ChatGPT

SQL Functions and Their Use Cases

This guide summarizes the most important SQL functions, operators, and constructs we discussed, along with clear examples and use cases.

---

1. String Functions

1.1 `CONCAT()`

- Joins two or more strings.

```
SELECT CONCAT(first_name, ' ', last_name) AS full_name FROM users;
```

1.2 `GROUP_CONCAT()`

- Returns a comma-separated string of values.
- Supports `DISTINCT`, `ORDER BY`, and `SEPARATOR`.

```
SELECT GROUP_CONCAT(DISTINCT product ORDER BY product SEPARATOR ',') FROM
orders;
```

1.3 `SUBSTRING_INDEX()`

- Extracts substring before or after a delimiter.

```
SELECT SUBSTRING_INDEX(mail, '@', 1) AS prefix,
       SUBSTRING_INDEX(mail, '@', -1) AS domain
FROM users;
```

1.4 `SUBSTRING()` / `MID()`

- Returns part of a string starting from a given position.

```
SELECT SUBSTRING('abcdef', 2, 3);  -- Returns 'bcd'
SELECT MID('abcdef', 2, 3);        -- Same as SUBSTRING
```

1.5 `LEFT()` / `RIGHT()`

- Returns a specific number of characters from the start or end.

```
SELECT LEFT('abcdef', 3);   -- Returns 'abc'
SELECT RIGHT('abcdef', 2);  -- Returns 'ef'
```

1.6 `UPPER()`, `LOWER()`

- Converts string to uppercase or lowercase.

```
SELECT UPPER(name), LOWER(name) FROM employees;
```

1.7 `INITCAP()` or Title Case

- **Not available in MySQL natively**, but can be achieved using custom logic or functions in other databases (like PostgreSQL or Oracle).

1.8 `REGEXP`

- Pattern matching using regular expressions.

```
-- Match valid emails starting with a letter and specific characters before
'@leetcode.com'
SELECT * FROM users
WHERE mail LIKE '%@leetcode.com'
  AND SUBSTRING_INDEX(mail, '@', 1) REGEXP '^[a-zA-Z][a-zA-Z0-9._-]*$';
```

---

2. Window Functions

2.1 `ROW_NUMBER()` / `RANK()` / `DENSE_RANK()`

- Assigns a unique number based on a specific order.

```
SELECT id, name, ROW_NUMBER() OVER (PARTITION BY dept ORDER BY salary
DESC) AS row_num
FROM employees;
```

2.2 `LAG()` / `LEAD()`

- Accesses data from previous or next rows.

```
SELECT id, weight, LAG(weight) OVER (ORDER BY turn) AS previous_weight
FROM queue;
```

2.3 `SUM(...) OVER(...)`

- Used for running totals or moving averages.

```
SELECT visited_on,
       SUM(amount) OVER (ORDER BY visited_on ROWS BETWEEN 6 PRECEDING AND
CURRENT ROW) AS total,
       ROUND(AVG(amount) OVER (ORDER BY visited_on ROWS BETWEEN 6
PRECEDING AND CURRENT ROW), 2) AS average_amount
FROM customer;
```

---

3. Aggregate Functions

3.1 `COUNT()`, `SUM()`, `AVG()`, `MAX()`, `MIN()`

- Performs aggregation over groups or entire table.

```
SELECT COUNT(*), AVG(salary), MAX(salary), MIN(salary)
FROM employees;
```

3.2 `NULLIF()`

- Returns NULL if two expressions are equal.

```
SELECT NULLIF(salary, '') AS salary FROM employees;
```

3.3 `COALESCE()`

- Returns first non-null value.

```
SELECT COALESCE(manager_id, 'No Manager') FROM employees;
```

---

4. Date Functions

4.1 `DATEDIFF(date1, date2)`

- Returns the difference in days.

```
SELECT DATEDIFF('2020-12-31', '2020-01-01');
```

4.2 `NOW()`, `CURDATE()`

- Returns current timestamp or date.

4.3 Filtering by specific date pattern

```
SELECT * FROM movierating WHERE created_at LIKE '2020-02-%';
```

---

5. Common Table Expressions (CTEs)

`WITH` Clause

- Useful for breaking down complex logic.

```
WITH ranked AS (
  SELECT *, ROW_NUMBER() OVER (PARTITION BY employee_id ORDER BY
primary_flag = 'Y' DESC) AS rn
  FROM employee
)
SELECT employee_id, department_id FROM ranked WHERE rn = 1;
```

---

6. DELETE with Duplicates

```
DELETE p1 FROM person p1
JOIN person p2
ON p1.email = p2.email AND p1.id > p2.id;
```

---

7. Pagination

`LIMIT` and `OFFSET`

```sql
SELECT salary FROM employee ORDER BY salary DESC LIMIT 1 OFFSET 1;
```

---

8. Miscellaneous

Get prefix or suffix in string

```sql
-- Prefix before '@'
SUBSTRING_INDEX(mail, '@', 1)
-- Suffix after '@'
SUBSTRING_INDEX(mail, '@', -1)
```

This guide covers a wide range of SQL capabilities including string handling, ranking, running totals, email validation, date filtering, and deletion logic for duplicates. Each function and construct is demonstrated with concise use cases that apply to real-world database tasks.