Lab Exercise

A-1:

1. SQL (Structured Query Language) is a standard language used to create, manage, and manipulate databases. It is essential because it allows users to efficiently retrieve, insert, update, and delete data in databases.

2. **DBMS** stores data as files and supports only single-user access. **RDBMS** stores data in tables with relationships and supports multiple users with data integrity and constraints.

3. SQL is used to define data structures (DDL), manipulate data (DML), control access (DCL), and query data (DQL) in relational databases.

4.
    - Data definition and manipulation
    - Querying using SELECT
    - Data integrity and constraints
    - Transaction control
    - Security and access control

A-2:

1. Keywords (e.g., SELECT, FROM, WHERE)
   Identifiers (e.g., table and column names)
   Operators (e.g., =, >, <)
   Clauses (e.g., WHERE, ORDER BY)
   Expressions and Statements

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

3. Clauses define specific conditions and operations within SQL statements, such as filtering (WHERE), sorting (ORDER BY), grouping (GROUP BY), and joining tables (JOIN).

A-3:

1.
   - **NOT NULL** – Ensures a column cannot have NULL values.
   - **UNIQUE** – Ensures all values in a column are unique.
   - **PRIMARY KEY** – Uniquely identifies each record; combines NOT NULL and UNIQUE.
   - **FOREIGN KEY** – Links a column to the PRIMARY KEY of another table.
   - **CHECK** – Ensures values meet a condition.
   - **DEFAULT** – Assigns a default value if none is provided.

2.
   - **PRIMARY KEY** uniquely identifies records within a table.
   - **FOREIGN KEY** maintains a link between two tables by referencing the PRIMARY KEY in another table.

3.
   - **NOT NULL** ensures a column always has a value.
   - **UNIQUE** ensures all values in a column are different, avoiding duplicates.

A-4:

1. DDL is a subset of SQL used to define and manage database structures like tables, schemas, and indexes.

2. The CREATE command is used to create new database objects such as tables.
   Syntax:
   CREATE TABLE table_name (

   column1 datatype constraint,
   column2 datatype constraint,
   ...
   );

3. **Data types** define the kind of data (e.g., INT, VARCHAR) each column can store.

   Constraints ensure data integrity, accuracy, and validity (e.g., NOT NULL, PRIMARY KEY).

A-5:

1. The ALTER command is used to modify the structure of an existing table, such as adding, modifying, or deleting columns and constraints.

2. **Add a column:** ALTER TABLE table_name ADD column_name datatype;

   Modify a column: ALTER TABLE table_name MODIFY column_name new_datatype;

Drop a column: ALTER TABLE table_name DROP COLUMN column_name;

A-6:

1. The DROP command is used to permanently delete a database object like a table, view, or database.

2.
    - All data and structure of the table are permanently removed.
    - All relationships, constraints, and indexes linked to the table are lost.
    - The action **cannot be undone**, so it should be used with caution.

A-7:

1.
INSERT: Adds new records into a table.
INSERT INTO table_name (columns) VALUES (values);

UPDATE: Modifies existing records in a table.
UPDATE table_name SET column = value WHERE condition;

DELETE: Removes records from a table.
DELETE FROM table_name WHERE condition;

2. The WHERE clause specifies which records to update or delete. Without it, all records in the table may be affected.

A-8:

1. The SELECT statement is used to retrieve data from one or more tables. It allows you to specify which columns to display and can include conditions to filter results.
   Example:
   SELECT column1, column2 FROM table_name;

2. **WHERE** filters records based on specific conditions.
   SELECT * FROM table_name WHERE condition;

   ORDER BY sorts the result set in ascending (ASC) or descending (DESC) order.
   SELECT * FROM table_name ORDER BY column ASC;

A-9:

1. **GRANT** gives specific privileges (like SELECT, INSERT) to users.

   REVOKE removes previously granted privileges from users.

2. **GRANT Example:**
   GRANT SELECT, INSERT ON table_name TO user_name;

   REVOKE Example:
   REVOKE INSERT ON table_name FROM user_name;

A-10:

1. **COMMIT** saves all changes made during the current transaction permanently.
   ROLLBACK undoes all changes made during the current transaction.

