📘 SQL Short Notes + Example Queries

## 1. Database & Table Creation

CREATE DATABASE School;  
USE School;  
  
CREATE TABLE Students (  
 ID INT PRIMARY KEY AUTO\_INCREMENT,  
 Name VARCHAR(50),  
 Age INT,  
 Grade VARCHAR(10)  
);  
📝 Creates a database and a table with fields and a primary key.

## 2. Insert Data

INSERT INTO Students (Name, Age, Grade)  
VALUES ('Alice', 15, '10th');  
📝 Adds a new record to the table.

## 3. Select Data

SELECT \* FROM Students; -- all columns  
SELECT Name, Grade FROM Students; -- specific columns  
📝 Retrieves data from the table.

## 4. WHERE Clause (Filtering)

SELECT \* FROM Students WHERE Age > 14;  
📝 Filters records based on a condition.

## 5. UPDATE Records

UPDATE Students SET Grade = '11th' WHERE Name = 'Alice';  
📝 Modifies existing records.

## 6. DELETE Records

DELETE FROM Students WHERE Age < 10;  
📝 Removes records from the table.

## 7. ORDER BY (Sorting)

SELECT \* FROM Students ORDER BY Age DESC;  
📝 Sorts results by a column.

## 8. LIMIT (Top N Records)

SELECT \* FROM Students LIMIT 3;  
📝 Limits number of records shown.

## 9. Aggregate Functions

SELECT COUNT(\*) FROM Students;  
SELECT AVG(Age) FROM Students;  
SELECT MAX(Age), MIN(Age) FROM Students;  
📝 Performs calculations on data.

## 10. GROUP BY (With Aggregates)

SELECT Grade, COUNT(\*) FROM Students GROUP BY Grade;  
📝 Groups rows by a column and applies aggregates.

## 11. HAVING (Group Filter)

SELECT Grade, COUNT(\*) FROM Students GROUP BY Grade HAVING COUNT(\*) > 2;  
📝 Filters groups after aggregation.

## 12. JOINs (Combining Tables)

-- Assuming a new table:  
CREATE TABLE Marks (  
 StudentID INT,  
 Subject VARCHAR(30),  
 Score INT  
);  
  
-- INNER JOIN  
SELECT Students.Name, Marks.Subject, Marks.Score  
FROM Students  
JOIN Marks ON Students.ID = Marks.StudentID;  
📝 Combines rows from two tables using a common field.

## 13. Subquery

SELECT Name FROM Students  
WHERE Age = (SELECT MAX(Age) FROM Students);  
📝 Nested query inside another query.

## 14. Aliases

SELECT Name AS StudentName, Age AS StudentAge FROM Students;  
📝 Renames columns in result for better readability.

## 15. DISTINCT (Remove Duplicates)

SELECT DISTINCT Grade FROM Students;  
📝 Returns unique values only.

## 16. IN, NOT IN

SELECT \* FROM Students WHERE Grade IN ('9th', '10th');  
📝 Filters using multiple values.

## 17. BETWEEN

SELECT \* FROM Students WHERE Age BETWEEN 13 AND 16;  
📝 Checks if a value is in a range.

## 18. LIKE (Pattern Matching)

SELECT \* FROM Students WHERE Name LIKE 'A%'; -- Starts with A  
📝 Pattern-based filtering using wildcards (% and \_).

## 19. IS NULL / IS NOT NULL

SELECT \* FROM Students WHERE Grade IS NULL;  
📝 Checks for missing (null) values.

## 20. ALTER TABLE

ALTER TABLE Students ADD Email VARCHAR(100);  
📝 Modifies table structure (e.g., add or drop column).

## 21. DROP

DROP TABLE Students;  
DROP DATABASE School;  
📝 Deletes a table or database permanently.

🎯 SQL Interview Questions & Answers

## 1. What is the difference between WHERE and HAVING?

✅ WHERE filters rows before GROUP BY.  
✅ HAVING filters after GROUP BY (used with aggregates).  
  
