Module 4 – Introduction to DBMS

# Introduction to SQL

1. **What is SQL, and why is it essential in database management?**

SQL stands for **Structured Query Language**.  
It is a **standard programming language** used to communicate with and manage data in a **Relational Database Management System (RDBMS)**.

* **Why is SQL Essential in Database Management?**

1. **Data Retrieval:**  
   SQL makes it easy to fetch only the required data from large databases using queries.
2. **Data Manipulation:**  
   It allows inserting, updating, and deleting records in a database.
3. **Data Definition:**  
   SQL helps in creating and modifying tables, views, indexes, and other database structures.
4. **Standardization:**  
   SQL is a standardized language supported by almost all popular databases (MySQL, Oracle, SQL Server, PostgreSQL, SQLite, etc.).
5. **Security & Access Control:**  
   SQL provides features to control user access and protect data from unauthorized usage.
6. **Transaction Management:**  
   SQL supports transactions (like COMMIT and ROLLBACK) to ensure **data accuracy, integrity, and consistency**.
7. **Explain the difference between DBMS and RDBMS.**

* **DBMS** is software that stores and manages data, usually in files or simple databases. It does not support relations between data, has high redundancy, less security, and limited functionality.
* **RDBMS** is an advanced DBMS based on the relational model. It stores data in **tables (rows & columns)**, supports relationships using **primary key & foreign key**, allows **normalization** to remove redundancy, provides **transaction management (ACID properties)**, and offers better **security**.

1. **Describe the role of SQL in managing relational databases.**

**Roles of SQL:**

1. **Data Definition (DDL):** Create and modify database structures like tables, views, and indexes (CREATE, ALTER, DROP).
2. **Data Manipulation (DML):** Insert, update, delete, and retrieve data (INSERT, UPDATE, DELETE, SELECT).
3. **Data Control (DCL):** Manage user permissions and security (GRANT, REVOKE).
4. **Transaction Control (TCL):** Ensure data consistency and reliability using COMMIT, ROLLBACK, and SAVEPOINT.
5. **Querying Data:** Retrieve meaningful information from large datasets with conditions, joins, and sorting.
6. **What are the key features of SQL**

Key Features of SQL

1. **Data Definition –** SQL can create, modify, and delete database structures using DDL commands
2. **Data Manipulation –** It can insert, update, delete, and retrieve data using DML commands
3. **Data Querying –** Powerful SELECT statements allow filtering, sorting, and joining data from multiple tables.
4. **Transaction Control –** Supports transactions with COMMIT, ROLLBACK, and SAVEPOINT to maintain data consistency.
5. **Security & Access Control –** Provides permissions to users with DCL commands
6. **Portability –** SQL is a standardized language supported by most RDBMS
7. **High Performance –** Handles large volumes of data efficiently with indexing and optimization.

# 2. SQL Syntax

**1. What are the basic components of SQL syntax?**

* Keywords → Predefined words that perform specific tasks
* Identifiers → Names given to database objects like tables, columns, or views
* Clauses → Provide conditions or instructions
* Expressions → Combinations of values, operators, and functions that return a single value.
* Predicates → Conditions that filter data
* Statements → Complete SQL commands formed using the above components

1. **Write the general structure of an SQL SELECT statement.**

SELECT column1, column2, ...

FROM table\_name

WHERE condition

GROUP BY column

HAVING condition

ORDER BY column ASC|DESC;

**🔹 Explanation of Clauses:**

* **SELECT** → Specifies the columns to retrieve.
* **FROM** → Specifies the table(s).
* **WHERE** → Filters rows based on condition.
* **GROUP BY** → Groups rows based on a column.
* **HAVING** → Filters groups (used with GROUP BY).
* **ORDER BY** → Sorts the result (ascending/descending).

**3. Explain the role of clauses in SQL statements.**

* **Role of Clauses in SQL Statements**

Clauses in SQL are **building blocks** that add conditions, structure, or rules to an SQL statement. They help in refining queries and controlling how data is selected, grouped, or ordered.

