SQL NOTES

1. WHAT IS SQL?

SQL (Structured Query Language) is a **standard language** used to:

- Store data in a database
- Retrieve data from a database
- Update data
- Delete data
- Manage the structure of the database
- -SQL stand for [STRUCTURED QUERY LANGUAGE]
- -SQL was earlier known as SEQUEL [STRUCTURED ENGLISH QUERY LANGUAGE]
- -SQL is programming language used to interact with relational DB
- -it is used to perform **CRUD** operations:
- Create
- Read
- Update
- Delete

-

👉 In short: **SQL is used to talk with databases.**

---

2. USES OF SQL

1. **Create databases & tables** (structure).
2. **Insert, update, delete** data.
3. **Fetch/query** data (e.g., find students with marks > 80).
4. **Manage permissions** (who can access what).
5. **Work with transactions** (bank money transfer = debit + credit).

---

3. TYPES OF DATA IN SQL (DATA TYPES)

Data types decide **what kind of data** a column can store.

1. Numeric Data Types

- INT → whole numbers (e.g., 10, 200)
- DECIMAL(p,s) → exact numbers with decimals (e.g., 123.45)
- FLOAT, DOUBLE → approximate decimal values

2. String/Text Data Types

- CHAR(n) → fixed-length string (e.g., 'ABC', always n characters)
- VARCHAR(n) → variable-length string (e.g., names, addresses)
- TEXT → long text data

3. Date/Time Data Types

- DATE → stores date (YYYY-MM-DD)
- TIME → stores time (HH:MM:SS)
- DATETIME → date + time (YYYY-MM-DD HH:MM:SS)
- TIMESTAMP → date + time with automatic update

4. Other Data Types

- BOOLEAN → true/false
- BLOB → Binary Large Object (images, files)

🔹 **Example**

```
CREATE TABLE students (
   roll_no INT PRIMARY KEY,
   name VARCHAR(50),
   marks DECIMAL(5,2),
   admission_date DATE,
   is_active BOOLEAN
```

);

Here:

- roll_no → integer
- name → string
- marks → decimal
- admission_date → date
- is_active → boolean

---

4. SQL PROCESS (EASY VERSION FOR INTERVIEW)

When we write an SQL query, the database follows 5 steps:

1. **Parsing** → Checks if the query is written correctly (syntax, table names, etc.).
2. **Optimization** → Finds the fastest way to run the query.
3. **Plan** → Creates a step-by-step plan to fetch the data.
4. **Execution** → Runs the query using that plan.
5. **Result** → Sends the output back to the user.

5. STRUCTURE OF SQL

is used to communicate with a database. Its structure is basically commands that are grouped based on what they do.
SQL is divided into different types of commands, based on their purpose:
DDL, DML, DCL, TCL, and DQL – each handling different aspects like defining database, manipulating data, controlling access, and managing transactions.

6. WINDOWS IN SQL (SHORT VERSION)

- A **window** is a set of rows **related to the current row**.
- **Window functions** calculate values (like sum, average, rank) **without reducing rows**.
- Common functions: ROW_NUMBER(), RANK(), DENSE_RANK(), SUM() OVER(), AVG() OVER().
- **Syntax:**

<function>() OVER (PARTITION BY column ORDER BY column)

- **PARTITION BY** → Groups rows
- **ORDER BY** → Order rows inside each group

Example: Running total per employee:

SUM(Amount) OVER (PARTITION BY Employee ORDER BY ID) AS RunningTotal

Keeps all rows and calculates values **per group or overall**.

---

7. DATABASE

Database is a collection of data in a format that can be easily accessed(digital)

---

8. DATABASE MANAGEMENT SYSTEM

a software application used to manage our DB is called DBMS
-we cannot directly access DB
-we use DBMS to access DB
-we use SQL (DMBS)

- **TYPE OF DBMS**

1. Relational DB
2. Non Relational DB

1.RELATIONAL DB (RDMS)
-Data stores in table (Rows, columns)

