**Module 4 – Introduction to DBMS**

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

* SQL (Structured Query Language) is a standardized programming language used to manage and manipulate relational databases. It allows users to create, read, update, and delete (CRUD) data stored in databases.

Why SQL is Essential in Database Management:

* Data Manipulation:  
  SQL helps in inserting, updating, deleting, and retrieving data efficiently.
* Data Definition:  
  You can create, modify, and delete database structures like tables and schemas.
* Data Control:  
  SQL provides commands to control access to data through permissions (like GRANT and REVOKE).
* Standardized Language:  
  SQL is universally used and supported by most database systems (MySQL, PostgreSQL, Oracle, SQL Server, etc.).
* Efficient Querying:  
  It allows complex queries to filter, sort, group, and join data from multiple tables.

1. **Explain the difference between DBMS and RDBMS.**

* Difference:

|  |  |
| --- | --- |
| DBMS | RDBMS |
| Stores data as files, often in hierarchical or network formats | Stores data in tables (rows and columns) |
| Not supported or rarely used | Supports normalization to reduce redundancy |
| Microsoft Access, XML, file systems | MySQL, PostgreSQL, Oracle, SQL Server |
| Limited or not support | Full support for multi-user access and transactions |
| Limited support for constraints and integrity | Enforces data integrity using keys and constraints |

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

* SQL (Structured Query Language) plays a central role in managing relational databases. One of its main functions is data definition, where SQL is used to create and modify the structure of database objects like tables, views, and indexes using commands such as CREATE, ALTER, and DROP.
* Another important role of SQL is datamanipulation. It allows users to insert new records, update existing data, delete records, and retrieve specific information using commands like INSERT, UPDATE, DELETE, and SELECT.
* SQL also provides datacontrol, which manages user access and permissions. With commands like GRANT and REVOKE, database administrators can control who can view or change data, ensuring security and privacy.
* In addition, SQL supports transactioncontrol to maintain the integrity and consistency of the database. Commands like COMMIT, ROLLBACK, and SAVEPOINT help manage changes to the database, especially when performing multiple related operations.

1. **What are the key features of SQL?**

* Key features:
* Data Definition Language (DDL):  
  SQL allows creating, altering, and deleting database structures using commands like CREATE, ALTER, DROP.
* Data Manipulation Language (DML):  
  SQL supports inserting, updating, deleting, and retrieving data using INSERT, UPDATE, DELETE, and SELECT.
* Data Control Language (DCL):  
  It provides control over access to data using GRANT and REVOKE.
* Transaction Control Language (TCL):  
  Manages transactions using commands like COMMIT, ROLLBACK, and SAVEPOINT.
* High-level Language:  
  SQL is declarative; you describe what you want, not how to do it.

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

* Keywords
* Keywords:  
  Reserved words that perform specific tasks (e.g., SELECT, FROM, WHERE, INSERT, UPDATE, DELETE, CREATE).
* Identifiers:  
  Names of database objects like tables, columns, databases (e.g., students, employee\_id).
* Clauses:  
  Building blocks of SQL statements that define actions (e.g., SELECT, FROM, WHERE, GROUP BY, ORDER BY, HAVING).
* Expressions:  
  Combinations of values, operators, and functions that return a result (e.g., price \* quantity, salary + bonus).
* Predicates:  
  Conditions used in WHERE, HAVING to filter records (e.g., age > 18, name = 'John').
* Literals:  
  Fixed data values like numbers (100), strings ('Hello'), or dates ('2023-01-01').
* Functions:  
  Built-in SQL functions for performing operations (e.g., SUM (), COUNT (), NOW (), UPPER ()).
* Semicolon (;):  
  Used to terminate a SQL statement (especially in systems that allow multiple queries).

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

* **Structure**

SELECT column1, column2, ...

FROM table\_name

WHERE condition

GROUP BY column

HAVING condition

ORDER BY column ASC|DESC;

* SELECT – Specifies the columns to retrieve.
* FROM – Specifies the table(s) to fetch data from.
* WHERE – Filters rows based on a condition (optional).
* GROUP BY – Groups rows that have the same values in specified columns (optional).
* HAVING – Filters groups after GROUP BY (optional).
* ORDER BY – Sorts the result set (optional).