**Important Clauses and Their Roles:**

1. **SELECT clause** → Specifies which columns or expressions to display.
2. **FROM clause** → Indicates the table(s) from which data is retrieved.
3. **WHERE clause** → Filters rows based on conditions.
4. **GROUP BY clause** → Groups rows that have the same values in specified columns.
5. **HAVING clause** → Filters groups after grouping is applied.
6. **ORDER BY clause** → Sorts the result set in ascending or descending order.

# *3.* SQL Constraints

**1. What are constraints in SQL? List and explain the different types of constraints.**

Constraints in SQL

Constraints are rules applied on table columns to maintain the accuracy, validity, and integrity of data in a database. They prevent invalid data from being entered and ensure consistency.

**🔹 Types of Constraints (Theory)**

1. **NOT NULL** → Ensures that a column cannot have empty (NULL) values.
2. **UNIQUE** → Ensures that all values in a column are different.
3. **PRIMARY KEY** → Uniquely identifies each record in a table (cannot be NULL and must be unique).
4. **FOREIGN KEY** → Maintains relationship between two tables and ensures referential integrity.
5. **CHECK** → Ensures that values in a column meet a specific condition.
6. **DEFAULT** → Assigns a default value to a column if no value is provided.

**2.How do PRIMARY KEY and FOREIGN KEY constraints differ?**

 A **Primary Key** is a constraint that **uniquely identifies each record** in a table. It does not allow duplicate values or NULLs. Each table can have only one primary key, which ensures the uniqueness and identity of its records.

 A **Foreign Key** is a constraint that creates a **link between two tables**. It refers to the **Primary Key of another table**, maintaining referential integrity. A foreign key can have duplicate values and can also contain NULL (if not restricted). A table can have multiple foreign keys.

**3. What is the role of NOT NULL and UNIQUE constraints?**

Role of NOT NULL and UNIQUE Constraints

* The NOT NULL constraint is used to ensure that a column must always have a value. It prevents empty (NULL) entries in a column and guarantees that essential data is never missing.
* The UNIQUE constraint ensures that all the values in a column are different from each other. It prevents duplication of data and maintains uniqueness. Unlike PRIMARY KEY, a UNIQUE column can accept one NULL value.

# 4. Main SQL Commands and Sub-commands (DDL)

**1. Define the SQL Data Definition Language (DDL).**

**Data Definition Language (DDL)** in SQL is a set of commands used to **define, create, modify, and delete the structure of database objects** such as tables, views, indexes, and schemas.

**🔹 Common DDL Commands**

1. **CREATE** → Creates a new database object (table, view, index, etc.).
2. **ALTER** → Modifies the structure of an existing database object.
3. **DROP** → Deletes database objects permanently.
4. **TRUNCATE** → Removes all records from a table but keeps its structure.
5. **RENAME** → Renames a database object.

**2. Explain the CREATE command and its syntax.**

The CREATE command in SQL is a DDL (Data Definition Language) statement used to create new database objects, such as databases, tables, views, or indexes. The most common use is to create a table where data can be stored.

CREATE TABLE table\_name (

column1 datatype constraints,

column2 datatype constraints,

...

);

**🔹 Example**

CREATE TABLE Students (

StudentID INT PRIMARY KEY,

Name VARCHAR(50) NOT NULL,

Age INT CHECK (Age >= 18),

Email VARCHAR(100) UNIQUE

);

This command creates a table named **Students** with columns StudentID, Name, Age, and Email, along with constraints.

1. **What is the purpose of specifying data types and constraints during table creation?**

**=> Purpose of Specifying Data Types and Constraints during Table Creation**

When creating a table in SQL, data types and constraints are specified to ensure the database stores data accurately, efficiently, and consistently.

