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

Ans : SQL (Structured Query Language) is a standard programming language used to interact with and manage data stored in relational databases. It allows users to create, retrieve, update, and delete data efficiently.

SQL is essential in database management because it provides a simple and powerful way to communicate with databases. It helps ensure data accuracy, consistency, and security while enabling fast data retrieval and manipulation. Almost all modern applications rely on SQL to store and manage their data.

Q : 2 Explain the difference between DBMS and RDBMS.

Ans : A DBMS (Database Management System) is software used to store, manage, and retrieve data. It does not necessarily follow a relational structure and may store data as files or simple tables.

An RDBMS (Relational Database Management System) is an advanced type of DBMS that stores data in structured tables with relationships between them. It uses keys (primary and foreign keys) to maintain relationships and ensures data integrity through constraints.

In short, DBMS focuses on basic data storage, while RDBMS focuses on structured data with relationships and rules.

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

Ans : SQL plays a central role in managing relational databases by providing commands to define database structures, manipulate data, and control access. It is used to create tables, insert and update records, retrieve data using queries, and manage user permissions.

SQL also helps maintain relationships between tables and ensures data integrity through constraints such as primary keys and foreign keys. Without SQL, interacting with relational databases would be difficult and inefficient.

Q : 4 What are the key features of SQL?

Ans :

- It is easy to learn and use due to its simple syntax.
- It supports data definition, data manipulation, and data control operations.
- It allows complex queries using conditions, joins, and functions.
- It ensures data integrity through constraints like primary keys and foreign keys.
- It provides security by controlling user access and permissions.
- It is platform-independent and works with most relational database systems.

Q : 5 Create a new database named school_db and a table called students with the following columns: student_id, student_name, age, class, and address.

Insert five records into the students table and retrieve all records using the SELECT statement.

Ans :

- ```
  CREATE DATABASE school_db;
  ```
- ```
  CREATE TABLE students (
      student_id INT PRIMARY KEY,
      student_name VARCHAR(50),
      age INT,
      class VARCHAR(20),
      address VARCHAR(100)
  );
  ```
- ```
  INSERT INTO students VALUES
  (1, 'Rahul Sharma', 14, '8A', 'Delhi'),
  (2, 'Anita Verma', 13, '7B', 'Mumbai'),
  (3, 'Karan Patel', 15, '9A', 'Ahmedabad'),
  (4, 'Sneha Gupta', 14, '8B', 'Jaipur'),
  (5, 'Arjun Singh', 13, '7A', 'Lucknow');
  ```
- ```
  SELECT * FROM students;
  ```

Q : 6 What are the basic components of SQL syntax?

Ans :

The basic components of SQL syntax are the building blocks used to write SQL queries. These include:

- Keywords – Reserved words that perform specific operations, such as SELECT, INSERT, UPDATE, DELETE, CREATE, and WHERE.
- Identifiers – Names of database objects like tables, columns, and schemas.
- Operators – Used to perform operations on data, such as =, >, <, AND, OR, and LIKE.
- Clauses – Parts of a query that define conditions and data manipulation, such as FROM, WHERE, GROUP BY, and ORDER BY.
- Expressions – Combinations of values, columns, and operators that produce a result.
- Literals – Fixed values such as numbers, strings, or dates used in queries.

Q : 7 Write the general structure of an SQL SELECT statement.

Ans : `SELECT column1, column2, ...FROM table_name WHERE condition GROUP BY column`

```
HAVING condition ORDER BY column;
```

Q : 8 Explain the role of clauses in SQL statements.

Ans : Clauses in SQL statements define how data should be processed and retrieved from a database. They act as instructions that control different parts of a query.

For example, the FROM clause specifies the table to query, the WHERE clause filters records, GROUP BY organizes data into groups, and ORDER BY sorts the results. Clauses make SQL queries more precise, flexible, and powerful by allowing users to apply conditions, grouping, and sorting to the data.

Q : 9 Lab 1: Write SQL queries to retrieve specific columns (student_name and age) from the students table.

Lab 2: Write SQL queries to retrieve all students whose age is greater than 10.

