Module 4 – Introduction to DBMS

1. Introduction to SQL

   Theory Questions:

   1) What is SQL, and why is it essential in database management?
      - SQL, or Structured Query Language, is a standard programming language specifically designed for managing and manipulating relational databases. It allows users to interact with databases by performing tasks such as creating, reading, updating, and deleting data.
      - Data Organization: SQL helps organize data into structured formats using tables, making it easier to store and retrieve information
      - Query Efficiency: It provides powerful querying capabilities to fetch specific data or perform complex analyses quickly.
      - Standardization: SQL is widely used and standardized across most relational database management systems (RDBMS) like MySQL, PostgreSQL, Oracle, and Microsoft SQL Server, ensuring compatibility and consistency.
      - Data Integrity and Security: It enforces rules like constraints and permissions to maintain data accuracy and restrict unauthorized access.
      - Scalability: SQL supports large datasets, making it indispensable for businesses that deal with growing volumes of data.
   2) Explain the difference between DBMS and RDBMS.
      - Structure of Data
        - DBMS: Data is stored as files, and the relationships between the data elements aren't inherently defined.
        - RDBMS: Data is stored in a tabular form (rows and columns), and relationships between tables are established using keys (primary and foreign).
      - Data Integrity
        - DBMS: It lacks robust mechanisms for maintaining data integrity.
        - RDBMS: It enforces data integrity using constraints like Primary Key, Foreign Key, Unique, and Not Null.
      - Support for Relationships
        - DBMS: Does not inherently support relationships between data, as it's more file-based.
        - RDBMS: Explicitly supports relationships between data by implementing relational models.

- Efficiency with Large Datasets
    - DBMS: Suitable for smaller datasets due to limited scalability.
    - RDBMS: Designed to handle large-scale datasets efficiently.
- ACID Compliance
    - DBMS: May not guarantee ACID (Atomicity, Consistency, Isolation, Durability) properties, depending on the system.
    - RDBMS: Strictly adheres to ACID properties, ensuring transactional integrity.

3) Describe the role of SQL in managing relational databases.
    - Data Definition
        - SQL allows users to define the structure of their data within relational databases. This is done using commands like create, alter and drop to set up tables, columns, and relationships.
    - Data Manipulation
        - Insert Data: insert into is used to add new records
        - Retrieve Data: Select queries help fetch specific data based on conditions.
        - Update Data: Update modifies existing records.
        - Delete Data: Delete removes unwanted records.
    - Querying and Reporting
        - SQL's querying capabilities allow users to retrieve data in a structured and meaningful way. Complex queries can analyze, filter, and sort large datasets for reporting and decision-making.
    - Data Relationships
        - SQL establishes and manages relationships between tables using keys. For instance, Primary Keys uniquely identify records, and Foreign Keys create links between tables.
    - Data Integrity and Security
        - SQL enforces constraints like Not null, Unique , Check, and Foreign key to maintain data accuracy and relationships. It also includes security features like user authentication and permissions to restrict unauthorized access.
    - Transaction Management
        - SQL ensures transactional integrity with commands like Commit and Rollback, supporting ACID properties.
    - Performance Optimization
        - SQL provides tools like Index for speeding up queries and optimizing database performance.
4) What are the key features of SQL?

- Data Definition Language (DDL): SQL allows you to define and modify the structure of a database, including creating, altering, and deleting tables and schemas.
- Data Manipulation Language (DML): You can insert, update, delete, and retrieve data using SQL commands like Insert, Update, Delete, and Select.
- Data Control Language (DCL): It enables controlling access to data through permissions and roles, using commands like Grant and Revoke.
- Querying Capabilities: SQL's Select statement allows complex data queries with filters, sorting, grouping, and aggregation.
- Relational Database Management: SQL is designed for relational databases, where data is stored in tables and relationships between tables are defined.
- Portability: SQL is supported by almost all major database systems (e.g., MySQL, PostgreSQL, Microsoft SQL Server), making it highly portable across platforms.
- Joins and Subqueries: SQL allows combining data from multiple tables through various join types (INNER, OUTER, LEFT, RIGHT) and using subqueries for complex operations.
- Built-in Functions: It offers a variety of built-in functions for mathematical calculations, string manipulation, date handling, and more.
- Indexing: SQL enables indexing to improve database performance by speeding up data retrieval.