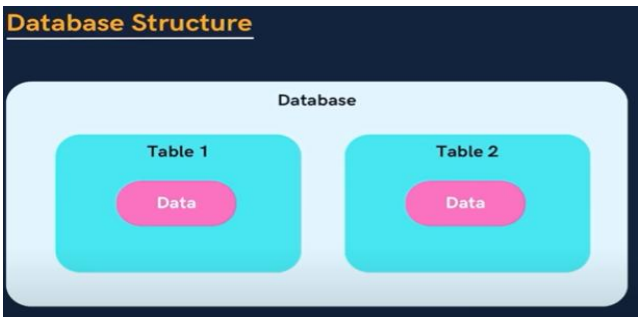
-we use SQL to work with relational DBMS
-example MYSQL, Oracle, SQL server

2.NON RELATIONAL DB
-Data not stored in table
-also known as NoSQL
-example : MongoDB

---

9. DATABASE STRUCTURE

It is the **organized way of storing data** using **tables**, **columns**, **rows**, and **keys** to **arrange, manage, and connect** information inside a database.



Example:
- **Database:** School
- **Tables:** Students, Teachers, Classes
- **Columns:** Name, Age, Class
- **Rows:** Each student's details
- **Keys:** Roll_No, Class_ID to link info

---

10. WHAT IS TABLE

A **table** is a **collection of data** arranged in **rows and columns** — just like an **Excel sheet**.
Columns = define **what kind of data** (like Name, Age, Email), it's also known as schema/structure which is used to show the design
Rows = store the **actual data records** (like 'Mohini, 22, mohini@gmail.com') its individual data

---

11. WHAT IS QUERY

A **query** is a **command** you write to **ask the database to do something** — like **get data**, **add data**, **change data**, or **delete data**.

---

12. WHAT IS A SCHEMA IN SQL

A schema is the layout of the database — it tells what is inside and how things are connected.
Example:
A School schema might have:
Tables: Students, Teachers, Classes
Columns: Name, Age, Subject
Relationships: Which student is in which class

---

13. WHAT IS INDEX IN SQL

An index is like a shortcut that makes it faster to find data in a table.
It works like an index in a book — it helps you quickly look up information without scanning every page.
In simple words:
An index helps the database search and sort data faster.
An **Index** is a database object that improves the speed of data retrieval.

🎯 **Use**
- Faster SELECT queries
- Helps in searching and sorting

Types:
- **Clustered Index** → Reorders the table's physical data. Only one allowed per table.
- **Non-Clustered Index** → Creates a separate structure that points to table rows. Multiple allowed.

Example Query

```
-- Create a non-clustered index on 'name'
CREATE INDEX idx_name ON Employees(name);
```

Explanation

When you search using WHERE name = 'Amit', MySQL uses this index to find results faster.

---

14. DATA TYPES

Data types decide what kind of values a column can hold in a table.

Common SQL Data Types:

Data Type	What it stores	Example
INT	Whole numbers	10, 200
FLOAT / DECIMAL	Decimal numbers	45.67
CHAR(n)	Fixed-length text	'A123'
VARCHAR(n)	Text with variable length	'Mohini'
DATE	Calendar date	'2024-07-31'
TIME	Time of day	'14:30:00'
BOOLEAN	True/False	TRUE, FALSE

CHAR: Fixed length — always takes up the same space, adds spaces if needed.

CHAR(5) [C][A][T][ ][ ] → 2 empty spaces filled

VARCHAR: Variable length — only uses space for what you actually type.

CHAR(5) [C][A][T] → No empty space

SIGNED

- Can store **both positive and negative numbers**.
  - Example: -10, 0, 50

UNSIGNED

- Can store **only positive numbers** (and zero).
  - Example: 0, 50, 200 — **no negatives!**
-

```
age INT UNSIGNED  -- age can't be negative
balance INT SIGNED  -- balance can be negative or positive
```

---

15. TYPE OF OPERATOR

Operators are symbols or keywords that **help you do calculations**, **compare values**, or **combine multiple conditions** in your SQL queries.

In short:

- **Arithmetic:** Do math.
- **Comparison:** Check if something is equal, bigger, smaller.
- **Logical:** Join multiple checks together.

1 Arithmetic Operators — do math

Operator	Meaning	Example
+	Add	SELECT 5 + 3; → 8
-	Subtract	SELECT 10 - 2; → 8

Operator	Meaning	Example
*	Multiply	SELECT 4 * 2; → 8
/	Divide	SELECT 16 / 2; → 8

👉 **Use:** Calculate totals, prices, averages, etc.