2. **BEGIN TRANSACTION**: Starts a transaction.
   COMMIT: Confirms and saves changes.
   ROLLBACK: Cancels changes if an error occurs.
   Transactions ensure data integrity by following the **ACID** properties (Atomicity, Consistency, Isolation, Durability).

A-11:

1. **JOIN** combines rows from two or more tables based on a related column.

   INNER JOIN returns only matching rows in both tables.

   LEFT JOIN returns all rows from the left table and matched rows from the right table; unmatched right rows show NULL.

   RIGHT JOIN returns all rows from the right table and matched rows from the left table; unmatched left rows show NULL.

   FULL OUTER JOIN returns all rows when there is a match in either table, with NULLs for unmatched rows.

2. Joins link tables by matching values in related columns, allowing queries to retrieve combined data that is distributed across multiple tables.

A-12:

1. The GROUP BY clause groups rows that have the same values in specified columns. It is used with aggregate functions like SUM(), COUNT() etc., to perform calculations on each group separately.

2. **GROUP BY** groups rows to summarize data based on one or more columns.

   ORDER BY sorts the result set based on one or more columns in ascending or descending order.

A-13:

1. A stored procedure is a precompiled set of SQL statements saved in the database that can be executed repeatedly. Unlike a standard SQL query, it can include control-flow logic, accept parameters, and perform multiple operations in one call.

2. Improve performance by reducing network traffic. Enhance security by controlling access to data. Promote code reuse and easier maintenance. Allow complex operations and business logic inside the database.

A-14:

1. A **view** is a virtual table created by a stored SQL query that displays data from one or more tables. Unlike a table, it does not store data physically but shows data dynamically when queried.

2. Simplifies complex queries by hiding them behind a view. Enhances security by restricting access to specific data. Provides a consistent, customized representation of data.

A-15:

1. A **trigger** is a special stored procedure that automatically executes in response to specific events on a table (like insert, update, or delete).
   Types:
   - **BEFORE trigger:** Executes before the event occurs.
   - **AFTER trigger:** Executes after the event occurs.
   - **INSTEAD OF trigger:** Executes instead of the event (used with views).

2. **INSERT trigger:** Fires when a new row is inserted.

   UPDATE trigger: Fires when an existing row is modified.

   DELETE trigger: Fires when a row is deleted.

A-16:

1. **PL/SQL** (Procedural Language/SQL) is Oracle's extension of SQL that adds procedural features like loops, conditions, and variables, allowing more complex and powerful database operations.

2. **Improved performance:** Executes multiple SQL statements as a block.

   Modularity: Supports procedures, functions, and packages.

   Error handling: Allows exception handling for robust programs.

   Reusability: Code can be reused and maintained easily.

   Security: Provides controlled access through stored procedures.

A-17:

1. Control structures are commands used to control the flow of execution in PL/SQL programs.

   IF-THEN: Executes a block of code only if a condition is true.

   ```
   IF condition THEN
   END IF;
   ```

   LOOP: Repeats a block of code multiple times.

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

2. They allow conditional logic, repetition, and decision-making, making it possible to write dynamic, flexible, and powerful database programs beyond simple SQL queries.

A-18:

1. A **cursor** is a pointer that allows row-by-row processing of query results.
   - **Implicit Cursor:** Automatically created by PL/SQL for single-row queries
   - **Explicit Cursor:** Manually declared and used for processing multiple rows returned by a query.

2. Use an **explicit cursor** when you need to:

    - Handle **multiple rows** returned by a query.
    - Perform **custom row-by-row processing**.
    - Have better **control over fetching and looping** through the results.

A-19:

1. A **SAVEPOINT** marks a specific point in a transaction to which you can roll back later without affecting the entire transaction.

    - **ROLLBACK TO SAVEPOINT** undoes changes after the savepoint.
    - **COMMIT** ends the transaction and clears all savepoints.

2. Savepoints are useful when you want to **partially undo** changes in a long or complex transaction without rolling back everything, such as handling errors in one part while preserving others.