Module-5) Se - Introduction To Dbms (Theory)

Introduction to SQL

Theory Questions:

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

Ans: SQL (Structured Query Language) is a standard language used to store, retrieve, manage, and manipulate data in a database.

- It is essential because it provides a consistent way to:
- Create and modify database structures (tables, views).
- Insert, update, delete, and retrieve data.
- Control access and security of the database.
- Ensure data consistency and integrity.

2. Explain the difference between DBMS and RDBMS.

Ans:

Feature	DBMS (Database Management System)	RDBMS (Relational DBMS)
Data storage	Stores data as files or hierarchical/network forms	Stores data in tables (rows & columns)
Relationships	Does not enforce relationships	Supports relations using foreign keys
Data integrity	Limited	Enforces integrity with constraints
Examples	File systems, XML DB	MySQL, Oracle, PostgreSQL, SQL Server

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

Ans:

- Defining Data: Using DDL (Data Definition Language) commands (CREATE, ALTER, DROP) to design tables and schemas.
- Manipulating Data: Using DML (Data Manipulation Language) commands (INSERT, UPDATE, DELETE) to modify records.
- Querying Data: Using DQL (SELECT) to fetch information as per conditions.
- Controlling Access: Using DCL (GRANT, REVOKE) to manage permissions.
- Ensuring Transactions: Using TCL (COMMIT, ROLLBACK, SAVEPOINT) to maintain consistency and recover from errors.

4. What are the key features of SQL?

Ans:

- Data Definition – Ability to create, modify, and delete database structures.
- Data Manipulation – Insert, update, delete, and retrieve records.
- Data Control – Define access permissions and security.
- Transaction Control – Commit, rollback, and savepoints ensure reliability.
- Standardization – SQL is standardized (ANSI/ISO), so it works across most RDBMS.
- Declarative Nature – Users specify *what* they want, not *how* to get it.
- Support for Joins & Relationships – Easily combine data from multiple tables.

2. SQL Syntax

1. What are the basic components of SQL syntax?

Ans: The SQL syntax is made up of different keywords and constructs that allow users to interact with the database. The main components are:

1. **Keywords/Commands** → Reserved words that define the operation to be performed (e.g., SELECT, INSERT, UPDATE, DELETE, CREATE).
2. **Identifiers** → Names of database objects like tables, columns, views, indexes (e.g., students, course_id).
3. **Clauses** → Provide additional instructions or conditions to the SQL command (e.g., WHERE, GROUP BY, ORDER BY).
4. **Expressions** → Combinations of values, operators, and functions used to produce a result (e.g., salary * 0.1, age > 18).
5. **Predicates** → Conditions that evaluate to TRUE, FALSE, or UNKNOWN (e.g., WHERE age >= 20).
6. **Statements** → A complete SQL instruction ending with a semicolon (e.g., SELECT * FROM students;).

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

Ans:

```
SELECT column_list | *
FROM table_name
[WHERE condition]
[GROUP BY column_list]
[HAVING condition]
[ORDER BY column_list [ASC|DESC]];
```

- **SELECT** → Specifies the columns to be retrieved (or * for all).
- **FROM** → Identifies the table(s) from which to fetch the data.
- **WHERE** → Filters rows based on a condition.
- **GROUP BY** → Groups rows with the same values into summary rows.
- **HAVING** → Applies conditions to groups (used with GROUP BY).
- **ORDER BY** → Sorts the result in ascending (ASC) or descending (DESC) order.

3. Explain the role of clauses in SQL statements.

Ans: Clauses in SQL are special keywords that refine or extend the functionality of SQL commands. They control what data is selected, how it is grouped, and how it is ordered.

- **WHERE Clause: Filters rows before grouping. Example:**

  SELECT * FROM students WHERE age > 18;

- **GROUP BY Clause: Groups data based on one or more columns. Example:**
  - **SELECT course_id, COUNT(*)**
  - **FROM students**
  - **GROUP BY course_id;**
- **HAVING Clause: Applies conditions to groups (used after GROUP BY). Example:**
  - **SELECT course_id, COUNT(*)**
  - **FROM students**
  - **GROUP BY course_id**
  - **HAVING COUNT(*) > 5;**
- **ORDER BY Clause: Sorts the final result. Example:**
  - **SELECT student_name, age**
  - **FROM students**

    ORDER BY age DESC;

3. SQL Constraints

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

Answer:
Constraints in SQL are rules applied to the columns of a table to restrict the type of data that can be stored in them. They help maintain accuracy, reliability, and consistency of the data in a database.