2️⃣ Comparison Operators — compare values

Operator	Meaning	Example
=	Equals	age = 18
!= or <>	Not equal	salary != 50000
>	Greater than	marks > 75
<	Less than	age < 30
>=	Greater or equal	price >= 100
<=	Less or equal	score <= 50

👉 **Use:** Find rows matching certain conditions.

Example:

```
SELECT * FROM students WHERE marks > 75;
```

3️⃣ Logical Operators — combine conditions

Operator	Meaning	Example
AND	Both conditions true	age > 18 AND city = 'Mumbai'
OR	Either condition true	salary > 50000 OR dept = 'HR'
NOT	Opposite	NOT city = 'Delhi'

👉 **Use:** Build complex filters.

Example:

```
SELECT * FROM employees WHERE salary > 50000 AND dept = 'HR';
```

---

16. WHAT IS EXPRESSION

An expression **in SQL is** any combination of values, columns, operators, and functions **that** produces a single value**.**
An expression is anything that gives a result.
It can **calculate**, **compare**, **filter**, or **format** your data.

👉 **In simple words:**
Expression = a formula or calculation in SQL.

Examples of Expressions

1️⃣ Arithmetic Expression

```
SELECT price * quantity AS total FROM orders;
```

➡️ Here, price * quantity is an expression — it calculates total price.

2️⃣ Comparison Expression

```
SELECT * FROM students WHERE marks > 80;
```

➡️ marks > 80 is an expression — it checks a condition.

3️⃣ Logical Expression

```
SELECT * FROM employees WHERE salary > 50000 AND dept = 'HR';
```

➡️ salary > 50000 AND dept = 'HR' is an expression — it combines conditions.

4️⃣ Function Expression

```
SELECT UPPER(name) FROM students;
```

➡️ UPPER(name) is an expression — it changes text to uppercase.

17. SQL FUNCTIONS

Functions

📖 Definition

Functions are predefined SQL methods that perform operations and return a value.

🎯 Use

- For calculations, formatting, data manipulation

Types:

- **Aggregate Functions** → COUNT, SUM, AVG, MAX, MIN
- **Scalar Functions** → UPPER, LOWER, LENGTH, ROUND
- **Date Functions** → NOW, CURDATE, DATE_ADD
- **String Functions** → CONCAT, SUBSTRING

📝 Example Query

```
SELECT department, COUNT(*) AS total_employees, AVG(salary) AS avg_salary
FROM Employees
GROUP BY department;
```

🔎 Explanation

Here, COUNT gives number of employees in each department and AVG calculates their average salary.

Definition: Functions are built-in operations used to perform calculations, manipulate data, or return information.

Category	Functions	Query Example
Aggregate	COUNT, SUM, AVG, MAX, MIN	SELECT department, AVG(salary) FROM employees GROUP BY department;
String	CONCAT, LENGTH, UPPER, LOWER	SELECT CONCAT(first_name, ' ', last_name) FROM students;
Date	NOW, CURDATE, DATEDIFF	SELECT DATEDIFF(CURDATE(), hire_date) FROM employees;
Numeric	ROUND, CEIL, FLOOR	SELECT ROUND(salary, 0) FROM employees;
Window	ROW_NUMBER, RANK, LAG	SELECT name, RANK() OVER (ORDER BY salary DESC) FROM employees;

18. WHAT IS AGGREGATE FUNCTIONS IN SQL

Aggregate functions in SQL are special functions that **perform a calculation on a set of rows (a group of values) and return a single summarized result**.

They are mostly used with the GROUP BY clause to group data, but they can also be used without it.

✨ Example

- COUNT() → Counts rows
- SUM() → Adds all values
- AVG() → Finds average

- MIN() → Finds smallest value
- MAX() → Finds largest value

✅ **Simple Definition:**
AGG (Aggregate functions) in SQL are used to summarize data by performing calculations like total, average, minimum, maximum, or count on a column.

19. WHAT IS CONSTRAINT

A CONSTRAINT is a rule you put on a table to control what data can be stored in it.
It protects your data and keeps it accurate.
Constraint = Rule to keep your data correct, safe, and meaningful.