* Data Types define the kind of values a column can hold (e.g., INT for numbers, VARCHAR for text, DATE for dates). They ensure storage efficiency and prevent invalid data entry.
* Constraints are rules applied on columns (like NOT NULL, UNIQUE, PRIMARY KEY, CHECK) that maintain data integrity, accuracy, and validity by restricting invalid or duplicate values.

# 5. ALTER Command

**1. What is the use of the ALTER command in SQL?**

The **ALTER command** is used to **modify the structure of an existing table** without deleting its data. It allows adding or removing columns, changing column data types or sizes, renaming tables or columns, and adding or dropping constraints, providing **flexibility and adaptability** in database design.

**2. How can you add, modify, and drop columns from a table using ALTER?**

Using ALTER to Add, Modify, and Drop Columns

The ALTER command in SQL is used to change the structure of an existing table without affecting the data stored in it.

* Adding a column: You can add a new column to a table to store additional information as needed.
* Modifying a column: You can change the data type, size, or constraints of an existing column to meet updated requirements.
* Dropping a column: You can remove a column from a table if it is no longer needed.

# 6. DROP Command

**1. What is the function of the DROP command in SQL?**

**📌 Function of the DROP Command in SQL**

The DROP command in SQL is a DDL (Data Definition Language) statement used to permanently delete database objects such as tables, views, or databases. When an object is dropped, all its data and structure are removed and cannot be recovered.

**2. What are the implications of dropping a table from a database?**

Dropping a table in a database is a **Data Definition Language (DDL)** operation that has several important implications:

* **Data loss** → All records are permanently deleted.
* **Schema loss** → Table structure, indexes, constraints, triggers are removed.
* **Dependency issues** → Views, foreign keys, procedures linked to it may break.
* **Space freed** → Storage occupied is released.
* **No undo** → Can’t be rolled back (except via backup).

# 7. Data Manipulation Language (DML)

**1. Define the INSERT, UPDATE, and DELETE commands in SQL.**

* **INSERT Command** – Used to add new records into a table.  
  Example: INSERT INTO Students (id, name) VALUES (1, 'Raj');
* **UPDATE Command** – Used to modify existing records in a table.  
  Example: UPDATE Students SET name = 'Ravi' WHERE id = 1;
* **DELETE Command** – Used to remove records from a table.  
  Example: DELETE FROM Students WHERE id = 1;

**2. What is the importance of the WHERE clause in UPDATE and DELETE operations?**

* The **WHERE clause** in UPDATE and DELETE is used to **specify conditions** on which rows should be updated or deleted. Without the WHERE clause, the command affects **all rows** in the table, which may cause accidental data loss or unwanted changes. It ensures **safe, targeted modifications** instead of altering the entire table.

# 8. Data Query Language (DQL)

**1. What is the SELECT statement, and how is it used to query data?**

* The **SELECT statement** in SQL is a **Data Query Language (DQL) command** used to **retrieve data** from a database. It can display specific columns or all records and allows users to **filter rows (WHERE)**, **sort results (ORDER BY)**, **join multiple tables**, and apply **functions like COUNT, SUM, AVG, etc.** to organize and analyze data — all without modifying the database.

**2. Explain the use of the ORDER BY and WHERE clauses in SQL queries.**

The **WHERE clause** in SQL is used to **filter rows** based on specific conditions so that only the required records are retrieved, updated, or deleted, while the **ORDER BY clause** is used to **sort the results** in ascending or descending order. Together, they help in getting **accurate and well-organized query results** from the database.

# 9. Data Control Language (DCL)

**1. What is the purpose of GRANT and REVOKE in SQL?**

The **GRANT** and **REVOKE** commands in SQL are **DCL statements** used to manage database security: **GRANT** gives users specific privileges (like SELECT, INSERT, UPDATE, DELETE) on database objects, while **REVOKE** removes those privileges, ensuring **controlled access and protection of data**.

