Module 4- Introduction to DBMS

1. Introduction to SQL

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

Ans : SQL (Structured Query Language) is a standard programming language used to manage and manipulate relational databases. It is essential because it allows users to insert, update, delete, and retrieve data efficiently. SQL helps in maintaining the accuracy and integrity of data, supports security through permissions, and enables interaction with databases through simple, human-readable commands.

2. Explain the difference between DBMS and RDBMS.

Ans : DBMS(Database Management System)

→Stores data in files or non-relational formats

→Does not support relationships between data

→Suitable for small amounts of data

→Examples: Microsoft Access, file systems

RDBMS(Relational Database management System)

→Stores data in tables

→Supports relationships using foreign keys

→Suitable for large, complex data sets

→Examples: MySQL, Oracle, SQL Server

3. Describe the role of SQL in managing relational databases.

Ans : SQL plays a crucial role in relational databases by providing commands to create tables, insert data, update records, delete records, and query the database. It also helps in setting up relationships between tables, enforcing data constraints, managing user permissions, and ensuring data consistency and security.

4. What are the key features of SQL?

- Data Querying: Retrieve specific data using SELECT.
- Data Manipulation: Insert, update, delete data (INSERT, UPDATE, DELETE).
- Data Definition: Create or modify database objects (CREATE, ALTER, DROP).
- Data Control: Manage access using GRANT and REVOKE.
- Transaction Control: Support for COMMIT and ROLLBACK.
- Standardized Language: Widely accepted and used across different platforms.

2. SQL Syntex

1.What are the basic components of SQL syntex?

Ans : The basic components of SQL syntax include:

- Keywords: Reserved words like SELECT, FROM, WHERE, etc.
- Identifiers: Names of tables, columns, databases, etc.
- Expressions: Formulas used to calculate values.
- Clauses: Parts of a statement (e.g., WHERE, GROUP BY, ORDER BY).
- Predicates: Conditions used to filter records (e.g., age > 18).
- Statements: Complete SQL commands (e.g., SELECT * FROM students;).

2. Write the general structure of an SQL SELECT statement.

Ans :

```
SELECT column1, column2, ...
FROM table_name
WHERE condition
GROUP BY column
HAVING condition
ORDER BY column ASC|DESC;
```

3. Explain the role of clauses in SQL statements.

Ans : Clauses are building blocks of SQL statements. Each clause serves a specific purpose:

- SELECT clause: Specifies which columns to return.
- FROM clause: Indicates the source table(s).
- WHERE clause: Filters rows based on conditions.
- GROUP BY clause: Groups rows that share values in specified columns.
- HAVING clause: Applies conditions to groups created by GROUP BY.
- ORDER BY clause: Sorts the result set by one or more columns.

3. SQL Constraints

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

Ans : Constraints in SQL are rules applied to columns in a table to ensure the validity, accuracy, and integrity of the data.

Types of constraints:

- PRIMARY KEY: Uniquely identifies each row in a table; cannot be NULL.
- FOREIGN KEY: Ensures referential integrity by linking one table to another.

- NOT NULL: Ensures that a column cannot have a NULL value.
- UNIQUE: Ensures all values in a column are different.
- CHECK: Ensures that all values in a column satisfy a specific condition.
- DEFAULT: Assigns a default value if no value is provided.

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

Ans : PRIMARY KEY:

→Uniquely identifies each record in the table

→Cannot contain NULL values

→One PRIMARY KEY per table

→Ensures row identity

FOREIGN KEY

→Refers to the PRIMARY KEY in another table

→Can contain NULL values

→Multiple FOREIGN KEYs allowed

→Ensures relationship integrity between tables

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

Ans :

- NOT NULL: Ensures that a column must always have a value; it cannot be left empty.
  Example: name VARCHAR(50) NOT NULL – every record must have a name.
- UNIQUE: Ensures that all values in a column are different from each other.
  Example: email VARCHAR(100) UNIQUE – no two users can have the same email.

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

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

Ans : Data Definition Language (DDL) is a category of SQL commands used to define and manage database structure or schema. It deals with creating, altering, and deleting tables, indexes, and other database objects.

