20
4	Mike Jones	2017-03-15

Query:

```
SELECT name, DATEDIFF(CURDATE(), hire_date) AS days_worked FROM
employees_with_dates;
```

Explanation: This query calculates the number of days each employee has worked since their hire date.

Result:

name	days_worked
John Doe	1342
Jane Smith	1578
Sam Brown	1750
Mike Jones	2360

---

25. `DATEADD()`

Query:

```
SELECT name, DATE_ADD(hire_date, INTERVAL 1 YEAR) AS anniversary FROM
employees_with_dates;
```

Explanation: This query adds one year to the `hire_date` for each employee.

Result:

name	anniversary
John Doe	2021-01-01

Jane Smith	2020-05-10
Sam Brown	2019-11-20
Mike Jones	2018-03-15

---

26. `NOW()`

Query:

```
SELECT name, NOW() AS current_time FROM employees;
```

Explanation: This query returns the current date and time.

Result:

name	current_time
John Doe	2024-09-04 15:35:00
Jane Smith	2024-09-04 15:35:00
Sam Brown	2024-09-04 15:35:00
Mike Jones	2024-09-04 15:35:00

---

27. `IFNULL()`

Example Table: `employees_with_phone`

id	name	phone
1	John Doe	123456789
2	Jane Smith	NULL
3	Sam Brown	987654321

Query:

```
SELECT name, IFNULL(phone, 'No Phone') AS phone_number FROM employees_with_phone;
```

Explanation: This query returns the phone number if it's not NULL; otherwise, it returns 'No Phone'.

Result:

name	phone_number
John Doe	123456789
Jane Smith	No Phone
Sam Brown	987654321

---

28. UNION / UNION ALL

Example Tables: **employees** and **managers**

employees Table:

id	name
1	John Doe
2	Jane Smith
3	Sam Brown

managers Table:

id	name
1	Alice Green
2	Bob White

Query:

```
SELECT name FROM employees
UNION
SELECT name FROM managers;
```

Explanation: This query returns a combined list of names from both the employees and managers tables, removing duplicates.

Result:

name
John Doe
Jane Smith
Sam Brown
Alice Green
Bob White

Using `UNION ALL`:

```
SELECT name FROM employees
UNION ALL
SELECT name FROM managers;
```

Explanation: This query returns all names from both tables, including duplicates.

Result:

name
John Doe
Jane Smith
Sam Brown
Alice Green
Bob White

---

29. `EXISTS`

Example Table: `departments`

id	department_name
1	Sales
2	Marketing

3	IT
4	HR

Query:

```
SELECT department_name
FROM departments d
WHERE EXISTS (
    SELECT 1
    FROM employees e
    WHERE e.department_id = d.id
);
```

Explanation: This query returns only those departments that have at least one employee.

Result:

department_name
Sales
Marketing
IT

---

30. `IN`

Query:

```
SELECT name, department_id
FROM employees
WHERE department_id IN (1, 3);
```

Explanation: This query returns the names and department IDs of employees who belong to either department 1 or 3.

Result:

name	department_id
John Doe	1
Sam Brown	1
Mike Jones	3