The different types of constraints are:

1. NOT NULL Constraint
    - Ensures that a column cannot have a NULL value.
    - Example: name VARCHAR(50) NOT NULL → name column must always have a value.
2. UNIQUE Constraint
    - Ensures that all the values in a column are different.
    - Example: email VARCHAR(100) UNIQUE → no two rows can have the same email address.
3. PRIMARY KEY Constraint
    - A combination of NOT NULL and UNIQUE.
    - It uniquely identifies each row in a table.
    - A table can have only one primary key, which may consist of single or multiple columns (composite key).
4. FOREIGN KEY Constraint
    - Establishes a relationship between two tables.
    - It ensures that the value in one table (child table) must match a value in another table (parent table).
    - Maintains referential integrity.
5. CHECK Constraint
    - Ensures that the values in a column meet a specific condition.
    - Example: age INT CHECK(age >= 18) → only values greater than or equal to 18 are allowed.
6. DEFAULT Constraint
    - Provides a default value for a column if no value is specified by the user.
    - Example: status VARCHAR(10) DEFAULT 'Active'.

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

Answer:
The differences between PRIMARY KEY and FOREIGN KEY are:

Feature	PRIMARY KEY	FOREIGN KEY
Purpose	Uniquely identifies each record in a table.	Establishes a relationship between two tables.
Uniqueness & Nulls	Must be unique and NOT NULL.	Can contain duplicate values and NULLs.
Number in a Table	Only one primary key per table.	A table can have multiple foreign keys.
Location	Defined in the same table.	Defined in child table, referencing parent table.
Integrity Maintained	Entity Integrity.	Referential Integrity.

Example:

- PRIMARY KEY → student_id in Students table.
- FOREIGN KEY → student_id in Marks table (refers to Students table).

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

Answer:

- NOT NULL Constraint
    - Ensures that a column must always have a value.
    - It prevents inserting records with missing or undefined values in that column.
    - Example: In a users table, username VARCHAR(50) NOT NULL ensures every user must have a username.
- UNIQUE Constraint
    - Ensures that all values in a column are distinct.
    - It prevents duplicate entries in a column but allows NULL (except when combined with NOT NULL).
    - Example: In an employees table, email VARCHAR(100) UNIQUE ensures no two employees have the same email.

Role Together:

- When used together, NOT NULL + UNIQUE ensures that a column must have a value and that value must be different from all others.
- Example: In a login system, username VARCHAR(50) NOT NULL UNIQUE ensures every username is mandatory and unique.

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

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

Answer:
SQL Data Definition Language (DDL) is a category of SQL commands used to define, modify, and manage the structure of database objects such as tables, views, indexes, and schemas.

- DDL commands deal with schema definition rather than data manipulation.
- Common DDL commands are:
    - CREATE → creates new database objects.
    - ALTER → modifies existing objects.
    - DROP → deletes database objects.
    - TRUNCATE → removes all records from a table but keeps its structure.
    - RENAME → renames database objects.

Thus, DDL is mainly used to design and define the structure of a database.

2. Explain the CREATE command and its syntax.

Answer:
The CREATE command in SQL is used to create new database objects, such as a database, table, or view.

- The most common use is to create a table where data will be stored.

Syntax of CREATE TABLE:

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

Explanation:

- table_name → name of the new table.
- column1, column2 ... → names of the table's columns.

- datatype → defines the type of data (e.g., INT, VARCHAR, DATE).
- [constraint] → optional rules applied to columns (e.g., NOT NULL, UNIQUE).

Example:

```
CREATE TABLE Students (
  student_id INT PRIMARY KEY,
  name VARCHAR(50) NOT NULL,
  age INT CHECK(age >= 18),
  email VARCHAR(100) UNIQUE
);
```

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

Answer:
Specifying data types and constraints during table creation ensures data integrity, accuracy, and efficiency in the database.