**2. How do you manage privileges using these commands?**

Privileges in SQL are managed using GRANT and REVOKE commands. With GRANT, the database administrator can assign permissions (such as SELECT, INSERT, UPDATE, DELETE) to users or roles on specific database objects. With REVOKE, those permissions can be taken back when no longer needed. 👉 This ensures proper security, controlled access, and data protection in the database.

# 10. Transaction Control Language (TCL)

**1. What is the purpose of the COMMIT and ROLLBACK commands in SQL?**

The **COMMIT** and **ROLLBACK** commands in SQL are used for **transaction control**, where **COMMIT** permanently saves all changes made during a transaction, and **ROLLBACK** undoes those changes to restore the database to its previous state, ensuring **data consistency, integrity, and control**.

**2. Explain how transactions are managed in SQL databases.**

Transactions in SQL are managed using **Transaction Control Language (TCL) commands** to ensure **data consistency and reliability**. A transaction is a sequence of operations treated as a single unit. SQL uses **BEGIN/START TRANSACTION** to start, **COMMIT** to permanently save changes, and **ROLLBACK** to undo changes if an error occurs. Transactions follow the **ACID properties** (Atomicity, Consistency, Isolation, Durability) to guarantee accuracy, reliability, and safe recovery of data.

# 11. SQL Joins

**1. Explain the concept of JOIN in SQL. What is the difference between INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL OUTER JOIN?**

A **JOIN** in SQL is used to **combine rows** from two or more tables based on a related column. It helps in retrieving data spread across multiple tables.

* **INNER JOIN** → Returns only the rows where there is a **match** in both tables.
* **LEFT JOIN (LEFT OUTER JOIN)** → Returns **all rows from the left table** and the matching rows from the right table; unmatched rows in the right table show as NULL.
* **RIGHT JOIN (RIGHT OUTER JOIN)** → Returns **all rows from the right table** and the matching rows from the left table; unmatched rows in the left table show as NULL.
* **FULL OUTER JOIN** → Returns **all rows from both tables**, with NULLs for non-matching rows on either side.

**2. How are joins used to combine data from multiple tables?**

Joins in SQL are used to **combine data from multiple tables** by matching rows through a **common column** (such as a primary key and foreign key). They help in retrieving **related information stored across different tables**, making it possible to view complete data in a single result set

# 12. SQL Group By

**1. What is the GROUP BY clause in SQL? How is it used with aggregate functions?**

The GROUP BY clause in SQL is used to group rows that have the same values in one or more columns into summary rows, helping organize data into categories instead of working with individual rows. It is used with aggregate functions like COUNT(), SUM(), AVG(), MIN(), and MAX() to perform calculations on each group separately.

**2. Explain the difference between GROUP BY and ORDER BY.**

The GROUP BY clause in SQL is used to group rows with the same values into summary rows, often with aggregate functions like COUNT(), SUM(), or AVG(), while the ORDER BY clause is used to sort the result set in ascending or descending order without grouping the data.

# 13. SQL Stored Procedure

**1. What is a stored procedure in SQL, and how does it differ from a standard SQL query?**

**Stored Procedure in SQL:-**

A stored procedure is a precompiled set of SQL statements stored in the database that can be executed repeatedly, accept parameters, and perform complex operations like insert, update, or delete.

**Difference:-**

A **stored procedure** in SQL is a **precompiled set of SQL statements** stored in the database that can be executed repeatedly with or without parameters and can include complex logic like loops and conditions, whereas a **standard SQL query** is a **single SQL statement** executed on demand, usually for a one-time operation, without precompilation or reusable structure.

**2. Explain the advantages of using stored procedures.**

**Advantages of Stored Procedures:-**

"Stored procedures are precompiled SQL code that can be reused multiple times, improving performance and reducing network traffic. They enhance security by controlling access, are easier to maintain, and can handle complex operations using control-of-flow statements like IF and WHILE."