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

* Role of clauses
* SELECT Clause:  
  Specifies which columns to retrieve from a table.
* FROM Clause:  
  Defines the table(s) from which the data is selected.
* WHERE Clause:  
  Filters rows based on specified conditions (used to retrieve only matching records).
* GROUP BY Clause:  
  Groups rows that have the same values in specified columns, often used with aggregate functions like COUNT (), SUM (), etc.
* HAVING Clause:  
  Filters the grouped data after applying GROUP BY (similar to WHERE but for groups).
* ORDER BY Clause:  
  Sorts the result set by one or more columns in ascending (ASC) or descending (DESC) order.
* JOIN Clause:  
  Combines rows from two or more tables based on a related column.

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

* Constraints:
* NOT NULL

Ensures that a column cannot have a NULL value.

Example: name VARCHAR (50) NOT NULL

* UNIQUE

Ensures that all values in a column are different.

Example: email VARCHAR (100) UNIQUE

* + PRIMARY KEY

Uniquely identifies each row in a table.

Combines NOT NULL + UNIQUE.

Only one primary key per table (can be one or more columns).

Example: id INT PRIMARY KEY

* + FOREIGN KEY

Creates a link between two tables.

Enforces referential integrity.

The value in one table must match a value in another table’s primary key.

Example: student\_id INT,

FOREIGN KEY (student\_id) REFERENCES students(id)

* + CHECK

Ensures that values in a column meet a specific condition.

Example: age INT CHECK (age >= 18)

* + DEFAULT

Sets a default value for a column if no value is provided.

Example: status VARCHAR (10) DEFAULT 'active'

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

* Difference

|  |  |
| --- | --- |
| Primary key | Foreign key |
| Uniquely identifies each record in a table | Establishes a relationship between two tables |
| Must be unique for every row | Can have duplicate values |
| Cannot be NULL | Can contain NULL values |
| Defined in the sametable | Refers to the primary key in another table |
| Entity integrity | Referential integrity |

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

* Role
* NOT NULL Constraint
* Purpose: Ensures that a column cannot have NULL (empty) values.
* Use Case: When a column must always have data.
* Enforces: Mandatory data entry in that column**.**

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

* DDL (Data Definition Language) is a subset of SQL used to define, create, modify, and delete the structure of database objects such as tables, schemas, indexes, and views.
* It deals with database schema definition, not the data itself.

|  |  |
| --- | --- |
| * CREATE | Creates new database objects (e.g., tables, views). |

|  |  |
| --- | --- |
| * ALTER | Modifies existing database structures (e.g., add/remove columns). |

|  |  |
| --- | --- |
| * DROP | Deletes database objects permanently. |

|  |  |
| --- | --- |
| * TRUNCATE | Removes all records from a table, but not the structure. |

|  |  |
| --- | --- |
| * RENAME | Renames a database object (like a table). |

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

* The CREATE command in SQL is part of DDL (Data Definition Language). It is used to create new database objects like:
* Tables
* Databases
* Views
* Indexes
* Stored Procedures

Example:

CREATE TABLE table\_name (

column1 datatype [constraint],

column2 datatype [constraint],

...

);

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

* Purpose **:**
* Purpose:  
  Define what type of data can be stored in each column (e.g., numbers, text, dates).
* Why it matters:

Prevents invalid data (e.g., storing text in a number column).

Optimizes storage and performance.

Enables proper comparisons and calculations.

* Example:

age INT , -- Only whole numbers

name VARCHAR (50) -- Up to 50 characters

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

* The ALTER command in SQL is used to modify the structure of an existing database object — most commonly a table.
* Uses of ALTER Command:
* Add a new column:

ALTER TABLE Employees ADD age INT;

* Modify a column’s data type:

ALTER TABLE Employees MODIFY age SMALLINT;

* Drop a column:

ALTER TABLE Employees DROP COLUMN age;

* Rename a table (MySQL):

RENAME TABLE Employees TO Staff;

* Add a constraint**:**

ALTER TABLE Employees ADD CONSTRAINT email\_unique UNIQUE(email);

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

* Describe below
* Add a new column:

ALTER TABLE Employees ADD age INT;

* Modify a column’s data type:

ALTER TABLE Employees MODIFY age SMALLINT;

* Drop a column:

ALTER TABLE Employees DROP COLUMN age;

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

* The DROP command is used to permanently delete database objects such as tables, databases, views, indexes, etc.
* Drop table