2. SQL Syntax

Theory Questions:

1) What are the basic components of SQL syntax?
    - The basic components of SQL syntax include keywords, clauses, expressions, predicates, and queries, which are used to interact with and manipulate data in relational database.
    - Keywords: These are reserved words that have specific meanings in SQL,like SELECT, FROM, WHERE, INSERT, UPDATE, DELETE, CREATE, ALTER, and DROP.
    - Clauses: These are groups of words that are used to specify conditions or actions within a query, such as WHERE (for filtering data), ORDER BY (for sorting), and GROUP BY (for grouping data).

- Expressions: These are combinations of values, operators, and functions that evaluate to a single value, used to perform calculations or comparisons.
- Queries: These are the SQL statements that instruct the database on what data to retrieve or manipulate, built by combining keywords, clauses, expressions, and predicates.
- Data Types: SQL uses various data types to represent different kinds of data, such as INT for integers, VARCHAR for text, and DATE for dates.
- Tables: Tables are the fundamental building blocks of a database, consisting of rows and columns that store data.
- Databases: Databases are collections of tables and other objects, organized to store and manage data.
- Indexes: Indexes are used to speed up data retrieval by creating a lookup table for specific columns.

2) 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];
3) Explain the role of clauses in SQL statements.
   - SELECT Clause:
     - Specifies the columns to retrieve from the databas
     - The main clause for fetching data, defining what you want to see in the result set.
   - FROM Clause:
     - Indicates the table(s) from which the data will be retrieved.
     - Acts as the source of data for the query.
   - WHERE Clause:
     - Filters rows based on specific conditions.
     - Helps to narrow down the dataset to only relevant records.
   - GROUP BY Clause:
     - Groups rows with the same values in specified columns.
     - Used in conjunction with aggregate functions.
   - HAVING Clause:
     - Filters the grouped data after applying the Group by clause.
     - Similar to Where , but specifically used for group-level conditions.

3. SQL Constraints

Theory Questions

1) What are constraints in SQL? List and explain the different types of constraints.
    - Constraints are rules that limit the type of data that can be stored in a table, ensuring data integrity and accuracy.
    - The main types are: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK, DEFAULT, and INDEX.
    - NOT NULL: Ensures a column cannot contain NULL values.
    - UNIQUE: Guarantees that all values in a column are distinct, preventing duplicates.
    - PRIMARY KEY: Uniquely identifies each row in a table and cannot contain NULL values.
    - FOREIGN KEY: Establishes a relationship between two tables by referencing the primary key of another table.
    - CHECK: Defines a condition that a column must meet, ensuring data validity.
    - DEFAULT: Specifies a default value for a column if no value is provided during insertion.
    - INDEX: Used to create indexes in the table, which helps in creating and retrieving data from the database very quickly.
2) How do PRIMARY KEY and FOREIGN KEY constraints differ?
    - A primary key uniquely identifies each row in a table, while a foreign key establishes a link between two tables, referencing the primary key of another table.
    - Primary Key:
        - Purpose: To uniquely identify each row within a table.
        - Uniqueness: Cannot contain duplicate values; each row must have a unique identifier.
        - Null Values: Does not allow null values.
        - Number: A table can only have one primary key.
    - Foreign Key:
        - Purpose: To establish a relationship between two tables by referencing the primary key of another table.
        - Uniqueness: Can contain duplicate values within the table, as it's designed to link to the primary key of another table.
        - Null Values: Can allow null values, depending on the relationship type.
        - Number: A table can have multiple foreign keys.
3) What is the role of NOT NULL and UNIQUE constraints?