Common DDL commands include:

- CREATE – to create new tables or databases
- ALTER – to modify existing database objects
- DROP – to delete objects
- TRUNCATE – to remove all data from a table

2. Explain the CREATE command and its syntax.

Ans : The CREATE command is used to create a new database or table in SQL.

Syntax to create a table:

```
CREATE TABLE table_name (
    column1 datatype constraint,
    column2 datatype constraint,
    ...
);
```

Example:

```
CREATE TABLE Students (
    ID INT PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Age INT,
    Email VARCHAR(100) UNIQUE
);
```

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

Ans :

- Data Types define the kind of data a column can store (e.g., INT, VARCHAR, DATE). This helps in saving storage, performing valid operations, and maintaining data accuracy.
- Constraints ensure data integrity, consistency, and validity by enforcing rules like uniqueness (UNIQUE), mandatory values (NOT NULL), and relationships (FOREIGN KEY).

5. ALTER Command

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

Ans : The ALTER command in SQL is used to modify the structure of an existing table without deleting it. It allows you to:

- Add new columns
- Modify existing columns (e.g., change data type or size)
- Drop (delete) columns
- Rename columns or the table
- Add or drop constraints

This command is part of DDL (Data Definition Language).

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

Ans :

- Add a column:

  ALTER TABLE table_name

  ADD column_name datatype;

  Example:

  ALTER TABLE Students

  ADD Address VARCHAR(100);

- Modify a column:

  ALTER TABLE table_name

  MODIFY column_name new_datatype;

  Example:

  ALTER TABLE Students

  MODIFY Name VARCHAR(100);

- Drop a column:

  ALTER TABLE table_name

  DROP COLUMN column_name;

  Example:

  ALTER TABLE Students

  DROP COLUMN Address;

6. DROP Command

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

Ans : The DROP command in SQL is used to permanently delete database objects such as tables, databases, views, indexes, or stored procedures.

Syntax:

DROP TABLE table_name;

Once executed, the object is completely removed from the database and cannot be recovered unless a backup exists.

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

Ans : Dropping a table has serious consequences:

- Permanent Data Loss: All data stored in the table is deleted permanently.
- Loss of Structure: The table schema (column names, data types, constraints) is also erased.
- Breaks Relationships: If the table is referenced by foreign keys in other tables, it may cause errors or require those relationships to be removed first.
- Irreversible Action: DROP does not go to a "recycle bin" — you cannot undo it without restoring from a backup.

7. Data Manipulation Language(DML)

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

- INSERT:
  The INSERT command is used to add new records (rows) into a table.
  Syntax:

  INSERT INTO table_name (column1, column2, ...)

  VALUES (value1, value2, ...);

- UPDATE:
  The UPDATE command is used to modify existing records in a table.
  Syntax:

  UPDATE table_name

  SET column1 = value1, column2 = value2, ...

  WHERE condition;

- DELETE:
  The DELETE command is used to remove existing records from a table.
  Syntax:

  DELETE FROM table_name

  WHERE condition;

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

Ans :

- The WHERE clause is crucial because it filters which records will be updated or deleted.
- Without a WHERE clause:
  - UPDATE will change all rows in the table.
  - DELETE will remove all rows, effectively emptying the table.

Example with WHERE:

DELETE FROM Students WHERE ID = 5;

This deletes only the student with ID 5.

Example without WHERE:

DELETE FROM Students;

This deletes all students from the table.

8. Data Query language(DQL)

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

Ans : The SELECT statement is used in SQL to retrieve data from one or more tables in a database. It allows you to specify the columns you want and apply filters, sorting, or calculations.

Basic Syntax:

SELECT column1, column2, ...

FROM table_name;

Example:

SELECT name, age FROM Students;

This retrieves the name and age columns from the Students table.

You can also use:

- * to select all columns
- WHERE to filter results
- ORDER BY to sort the results

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

Ans : WHERE Clause:
Filters the rows returned by the query based on specified conditions.

