SQL Notes — (Simple & Practical)

Intro: SQL (Structured Query Language) ek language hai jo databases se baat karne ke liye use hoti hai. Agar tum data store karte ho — users, courses, orders — SQL se tum woh data add, read, update, aur delete kar sakte ho.

---

1) Database aur Table kya hote hain?

- **Database = ek jagah jahan data rakha hota hai (jaise school ki file cabinet).**
- **Table = database ke andar ek sheet jisme rows (records) aur columns (fields) hoti hain.**

Example: students table:

id	name	email	age

---

2) Popular SQL Engines

- **MySQL / MariaDB**
- **PostgreSQL**
- **SQLite (lightweight)**
- **SQL Server (Microsoft)**

---

3) Basic Commands (CRUD)

Create (INSERT)

INSERT INTO students (name, email, age) VALUES ('Aman','aman@gmail.com',20);

Read (SELECT)

SELECT * FROM students; -- sab rows

SELECT name, email FROM students; -- specific columns

SELECT * FROM students WHERE age > 18; -- condition

Update

UPDATE students SET age = 21 WHERE id = 1;

Delete

SQL Notes — (Simple & Practical)

DELETE FROM students WHERE id = 3;

---

4) Creating Tables (DDL)

CREATE TABLE students (

id INT AUTO_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(150) UNIQUE,

age INT,

created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP

);

- **PRIMARY KEY unique identifier.**
- **AUTO_INCREMENT automatically bada number deta hai.**
- **NOT NULL column khali nahi ho sakta.**

---

5) Data Types (Common)

- **INT, BIGINT (numbers)**
- **VARCHAR(n) (text, max n chars)**
- **TEXT (long text)**
- **DATE, DATETIME, TIMESTAMP**
- **DECIMAL(10,2) (money)**
- **BOOLEAN (true/false)**

---

6) WHERE, AND, OR, NOT

SELECT * FROM users WHERE age > 18 AND city = 'Delhi';

SELECT * FROM users WHERE NOT active;

SQL Notes — (Simple & Practical)

7) ORDER BY, LIMIT

SELECT * FROM courses ORDER BY created_at DESC LIMIT 5; -- latest 5

---

8) Aggregate Functions (GROUP BY)

- **COUNT(), SUM(), AVG(), MIN(), MAX()**

SELECT COUNT(*) AS total_students FROM students;

SELECT city, COUNT(*) FROM students GROUP BY city;

SELECT course_id, AVG(score) FROM results GROUP BY course_id;

---

9) Joins (Bahut important)

Tables ko join karke combined data laate hain.

- **INNER JOIN: dono tables me matching record chahiye.**
- **LEFT JOIN (LEFT OUTER): left table ke saare rows, right ke matching agar ho to.**
- **RIGHT JOIN: right ka saara, left matching agar ho to.**
- **FULL JOIN (Postgres): dono tables ke sab rows, matching jahan ho.**

Example (students + enrollments):

SELECT s.name, e.course_id

FROM students s

INNER JOIN enrollments e ON s.id = e.student_id;

-- Left join example:

SELECT s.name, e.course_id

FROM students s

LEFT JOIN enrollments e ON s.id = e.student_id;

---

10) Subqueries

SQL Notes — (Simple & Practical)

Query ke andar query.

SELECT name FROM students WHERE id IN (SELECT student_id FROM enrollments WHERE course_id=2);

---

11) Views (Virtual Tables)

View ek saved query hai jise table ki tarah treat kar sakte ho.

CREATE VIEW student_emails AS

SELECT id, name, email FROM students;

SELECT * FROM student_emails;

---

12) Indexes (Performance)

Index search fast karte hain.

CREATE INDEX idx_email ON students(email);

- **Lekin index space leta hai, aur writes me thoda slow kar sakta hai — zaroorat par hi banao.**

---

13) Transactions (ACID)

Multiple queries ko ek group me run karna.

START TRANSACTION;

UPDATE accounts SET balance = balance - 100 WHERE id = 1;

UPDATE accounts SET balance = balance + 100 WHERE id = 2;

COMMIT; -- changes save ho gaye

-- agar koi error ho to ROLLBACK;

---

14) Constraints (Rules)

- **PRIMARY KEY**

SQL Notes — (Simple & Practical)

- **UNIQUE**
- **NOT NULL**
- **CHECK (value condition) — supported in Postgres/MySQL newer versions**
- **FOREIGN KEY (referential integrity)**

ALTER TABLE enrollments

ADD CONSTRAINT fk_student FOREIGN KEY (student_id) REFERENCES students(id);

---

15) Normalization (Simple explanation)

- **Data ko duplicate se bachane ke liye tables ko theek tarike se todte hain.**
- **1NF, 2NF, 3NF common rules. Example: students aur addresses ko alag tables me rakhna.**

---

16) Stored Procedures & Functions

Server-side reusable code.

```
-- MySQL procedure example
DELIMITER $$
CREATE PROCEDURE getStudentCount()
BEGIN
SELECT COUNT(*) FROM students;
END$$
DELIMITER ;

CALL getStudentCount();
```

---

17) Triggers

Automatic action jab table me insert/update/delete hota hai.

SQL Notes — (Simple & Practical)

CREATE TRIGGER before_student_insert

BEFORE INSERT ON students

FOR EACH ROW

SET NEW.created_at = NOW();

---

18) Backup & Restore (Basic)

MySQL (command line):

- **Backup: mysqldump -u root -p dbname > backup.sql**
- **Restore: mysql -u root -p dbname < backup.sql**

---

19) Security Best Practices

1. **Use least privilege: user ko sirf zaroori permissions do.**
2. **Avoid SQL injection: use prepared statements / parameterized queries.**
3. **Encrypt sensitive data where needed.**
4. **Regular backups aur strong passwords.**

Example (prepared statement in PHP PDO):

$stmt = $pdo->prepare('SELECT * FROM users WHERE email = ?');

$stmt->execute([$_POST['email']]);

---

20) Common Errors & Tips

- **Duplicate entry → UNIQUE constraint violation.**
- **Foreign key constraint fails → referenced record missing.**
- **Syntax error → check commas, semicolons.**
- **Use EXPLAIN SELECT ... to see query plan (performance).**

---

21) Useful Commands Quick Cheat-sheet

SQL Notes — (Simple & Practical)

- **SHOW TABLES;**
- **DESCRIBE students; or SHOW COLUMNS FROM students;**
- **EXPLAIN SELECT * FROM enrollments;**
- **SHOW INDEX FROM students;**

---

22) Practice Projects (Try these)

1. **Simple Student Management (CRUD) with MySQL.**
2. **Course enrollment system: students, courses, enrollments.**
3. **Reports: top enrolled courses, monthly signups.**
4. **Implement transactions: transfer balance between users.**