Syntax: DROP TABLE table\_name;

Example: DROP students;

* Drop Database

Syntax: DROP DATABASE database\_name;

Example: DROP school\_db

* Drop other objects

Example: DROP VIEW student\_view;

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

* Here are the main implications:
* Dropping a table permanently deletes all the data stored in it, with no way to recover it unless a backup exists.
* The structure or schema of the table is also lost, including column names, data types, and constraints.
* Any foreign key relationships involving the table may break, causing errors in related tables.
* All dependent objects like views, indexes, triggers, and stored procedures that use the table will become invalid or stop functioning.
* It can free up storage space, which may be useful in cleaning up unused or obsolete data structures.
* Careless use of DROP TABLE can lead to data loss and application failures, so it should be used with caution.

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

* Commands:
* **INSERT** Command

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

Syntax: INSERT INTO table\_name (column1, column2, ...)

VALUES (value1, value2, ...);

Example: INSERT INTO students (name, age, grade)

VALUES ('Ravi', 18, 'A');

* **UPDATE** Command

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

Syntax: UPDATE table\_name

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

WHERE condition;

Example: UPDATE students

SET grade = 'B'

WHERE name = 'Ravi';

* **DELETE** Command

The DELETE command is used to remove one or more records from a table.

Syntax: DELETE FROM table\_name

WHERE condition;

Example: DELETE FROM students

WHERE name = 'Ravi';

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

* The WHERE clause is crucial when performing UPDATE and DELETE commands because it filters the rows that should be updated or deleted.
* Why it is Important:
* Prevents Unintended Changes:  
  Without a WHERE clause, all rows in the table will be updated or deleted.
* Targets Specific Records:  
  It allows you to change or remove only the rows that match certain conditions.
* Maintains Data Integrity:  
  Ensures that only relevant data is modified, protecting the rest of the table from accidental data loss or corruption.

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

* The SELECT statement is used to retrieve data from one or more tables in a database. It is the most commonly used SQL command for querying and displaying records.
* Basic syntax: SELECT column1, column2, ...

FROM table name

WHERE condition;

* Retrieve All Columns

SELECT \* FROM students;

* Retrieve specific columns

SELECT name, age FROM students;

* Filter data with WHERE clause

SELECT \* FROM students

WHERE grade = 'A';

* Sort Results with ORDER BY

SELECT name, age FROM students

ORDER BY age DESC;

* Use of Functions

SELECT COUNT (\*) FROM students;

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

* Explain below
* WHERE clause

The WHERE clause is used to filter rows in a table based on a specific condition.

To select only those records that meet a certain condition.

It is used with commands like SELECT, UPDATE, and DELETE.

Example: SELECT \* FROM students

WHERE grade = 'A';

* ORDER BY clause

The ORDER BY clause is used to sort the result of a query in ascending (ASC) or descending (DESC) order based on one or more columns.

To organize the result set in a meaningful order.

Default order is ascending (A-Z or 0-9) unless specified as DESC.

Example: SELECT name, age FROM students

ORDER BY age DESC;

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

* Purpose of GRANT and REVOKE in SQL:

The GRANT and REVOKE commands are used for managing user permissions in a database. They help control who can access or modify database objects like tables, views, procedures, etc.

* GRANT Command

The GRANT command is used to give specific privileges (permissions) to users.

Purpose:

Allow users to read, insert, update, or delete data.

Grant access to execute functions, create tables, or use resources.

Syntax: GRANT privilege\_type ON object\_name TO user\_name;

* REVOKE Command

The REVOKE command is used to remove previously granted permissions from a user. Restrict access when it's no longer needed or authorized.

Enhances security by removing unnecessary rights.

Syntax: REVOKE privilege\_type ON object\_name FROM user\_name;

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

* Managing privileges in SQL means controlling what actions users can perform on database objects (like tables, views, procedures, etc.). This is done using the GRANT and REVOKE commands**.**
* Using GRANT to Give Privileges

You use the GRANT command to allow a user to perform certain operations.

Syntax: GRANT privilege1, privilege2 ON object TO user;

Example: REVOKE INSERT ON students FROM user1;

* Using REVOKE to Remove Privileges

You use the REVOKE command to take away previously given privileges.