- Purpose of Data Types:
    1. Define the kind of values a column can store (e.g., numbers, text, date).
    2. Optimize storage space (e.g., INT uses less space than VARCHAR).
    3. Improve performance by allowing the database to process queries more efficiently.
    - Example: age INT ensures only numeric values are stored in the age column.
- Purpose of Constraints:
    1. Enforce rules to maintain valid and consistent data.
    2. Prevent invalid entries (e.g., NULL values, duplicates, out-of-range values).
    3. Maintain relationships between tables (using PRIMARY KEY, FOREIGN KEY).
    - Example: email VARCHAR(100) UNIQUE ensures no duplicate email addresses are entered.

5. ALTER Command

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

Answer:
The ALTER command in SQL is a DDL command used to modify the structure of an existing table without affecting the stored data. It is commonly used to:

- Add new columns,
- Modify existing columns,
- Drop (remove) columns,
- Add or remove constraints.

This makes ALTER useful when changes are required in a table design after its creation.

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

Answer:

- Add a Column:
- ALTER TABLE Students ADD phone VARCHAR(15);

→ Adds a new column phone to the Students table.

- Modify a Column:
- ALTER TABLE Students MODIFY name VARCHAR(100);

→ Changes the size of the name column to 100 characters.

- Drop a Column:
- ALTER TABLE Students DROP COLUMN age;

→ Removes the age column from the Students table.

6. DROP Command

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

Answer:
The DROP command in SQL is a DDL command used to delete database objects permanently such as tables, views, or databases.

- When a table is dropped, its structure along with all the stored data, indexes, and constraints are permanently removed.
- Syntax:
- DROP TABLE table_name;

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

Answer:
Dropping a table has the following implications:

1. The table structure and all its data are permanently deleted.
2. All constraints (PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK) defined on the table are lost.
3. Any relationships with other tables (via FOREIGN KEY) will be broken.
4. The action cannot be undone unless a backup exists.

Thus, the DROP command should be used with caution as it results in permanent loss of data and schema.

7. Data Manipulation Language (DML)

Q1. Define the INSERT, UPDATE, and DELETE commands in SQL.

Answer:

- INSERT: Adds new rows of data into a table.
- INSERT INTO Students (id, name, age) VALUES (1, 'Amit', 20);
- UPDATE: Modifies existing data in a table.
- UPDATE Students SET age = 21 WHERE id = 1;
- DELETE: Removes rows of data from a table.
- DELETE FROM Students WHERE id = 1;

Q2. What is the importance of the WHERE clause in UPDATE and DELETE operations?

Answer:
The WHERE clause specifies the condition for selecting rows.

- Without WHERE → all rows get updated or deleted.
- With WHERE → only specific rows are affected.

Example:

- DELETE FROM Students; → deletes all rows.
- DELETE FROM Students WHERE id = 1; → deletes only student with id = 1.

8. Data Query Language (DQL)

Q1. What is the SELECT statement, and how is it used to query data?

Answer:
The SELECT statement is used to retrieve data from one or more tables.
Syntax:

SELECT column1, column2 FROM table_name;

Example:

SELECT name, age FROM Students;

This retrieves name and age of all students.

Q2. Explain the use of the ORDER BY and WHERE clauses in SQL queries.

Answer:

- WHERE Clause: Filters rows based on a condition.
- SELECT * FROM Students WHERE age > 18;
- ORDER BY Clause: Sorts the result set in ascending (ASC) or descending (DESC) order.

SELECT * FROM Students ORDER BY age DESC;

9. Data Control Language (DCL)

Q1. What is the purpose of GRANT and REVOKE in SQL?

Answer:

- **GRANT:** Provides specific privileges to users (e.g., SELECT, INSERT).
- **REVOKE:** Removes previously granted privileges.

Q2. How do you manage privileges using these commands?

Answer:
Privileges are managed by assigning or removing permissions.
Example:

GRANT SELECT, INSERT ON Students TO user1;

REVOKE INSERT ON Students FROM user1;

10. Transaction Control Language (TCL)

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

Answer:

- COMMIT and ROLLBACK are Transaction Control Language (TCL) commands in SQL.
- They are used to manage transactions, ensuring data integrity and consistency.
- COMMIT:
    - Saves all the changes made by a transaction permanently in the database.
    - Once committed, changes cannot be undone.
- ROLLBACK:
    - Cancels all the changes made by a transaction before it is committed.
    - Restores the database to the state before the transaction began.