# 14. SQL View

**1. What is a view in SQL, and how is it different from a table?**

**View in SQL:-**

A view in SQL is a virtual table created from one or more tables using a query. It does not store data physically and displays data dynamically, simplifying queries, enhancing security, and formatting data as needed.

**Difference Between a View and a Table:-**

A table stores data physically and is fully updatable, while a view is a virtual table created using a query, may be read-only, depends on underlying tables, and is used to simplify queries, provide security, and present data in a specific format.

**2. Explain the advantages of using views in SQL databases.**

**Advantages of Views in SQL**

1. **Simplify Complex Queries:** Views can combine multiple tables and present data in a simplified way, so users don’t need to write complex joins repeatedly.
2. **Data Security:** Access can be restricted to specific columns or rows through views, protecting sensitive data.
3. **Consistency and Abstraction:** Views provide a consistent interface to the data even if the underlying tables change, hiding the complexity from users.
4. **Reusability:** A view can be used multiple times in queries without rewriting the original complex SQL.
5. **Custom Data Presentation:** Views can present data in a format or structure suitable for reporting or application use.

# 15. SQL Triggers

**1. What is a trigger in SQL? Describe its types and when they are used.**

**Trigger in SQL?**

A **trigger** is a **special type of stored procedure** that automatically executes in response to certain events on a table or view, such as **INSERT, UPDATE, or DELETE**. Triggers are used to enforce business rules, maintain audit trails, and ensure data integrity.

**Types of Triggers**

1. **Before Trigger:**
   * Executes **before** an INSERT, UPDATE, or DELETE operation on a table.
   * Used to **validate or modify data** before it is saved.
2. **After Trigger:**
   * Executes **after** an INSERT, UPDATE, or DELETE operation.
   * Used for **audit logs, notifications, or cascading changes**.
3. **Instead of Trigger:**
   * Executes **instead of** the triggering action (common on views).
   * Used to **modify how data is inserted, updated, or deleted** in complex views.

**When Triggers are Used**

* To **enforce complex business rules** automatically.
* To **maintain audit trails** of changes.
* To **automate system tasks** without manual intervention.

**2. Explain the difference between INSERT, UPDATE, and DELETE triggers.**

* INSERT Trigger
  + Executes automatically when a new row is inserted into a table
  + Can validate or modify data before/after insertion, or maintain audit logs of new records
* UPDATE Trigger
  + Executes automatically when an existing row is updated
  + Can enforce business rules, validate changes, or track modifications in audit tables
* DELETE Trigger
  + Executes automatically when a row is deleted from a table
  + Can prevent deletion of critical data, log deletions, or cascade deletions to related tables

# 16. Introduction to PL/SQL

**1. What is PL/SQL, and how does it extend SQL's capabilities?**  **PL/SQL:-**

PL/SQL (Procedural Language/SQL) is Oracle’s procedural extension of SQL that combines SQL with programming features like variables, loops, conditions, and exception handling to perform complex and powerful database operations.

**How PL/SQL Extends SQL’s Capabilities**

1. **Procedural Logic:** Unlike standard SQL, PL/SQL supports **IF-ELSE, loops, and case statements**, enabling conditional and iterative processing.
2. **Variables and Constants:** You can store values in **variables**, use **constants**, and manipulate them in your programs.
3. **Error Handling:** PL/SQL allows **exception handling** to manage runtime errors gracefully.
4. **Modular Programming:** Supports **procedures, functions, packages, and triggers** for reusable and organized code.
5. **Improved Performance:** Blocks of PL/SQL can be executed together, reducing network traffic and improving efficiency compared to multiple separate SQL statements.

**2. List and explain the benefits of using PL/SQL.**

* **Tight SQL Integration:** Execute SQL statements directly.
* **Procedural Capabilities:** Supports loops, conditions, and control-of-flow.
* **Modularity:** Organize code into procedures, functions, packages, and triggers.
* **Error Handling:** Manage runtime errors with exceptions.
* **Improved Performance:** Execute multiple SQL statements in a single block.
* **Security:** Control access and hide direct table operations.