Constraint	What it does	Example
PRIMARY KEY	**Uniquely identifies each row**	**id must be unique and not null**
FOREIGN KEY	**Links tables together**	**student_id in marks table refers to id in students table (it can be duplicate)**
UNIQUE	**Makes sure values are unique**	**Email must be unique**
NOT NULL	**Value cannot be empty**	**Name cannot be blank**
CHECK	**Checks a condition**	**Age must be > 18**
DEFAULT	**Sets a default value**	**If no city given, default is 'Mumbai'**

FOREIGN KEY :

```
CREATE TABLE TEMP (
Students_id int,
FOREIGN KEY (students_id) references student(ID));
```

---

20. COMMANDS IN SQL

Type	Full Form	Definition
DDL	Data Definition Language	Commands to create or change structure of tables & databases.
DML	Data Manipulation Language	Commands to add, change, or delete data in tables.
DQL	Data Query Language	Commands to fetch data from tables. (Mostly SELECT)
DCL	Data Control Language	Commands to give or remove user permissions.
TCL	Transaction Control Language	Commands to manage transactions (save or undo changes).

Main Types of SQL Commands

1️⃣ DDL — Data Definition Language

👉 These commands **define or change the structure** (table, database).
CREATE → make a new table or database
ALTER → change a table (add/remove columns)
DROP → delete a table or database
TRUNCATE → delete all rows, but keep table

---

2 DML — Data Manipulation Language

👉 These commands **work with the data inside tables**.

INSERT → add new data

UPDATE → change existing data

DELETE → remove data

SELECT → get data (some people call SELECT DQL too)

---

3 DCL — Data Control Language

👉 These commands **control permissions**.

GRANT → give access

REVOKE → take back access

---

4 TCL — Transaction Control Language

👉 These commands **control transactions** (save or undo changes).

COMMIT → save changes permanently

ROLLBACK → undo changes

SAVEPOINT → set a point to rollback to later

⭐ In short:

DDL → Structure

DML → Data

DCL → Access

TCL → Transactions

---

21. CREATING NEW DATABASE

- CREATE means "make something new" in the database.

```
CREATE DATABASE school;                    -- here ; work as full stop
```

- DROP means delete it completely from the database.

```
DROP DATABASE school;                      -- drop means to delete
```

---

22. CREATE OUR FIRST TABLE

- The USE command is used to select which database you want to work with.

```
USE school;
CREATE TABLE student (
Id INT PRIMARY KEY,
Name VARCHAR(50),
Age INT NOT NULL );

INSERT INTO students VALUES ( 1,"Mohini",23 );
SELECT * FROM student;
```

-- The INSERT INTO command is used to add new data (rows) into a table.

---

23. DATABASE RELATED QUERIES

```
CREATE DATABASE school;
CREATE DATABASE IF NOT EXISTS school;          -- if it will exists it will show warning
```

```
DROP DATABASE school;
DROP DATABASE IF EXISTS school;                -- if it exists it will drop it
```

```
SHOW DATABASE;                 -- to show all database
```

---

24. TABLE RELATED QUERIES

INSERT INTO students VAULSE (2,”mona”,21); -- to insert value in table (in order)
INSERT INTO students (ID,Age,Name) VALUES (3,22,”rohit”) – insert value in unordered
INSERT INTO students VALUES
(4,”pooja”,24),
(5,”priya”,20); -- to insert multiple values at the time