SELECT * FROM Students WHERE age > 18;

→ Returns only students older than 18.

- ORDER BY Clause:
  Sorts the result set in ascending (ASC) or descending (DESC) order by one or more columns.

SELECT * FROM Students ORDER BY age DESC;

→ Returns all students sorted by age in descending order.

Combined Example:

SELECT name, age FROM Students

WHERE age >= 18

ORDER BY name ASC;

9. Data Control language(DCL)

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

Ans :

- GRANT is used to give permissions or privileges to users so they can perform specific actions on database objects (like tables, views, procedures).
- REVOKE is used to take back those permissions.

These commands help in controlling access and maintaining security in a database system.

2. How do you manage privileges using these commands?

Ans :

Using GRANT:

Grants specific privileges to a user.

GRANT SELECT, INSERT ON Students TO user_name;

→ Allows the user to view (SELECT) and add (INSERT) records in the Students table.

You can also use:

GRANT ALL ON Students TO user_name;

10. Transaction Control Language(TCL)

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

Ans: COMMIT:
This command is used to save all the changes made during the current transaction permanently to the database.

Example:

INSERT INTO Students VALUES (101, 'Amit', 20);

COMMIT;

→ The new student record is permanently saved.

- ROLLBACK:
  This command is used to undo changes made in the current transaction and restore the database to its previous state.

2. Explain how transactions are managed in SQL databases.

Ans : A transaction is a sequence of one or more SQL operations that are treated as a single logical unit of work. Transactions follow the ACID properties:

- A – Atomicity: All operations succeed or none.
- C – Consistency: Database remains in a valid state.
- I – Isolation: Transactions do not interfere with each other.
- D – Durability: Once committed, changes are permanent.

Transaction Management Commands:

- BEGIN or START TRANSACTION: Starts a new transaction.
- COMMIT: Saves changes permanently.
- ROLLBACK: Cancels all changes made in the transaction.
- SAVEPOINT: Creates a checkpoint within a transaction to roll back to.
- ROLLBACK TO SAVEPOINT: Undoes part of the transaction up to the savepoint.

Example of a Transaction:

START TRANSACTION;

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

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

COMMIT; -- Both updates are saved together

If something goes wrong, use:

ROLLBACK;

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?

Ans : JOIN:

A JOIN in SQL is used to combine rows from two or more tables based on a related column between them, usually using primary and foreign keys.

Types of JOIN:

1.INNER JOIN

Description: Returns only matching rows from both tables.

Example : Records that have matching values in both tables.

2.LEFT JOIN

Description : Returns all rows from the left table, and matched rows from the right. Unmatched right rows will be NULL.

Example : All records from left + matched ones from right.

3.RIGHT JOIN

Description : Returns all rows from the right table, and matched rows from the left. Unmatched left rows will be NULL.

Example : All records from right + matched ones from left.

4.FULL OUTER JOIN

Description : Returns all rows when there is a match in one of the tables. If there's no match, the result is NULL from the missing side.

Example : All records from both tables.

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

Ans : Joins allow you to:

- Combine related information (e.g., student details with their marks).

- Avoid data duplication by normalizing your database (splitting data into separate related tables).
- Perform complex queries across multiple tables as if they were one.

Example Use Case:
If you store students in one table and their marks in another, a JOIN lets you list each student's name along with their marks.

12. SQL Group By

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

Ans : GROUP BY:

The GROUP BY clause in SQL is used to group rows that have the same values in specified columns. It is commonly used with aggregate functions to perform calculations on each group of rows.

Aggregate functions include:

- SUM() – total value
- AVG() – average
- COUNT() – number of items
- MAX() – highest value
- MIN() – lowest value

Syntax:

SELECT column, AGGREGATE_FUNCTION(column)

FROM table

GROUP BY column;

Example:

SELECT department, AVG(salary)

FROM Employees

GROUP BY department;

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

Ans : GROUP BY

Purpose : Groups rows for aggregate calculations

Used with : Aggregate functions like SUM, COUNT, etc.