Ans :

- ```
  SELECT student_name, age
  FROM students;
  ```

- ```
  SELECT *
  FROM students
  WHERE age > 10;
  ```

Q : 10 What are constraints in SQL? List and explain the different types of constraints.

Ans :

Constraints in SQL are rules applied to table columns to ensure the accuracy, reliability, and integrity of data stored in a database. They prevent invalid data from being inserted and help maintain consistency.

Types of SQL Constraints

- PRIMARYKEY
  Ensures each record is unique and not null. It uniquely identifies each row in a table.
- FOREIGNKEY
  Establishes a relationship between two tables by referencing the primary key of another table.
- NOTNULL
  Ensures that a column cannot have a null (empty) value.
- UNIQUE
  Ensures all values in a column are different from each other.
- CHECK
  Ensures that values in a column meet a specific condition.
- DEFAULT
  Assigns a default value to a column when no value is provided.

Q : 11 How do PRIMARY KEY and FOREIGN KEY constraints differ?

Ans : A PRIMARY KEY uniquely identifies each record in a table and does not allow null or duplicate values. There can be only one primary key in a table.

A FOREIGN KEY is used to create a relationship between two tables. It refers to the primary key of another table and ensures referential integrity. A table can have multiple foreign keys.

In simple terms, a primary key identifies records within a table, while a foreign key links one table to another.

Q : 12 What is the role of NOT NULL and UNIQUE constraints?

Ans : The NOT NULL constraint ensures that a column must always contain a value and cannot be left empty. It is used when a field is mandatory.

The UNIQUE constraint ensures that all values in a column are different, preventing duplicate entries. It is commonly used for fields like email or username.

Together, these constraints help maintain data accuracy and prevent incomplete or duplicate records.

Q : 13 Lab 1: Create a table teachers with the following columns: teacher_id (Primary Key), teacher_name (NOT NULL), subject (NOT NULL), and email (UNIQUE).

Lab 2: Implement a FOREIGN KEY constraint to relate the teacher_id from the teachers table with the students table.

Ans :

- ```
  CREATE TABLE teachers (
      teacher_id INT PRIMARY KEY,
      teacher_name VARCHAR(50) NOT NULL,
      subject VARCHAR(50) NOT NULL,
      email VARCHAR(100) UNIQUE
  );
  ```
- ```
  ALTER TABLE students ADD teacher_id INT;
  ```
- ```
  ALTER TABLE students
  ADD CONSTRAINT fk_teacher
  FOREIGN KEY (teacher_id)
  REFERENCES teachers(teacher_id);
  ```

Q : 14 Define the SQL Data Definition Language (DDL).

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

Common DDL commands include:
  ```

- CREATE – Creates database objects
- ALTER – Modifies existing database objects
- DROP – Deletes database objects
- TRUNCATE – Removes all records from a table

DDL focuses on the design and structure of the database rather than the data itself.

Q : 15 Explain the CREATE command and its syntax.

Ans : The CREATE command is used to create new database objects such as databases, tables, views, and indexes.

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

Q : 16 What is the purpose of specifying data types and constraints during table creation?

Ans : Data types define the type of data that can be stored in each column, such as integers, text, or dates. They ensure that only valid data is entered into the database.

Constraints are rules applied to columns to maintain data accuracy and integrity. They prevent invalid, duplicate, or missing values and help enforce relationships between tables.

Together, data types and constraints ensure that the database remains consistent, reliable, and efficient.

Q : 17 Lab 1: Create a table courses with columns: course_id, course_name, and course_credits. Set the course_id as the primary key.

- Lab 2: Use the CREATE command to create a database university_db.

Ans :

- ```
  CREATE TABLE courses (
      course_id INT PRIMARY KEY,
      course_name VARCHAR(50),
      course_credits INT
  );
  ```
- ```
  CREATE DATABASE university_db;
  ```

Q : 18 What is the use of the ALTER command in SQL?

Ans : The ALTER command in SQL is used to modify the structure of an existing database object, such as a table. It allows users to make changes without deleting the table or losing existing data.

With the ALTER command, you can add new columns, change column data types, rename columns, add or remove constraints, and delete columns. It is mainly used when the database structure needs to be updated or improved.

Q : 19 How can you add, modify, and drop columns from a table using ALTER?

Ans :