2. Explain how transactions are managed in SQL databases.

Answer:

- A transaction is a group of one or more SQL operations executed as a single logical unit of work.
- Transactions follow the ACID properties:
    - Atomicity → all or nothing execution.
    - Consistency → maintains database rules.
    - Isolation → transactions run independently.
    - Durability → once committed, changes persist.

Transaction Flow:

1. Start a transaction (implicitly or with BEGIN).
2. Perform SQL operations (INSERT, UPDATE, DELETE).
3. Use COMMIT to save changes OR ROLLBACK to undo.

11. SQL Joins

Q1. Explain the concept of JOIN in SQL. What is the difference between INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL OUTER JOIN?

Answer:
A JOIN in SQL is used to combine rows from two or more tables based on a related column (usually a Primary Key–Foreign Key relationship).

Types of Joins:

Type of Join	Description	Example Result
INNER JOIN	Returns only rows that have matching values in both tables.	Students who have marks recorded.
LEFT JOIN	Returns all rows from the left table and matching rows from the right. Non-matching rows from the right appear as NULL.	All students, with marks if available.
RIGHT JOIN	Returns all rows from the right table and matching rows from the left. Non-matching rows from the left appear as NULL.	All marks, even if no matching student record.
FULL OUTER JOIN	Returns all rows from both tables. Non-matching rows are filled with NULL values.	All students and all marks, matched where possible.

Q2. How are joins used to combine data from multiple tables?

Answer:

- Joins link tables through related columns.
- They allow combining data into a single result set.

Example:

```
SELECT Students.name, Marks.score
FROM Students
INNER JOIN Marks ON Students.id = Marks.student_id;
```

12. SQL Group By

Q1. What is the GROUP BY clause in SQL? How is it used with aggregate functions?

Answer:

- The GROUP BY clause groups rows having the same values into summary rows.
- It is often used with aggregate functions like COUNT(), SUM(), AVG(), MIN(), and MAX().

Example:

SELECT department, AVG(salary)

FROM Employees

GROUP BY department;

Gives average salary for each department.

Q2. Explain the difference between GROUP BY and ORDER BY.

Feature	GROUP BY	ORDER BY
Purpose	Groups rows based on column values.	Sorts rows in ascending/descending order.
Use with	Aggregate functions (SUM, COUNT, AVG).	Any columns (numeric, text, etc.).
Example	GROUP BY department → groups employees by department.	ORDER BY salary DESC → sorts employees by salary.

13. SQL Stored Procedure

Q1. What is a stored procedure in SQL, and how does it differ from a standard SQL query?

Answer:

- A stored procedure is a precompiled collection of one or more SQL statements stored in the database and executed as a single unit.
- Unlike a standard SQL query (written and executed each time), a stored procedure is saved in the database and can be called repeatedly.

Example:

```
CREATE PROCEDURE GetStudents()

AS

SELECT * FROM Students;
```

Q2. Explain the advantages of using stored procedures.

- Reusability: Can be called multiple times.
- Performance: Precompiled, so execution is faster.
- Security: Access can be restricted via GRANT/REVOKE.
- Maintainability: Business logic stored in one place, easy to update.
- Reduces Network Traffic: Multiple SQL statements executed in one call.

14. SQL View

Q1. What is a view in SQL, and how is it different from a table?

Answer:

- A view is a virtual table based on the result of a SQL query.
- It does not store data physically; instead, it displays data stored in one or more tables.

Difference from a table:

- Table → physically stores data.
- View → logical representation of data (acts like a saved query).

Example:

CREATE VIEW StudentMarks AS

SELECT Students.name, Marks.score

FROM Students INNER JOIN Marks ON Students.id = Marks.student_id;

Q2. Explain the advantages of using views in SQL databases.

- Data Security: Users can be given access to views instead of full tables.
- Simplicity: Simplifies complex queries by saving them as views.
- Consistency: Provides a consistent way to access data.
- Reusability: Frequently used queries can be stored as views.
- Logical Independence: Changes in table structure may not affect the view.