1 **SELECT** — To **fetch data** from a table. **Example:** SELECT * FROM students;
2 **DISTINCT** — To get **unique values**, removing duplicates. **Example:** SELECT DISTINCT city FROM customers;
3 **WHERE** — To **filter** data with a condition. **Example:** SELECT * FROM students WHERE age > 18;
4 **ORDER BY** — To **sort** the result (ASC or DESC). **Example:** SELECT * FROM students ORDER BY name ASC;
5 **GROUP BY** — To **group rows** having the same values. **Example:** SELECT city, COUNT(*) FROM customers GROUP BY city;
6 **HAVING** — To filter groups (like WHERE but after GROUP BY). **Example:** SELECT city, COUNT(*) FROM customers GROUP BY city HAVING COUNT(*) > 2;
7 **JOIN** — To **combine data** from two or more tables. **Example:** SELECT * FROM orders JOIN customers ON orders.cust_id = customers.id;
8 **INSERT INTO** — To **add new rows** to a table. **Example:** INSERT INTO students (name, age) VALUES ('Amit', 20);
9 **UPDATE** — To **change existing data** in a table. **Example:** UPDATE students SET age = 21 WHERE name = 'Amit';
10 **DELETE** — To **remove rows** from a table. **Example:** DELETE FROM students WHERE age < 18;
1 **CREATE TABLE** — To **create a new table**. **Example:** CREATE TABLE students (id INT, name VARCHAR(50));
2 **DROP TABLE** — To **delete an entire table**. **Example:** DROP TABLE students;
3 **ALTER TABLE** — To **modify a table** (add, remove columns, etc.). **Example:** ALTER TABLE students ADD email VARCHAR(50);
4 **LIMIT** — To **limit the number of rows** returned. **Example:** SELECT * FROM students LIMIT 5;
5 **UNION** — To **combine results** of two queries. **Example:** SELECT city FROM customers UNION SELECT city FROM suppliers;
6 **SHOW** — To **display database or table info**. **Example:** SHOW TABLES; — lists all tables in the database. **Example:** SHOW DATABASES; — lists all databases. **Example:** SHOW COLUMNS FROM students; — shows columns in a table.
7 **DROP** — To **delete** a database, table, or column **permanently**. **Example:** DROP TABLE students; — deletes the whole students table. **Example:** DROP DATABASE school; — deletes the school database.
8 **TRUNCATE** — To **remove all rows** from a table but **keep its structure**. **Example:** TRUNCATE TABLE students; — clears all data, table stays.

9️⃣ **DESCRIBE** — To see the **structure** of a table (columns, types). **Example:** DESCRIBE students;
0️⃣ **USE** — To **select which database** to work on. **Example:** USE school;
1️⃣ **RENAME** — To **rename a table**. **Example:** RENAME TABLE students TO alumni;
2️⃣ **BACKUP** — (Not a command but done using tools or mysqldump) — to **save a copy** of a database.
3️⃣ **INDEX** — To **create an index** to speed up search. **Example:** CREATE INDEX idx_name ON students(name);

---

25. VIEW IN MYSQL

- A **View** is a **virtual table** based on the result of a SQL query.
- It does **not store data physically**, it just saves the **query**.
- When you call the view, it runs the query and shows the result.

Definition:

👉 A View is a saved SQL query that can be treated as a table for easier data retrieval and security.

Example:

```
-- Create a view to show only high-salary employees
CREATE VIEW high_salary AS
SELECT name, salary
FROM employees
WHERE salary > 50000;

-- Use the view
SELECT * FROM high_salary;
```

---

26. SUBQUERY IN MYSQL

- A **Subquery** is a query **inside another query**.
- It is used when one query depends on the result of another query.
- It can be in the WHERE, FROM, or SELECT clause.

Definition:

👉 A Subquery is a nested SQL query that provides results to the main query.

Example:

```
-- Find employees who earn more than the average salary
SELECT name, salary
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);
```

☑ **In short:**

- **View** = saved query (virtual table).
- **Subquery** = query inside another query (temporary, used within a statement).

---

27. WHAT IS A JOIN?

👉 **A JOIN is used to combine rows from two or more tables based on a related column between them.**

◇ 1. INNER JOIN

Definition:

Returns only the rows that have **matching values in both tables**.

Example:

```
SELECT e.name, d.dept_name
FROM employees e
INNER JOIN departments d
ON e.dept_id = d.dept_id;
```

☑ Shows employees who belong to a valid department.

◇ 2. LEFT JOIN (LEFT OUTER JOIN)

Definition:

Returns **all rows from the left table**, and the matching rows from the right table. If no match, returns **NULL**.

Example:

```
SELECT e.name, d.dept_name
FROM employees e
LEFT JOIN departments d
ON e.dept_id = d.dept_id;
```

☑ Shows all employees, even if some don't have a department.

◇ 3. RIGHT JOIN (RIGHT OUTER JOIN)

Definition:

Returns **all rows from the right table**, and the matching rows from the left table. If no match, returns **NULL**.

Example:

```
SELECT e.name, d.dept_name
FROM employees e
RIGHT JOIN departments d
ON e.dept_id = d.dept_id;
```

☑ Shows all departments, even if no employees are assigned.

◇ 4. FULL JOIN (FULL OUTER JOIN)

Definition:

Returns **all rows from both tables** (matched + unmatched).

⚠ MySQL does not directly support FULL JOIN, but we can simulate it using UNION.