- NOT NULL:
    - Purpose: Prevents a column from accepting NULL values, ensuring that every row in the table has a value for that column.
    - If you define a column as NOT NULL, you cannot insert a new record or update an existing one without providing a value for that column.
    - Use Cases: Ensuring that required information is always present, like a user's ID or email address.
        - Maintaining data integrity by preventing missing or unknown data.
- UNIQUE:
    - Purpose: Enforces that all values within a column are unique, meaning no two rows can have the same value in that column.
    - If you define a column as UNIQUE, you cannot insert a new record with a value that already exists in that column.
    - Use Cases: Ensuring the uniqueness of identifiers, like user IDs or product codes.
        - Preventing duplicate entries in a table.

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

Theory Questions:

1) Define the SQL Data Definition Language (DDL).
    - Data Definition Language (DDL) is a set of commands used to define and modify the structure of database objects like tables, indexes, and schemas.
    - It's responsible for creating, modifying, and deleting the **structure** of your database.
    - CREATE: This command is used to build the foundation of your database. You can create tables, views, indexes, and even users with this command.
    - ALTER: This command allows you to make changes to your existing database structures. You can modify tables, columns, indexes, and constraints.
    - DROP: This command is used to remove elements from your database. You can drop tables, views, indexes, users, or even the entire database itself.
2) Explain the CREATE command and its syntax.
    - The CREATE command in SQL is a Data Definition Language (DDL) command used to create new database objects like tables, indexes, views, or database.
    - Syntax: CREATE [OBJECT_TYPE] [OBJECT_NAME] [OPTIONS];

- CREATE: This keyword initiates the creation of a new database object.
- [object_type]: Specifies the type of object to be created, such as TABLE, DATABASE, INDEX, or VIEW.
- [object_name]: The user-defined name for the new object.
- OPTIONS: These are optional parameters that define the characteristics of the object. For example, you can specify column names, data types, constraints, and more.

3) What is the purpose of specifying data types and constraints during table creation?
    - Specifying data types and constraints during table creation ensures data integrity, accuracy, and consistency by defining the structure and rules for the data stored within the table.
    - Data Types: Data types define the kind of data a column can store.
    - Constraints: Constraints are rules that restrict the data that can be inserted or updated in a table, ensuring data accuracy and reliability.

5. ALTER Command

   Theory Questions:

1) What is the use of the ALTER command in SQL?
    - The purpose of the SQL ALTER TABLE statement is to modify the structure of an existing table, allowing you to add, delete, or change columns, constraints, and other structural elements.
2) How can you add, modify, and drop columns from a table using ALTER?
    - ALTER TABLE table_name ADD column_name data_type; for add new column.
    - ALTER TABLE table_name MODIFY column_name new_data_type; for modify data.
    - ALTER TABLE table_name DROP COLUMN column_name; for delete column.

6. DROP Command

   Theory Questions

1) What is the function of the DROP command in SQL?
    - The primary function of the DROP command is to delete database objects.
    - Tables: Deletes a table and all its data, structure, and associated constraints.
    - Databases: Completely removes a database and all its objects.
    - Functions: Deletes the definition of a function.
    - Procedures: Removes stored procedures.
2) What are the implications of dropping a table from a database?

- Dropping a table permanently deletes both the table structure and all its data, along with any associated indexes, triggers, and constraints, making the operation irreversible unless a backup is available.
- Permanent Data Loss: Dropping a table means the data stored within it is permanently deleted and cannot be recovered unless a backup exists.
- Table Structure Removal: The table definition, including column definitions, data types, and constraints, is also removed from the database schema.
- Index and Constraint Removal: All indexes and constraints associated with the dropped table are also removed.