Example:  
SELECT \* FROM Students WHERE Age > 15;  
SELECT Grade, COUNT(\*) FROM Students GROUP BY Grade HAVING COUNT(\*) > 2;

## 2. Difference between INNER JOIN, LEFT JOIN, and RIGHT JOIN

- INNER JOIN: Returns rows when there's a match in both tables.  
- LEFT JOIN: All records from the left table + matching ones from the right.  
- RIGHT JOIN: All records from the right table + matching ones from the left.  
  
Example:  
SELECT s.Name, m.Score FROM Students s INNER JOIN Marks m ON s.ID = m.StudentID;

## 3. How to find the second highest score?

SELECT MAX(Score) FROM Marks  
WHERE Score < (SELECT MAX(Score) FROM Marks);

## 4. Count number of students in each grade

SELECT Grade, COUNT(\*) AS Total FROM Students GROUP BY Grade;

## 5. Retrieve students with names starting with 'A'

SELECT \* FROM Students WHERE Name LIKE 'A%';

## 6. What is DISTINCT used for?

Removes duplicate values.  
  
Example:  
SELECT DISTINCT Grade FROM Students;

## 7. What is a subquery?

A query nested inside another query.  
  
Example:  
SELECT Name FROM Students WHERE Age = (SELECT MAX(Age) FROM Students);

## 8. What is normalization?

Normalization is the process of organizing data to reduce redundancy and improve data integrity.  
- 1NF: Atomic values  
- 2NF: Full functional dependency  
- 3NF: No transitive dependency

## 9. What are constraints in SQL?

Constraints enforce rules:  
- PRIMARY KEY: Uniquely identifies records  
- FOREIGN KEY: Enforces referential integrity  
- UNIQUE: Ensures unique values  
- NOT NULL: Disallows null values  
- CHECK: Enforces conditions on column values

## 10. DELETE vs TRUNCATE vs DROP

| Command | Removes Data | Rollback | Affects Structure |  
|----------|--------------|----------|-------------------|  
| DELETE | Yes | Yes | No |  
| TRUNCATE | Yes (All) | No | No |  
| DROP | Yes (All) | No | Yes (Deletes Table)|

## Some of The Most Important SQL Commands

* SELECT - extracts data from a database
* UPDATE - updates data in a database
* DELETE - deletes data from a database
* INSERT INTO - inserts new data into a database
* CREATE DATABASE - creates a new database
* ALTER DATABASE - modifies a database
* CREATE TABLE - creates a new table
* ALTER TABLE - modifies a table
* DROP TABLE - deletes a table
* CREATE INDEX - creates an index (search key)
* DROP INDEX - deletes an index

📘 SQL Stored Procedures: IN, OUT, INOUT

# What is a Stored Procedure?

A stored procedure is a set of SQL statements that can be stored and reused in the database. It supports parameters: IN, OUT, and INOUT.

## Difference Between IN, OUT, INOUT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameter Type | Purpose | Input Value | Output Value | Example Use |
| IN | Passes value into procedure | Yes | No | Filter by country |
| OUT | Returns value from procedure | No | Yes | Return total customers |
| INOUT | Takes and returns value | Yes | Yes | Update discount rate |

## Example: IN Parameter

DELIMITER //  
  
CREATE PROCEDURE GetCustomersByCountry(IN countryName VARCHAR(50))  
BEGIN  
 SELECT \* FROM Customers WHERE Country = countryName;  
END //  
  
DELIMITER ;

Usage: CALL GetCustomersByCountry('India');

## Example: OUT Parameter

DELIMITER //  
  
CREATE PROCEDURE GetCustomerCount(OUT total INT)  
BEGIN  
 SELECT COUNT(\*) INTO total FROM Customers;  
END //  
  
DELIMITER ;

Usage: CALL GetCustomerCount(@total); SELECT @total;

## Example: INOUT Parameter

DELIMITER //  
  
CREATE PROCEDURE UpdateDiscount(INOUT discountRate DECIMAL(5,2))  
BEGIN  
 SET discountRate = discountRate + 5.00;  
END //  
  
DELIMITER ;

Usage: SET @rate = 10.00; CALL UpdateDiscount(@rate); SELECT @rate;

📌 DELIMITER // is used to tell MySQL that you're writing a block.