Output : One row per group

Order Affected : No specific order unless used with ORDER BY

ORDER BY

Purpose : Sorts the result set

Used With : Any query, with or without aggregates

Output : Rows sorted, not grouped

Order Affected : Determines ascending/descending row order

13. SQL Stored Procedure

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

Ans : A stored procedure is a precompiled collection of SQL statements (like SELECT, INSERT, UPDATE, etc.) stored in the database and executed as a single unit.

Stored procedure:

Definition: Named and stored in the database

Reusability : Can be reused multiple times

Logic : Supports programming logic

Performance : Precompiled, thus faster on repeated use

Input Parameters : Can accept parameters (IN, OUT, INOUT)

Standard SQL query:

Definition : Written and executed directly

Reusability : Written each time you need it

Logic : Limited to single SQL operation

Performance : Compiled each time it runs

Input Parameters : Typically has hardcoded values

2. Explain the advantages of using stored procedures.

Ans : Advantages:

1. Improved Performance
   →Stored procedures are compiled once and stored, reducing execution time for repeated use.
2. Reusability
   →Procedures can be called multiple times from different applications or parts of code.
3. Security & Access Control
   →You can give users permission to execute a procedure without giving them direct access to the underlying tables.
4. Centralized Logic
   →Business logic can be maintained in one place inside the database, which is easier to manage and update.
5. Reduced Network Traffic
   →Instead of sending multiple SQL statements, just the procedure call is sent — which is lighter and faster.
6. Supports Parameters
   →Procedures can take inputs and return outputs, making them dynamic and flexible.

14. SQL View

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

Ans : A view is a virtual table that is based on the result of a SELECT query. It does not store data physically but displays data from one or more tables.

View

Definition : Virtual table based on a SELECT query

Data Storage : Does not store data physically

Modification : Can be updatable

Dependency : Depend on underlying tables

Purpose : Used for simplified or secure data access

Table

Definition : Physical table that stores data

Data Storage : Stores data in the database

Modification : Fully moddifiable

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

Ans : Advantages of Views:

1. Simplify Complex Queries
   →Views can hide complex joins and calculations, making queries easier to write and understand.

2. Enhance Security
   →Views can restrict user access to specific columns or rows, protecting sensitive data.
3. Data Abstraction
   →Users interact with views without knowing the underlying table structure.
4. Reusability
   →A view can be reused in multiple queries or applications.
5. Logical Data Independence
   →If the underlying table changes, the view can remain the same, reducing impact on applications.

6.Custom Representation

→Views can present data in a format suitable for reporting or analysis (e.g., renaming columns, calculated fields).

15. SQL Triggers

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

Ans : Trigger in SQL:

A trigger is a set of instructions in SQL that automatically executes in response to specific events (like INSERT, UPDATE, or DELETE) on a table or view.

Types of Triggers:

1.Before Trigger

2.After Trigger

3.INSTEAD Of Trigger

Triggers Used:

- Automatically update audit/log tables.
- Validate or modify data before insert/update.
- Prevent invalid transactions.
- Enforce business rules automatically.

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

Ans :

INSERT Trigger:

Event : On new row insertion

Purpose : Executes when a new record is inserted into a table.

UPDATE Trigger :

Event : On data modification

Purpose : Executes when an existing record is modified.

DELETE Trigger :

Event : On record deletion

Purpose : Executes when a record is removed from the table.

16. Introduction to PL/SQL

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

Ans : PL/SQL (Procedural Language/Structured Query Language)

PL/SQL is Oracle's procedural extension to SQL. While SQL is used for querying and manipulating data, PL/SQL adds programming features like:

- Variables
- Loops
- Conditions (IF, CASE)
- Procedures & Functions
- Exception Handling

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

Ans : Benefits of Using PL/SQL (Simple Bullet Form)

- Modular Programming

  ➤ Write once, use many times — PL/SQL supports procedures, functions, and packages to organize code efficiently.

- Better Performance

  ➤ Sends a block of code to the server at once instead of multiple individual SQL statements — saves time and resources.