- ```
  ALTER TABLE table_name
  ADD column_name datatype;
  ```

- ```
  ALTER TABLE table_name
  MODIFY column_name datatype;
  ```

- ```
  ALTER TABLE table_name
  DROP COLUMN column_name;
  ```

Q : 20 What is the function of the DROP command in SQL?

Ans : The DROP command in SQL is used to permanently remove database objects such as tables, databases, indexes, or views. When an object is dropped, its structure and all the data stored in it are deleted completely from the database.

The DROP command is part of the Data Definition Language (DDL) and should be used carefully because the operation cannot usually be undone.

Q : 21 What are the implications of dropping a table from a database?

Ans : Dropping a table removes both the table structure and all its records permanently. This means the data cannot be recovered unless a backup exists.

It can also affect other tables that depend on it through foreign key relationships, potentially causing errors or breaking database integrity. Therefore, dropping a table should only be done when it is no longer needed.

Q : 22 Lab 1: Drop the teachers table from the school_db database.

• Lab 2: Drop the students table from the school_db database and verify that the table has been removed.

Ans :

- ```
  DROP TABLE school_db.teachers;
  ```

- `DROP TABLE school_db.students;`

- `SHOW TABLES FROM school_db;`

Q : 23 Define the INSERT, UPDATE, and DELETE commands in SQL.

Ans : INSERT

The INSERT command is used to add new records (rows) into a table. It allows users to enter data into one or more columns of a table.

UPDATE

The UPDATE command is used to modify existing records in a table. It changes the values of specified columns based on a condition.

DELETE

The DELETE command is used to remove one or more records from a table. It deletes rows that match a specified condition.

Together, these commands are part of the Data Manipulation Language (DML) and are used to manage data within a database.

Q : 24 What is the importance of the WHERE clause in UPDATE and DELETE operations?.

Ans : The WHERE clause specifies which records should be updated or deleted. It ensures that only the intended rows are affected.

Without the WHERE clause:

- UPDATE will modify all rows in the table
- DELETE will remove all records from the table

Therefore, the WHERE clause is important to prevent accidental data loss or unwanted changes.

Q : 25 Lab 1: Insert three records into the courses table using the INSERT command.

- Lab 2: Update the course duration of a specific course using the UPDATE command.
- Lab 3: Delete a course with a specific course_id from the courses table using the DELETE command.

Ans :

- ```
  INSERT INTO courses (course_id, course_name, course_duration)
  VALUES
  (101, 'Database Systems', '3 Months'),
  (102, 'Web Development', '4 Months'),
  (103, 'Machine Learning', '6 Months');
  ```

- `UPDATE courses`
  ```

```
SET course_duration = '5 Months'
WHERE course_id = 102;
```

- ```
  DELETE FROM courses
  WHERE course_id = 101;
  ```

Q : 25 What is the SELECT statement, and how is it used to query data?

Ans : The SELECT statement is used to retrieve data from one or more tables in a database. It allows users to specify which columns to display and can include conditions, sorting, and grouping.

The SELECT statement is the most commonly used SQL command because it helps users view and analyze stored data without modifying it. By combining it with clauses like WHERE, ORDER BY, and LIMIT, users can fetch specific and organized results.

Q : 26 Explain the use of the ORDER BY and WHERE clauses in SQL queries.

Ans : The WHERE clause is used to filter records based on a specified condition. It ensures that only the rows matching the condition are returned.

The ORDER BY clause is used to sort the result set in either ascending (ASC) or descending (DESC) order based on one or more columns. It helps organize data in a meaningful way.

Together, these clauses make SQL queries more precise and useful by allowing filtering and sorting of data.

Q : 27 Lab 1: Retrieve all courses from the courses table using the SELECT statement.

