Complete SQL CheatSheet

- Developer Shaurya

Important Links

- **SQL Cheatsheet :** https://developershaurya.com/sql-cheatsheet/

- **Website:** https://developershaurya.com/

- **All Cheatsheets :** https://developershaurya.com/cheatsheets/

- **Youtube:** https://www.youtube.com/@DeveloperShaurya

- **(Developer Shaurya) Community :** https://t.me/developer_shaurya (Telegram)

CheatSheet SQL

SQL Cheatsheet – A Quick Guide for Beginners & Developers

by Developer Shaurya • September 25, 2025 • 0

Content Table

1. **Basics**

```
SHOW DATABASES;
USE database_name;
SHOW TABLES;
```

2. **Data Types**

- Numbers → `INT`, `BIGINT`, `DECIMAL`, `FLOAT`
- Strings → `CHAR`, `VARCHAR`, `TEXT`
- Date/Time → `DATE`, `DATETIME`, `TIMESTAMP`
- Boolean → `BOOLEAN`

3. **Database & Table**

```
CREATE DATABASE db_name;
DROP DATABASE db_name;

CREATE TABLE users (
  id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(100) NOT NULL,
  email VARCHAR(100) UNIQUE,
  age INT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

ALTER TABLE users ADD phone VARCHAR(15);
ALTER TABLE users DROP COLUMN phone;
DROP TABLE users;
```

4. **Insert / Update / Delete**

```
INSERT INTO users (name, email, age)
VALUES ('Alice', 'alice@mail.com', 22);

UPDATE users SET age = 23 WHERE id = 1;

DELETE FROM users WHERE id = 1;
```

5. Select Queries

```
SELECT * FROM users;
SELECT name, age FROM users;
SELECT DISTINCT age FROM users;
SELECT name AS full_name FROM users;
```

6. WHERE Clause

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

-- Operators: =, !=, <>, >, <, <=, >=
-- Logical: AND, OR, NOT
-- Range: BETWEEN ... AND ...
-- List: IN (...)
-- Pattern: LIKE, ILIKE
```

7. Sorting & Limiting

```
SELECT * FROM users ORDER BY age ASC;
SELECT * FROM users ORDER BY age DESC;
SELECT * FROM users LIMIT 5 OFFSET 10;
```

8. Aggregate Functions

```
SELECT COUNT(*), AVG(age), MIN(age), MAX(age), SUM(age) FROM users;
```

9. GROUP BY / HAVING

```
SELECT age, COUNT(*) FROM users GROUP BY age;
SELECT age, COUNT(*) FROM users GROUP BY age HAVING COUNT(*) > 1;
```

10. **Joins**

```
-- Inner Join
SELECT u.name, o.order_date
FROM users u
INNER JOIN orders o ON u.id = o.user_id;

-- Left Join
SELECT u.name, o.order_date
FROM users u
LEFT JOIN orders o ON u.id = o.user_id;

-- Right Join
SELECT u.name, o.order_date
FROM users u
RIGHT JOIN orders o ON u.id = o.user_id;

-- Full Join (Postgres, MySQL 8+)
SELECT u.name, o.order_date
FROM users u
FULL OUTER JOIN orders o ON u.id = o.user_id;
```

11. **Subqueries**

```
SELECT * FROM users
WHERE age > (SELECT AVG(age) FROM users);
```

12. **Set Operations**

```
SELECT name FROM users
UNION
SELECT name FROM customers;

SELECT name FROM users
UNION ALL
SELECT name FROM customers;
```

13. **Constraints**

- PRIMARY KEY
- FOREIGN KEY
- UNIQUE
- NOT NULL
- DEFAULT

14. **Indexes**

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

15. **Views**

```
CREATE VIEW adult_users AS
SELECT * FROM users WHERE age >= 18;

SELECT * FROM adult_users;
DROP VIEW adult_users;
```

16. **Transactions**

```
START TRANSACTION;
UPDATE users SET age = age+1 WHERE id=1;
DELETE FROM users WHERE id=2;
COMMIT;
ROLLBACK;
```

17. **Privileges**

```
GRANT ALL PRIVILEGES ON db_name.* TO 'user'@'localhost';
REVOKE ALL PRIVILEGES ON db_name.* FROM 'user'@'localhost';
```

Advanced SQL

18. **Stored Procedures**

```
DELIMITER //
CREATE PROCEDURE GetUsers()
BEGIN
  SELECT * FROM users;
END //
DELIMITER ;

CALL GetUsers();
```

19. **Functions**

```
CREATE FUNCTION UserCount()
RETURNS INT
DETERMINISTIC
RETURN (SELECT COUNT(*) FROM users);
```

20. **Triggers**

```
CREATE TRIGGER before_insert_user
BEFORE INSERT ON users
FOR EACH ROW
SET NEW.created_at = NOW();
```

21. **Window Functions** (MySQL 8+, PostgreSQL)

```
SELECT name, age,
       RANK() OVER (ORDER BY age DESC) AS rank,
       ROW_NUMBER() OVER (PARTITION BY age ORDER BY name) AS row_num
FROM users;
```

22. **CTEs (Common Table Expressions)**

```
WITH avg_age AS (
   SELECT AVG(age) AS avg FROM users
)
SELECT * FROM users WHERE age > (SELECT avg FROM avg_age);
```

23. **Recursive CTEs**

```
WITH RECURSIVE numbers AS (
  SELECT 1 AS n
  UNION ALL
  SELECT n+1 FROM numbers WHERE n < 10
)
SELECT * FROM numbers;
```

24. **JSON Data**

```
-- MySQL JSON Example
SELECT JSON_EXTRACT(details, '$.address') FROM users;
SELECT details->>'$.phone' FROM users;
```

25. **Temporary Tables**

```
CREATE TEMPORARY TABLE temp_users AS
SELECT * FROM users WHERE age > 20;
```

26. **Backup & Restore**

```
-- Export
mysqldump -u user -p db_name > backup.sql

-- Import
mysql -u user -p db_name < backup.sql
```