7. Data Manipulation Language (DML)

   Theory Questions:

   1) Define the INSERT, UPDATE, and DELETE commands in SQL.
      - INSERT:
        - The INSERT statement is used to add new rows (records) into a table.
        - It allows you to specify the columns to insert data into and the values to insert.
      - UPDATE:
        - The UPDATE statement is used to modify existing data within a table.
        - You can specify the columns to update and the new values, along with a WHERE clause to target specific rows.
      - DELETE:
        - The DELETE statement is used to remove rows from a table.
        - You can use a WHERE clause to specify which rows to delete.
   2) What is the importance of the WHERE clause in UPDATE and DELETE operations?
      - The WHERE clause is crucial in UPDATE and DELETE operations because it specifies which rows should be modified or removed, preventing unintended changes to the entire table.
      - UPDATE: When updating data, the WHERE clause ensures that only the rows meeting the specified condition are modified, leaving other rows untouched.
      - DELETE: Similarly, in DELETE operations, the WHERE clause is essential for removing only the desired records.

8. Data Query Language (DQL)

   Theory Questions:

1) What is the SELECT statement, and how is it used to query data?
    - The SELECT statement in SQL is used to retrieve data from a database, allowing you to query and retrieve specific information from one or more tables, returning the results as a result set or table.
    - SELECT [column_list] FROM [table_name] [WHERE condition] [ORDER BY column_name [ASC|DESC]];
    - SELECT: This keyword initiates the query, indicating that you want to retrieve data.
    - FROM: Specifies the table from which the data will be retrieved.
    - ;: This semicolon indicates the end of the SQL statement.
2) Explain the use of the ORDER BY and WHERE clauses in SQL queries.
    - The WHERE clause filters rows based on a condition, while the ORDER BY clause sorts the resulting rows, either ascending or descending, based on one or more column.
    - WHERE Clause: The WHERE clause is used to filter records (rows) from a table based on a specified condition.
        - SELECT column1, column2 FROM table_name WHERE condition;
    - ORDER BY Clause: The ORDER BY clause sorts the result set of a SELECT statement in ascending or descending order based on one or more columns.
        - SELECT column1, column2 FROM table_name WHERE condition ORDER BY column_name [ASC | DESC];

9. Data Control Language (DCL)

Theory Questions:

1) What is the purpose of GRANT and REVOKE in SQL?
    - GRANT provides users with specific privileges (like SELECT, INSERT, UPDATE, DELETE) on database objects, while REVOKE removes those previously granted permissions, ensuring controlled and secure access to the database.
    - The GRANT statement is used to assign privileges to users, groups, or roles, allowing them to perform specific actions on database objects like tables, views, or stored procedures.
    - The REVOKE statement is used to remove privileges that were previously granted to users, groups, or roles.
    - You can revoke privileges from a single user, multiple users, or a role.
2) How do you manage privileges using these commands?

- To manage privileges in a database system like MySQL, you use the GRANT and REVOKE commands. GRANT assigns privileges to a user, while REVOKE removes them.
- GRANT Command: Grants specific privileges (like SELECT, INSERT, UPDATE, DELETE, or ALL) on a database or object (table, view, etc.) to a user.
    - GRANT <privileges> ON <database>.<object> TO '<user>'@'<host>'
- REVOKE Command: Removes privileges from a user.
    - REVOKE <privileges> ON <database>.<object> FROM '<user>'@'<host>'

10. Transaction Control Language (TCL)

Theory Questions:

1) What is the purpose of the COMMIT and ROLLBACK commands in SQL?
    - In SQL, COMMIT makes changes made during a transaction permanent, while ROLLBACK cancels those changes, reverting the database to its state before the transaction started.
    - COMMIT: The COMMIT command is used to save all the changes made during a transaction.
        - Once a COMMIT command is executed, the changes become permanent and visible to other users and sessions.
        - After a COMMIT, the transaction state becomes permanently visible to all users.
    - ROLLBACK: The ROLLBACK command is used to undo all changes made during the current transaction since the last COMMIT.
        - It reverts the database to its state before the transaction started.
        - If a transaction fails or errors occur, ROLLBACK can be used to undo the changes and prevent data corruption.
2) Explain how transactions are managed in SQL databases.
    - In SQL databases, transactions are managed using SQL commands like BEGIN TRANSACTION, COMMIT, and ROLLBACK to ensure data integrity by treating a sequence of operations as a single, indivisible unit of work.
    - BEGIN TRANSACTION: This command marks the beginning of a transaction, signaling that all subsequent SQL statements will be part of this transaction until a COMMIT or ROLLBACK statement is issued.
    - COMMIT: This statement finalizes a transaction, making all changes within it permanent. If the transaction was successful, the changes are saved; otherwise, they are discarded.