15. SQL Triggers

Q1. What is a trigger in SQL? Describe its types and when they are used.

Answer:
A trigger in SQL is a stored program that is automatically executed (fired) in response to certain events on a table or view.

- Triggers help enforce business rules, maintain audit logs, and ensure data integrity.

Types of Triggers:

1. BEFORE Trigger – Executes before an INSERT, UPDATE, or DELETE operation.
   Use: Validate data before inserting or updating.
2. AFTER Trigger – Executes after an INSERT, UPDATE, or DELETE operation.
   Use: Maintain logs, update related tables.
3. INSTEAD OF Trigger – Executes in place of an INSERT, UPDATE, or DELETE (mainly for views).
   Use: Modify complex views where direct DML is not possible.

Q2. Explain the difference between INSERT, UPDATE, and DELETE triggers.

Trigger Type	Fired When	Example Use Case
INSERT Trigger	When a new row is inserted.	Automatically add entry into audit log.
UPDATE Trigger	When an existing row is modified.	Track old vs. new values for auditing.
DELETE Trigger	When a row is deleted.	Store deleted data in an archive table.

16. Introduction to PL/SQL

Q1. What is PL/SQL, and how does it extend SQL's capabilities?

Answer:

- PL/SQL (Procedural Language/SQL) is Oracle’s procedural extension to SQL.
- It adds programming features (variables, loops, conditions, procedures, functions) to SQL.
- While SQL can only query/manipulate data, PL/SQL allows writing full programs with control flow.

Q2. List and explain the benefits of using PL/SQL.

- Improved Performance → Multiple SQL statements can be executed in a block.
- Modularity → Supports procedures, functions, and packages.
- Error Handling → Provides exception handling features.
- Portability → PL/SQL programs run on any Oracle database.
- Security → Business logic can be stored in the database and protected from users.

17. PL/SQL Control Structures

Q1. What are control structures in PL/SQL? Explain the IF-THEN and LOOP control structures.

Answer:
Control structures allow conditional execution and repetition in PL/SQL.

- IF-THEN (Decision Making): Executes a block of code if a condition is true.

```
IF salary > 50000 THEN
bonus := 5000;
END IF;
```

LOOP (Iteration): Executes a block repeatedly until explicitly exited.

```
LOOP
counter := counter + 1;
EXIT WHEN counter > 10;
END LOOP;
```

Q2. How do control structures in PL/SQL help in writing complex queries?

- They allow decision-making (IF-THEN-ELSE).
- They support iteration (LOOP, WHILE, FOR).
- They help combine SQL with procedural logic, enabling advanced operations like validations, calculations, and automation.

18. SQL Cursors

Q1. What is a cursor in PL/SQL? Explain the difference between implicit and explicit cursors.

Answer:
A cursor is a pointer that allows row-by-row processing of query results.

- Implicit Cursor:
    - Created automatically by Oracle for single-row queries (SELECT INTO).
    - Simple and does not require declaration.
- Explicit Cursor:
    - Declared by the programmer to handle multi-row queries.
    - Requires OPEN, FETCH, and CLOSE statements.

Q2. When would you use an explicit cursor over an implicit one?

- Use explicit cursor when:
    1. Query returns multiple rows.
    2. You need to process each row individually.
    3. More control is required over fetching and iteration.

19. Rollback and Commit Savepoint

Q1. Explain the concept of SAVEPOINT in transaction management. How do ROLLBACK and COMMIT interact with savepoints?

Answer:

- A SAVEPOINT is a marker within a transaction that allows partial rollback.
- COMMIT makes all changes permanent (ignores savepoints).
- ROLLBACK TO SAVEPOINT undoes changes only up to that savepoint.

Example:

SAVEPOINT A;

UPDATE Accounts SET balance = balance - 100;

SAVEPOINT B;

UPDATE Accounts SET balance = balance - 200;

ROLLBACK TO B; -- cancels second update only

Q2. When is it useful to use savepoints in a database transaction?

- When a transaction has multiple steps and only part of it needs to be undone.
- Useful in long transactions where errors may occur at intermediate steps.
- Helps in fine-grained control of rollback without canceling the entire transaction.