- Lab 2: Sort the courses based on course_duration in descending order using ORDER BY.
- Lab 3: Limit the results of the SELECT query to show only the top two courses using LIMIT.

Ans :

- ```
  SELECT * FROM courses;
  ```

- ```
  SELECT * FROM courses
  ORDER BY course_duration DESC;
  ```

- ```
  SELECT * FROM courses
  LIMIT 2;
  ```

Q : 28 What is the purpose of GRANT and REVOKE in SQL?

Ans : The GRANT and REVOKE commands are used to manage user permissions in a database.

- GRANT is used to give specific privileges to users or roles, such as the ability to read, insert, update, or delete data.

- REVOKE is used to remove previously granted privileges from users or roles.

These commands help maintain database security by ensuring that users only have access to the data and operations they are authorized to use.

Q : 29 How do you manage privileges using these commands?

Ans : Privileges are managed by assigning or removing specific permissions on database objects like tables, views, or databases.

Using the GRANT command, a database administrator can allow users to perform actions such as SELECT, INSERT, UPDATE, or DELETE. With the REVOKE command, these permissions can be withdrawn when they are no longer needed.

This control ensures data security, prevents unauthorized access, and helps enforce role-based access control within a database system.

Q : 30 Lab 1: Create two new users user1 and user2 and grant user1 permission to SELECT from the courses table.

- Lab 2: Revoke the INSERT permission from user1 and give it to user2.

Ans :

- ```
  CREATE USER user1 IDENTIFIED BY 'password1';
  CREATE USER user2 IDENTIFIED BY 'password2';

  GRANT SELECT ON courses TO user1;
  ```

- ```
  REVOKE INSERT ON courses FROM user1;
  GRANT INSERT ON courses TO user2;
  ```

Q : 31 What is the purpose of the COMMIT and ROLLBACK commands in SQL?

Ans : The COMMIT and ROLLBACK commands are used to control transactions in SQL.

- COMMIT saves all the changes made during a transaction permanently in the database. Once a COMMIT is executed, the changes cannot be undone.
- ROLLBACK is used to undo changes made during a transaction. It restores the database to its previous state before the transaction began or before a savepoint.

These commands help maintain data consistency and ensure that only correct and verified changes are stored in the database.

Q : 32 Explain how transactions are managed in SQL databases.

Ans : A transaction is a sequence of SQL operations performed as a single unit of work. Transactions ensure that all operations are completed successfully; otherwise, the changes are rolled back.

Transactions are managed using commands such as:

- BEGIN / START TRANSACTION – Starts a transaction
- COMMIT – Saves changes permanently
- ROLLBACK – Reverts changes
- SAVEPOINT – Creates a point within a transaction to roll back to

SQL transactions follow the ACID properties (Atomicity, Consistency, Isolation, Durability), which guarantee reliable and secure data processing.

Q : 33 Lab 1: Insert a few rows into the courses table and use COMMIT to save the changes.

• Lab 2: Insert additional rows, then use ROLLBACK to undo the last insert operation.

• Lab 3: Create a SAVEPOINT before updating the courses table, and use it to roll back specific changes.

Ans :

- ```
  START TRANSACTION;
  INSERT INTO courses VALUES (201, 'Data Structures', '3 Months');
  INSERT INTO courses VALUES (202, 'Operating Systems', '4 Months');
  COMMIT;
  ```

- ```
  START TRANSACTION;
  INSERT INTO courses VALUES (203, 'Computer Networks', '3 Months');
  ROLLBACK;
  ```

- ```
  START TRANSACTION;
  INSERT INTO courses VALUES (204, 'Cloud Computing', '5 Months');
  SAVEPOINT before_update;
  UPDATE courses
  SET course_duration = '6 Months'
  WHERE course_id = 204;
  ROLLBACK TO before_update;
  COMMIT;
  ```