- ROLLBACK: This statement cancels a transaction, undoing all changes made within it. It's used when an error occurs or when the transaction violates some condition.
- SAVEPOINT: This allows you to set points within a transaction to which you can later roll back, useful for partially undoing changes within a transaction.

11. SQL Joins

Theory Questions:

1) Explain the concept of JOIN in SQL. What is the difference between INNER JOIN, LEFTJOIN, RIGHT JOIN, and FULL OUTER JOIN?
    - The main types of joins are INNER JOIN (returns only matching records), LEFT JOIN (returns all records from the left table and matching from the right), RIGHT JOIN (returns all records from the right table and matching from the left), and FULL OUTER JOIN (returns all records from both tables, including unmatched ones.
    - INNER JOIN: Returns only the rows where there's a match in both tables based on the join condition.
        - It's like finding the intersection of the two tables.
        - EX.: SELECT * FROM Customers INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
    - LEFT JOIN: Returns all rows from the left table, and the matching rows from the right table.
        - If there's no match in the right table, the right table columns will contain NULL values.
        - EX.: SELECT * FROM Customers LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
    - RIGHT JOIN: Returns all rows from the right table, and the matching rows from the left table.
        - If there's no match in the left table, the left table columns will contain NULL values.
        - EX.: SELECT * FROM Customers RIGHT JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
    - FULL JOIN: Returns all rows from both tables, including both matched and unmatched rows.
        - If there's no match in one of the tables, the corresponding columns will contain NULL value
        - EX.: SELECT * FROM Customers FULL OUTER JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
2) How are joins used to combine data from multiple tables?

- Joins establish logical relationships between tables, allowing you to access data from multiple tables simultaneously.
- They use common key values (columns) shared across different tables to link the data.
- The JOIN keyword, combined with ON, is used to specify the condition for joining the table.
- The ON clause specifies the condition for joining, often based on a primary key in one table and a foreign key in another.

12. SQL Group By

Theory Questions:

1) What is the GROUP BY clause in SQL? How is it used with aggregate functions?
    - The SQL GROUP BY clause groups rows with the same values in one or more columns, and it's commonly used with aggregate functions (like COUNT, SUM, AVG, MAX, MIN) to perform calculations on these groups.
    - It takes one or more column names as arguments.
    - It then groups rows where the values in those specified columns are identical.
    - Aggregate Functions:
        - COUNT(): Counts the number of rows in each group
        - SUM(): Calculates the sum of values in a specified column for each group.
        - AVG(): Calculates the average of values in a specified column for each group.
        - MAX(): Finds the maximum value in a specified column for each group.
        - MIN(): Finds the minimum value in a specified column for each group.
2) Explain the difference between GROUP BY and ORDER BY.
    - GROUP BY: To group rows that have the same values in one or more column.
        - Typically used with aggregate functions to calculate summary statistics for each groups.
    - ORDER BY: To sort the result set of a SQL query based on one or more columns.
        - Arranges the data in either ascending (default) or descending order.

13. SQL Stored Procedure

Theory Questions:

1) What is a stored procedure in SQL, and how does it differ from a standard SQL query?
    - A stored procedure is a pre-compiled set of SQL statements stored in the database, while a standard SQL query is a single, ad-hoc statement executed on demand.
    - Stored procedures offer reusability, performance benefits, and security advantages over standard queries.
2) Explain the advantages of using stored procedures.
    - **Advantages :**
        - Better Performance – The procedure calls are quick and efficient as stored procedures are compiled once and stored in executable form.Hence the response is quick. The executable code is automatically cached, hence lowers the memory requirements.
        - Higher Productivity – Stored Procedure helps to encapsulate the SQL logic and business logic due to which it provide reusability and modularity.
        - Ease of Use – To create a stored procedure, one can use any Java Integrated Development Environment (IDE). Then, they can be deployed on any tier of network architecture.
        - Scalability – Stored procedures increase scalability by isolating application processing on the server.
        - Maintainability – Maintaining a procedure on a server is much easier then maintaining copies on various client machines, this is because scripts are in one location.
        - Security – Access to the Oracle data can be restricted by allowing users to manipulate the data only through stored procedures that execute with their definer's privileges.