Example:

```
SELECT e.name, d.dept_name
FROM employees e
LEFT JOIN departments d
ON e.dept_id = d.dept_id
UNION
SELECT e.name, d.dept_name
FROM employees e
RIGHT JOIN departments d
ON e.dept_id = d.dept_id;
```

☑ Shows all employees and all departments, including those without matches.

◇ 5. CROSS JOIN

Definition:

Returns the **cartesian product** of both tables (every row of one table combined with every row of the other table).

Example:

```
SELECT e.name, d.dept_name
FROM employees e
CROSS JOIN departments d;
```

☑ If employees = 3 and departments = 3 → result = 9 rows.

◇ 6. SELF JOIN

Definition:

A **Self Join** is a regular join where a table is **joined with itself**.

- It is used to compare rows within the same table.
- Useful for finding **hierarchies** or **relationships** in one table (like employees & managers).

Example:

Suppose we have an employees table:

emp_id	name	manager_id
1	Riya	NULL
2	Amit	1
3	Neha	1
4	Rahul	2

👉 We want to find each employee with their manager's name.

```
SELECT e.name AS Employee, m.name AS Manager
FROM employees e
LEFT JOIN employees m
ON e.manager_id = m.emp_id;
```

☑ Result:

Employee	Manager
Riya	NULL
Amit	Riya
Neha	Riya
Rahul	Amit

🔑 **Final List of Joins in MySQL (with definitions):**

1. **INNER JOIN** → only matching rows.
2. **LEFT JOIN** → all left + matches from right.
3. **RIGHT JOIN** → all right + matches from left.
4. **FULL JOIN** → all rows from both (use UNION in MySQL).
5. **CROSS JOIN** → cartesian product (all combinations).
6. **SELF JOIN** → join a table with itself (hierarchies).

---

28. CASE IN SQL

Works like IF-ELSE condition.

Example:

```
SELECT name,
CASE
  WHEN marks >= 90 THEN 'A'
  WHEN marks >= 75 THEN 'B'
  ELSE 'C'
END AS grade
FROM students;
```

29. EXISTS

Checks if a **subquery returns any rows**.

Returns **TRUE** if at least one row exists, otherwise **FALSE**.

Syntax:

```
SELECT column_name
FROM table
WHERE EXISTS (subquery);
```

30. SELECT INTO in SQL

SELECT INTO is used to **create a new table and copy data into it** from another table.

Table must **not exist** before.

```
SELECT column1, column2, …
INTO new_table
FROM old_table
WHERE condition;
```

31. INSERT INTO SELECT

- Used when the table **already exists**.
- It copies data from one table to another existing table.

Example:

```
INSERT INTO employees_archive (id, name, salary)
SELECT id, name, salary
FROM employees
WHERE salary > 50000;
```

32. NORMALIZATION

Normalization (Definition)

Normalization is the process of **organizing data in a database** so that:

1. There is **no repeated data** (no redundancy).
2. Data is **stored in the correct tables**.
3. The database is **efficient, accurate, and easy to maintain**.

In simple words:

Normalization = **Organize data → Remove duplicates → Keep it clean and manageable**.

Types of Normalization (Normal Forms)

1. **1NF (First Normal Form):**
   - Each column contains **only one value**.
   - No lists or multiple values in a single column.
2. **2NF (Second Normal Form):**
   - Table is in 1NF.
   - Each column depends on the **whole primary key**, not part of it.
3. **3NF (Third Normal Form):**
   - Table is in 2NF.
   - No column depends on **another non-key column**.
4. **BCNF (Boyce-Codd Normal Form):**
   - Stronger version of 3NF.
   - Every column that determines another column must be a **key**.
5. **4NF (Fourth Normal Form):**
   - Table is in BCNF.
   - No column contains **multiple independent lists**.
6. **5NF (Fifth Normal Form):**
   - Table is in 4NF.
   - Split tables so that **all redundancy is removed**; data can be combined only using **joins**.

---

33. TRANSACTIONS & ACID

Definition

A **Transaction** is a set of SQL statements executed as a single unit of work.

Use

- Ensures **data consistency**
- Used in **banking, payments, booking systems**

ACID Properties

- **A**tomicity → All or nothing
- **C**onsistency → Data must be valid
- **I**solation → Transactions don't affect each other
- **D**urability → Once committed, changes are permanent

Example Query

```
START TRANSACTION;

UPDATE Accounts SET balance = balance - 1000 WHERE acc_id = 1;
UPDATE Accounts SET balance = balance + 1000 WHERE acc_id = 2;

COMMIT;
```

Explanation

If money is deducted but not added (failure), **ROLLBACK** will undo changes. If all succeed, **COMMIT** makes them permanent.