Syntax: REVOKE privilege1, privilege2 ON object FROM user;

Example: REVOKE privilege1, privilege2 ON object FROM user;

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

* The COMMIT and ROLLBACK commands are part of transaction control in SQL. They help ensure data integrity and allow you to control changes made during a transaction.
* COMMIT Command

The COMMIT command is used to permanently save all changes made during the current transaction.

To make changes permanent in the database.

Ends the current transaction successfully.

example: UPDATE students SET grade = 'A' WHERE student\_id = 101;

COMMIT;

* + - ROLLBACK Command

The ROLLBACK command is used to undo changes made during the current transaction.

To cancel changes if something goes wrong (e.g., error, wrong update).

Restores the data to its previous state.

example: UPDATE students SET grade = 'F' WHERE student\_id = 101;

ROLLBACK;

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

* A transaction is a group of one or more SQL statements executed as a single unit of work. Transactions are used to maintain data accuracy, consistency, and integrity.
* ACID Properties of Transactions:
* Atomicity – All operations in a transaction succeed together or fail together.
* Consistency – Ensures that the database remains in a valid state before and after the transaction.
* Isolation – Multiple transactions can occur independently without affecting each other.
* Durability – Once a transaction is committed, the changes are permanently saved, even after a crash.
* Transaction Control Commands:
* BEGIN or START TRANSACTION – Starts a new transaction.
* COMMIT – Saves all changes made during the transaction permanently.
* ROLLBACK – Cancels all changes made during the transaction.
* SAVEPOINT – Sets a point to partially roll back a transaction to a specific step.

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

* The JOIN clause in SQL is used to combine rows from two or more tables based on a related column between them (usually a foreign key).
* Types of array :

1. INNER JOIN

Returns only matching rows from both tables.  
Rows are included only when a match exists in both tables.

Example: SELECT \* FROM A

INNER JOIN B ON A.id = B.a\_id;

1. LEFT JOIN (LEFT OUTER JOIN)

Returns all rows from the left table, even if there is no match in the right table.  
If no match, columns from the right table will show NULL.

Example: SELECT \* FROM A

LEFT JOIN B ON A.id = B.a\_id;

1. RIGHT JOIN (RIGHT OUTER JOIN)

Returns all rows from the right table, even if there is no match in the left table.  
If no match, columns from the left table will show NULL.

Example: SELECT \* FROM A

RIGHT JOIN B ON A.id = B.a\_id;

1. FULL OUTER JOIN

Returns all rows from both tables.  
If there is a match, rows are joined.  
If there is no match, NULLs are filled in for missing values.

Example: SELECT \* FROM A

FULL OUTER JOIN B ON A.id = B.a\_id;

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

* Use of joins:
* Joins combine rows from two or more tables based on a related column (usually a foreign key).
* They are used when data is spread across multiple tables and needs to be viewed together.
* The join condition is usually written using the ON keyword, comparing related columns.
* Types of Joins:
* INNER JOIN
* RIGHT JOIN (Right Outer Join)
* FULL OUTER JOIN
* CROSS JOIN

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

* The GROUP BY clause is used to group rows that have the same values in one or more columns.
* How It Works:
* SQL groups the rows based on the specified column(s).  
  Then, aggregate functions are applied to each group, not the entire table.
* Common Aggregate Functions Used with GROUP BY:
* COUNT () → Counts rows in each group
* SUM () → Adds values in each group
* AVG () → Calculates average for each group
* MAX () → Finds highest value in each group
* MIN () → Finds lowest value in each group

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

* GROUP BY:
* Used to group rows that have the same values in one or more columns.
* Commonly used with aggregate functions like SUM (), COUNT (), AVG (), etc.
* Helps in data summarization (e.g., total sales per category).
* Groups data but does not sort it by default.
* ORDER BY:
* Used to sort the result in ascending (ASC) or descending (DESC) order.
* Does not group data.
* Can be used with or without GROUP BY.
* Affects presentation, not grouping or aggregation.