14. SQL View

Theory Questions:

1) What is a view in SQL, and how is it different from a table?
    - A view is a virtual table based on the result of a query, while a table is a physical structure that stores data.
    - Views don't store data themselves but dynamically retrieve data based on a pre-defined query each time they're accessed, whereas tables store data permanently.
2) Explain the advantages of using views in SQL databases.
    - Restrict database access
    - Represent data from different tables
    - Make complex query easy

- Calculate values of formula
- Represent different views of the same data
- Views can be created from other views

15. SQL Triggers

Theory Questions:

1) What is a trigger in SQL? Describe its types and when they are used.
    - A trigger is a special type of stored procedure that automatically executes in response to specific events (like INSERT, UPDATE, or DELETE) on a table or view.
    - DML Trigger:
        - Row-Level Triggers**:** Execute once for each row affected by the triggering event.
        - Statement-Level Triggers: Execute once for the entire statement, regardless of how many rows are affected.
    - Use Triggers:
        - Enforcing Business Rules
        - Maintaining Data Integrity
        - Auditing and Logging
        - Automating Tasks
2) Explain the difference between INSERT, UPDATE, and DELETE triggers.
    - INSERT, UPDATE, and DELETE triggers are database triggers that execute specific actions in response to INSERT, UPDATE, and DELETE operations on a table, respectively, allowing for data validation, modification, or other actions before or after these operations.
    - INSERT Trigger:
        - Fires when a new row is inserted into a table.
        - Can be used to validate data before insertion, update related tables, or perform other actions.
    - UPDATE Trigger:
        - Fires when an existing row in a table is modified.
        - Can be used to track changes, update related tables, or perform other actions based on the updated data.
    - DELETE Trigger:
        - Fires when a row is deleted from a table.
        - Can be used to archive deleted data, update related tables, or perform other actions.

16. Introduction to PL/SQL

Theory Questions:

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

- PL/SQL (Procedural Language/SQL) is a procedural language extension to SQL developed by Oracle, allowing developers to combine SQL's data manipulation with procedural programming constructs like loops, conditions, and error handling, enabling more complex and efficient database operations.

2) List and explain the benefits of using PL/SQL.
    - Benefits of Using PL/SQL
        - Improved Performance:
            - PL/SQL allows for the execution of multiple SQL statements in a single block, reducing the number of context switches between the SQL and PL/SQL engines. This leads to faster execution times.
        - Code Reusability:
            - You can create reusable code blocks (procedures and functions) that can be called multiple times throughout your applications. This modular approach simplifies maintenance and enhances productivity.
        - Enhanced Security:
            - PL/SQL provides a way to encapsulate business logic and restrict direct access to the underlying data. You can grant users permission to execute specific procedures without giving them access to the underlying tables.
        - Error Handling:
            - PL/SQL includes robust error handling capabilities through the use of exceptions. This allows developers to manage errors gracefully and maintain the integrity of the database.
        - Integration with SQL:
            - PL/SQL seamlessly integrates with SQL, allowing you to execute SQL statements directly within PL/SQL blocks. This makes it easy to manipulate data and perform complex queries.
        - Support for Complex Business Logic:
            - You can implement complex business rules and logic directly in the database, which can lead to more efficient applications and reduced data transfer between the application and the database.
        - Portability:
            - PL/SQL code is portable across different platforms that support Oracle databases. This means you can develop

applications that can run on various systems without significant changes.

