Famous words In SQL

INSERT :→ Database me naya data dalna.

```
INSERT INTO table_name (column1, column2) VALUES (value1, value2);
```

SELECT :→ Database se data lena (padhna)

```
SELECT column1, column2 FROM table_name WHERE condition;
```

FROM :→ Kis table se data lena hai, yeh batata hai

```
SELECT * FROM users;
```

WHERE :→Sirf specific data chahiye toh condition lagane ke liye.

```
SELECT * FROM users WHERE age > 18;
```

INSERT INTO :→Kis table me data daalna hai, yeh batata hai.

```
INSERT INTO users (name, email) VALUES ('Ali', 'ali@example.com');
```

VALUES : →Jo naya data insert kar rahe ho, uske actual values.

```
INSERT INTO users (name, age) VALUES ('Sara', 25);
```

UPDATE :→Pehle se jo data hai usko badalna (update karna).

```
UPDATE users SET age = 30 WHERE name = 'Sara';
```

SET :→Kisi column ya variable ki value change karne ke liye.

```
SET @age = 25;
UPDATE users SET age = 30 WHERE name = 'Sara';
```

ORDER BY :→ Data ko sort (arrange) karne ke liye.

```
SELECT * FROM users ORDER BY age;
```

DESC :⟶ Data ko descending order (bade se chhota) me sort karna.

```
SELECT * FROM users ORDER BY age DESC;
```

NOT IN :- Jo values chahiye hi nahi, unko exclude karne ke liye.

```
SELECT * FROM users WHERE city NOT IN ('Delhi', 'Mumbai');
```

LEFT JOIN :⟶Do tables ko jodta hai, lekin left table ka pura data dikhata hai.

```
SELECT users.name, orders.amount FROM users LEFT JOIN orders ON users.id =
orders.user_id;
```

ON :⟶Jab tables join karte hain, to kaunsa column match karega yeh batata hai.

```
SELECT users.name, orders.amount FROM users
LEFT JOIN orders ON users.id = orders.user_id;
```

FIND_IN_SET :⟶Ek column me comma-separated values ho to usme se specific value dhoondhne ke liye.

```
SELECT * FROM users WHERE FIND_IN_SET('admin', roles);
```

DELETE :⟶Database se data hataane ke liye.

```
DELETE FROM users WHERE id = 5;
```

AS :⟶Column ya table ka temporary naam (alias) dene ke liye.

```
SELECT name AS userName FROM users;
```

IN ⟶Multiple values ke liye condition lagane ke liye.

```
SELECT * FROM users WHERE city IN ('Delhi', 'Mumbai');
```

INNER JOIN :⟶Do tables ko match karta hai aur sirf wahi rows dikhata hai jo dono tables me match hoti hain.

```
SELECT users.name, orders.amount FROM users
INNER JOIN orders ON users.id = orders.user_id;
```

UPPER :-Text ko uppercase (bade letters) me convert karne ke liye.

```sql
SELECT UPPER(name) FROM users;
```

DISTINCT → Duplicate values ko remove karne ke liye.

```sql
SELECT DISTINCT city FROM users;
```

LIMIT → Kitne rows dikhane hain, uska restriction lagata hai.

```sql
SELECT * FROM users LIMIT 5;
```

HAVING → `GROUP BY` ke sath filtering karne ke liye.

```sql
SELECT city, COUNT(*) FROM users GROUP BY city HAVING COUNT(*) > 10;
```

CASE → Conditional logic lagane ke liye (like `if-else`).

```sql
SELECT name,
       CASE
           WHEN age < 18 THEN 'Minor'
           ELSE 'Adult'
       END AS age_category
FROM users;
```

COALESCE → Null values ke liye default value set karne ke liye.

```sql
SELECT name, COALESCE(email, 'No Email') FROM users;
```

GROUP_CONCAT → Multiple rows ki values ko ek row me combine karne ke liye.

```sql
SELECT city, GROUP_CONCAT(name) FROM users GROUP BY city;
```

SUBQUERY → Query ke andar ek aur query likhne ke liye.

```sql
SELECT * FROM users WHERE id IN (SELECT user_id FROM orders);
```

EXISTS → Check karne ke liye ki koi record exist karta hai ya nahi.

```sql
SELECT * FROM users WHERE EXISTS (SELECT 1 FROM orders
WHERE orders.user_id = users.id);
```

BETWEEN → Value ka range specify karne ke liye.

```sql
SELECT * FROM users WHERE age BETWEEN 18 AND 30;
```