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

* A stored procedure is a precompiled set of one or more SQL statements stored in the database with a specific name.
* It can be executed (called) whenever needed, with or without parameters.
* Features of Stored Procedure:
* Written once and can be used multiple times.
* Supports input, output, and input-output parameters.
* Can include control flow logic (e.g., IF, WHILE).
* Improves performance due to precompilation.
* Helps in modularizing and securing SQL code.
* Syntax:

CREATE PROCEDURE GetEmployeeByDept (IN dept\_id INT)

BEGIN

SELECT \* FROM employees WHERE department\_id = dept\_id;

END;

* To call procedure:

CALL GetEmployeeByDept (3);

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

* Advantages of stored procedure:
* Reusability  
  Stored once in the database and can be called multiple times from different programs.
* Improved Performance  
  They are precompiled, so execution is faster than ad-hoc SQL queries.
* Code Modularity  
  Makes it easier to organize and reuse code by dividing it into logical units.
* Better Security  
  Access to data can be controlled by granting users permission to execute procedures, without giving direct table access.
* Consistency  
  Ensures that the same business rules are applied everywhere the procedure is used.

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 an SQL query.
* It does not store data physically — it only displays data from one or more real tables.
* A view is created using the CREATE VIEW statement.
* Example :

CREATE VIEW EmployeeSalary AS

SELECT name, salary FROM employees WHERE salary > 50000;

* Difference Between View and Table:

|  |  |
| --- | --- |
| View | table |
| Virtual table (based on a query) | Physical table (stores actual data) |
| Does not store data | Stores data permanently |
| Sometimes read-only | Fully updatable |
| Slower for large data | Faster |
| Reflects latest data from base tables | Must be updated manually |

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

* Advantages**:**
* Simplifies Complex Queries  
  Views can wrap complex joins and calculations into a single, easy-to-use virtual table.
* Enhances Data Security  
  Restricts user access to specific columns or rows without exposing the entire table.
* Provides Logical Data Independence  
  The underlying table structure can change without affecting users who query the view.
* Improves Readability and Reusability  
  Once created, a view can be reused in multiple queries, improving consistency and reducing repetition.
* Supports Data Abstraction  
  Presents only the necessary data, hiding unnecessary details from the user.
* Facilitates Maintenance  
  Changes in business logic can be handled in the view definition instead of updating all queries.

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

* A trigger is a set of instructions that automatically executes (or "fires") in response to certain events on a table or view, such as INSERT, UPDATE, or DELETE.
* Types of triggers:
* BEFORE Trigger  
  Executes before the triggering SQL operation (INSERT, UPDATE, DELETE).  
  Use Case: To validate or modify data before it is committed to the table.  
  Example: Prevent negative values from being inserted into a salary column.
* AFTER Trigger  
  Executes after the triggering SQL operation has completed.  
  Use Case: For logging changes, updating audit tables, or cascading changes to other tables.  
  Example: After a new user is added, log the action in a history table.

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

* Here's the difference

1. INSERT Trigger

Fires when a new row is added to a table using an INSERT statement.  
Use Case: Automatically log new entries, set default values, or validate inserted data. Example Use: After inserting a new employee, add a record in the audit table. Set created\_at timestamp when a row is inserted.

1. UPDATE Trigger

Fires when an existing row is modified using an UPDATE statement.  
Use Case: Track changes, restrict certain updates, or compare old vs. new values.

Example Use:

Log old and new values for salary updates.

Prevent update if user tries to reduce product quantity below a certain threshold.

1. DELETE Trigger

Fires when a row is removed from a table using a DELETE statement.  
Use Case: Archive deleted data, enforce restrictions before deletion, or log deletions.

Example Use:

Move deleted customer records to a backup table.

Prevent deletion of rows that are referenced in another table.

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

* PL/SQL (Procedural Language/Structured Query Language) is Oracle's procedural extension to SQL. It combines the power of SQL with the features of procedural programming, such as loops, conditions, variables, and error handling.
* Procedural Logic  
   Allows use of control structures like IF, LOOP, and CASE, enabling complex decision-making and flow control.
* Variables & Constants  
   Supports declaration and use of variables and constants to store and manipulate data during program execution.
* Error Handling  
   Provides structured exception handling using EXCEPTION blocks to catch and manage runtime errors.
* Modular Programming  
   Enables creation of procedures, functions, packages, and triggers for reusable, organized, and maintainable code.
* Cursors  
   Allows row-by-row processing of query results using implicit and explicit cursors for precise data handling.
* Tight SQL Integration  
   Supports embedding SQL queries directly within procedural code for seamless interaction with the database.