- Built-in Functions and Packages:
  - PL/SQL comes with a rich set of built-in functions and packages that simplify common tasks, such as string manipulation, date handling, and mathematical calculations.
- Batch Processing:
  - PL/SQL supports batch processing, allowing you to process large volumes of data efficiently. This is particularly useful for data migration and transformation tasks.

17. PL/SQL Control Structures

Theory Questions:

1) What are control structures in PL/SQL? Explain the IF-THEN and LOOP control structures.
   - Control structures dictate the order of execution based on conditions or loops, enabling developers to create dynamic and flexible programs.
   - IF-THEN: Executes a block of code only if a condition is true.
     ```
     IF condition THEN
       -- Statements to execute if the condition is true
     END IF;
     ```
   - LOOP: Executes a block of code repeatedly until an EXIT statement is encounter.
     ```
     LOOP
       -- Statements to execute repeatedly
       EXIT WHEN condition; -- Optional EXIT statement
     END LOOP;
     ```
2) How do control structures in PL/SQL help in writing complex queries?
   - Control structures in PL/SQL, like IF-THEN-ELSE statements, CASE statements, and loops (FOR, WHILE), allow developers to create dynamic and complex queries by enabling conditional logic and iterative processing within a single SQL block, effectively tailoring the data retrieval based on specific criteria or situations, making the query logic more flexible and adaptable to different scenarios.
   - **IF-THEN-ELSE:** Use these statements to include or exclude data from a query based on specific conditions, allowing for fine-grained control over the results.
   - **CASE expressions:** Select different values or perform different actions based on conditions within a single query, simplifying complex logic.

18. SQL Cursors

Theory Questions:

1) What is a cursor in PL/SQL? Explain the difference between implicit and explicit cursors.
    - A cursor is a named control structure that allows you to retrieve and process data from a database one row at a time.
    - There are two types: implicit, automatically created by the database for single-row operations, and explicit, programmer-defined for multi-row processing and greater control.
    - Implicit Cursors:
        - Automatically creates an implicit cursor when you execute a SELECT INTO statement or a DML statement.
        - Implicit cursors are designed for single-row operations, meaning they are suitable for fetching a single row of data or performing DML operations on a single record.
        - The database handles the opening, fetching, and closing of implicit cursors automatically.
    - Explicit Cursors:
        - You explicitly declare and manage explicit cursors using the CURSOR keyword.
        - Explicit cursors are used to handle queries that return multiple rows, allowing you to process each row individually.
        - You must explicitly open, fetch, and close explicit cursors using OPEN, FETCH, and CLOSE statements.
2) When would you use an explicit cursor over an implicit one?
    - When you need fine-grained control over the cursor lifecycle, including opening, fetching, and closing, or when you need to reuse the cursor or its query within a PL/SQL block.

19. Rollback and Commit Save point

Theory Questions:

1) Explain the concept of SAVEPOINT in transaction management. How do ROLLBACK andCOMMIT interact with savepoints?
    - A savepoint acts as a marker within a transaction, allowing you to rollback to a specific point without aborting the entire transaction.
    - Savepoint:
        - A savepoint is a named entity that represents the state of data and schemas at a particular point within a unit of work.
        - t allows you to create markers within a transaction that you can rollback to, without preventing the transaction from being committed at a later point.

    - Savepoints are useful for implementing complex error recovery in database applications.
- Interact with ROLLBACK and COMMIT:
- SAVEPOINT:
    - Creates a named point within a transaction.
    - You can declare a savepoint using a command like SAVEPOINT savepoint_name.
- ROLLBACK TO SAVEPOINT:
    - Undoes any changes made to the database after the specified savepoint was created.
    - The savepoint remains valid and can be rolled back to again later, if needed.
- COMMIT:
    - Persists all changes made to the database since the beginning of the transaction.
    - Discards all savepoints created within the transaction.

2) When is it useful to use savepoints in a database transaction?
    - Savepoints are useful in database transactions, particularly for complex operations, as they allow you to rollback specific parts of a transaction without aborting the entire process, offering granular control over error recovery and partial commit.