Q : 34 Explain the concept of JOIN in SQL. What is the difference between INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL OUTER JOIN?
  ```

Ans : A JOIN in SQL is used to combine rows from two or more tables based on a related column between them, usually a primary key and a foreign key. Joins allow users to retrieve related data stored across multiple tables in a single query.

Types of JOINs

- INNER JOIN
  Returns only the rows that have matching values in both tables.
- LEFT JOIN (LEFT OUTER JOIN)
  Returns all rows from the left table and the matching rows from the right table. If there is no match, NULL values are returned for the right table.
- RIGHT JOIN (RIGHT OUTER JOIN)
  Returns all rows from the right table and the matching rows from the left table. If there is no match, NULL values are returned for the left table.
- FULL OUTER JOIN
  Returns all rows from both tables. If there is no match, NULL values are shown for the missing side.

Q : 35 How are joins used to combine data from multiple tables?

Ans : Joins are used when related data is stored in separate tables. By using a common column (such as an ID), SQL can combine rows from these tables into a single result set.

For example, an employee table may store employee details, while a departments table stores department names. Using a JOIN, we can display employee information along with their department in one query. This helps avoid data duplication and keeps databases organized.

Q : 36 Lab 1: Create two tables: departments and employees. Perform an INNER JOIN to display employees along with their respective departments.

- Lab 2: Use a LEFT JOIN to show all departments, even those without employees.

Ans :

```
CREATE TABLE departments (
    department_id INT PRIMARY KEY,
    department_name VARCHAR(50)
);
```

```
CREATE TABLE employees (
    employee_id INT PRIMARY KEY,
    employee_name VARCHAR(50),
    department_id INT,
    FOREIGN KEY (department_id) REFERENCES departments(department_id)
);
```

- ```
  SELECT employees.employee_name, departments.department_name
  FROM employees
  INNER JOIN departments
  ON employees.department_id = departments.department_id;
  ```

- ```
  SELECT departments.department_name, employees.employee_name
  FROM departments
  LEFT JOIN employees
  ON departments.department_id = employees.department_id;
  ```

Q : 37 What is the GROUP BY clause in SQL? How is it used with aggregate functions?

Ans : The GROUP BY clause in SQL is used to arrange rows into groups based on one or more columns that have the same values. It is commonly used with aggregate functions to perform calculations on each group rather than on the entire table.

Aggregate functions such as COUNT(), SUM(), AVG(), MIN(), and MAX() are applied to each group to produce summarized results. For example, GROUP BY can be used to count how many employees belong to each department or calculate the average salary per department.

Q : 38 Explain the difference between GROUP BY and ORDER BY.

Ans : The GROUP BY clause is used to group rows that share similar values so that aggregate functions can be applied to each group.

The ORDER BY clause is used to sort the result set in ascending or descending order based on one or more columns.

In simple terms, GROUP BY organizes data into groups for calculations, while ORDER BY arranges the final output in a sorted manner.

Q : 39 Lab 1: Group employees by department and count the number of employees in each department using GROUP BY.

- Lab 2: Use the AVG aggregate function to find the average salary of employees in each department.

Ans :

- ```
  SELECT department_id, COUNT(*) AS total_employees
  FROM employees
  BY department_id;
  ```

- ```
  SELECT department_id, AVG(salary) AS average_salary
  FROM employees
  GROUP BY department_id;
  ```

Q : 40 What is a stored procedure in SQL, and how does it differ from a standard SQL query?

Ans : A stored procedure is a precompiled collection of SQL statements that is stored in the database and executed as a single unit. It can accept parameters, perform operations, and return results.

A standard SQL query is executed directly by the user each time it is needed and is not stored in the database for reuse.

Difference:

- Stored procedures are reusable, precompiled, and stored in the database.
- Standard queries are written and executed manually each time.

Stored procedures improve performance and maintainability, while standard queries are mainly used for simple or one-time operations.

Q : 41 Explain the advantages of using stored procedures.

Ans :

- **Reusability** – The same procedure can be executed multiple times without rewriting code.
- **Better Performance** – Since they are precompiled, execution is faster.
- **Security** – Users can execute procedures without direct access to tables.
- **Maintainability** – Changes can be made in one place without modifying multiple queries.
- **Reduced Network Traffic** – Only the procedure call is sent instead of long SQL statements.

Q : 42 Lab 1: Write a stored procedure to retrieve all employees from the employees table based on department.

- Lab 2: Write a stored procedure that accepts course_id as input and returns the course details.

Ans :