# 17. PL/SQL Control Structures

**1. What are control structures in PL/SQL? Explain the IF-THEN and LOOP control structures.**

**Control Structures in PL/SQL**

**Control structures** in PL/SQL are programming constructs that **control the flow of execution** of statements in a program. They allow conditional execution, repetition, and branching in your PL/SQL code.

The main types are:

1. **Conditional Statements** (e.g., IF-THEN, IF-THEN-ELSE, CASE)
2. **Looping Statements** (e.g., LOOP, WHILE LOOP, FOR LOOP)

**IF-THEN Control Structure**

* **Purpose:** Executes a block of code **only if a specified condition is true**.
* **Syntax:**

IF condition THEN

-- statements to execute

END IF;

* Can be extended with **ELSE** or **ELSIF** for multiple conditions.

**LOOP Control Structure**

* **Purpose:** Repeats a block of code **multiple times** until explicitly exited.
* **Types:**
  1. **Simple LOOP:** Executes until an **EXIT** statement is encountered.
  2. **WHILE LOOP:** Executes as long as a condition is true.
  3. **FOR LOOP:** Executes a fixed number of times.
* **Syntax (Simple LOOP):**

LOOP

-- statements

EXIT WHEN condition;

END LOOP;

**2. How do control structures in PL/SQL help in writing complex queries?**

Control structures in PL/SQL help write complex queries by allowing conditional execution, repetition, and branching within your code.

* Conditional statements (IF-THEN, IF-ELSE, CASE) let you execute different SQL operations based on specific conditions.
* Looping statements (LOOP, WHILE, FOR) allow you to process multiple rows or perform repeated operations efficiently.
* They enable dynamic decision-making and automated processing that standard SQL alone cannot handle.

# 18. SQL Cursors

**1. What is a cursor in PL/SQL? Explain the difference between implicit and explicit cursors.**

**Cursor in PL/SQL?**

A **cursor** in PL/SQL is a **pointer that allows you to retrieve and manipulate rows returned by a query one at a time**. Cursors are used when a query returns **multiple rows**, and you need to process each row individually.

**Difference between implicit and explicit cursors:-**

Implicit cursors are automatically created by PL/SQL for single-row queries like INSERT, UPDATE, DELETE, and are managed internally with limited control. Explicit cursors, on the other hand, are declared and controlled by the programmer for queries that return multiple rows, allowing row-by-row processing with full control over opening, fetching, and closing the cursor.

**2. When would you use an explicit cursor over an implicit one?**

You would use an **explicit cursor** over an implicit one when you need to **process multiple rows returned by a query individually**, or when you require **more control over row-by-row processing**, such as **fetching, looping, and closing the cursor explicitly**. Implicit cursors are sufficient only for **single-row operations** where detailed control isn’t needed.

# 19. Rollback and Commit Savepoint

**1. Explain the concept of SAVEPOINT in transaction management. How do ROLLBACK and COMMIT interact with savepoints?**

**SAVEPOINT in Transaction Management**

A **SAVEPOINT** is a **marker within a transaction** that allows you to **partially undo** changes without rolling back the entire transaction. It helps manage complex transactions by giving control over which part of the transaction to undo if something goes wrong.

**Interaction with ROLLBACK and COMMIT**

1. **ROLLBACK TO SAVEPOINT**
   * Undoes changes **only up to the specified savepoint**, keeping the rest of the transaction intact.
2. **COMMIT**
   * Saves **all changes permanently**, including those before and after any savepoints.
   * After a COMMIT, savepoints are cleared and cannot be used.

**2. When is it useful to use savepoints in a database transaction?**

Savepoints are used in complex transactions to mark specific points within the transaction. They allow you to roll back only part of the transaction if an error occurs, without affecting earlier successful operations. Savepoints help in error recovery, testing critical steps, and managing multi-step or long-running database operations efficiently.