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

* **Benefits of Using PL/SQL:**
* Combines SQL with Procedural Logic  
  Enables writing complex business logic by combining SQL with programming features like loops, conditions, and variables.
* Improved Performance  
  Reduces network traffic by sending blocks of code to the database server instead of multiple individual SQL statements.
* Modular Code Structure  
  Supports procedures, functions, and packages, allowing code to be reused, organized, and easier to maintain.
* Portability  
  PL/SQL programs can run on any Oracle database, making applications easily portable across different systems.
* Security  
  Access to database objects can be controlled using PL/SQL packages, hiding internal logic and improving data protection.

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

* Control structures in PL/SQL are programming constructs used to control the flow of execution in a program. They help in making decisions, repeating actions, and branching logic.
* PL/SQL supports three main types of control structures:

Conditional Control (IF-THEN, IF-THEN-ELSE)

Iterative Control (LOOP, WHILE, FOR)

Sequential Control (GOTO, NULL)

* IF-THEN Control Structure

Used to execute a block of code only if a specified condition is true.

Syntax:

IF condition THEN

-- statements to execute if condition is true

END IF;

Example:

IF salary < 3000 THEN

salary := salary + 500;

END IF;

* LOOP Control Structure

Used to repeatedly execute a block of code until an EXIT condition is met. It's a basic loop without any predefined end.

Syntax:

LOOP

-- statements

EXIT WHEN condition;

END LOOP;

Example:

DECLARE

counter NUMBER: = 1;

BEGIN

LOOP

DBMS\_OUTPUT.PUT\_LINE ('Counter: ' || counter);

counter: = counter + 1;

EXIT WHEN counter > 5;

END LOOP;

END;

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

* Control structures in PL/SQL enhance the power of SQL by enabling conditional logic, repetition, and decision-making, which are essential for writing complex, dynamic, and intelligent database programs.
* Ways Control Structures Help in Writing Complex Queries:
* Conditional Execution  
  Use of IF-THEN, IF-THEN-ELSE, and CASE allows executing specific SQL queries only when certain conditions are met.
* Iterative Processing  
  LOOP, WHILE, and FOR loops allow repeating actions like executing queries multiple times or processing rows one by one using cursors.
* Error Handling  
  Using EXCEPTION blocks ensures complex queries can handle errors without crashing the program.
* Decision-Making Logic  
  Control structures allow embedding logic that makes runtime decisions to choose between multiple query paths.
* Modular Execution  
  Combine control structures inside procedures/functions to build reusable logic that simplifies complex query operations.

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

* A cursor in PL/SQL is a pointer that retrieves and processes rows returned by a SQL query. It helps in handling query result sets row by row, especially when a query returns multiple rows.
* Types of cursors:

1. Implicit cursor
2. Explicit cursor

* Difference
* Implicit cursors are automatically created by PL/SQL, while explicit cursors are manually declared by the programmer.
* Implicit cursors are used for single-row queries and DML operations, whereas explicit cursors handle multi-row SELECT queries.
* Implicit cursors are managed internally by PL/SQL, but explicit cursors require manual OPEN, FETCH, and CLOSE operations.
* Implicit cursors are accessed using the SQL keyword, while explicit cursors have user-defined names.
* Both support cursor attributes, but implicit cursors use SQL% and explicit cursors use cursor\_name%.

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

* You would use an explicit cursor over an implicit one in the following situations:
* When the query returns multiple rows
* When you need fine-grained control over data processing
* When you want to process each row individually
* When you need to use cursor attributes
* When performance or logic requires selective fetching

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

* SAVEPOINT is used in transaction management to set a marker or checkpoint within a transaction.
* It allows you to roll back part of a transaction without undoing the whole transaction.
* Syntax: SAVEPOINT savepoint\_name;
* ROLLBACK TO savepoint\_name undoes all changes made after that savepoint, while keeping earlier changes intact.
* COMMIT makes all changes permanent, and it releases all savepoints created in the transaction.
* SAVEPOINT helps in error handling, partial rollbacks, and maintaining control in complex transactions.
* Example:

UPDATE accounts SET balance = balance - 100 WHERE id = 1;

SAVEPOINT after\_withdraw;

UPDATE accounts SET balance = balance + 100 WHERE id = 2;

SAVEPOINT after\_deposit;

ROLLBACK TO after\_withdraw;

COMMIT;

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

* When performing multiple steps in a transaction
* When handling conditional logic or partial failures
* When dealing with complex business logic
* During batch operations
* In nested procedures or triggers