- ```
  CREATE PROCEDURE GetEmployeesByDepartment
    @dept_id INT
  AS
  BEGIN
    SELECT *
    FROM employees
    WHERE department_id = @dept_id;
  END;
  ```
- ```
  CREATE PROCEDURE GetCourseDetails
    @course_id INT
  AS
  BEGIN
    SELECT *
    FROM courses
    WHERE course_id = @course_id;
  ```

```
END;
```

Q : 42 What is a view in SQL, and how is it different from a table?

Ans : A view in SQL is a virtual table created from a SELECT query. It does not store data physically; instead, it displays data retrieved from one or more tables whenever it is queried.

A table, on the other hand, stores actual data in the database. Tables contain rows and columns where data is permanently stored.

Difference:

- A table stores real data.
- A view stores only the query and shows data dynamically.

Views are mainly used to simplify complex queries and restrict access to specific data.

Q : 43 Explain the advantages of using views in SQL databases.

Ans :

- Data Security – Users can access specific data without seeing the entire table.
- Simplified Queries – Complex joins and calculations can be stored in a view for easy reuse.
- Data Abstraction – Users do not need to know the underlying table structure.
- Consistency – The same query logic can be reused without rewriting it.
- Easy Maintenance – Changes can be made in one place instead of multiple queries.

Q : 44 Lab 1: Create a view to show all employees along with their department names.

- Lab 2: Modify the view to exclude employees whose salaries are below $50,000.

Ans :

- ```
  CREATE VIEW employee_department_view AS
  SELECT employees.employee_name,
         employees.salary,
         departments.department_name
  FROM employees
  INNER JOIN departments
  ON employees.department_id = departments.department_id;
  ```

- ```
  CREATE OR REPLACE VIEW employee_department_view AS
  SELECT employees.employee_name,
         employees.salary,
         departments.department_name
  FROM employees
  INNER JOIN departments
  ```

```
ON employees.department_id = departments.department_id
WHERE employees.salary >= 50000;
```

Q : 45 What is a trigger in SQL? Describe its types and when they are used.

Ans : A trigger in SQL is a special type of stored program that automatically executes (fires) when a specific event occurs on a table or view, such as an INSERT, UPDATE, or DELETE operation. Triggers are used to enforce business rules, maintain data integrity, and automate tasks.

Types of Triggers

- BEFORE Trigger
  Executes before the triggering event occurs. It is used to validate or modify data before it is saved.
- AFTER Trigger
  Executes after the event has occurred. It is commonly used for logging changes or updating related tables.
- INSTEAD OF Trigger
  Executes instead of the triggering operation. It is often used with views to control how data is inserted, updated, or deleted.

Triggers are useful when automatic actions are required without user intervention.

Q : 46 Explain the difference between INSERT, UPDATE, and DELETE triggers..

Ans :

- INSERT Trigger
  Fires automatically when a new record is added to a table. It is used for tasks like logging new entries or validating data.
- UPDATE Trigger
  Executes when an existing record is modified. It is used to track changes or update related data.
- DELETE Trigger
  Runs when a record is removed from a table. It is often used to maintain logs or enforce cascading rules.
  The main difference lies in the type of database operation that activates the trigger.

Q : 47 Lab 1: Create a trigger to automatically log changes to the employees table when a new employee is added.

• Lab 2: Create a trigger to update the last_modified timestamp whenever an employee record is updated.

Ans :

- ```
  CREATE TABLE employee_log (
      log_id INT AUTO_INCREMENT PRIMARY KEY,
      employee_id INT,
      action VARCHAR(20),
  ```
  ```

```
    action_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

- ```
  CREATE TRIGGER after_employee_insert
  AFTER INSERT ON employees
  FOR EACH ROW
  INSERT INTO employee_log (employee_id, action)
  VALUES (NEW.employee_id, 'INSERT');
  ```