DEFAULT → Column ki default value set karne ke liye.

```sql
CREATE TABLE users (
    id INT PRIMARY KEY,
    name VARCHAR(50),
    status VARCHAR(20) DEFAULT 'Active'
);
```

AUTO_INCREMENT → Column me automatically next number assign karne ke liye (mostly ID ke liye).

```sql
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50)
);
```

CHAR_LENGTH → String ke length count karne ke liye.

```sql
SELECT CHAR_LENGTH(name) FROM users;
```

REPLACE → Kisi string me koi word replace karne ke liye.

```sql
SELECT REPLACE('Hello World', 'World', 'MySQL');
```

RAND() → Random number generate karne ke liye.

```sql
SELECT * FROM users ORDER BY RAND() LIMIT 1;
```

NOW() → Current date & time dikhane ke liye.

```sql
SELECT NOW();
```

UNION & UNION ALL → Do queries ke results ko combine karta hai.

```sql
SELECT name FROM users
UNION
SELECT name FROM admins;
```

ALTER TABLE → Table ka structure change karne ke liye.

```sql
ALTER TABLE users ADD COLUMN phone VARCHAR(20);
```

INDEX → Query fast karne ke liye indexing lagana.

```sql
CREATE INDEX idx_name ON users(name);
```

TRUNCATE → Pura table clear karne ke liye (faster than DELETE).

```sql
TRUNCATE TABLE users;
```

EXISTS → Check karta hai ki subquery ka result exist karta hai ya nahi.

```sql
SELECT * FROM users WHERE EXISTS
(SELECT 1 FROM orders WHERE orders.user_id = users.id);
```

COALESCE → Agar value NULL ho to default value dene ke liye

```sql
SELECT name, COALESCE(email, 'No Email') FROM users;
```

GROUP_CONCAT → Multiple rows ko ek row me comma-separated values me combine karna.

```sql
SELECT city, GROUP_CONCAT(name) FROM users GROUP BY city;
```

LEAST & GREATEST → Smallest ya largest value find karne ke liye.

```sql
SELECT LEAST(10, 20, 5) AS smallest, GREATEST(10, 20, 5)
AS largest;
```

WINDOW FUNCTIONS (RANK, DENSE_RANK, ROW_NUMBER) → Row-wise ranking dene ke liye

```sql
SELECT name, salary, RANK() OVER (ORDER BY salary DESC)
AS rank FROM employees;
```

JSON Functions → JSON data handle karne ke liye.

```sql
SELECT JSON_EXTRACT('{"name": "Ali", "age": 25}', '$.name');
```

TRANSACTION (COMMIT & ROLLBACK) → Safe database updates ke liye.

```
START TRANSACTION;
UPDATE users SET balance = balance - 100 WHERE id = 1;
UPDATE users SET balance = balance + 100 WHERE id = 2;
COMMIT;
Example: "Agar dono updates sahi chale to COMMIT,
warna ROLLBACK karke pehle wala data wapas lao."
```

EVENTS (Scheduled Tasks) → Automatically koi task run karne ke liye.

```
CREATE EVENT delete_old_orders
ON SCHEDULE EVERY 1 DAY
DO DELETE FROM orders WHERE order_date < NOW() - INTERVAL 30 DAY;
```

ENUM & SET (Data Types) → Fixed choices wale data types.

```
CREATE TABLE users (
    id INT PRIMARY KEY,
    status ENUM('Active', 'Inactive', 'Suspended')
);
Example: "Users ka status sirf 'Active', 'Inactive',
ya 'Suspended' ho sakta hai."
```

GROUP BY

```
SELECT city, COUNT(*) AS total_users
FROM users
GROUP BY city;
```

Handling Time Zones in MySQL for Users in Different Continents
Problem:

- Users are from different time zones (America, Europe, Asia, etc.).
- **Database stores time in UTC** (recommended best practice).
- **Frontend doesn't send the user's timezone** explicitly.
- You need to **show timestamps in the user's local time** & also **filter date ranges properly**.
  Step 1: Always Store Time in UTC in MySQL

```
CREATE TABLE orders (
    id INT AUTO_INCREMENT PRIMARY KEY,
    order_time DATETIME DEFAULT UTC_TIMESTAMP() -- Always store in UTC
);
```

✔ This ensures all stored times are in **UTC format**.

Frontend time zone detect karne ke liye JavaScript ka `**Intl.DateTimeFormat().resolvedOptions().timeZone**` use kar sakte ho.

```
console.log(Intl.DateTimeFormat().resolvedOptions().timeZone);
```