- Strong Error Handling

  ➤ Handles errors smoothly using EXCEPTION blocks — helps catch and manage run-time problems effectively.

- Built-in SQL Support

  ➤ Allows SQL to be written directly within PL/SQL code — no need for switching between languages.

- Easy Maintenance

  ➤ Code is stored in the database and organized — easier to update, debug, and manage over time.

- Enhanced Security

  ➤ You can control access to specific code blocks using privileges — ensures only authorized users can run or modify them.

17.PL/SQL Control Structures

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

Ans : Control structures in PL/SQL are used to control the flow of execution in a program. They allow you to make decisions, repeat actions, and handle conditions.

a) IF-THEN Structure

Used for decision-making.

Syntax:

```
IF condition THEN
   -- statements to execute if condition is true
END IF;
```

Example:

```
IF salary > 50000 THEN
   bonus := 1000;
END IF;
```

b) LOOP Structure

Used for repeating a set of statements.

Syntax:

```
LOOP
   -- statements
```

```
  EXIT WHEN condition;
END LOOP;
```

Example:

```
i := 1;
LOOP
  DBMS_OUTPUT.PUT_LINE(i);
  i := i + 1;
  EXIT WHEN i > 5;
END LOOP;
```

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

Ans : Control structures allow you to:

- Add logic (e.g., perform actions only when certain conditions are true).
- Repeat operations (e.g., loop through rows or perform calculations).
- Make decisions dynamically, based on variables or data.

This helps in:

- Building complex business logic inside procedures and functions.

- Making queries dynamic and flexible.
- Improving code readability and maintainability.

18. SQL Cursors

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

Ans : Cursor in PL/SQL:

A cursor is a pointer or a memory area used to fetch and process rows returned by a SQL query, one row at a time.

PL/SQL automatically or manually uses cursors to handle query results.

Type of Cursors :

Implicit Cursor

Created By : Automatically by PL/SQL

When Used : For single-row SQL statement

Control : Limited control

Example : SELECT salary INTO var FROM employees WHERE id=101;

Explicit Cursor

Created By : Manually by the programmer

When Used : For multi-row SELECT queries

Control : Full control

Example : Declare and fetch rows from SELECT *FROM employees;

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

Ans : You would use an explicit cursor when:

- Your query returns multiple rows, and you need to process each row one by one.
- You need more control over row fetching (e.g., loop through rows, check conditions).
- You want to track cursor status or handle specific logic inside loops.

Example Use Case:

If you want to give a bonus to every employee with a salary below 30000:

```
CURSOR emp_cursor IS
SELECT emp_id, salary FROM employees WHERE salary < 30000;

BEGIN
  FOR emp IN emp_cursor LOOP
    UPDATE employees
```

```
    SET bonus = 1000
    WHERE emp_id = emp.emp_id;
  END LOOP;
END;
```

19. RollBack and Commit Savepoint

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

Ans : SAVEPOINT:

A SAVEPOINT is a marker or label that allows you to set a point within a transaction to which you can later roll back if needed, without affecting the entire transaction.

It helps manage large or complex transactions by breaking them into smaller parts.

COMMIT:

- Saves all changes made in the current transaction permanently.
- Once committed, you cannot roll back to any previous savepoints.

ROLLBACK TO SAVEPOINT:

- Reverts the changes only to the specified savepoint, not the entire transaction.
- Useful when a part of the transaction fails but you don't want to cancel the entire process.

Example:

```
BEGIN;

UPDATE accounts SET balance = balance - 1000 WHERE id = 1;

SAVEPOINT sp1;


UPDATE accounts SET balance = balance + 1000 WHERE id = 2;

-- Oops! Something went wrong here


ROLLBACK TO sp1; -- Undo second update only

COMMIT; -- Save the first update
```

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

Ans : You use SAVEPOINTs when:

- You have a long or complex transaction and want to safely handle parts of it.

- You want to partially undo changes without canceling the whole transaction.
- You're performing multiple dependent updates, and you need to recover from a failure in just one part.
- You're testing or validating data in steps and want the option to go back to a specific step if something fails.