- ```
  ALTER TABLE employees
  ADD last_modified TIMESTAMP;
  ```

- ```
  CREATE TRIGGER before_employee_update
  BEFORE UPDATE ON employees
  FOR EACH ROW
  SET NEW.last_modified = CURRENT_TIMESTAMP;
  ```

Q : 48 What is PL/SQL, and how does it extend SQL's capabilities?

Ans : PL/SQL (Procedural Language/SQL) is an extension of SQL developed by Oracle that combines SQL commands with procedural programming features such as variables, loops, conditions, and exception handling.

While SQL is used mainly for querying and manipulating data, PL/SQL allows developers to write complete programs that include logic and control structures. It enables complex operations like decision-making, iteration, and error handling directly within the database, making database applications more powerful and efficient.

Q : 49 List and explain the benefits of using PL/SQL.

Ans :

- Improved Performance – Multiple SQL statements can be executed in a single block, reducing communication between the application and database.
- Procedural Capabilities – Supports loops, conditions, and variables for complex logic.
- Error Handling – Provides exception handling to manage runtime errors effectively.
- Code Reusability – Procedures and functions can be reused multiple times.
- Enhanced Security – Business logic can be stored in the database, limiting direct access to tables.
- Modular Programming – Programs can be organized into blocks, procedures, and packages.

Q : 50 Write a PL/SQL block to print the total number of employees from the employees table.

- Lab 2: Create a PL/SQL block that calculates the total sales from an orders table.

Ans :
  ```

- ```
  DECLARE
      total_emp NUMBER;
  BEGIN
      SELECT COUNT(*) INTO total_emp FROM employees;
      DBMS_OUTPUT.PUT_LINE('Total Employees: ' || total_emp);
  END;
  ```

- ```
  DECLARE
      total_sales NUMBER;
  BEGIN
      SELECT SUM(order_amount) INTO total_sales FROM orders;
      DBMS_OUTPUT.PUT_LINE('Total Sales: ' || total_sales);
  END;
  ```

Q : 51 What are control structures in PL/SQL? Explain the IF-THEN and LOOP control structures.

Ans : Control structures in PL/SQL are programming constructs that control the flow of execution within a PL/SQL block. They allow the program to make decisions and repeat tasks based on specific conditions.

**IF-THEN Control Structure**

The IF-THEN statement is used to execute a block of code only when a specified condition is true. It helps in decision-making within a program.

**LOOP Control Structure**

A LOOP is used to execute a block of code repeatedly until a specified condition is met. PL/SQL supports different types of loops such as simple LOOP, WHILE LOOP, and FOR LOOP.

Q : 52 How do control structures in PL/SQL help in writing complex queries?

Ans : Control structures help manage complex database operations by allowing conditional logic and repeated execution of statements. They make it possible to process records one by one, apply different actions based on conditions, and automate repetitive tasks.

For example, IF-THEN statements can validate data before inserting it, while loops can process multiple rows or perform calculations. This improves flexibility, reduces manual effort, and enables the creation of dynamic and intelligent database programs.

Q : 53 Lab 1: Write a PL/SQL block using an IF-THEN condition to check the department of an employee.

- Lab 2: Use a FOR LOOP to iterate through employee records and display their names.

Ans :

- ```
  DECLARE
      dept_id NUMBER := 1;
      emp_dept NUMBER;
  ```
  ```

```
BEGIN
    SELECT department_id INTO emp_dept
    FROM employees
    WHERE employee_id = 1;

    IF emp_dept = dept_id THEN
        DBMS_OUTPUT.PUT_LINE('Employee belongs to the specified
department');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Employee belongs to a different
department');
    END IF;
END;
```