---

34. STORED PROCEDURES

Definition

A **Stored Procedure** is a set of SQL statements stored in the database and executed by calling it.

Use

- Code reusability
- Faster execution (precompiled)
- Secure business logic

Example Query

```
DELIMITER //
CREATE PROCEDURE GetEmployee(IN empid INT)
BEGIN
   SELECT * FROM Employees WHERE emp_id = empid;
END //
DELIMITER ;
```

Explanation

Here, GetEmployee(101) will return employee details with emp_id=101.

---

35. TRIGGERS

Definition

A **Trigger** is a block of SQL code that runs automatically when an event occurs (INSERT, UPDATE, DELETE).

Use

- Enforce rules
- Automatically update logs or history tables

Example Query

```
CREATE TRIGGER after_insert_employee
AFTER INSERT ON Employees
FOR EACH ROW
INSERT INTO EmployeeLogs(emp_id, action)
```

```
VALUES (NEW.emp_id, 'Employee Added');
```

Explanation

Whenever a new employee is added, it automatically logs the event in EmployeeLogs.

---

36. DIFFERENCE BETWEEN

1. SQL vs MySQL

SQL	MySQL
Language (Structured Query Language)	Database Management System (RDBMS)
Standard language to interact with databases	Software that uses SQL to manage data

---

2. DELETE vs TRUNCATE vs DROP

Command	Use	Can Rollback?	Removes Structure?
DELETE	Removes selected rows	☑ Yes	❌ Keeps table
TRUNCATE	Removes all rows (faster)	❌ No	❌ Keeps table
DROP	Removes table completely	❌ No	☑ Yes

---

3. WHERE vs HAVING

WHERE	HAVING
Used to filter rows before grouping	Used to filter groups after grouping
Works with any column	Works with aggregate functions (SUM, AVG, COUNT)

---

4. PRIMARY KEY vs FOREIGN KEY

PRIMARY KEY	FOREIGN KEY
Uniquely identifies each row in a table	References primary key of another table
Only one primary key per table	Multiple foreign keys allowed
Cannot be NULL	Can contain NULL

---

5. CHAR vs VARCHAR

CHAR	VARCHAR
Fixed length	Variable length
Faster, but wastes space	Efficient, saves space

---

6. UNION vs UNION ALL

UNION	UNION ALL
Combines results but removes duplicates	Combines results including duplicates
Slower (checks duplicates)	Faster

---

7. INNER JOIN vs OUTER JOIN

INNER JOIN	OUTER JOIN
Returns only matching rows	Returns matching + non-matching rows

8.EXISTS vs IN

EXISTS	IN
Checks if a subquery returns rows	Checks if a value exists in a list/subquery

EXISTS	IN
Faster for correlated subqueries	Better for small lists

Example:

```
SELECT name FROM students s
WHERE EXISTS (SELECT 1 FROM marks m WHERE s.id = m.student_id);
```

---

9. ANY vs ALL

ANY	ALL
True if condition matches at least one value	True if condition matches all values

Example:

```
SELECT * FROM employees
WHERE salary > ANY (SELECT salary FROM employees WHERE dept='HR');
```

10. View vs Table

Feature	View	Table
Storage	Virtual, no storage	Physically stores data
Updates	Limited	Full insert/update/delete
Use	Security, abstraction	Store data permanently

---

11. Procedure vs Function

Feature	Procedure	Function
Return Value	Can return multiple values	Must return one value
Use in Query	Cannot be used in SELECT	Can be used in SELECT
Purpose	Perform tasks	Compute and return result

---

12. Clustered vs Non-Clustered Index

Feature	Clustered	Non-Clustered
Data Storage	Reorders table	Separate structure
Count per Table	Only one	Multiple allowed
Speed	Faster for range queries	Slower for range but flexible

---

13. Comparison Tables

(a) Primary Key vs Unique Key

Feature	Primary Key	Unique Key
Null Values	Not allowed	One null allowed
Uniqueness	Must be unique	Must be unique
Count per Table	Only one	Multiple allowed