- ```
  BEGIN
      FOR emp_rec IN (SELECT employee_name FROM employees) LOOP
          DBMS_OUTPUT.PUT_LINE(emp_rec.employee_name);
      END LOOP;
  END;
  ```

Q : 54 What is a cursor in PL/SQL? Explain the difference between implicit and explicit cursors.

Ans : A cursor in PL/SQL is a pointer to a result set returned by a SQL query. It allows PL/SQL programs to retrieve and process rows one at a time instead of handling the entire result set at once.

Implicit Cursor

An implicit cursor is automatically created by Oracle whenever a SQL statement such as INSERT, UPDATE, DELETE, or SELECT INTO is executed. The user does not need to declare or manage it manually.

Explicit Cursor

An explicit cursor is declared by the programmer to retrieve multiple rows from a query. It must be opened, fetched, and closed manually, giving more control over row processing.

Difference:
Implicit cursors are automatic and simple, while explicit cursors provide more control and are used when processing multiple rows individually.

Q : 55 When would you use an explicit cursor over an implicit one?

Ans : An explicit cursor is used when a query returns multiple rows and you need to process each row individually. It is useful when applying logic to each record, performing row-by-row operations, or when greater control over the query execution is required.
  ```

Implicit cursors are suitable for simple operations, while explicit cursors are preferred for complex data processing tasks.

Q : 56 Lab 1: Write a PL/SQL block using an explicit cursor to retrieve and display employee details.

- Lab 2: Create a cursor to retrieve all courses and display them one by one.

Ans :

- ```
  DECLARE
      CURSOR emp_cursor IS
          SELECT employee_id, employee_name, salary FROM employees;

      emp_record emp_cursor%ROWTYPE;
  BEGIN
      OPEN emp_cursor;

      LOOP
          FETCH emp_cursor INTO emp_record;
          EXIT WHEN emp_cursor%NOTFOUND;

          DBMS_OUTPUT.PUT_LINE(
              emp_record.employee_id || ' - ' ||
              emp_record.employee_name || ' - ' ||
              emp_record.salary
          );
      END LOOP;

      CLOSE emp_cursor;
  END;
  ```

- ```
  DECLARE
      CURSOR course_cursor IS
          SELECT course_id, course_name FROM courses;

      course_rec course_cursor%ROWTYPE;
  BEGIN
      OPEN course_cursor;

      LOOP
          FETCH course_cursor INTO course_rec;
          EXIT WHEN course_cursor%NOTFOUND;

          DBMS_OUTPUT.PUT_LINE(
              course_rec.course_id || ' - ' || course_rec.course_name
          );
  ```

```
        END LOOP;

        CLOSE course_cursor;
    END;
```

Q : 57 Explain the concept of SAVEPOINT in transaction management. How do ROLLBACK and COMMIT interact with savepoints?

Ans : A SAVEPOINT is a point within a transaction that allows you to partially roll back changes instead of undoing the entire transaction. It acts like a checkpoint that you can return to if needed.

- ROLLBACK TO SAVEPOINT reverses all changes made after that specific savepoint but keeps the earlier changes intact.
- COMMIT saves all changes permanently and removes all savepoints created during the transaction.

In simple terms, savepoints provide more control over transactions by allowing selective undo operations.

Q : 58 When is it useful to use savepoints in a database transaction?

Ans : Savepoints are useful when a transaction involves multiple steps and you want the flexibility to undo only certain parts without cancelling the entire transaction.

They are commonly used when:

- Performing complex operations with multiple updates
- Handling errors in long transactions
- Testing partial changes before finalizing
- Maintaining data integrity during multi-step processes

Savepoints improve control, reduce risk, and make transactions more flexible.

Q : 59 Lab 1: Perform a transaction where you create a savepoint, insert records, then rollback to the savepoint.

- Lab 2: Commit part of a transaction after using a savepoint and then rollback the remaining changes.

Ans :

```
START TRANSACTION;
INSERT INTO courses VALUES (301, 'Cyber Security', '4 Months');
SAVEPOINT sp1;
INSERT INTO courses VALUES (302, 'Data Analytics', '5 Months');
ROLLBACK TO sp1;
```

```
COMMIT;
```

- ```
  START TRANSACTION;
  INSERT INTO courses VALUES (303, 'Artificial Intelligence', '6 Months');
  SAVEPOINT sp2;
  INSERT INTO courses VALUES (304, 'Big Data', '5 Months');
  ROLLBACK TO sp2;
  COMMIT;
  ```
  ```