**Batch: T2**

**Asssignment No: 0**

**Title of Assignment: Study of PL/SQL, T-SQL and Angular**

**Student Name: Shreeyash Shripad Dongarkar**

**Student PRN: 22510025**

**PL/SQL (Procedural Language/Structured Query Language)**

* An extension of SQL used in Oracle databases.
* Allows combining SQL with procedural programming features like loops, conditions, variables, and error handling, making it powerful for complex database operations.

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Hello, PL/SQL!');

END;

**1. Block Structure**

PL/SQL programs are organized into **blocks**, which can be **anonymous** or **named**. A block has three parts:

* **Declaration Section**: Defines variables, constants, and cursors.
* **Execution Section**: Contains the procedural code.
* **Exception Handling Section**: Handles runtime errors.

DECLARE

v\_message VARCHAR2(50) := 'Hello, PL/SQL!';

BEGIN

DBMS\_OUTPUT.PUT\_LINE(v\_message);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred!');

END;

**2. Procedural Constructs**

PL/SQL allows for procedural programming by including constructs such as:

* **Conditional Statements**: IF...THEN, ELSIF, CASE
* **Loops**: FOR, WHILE, and LOOP

**3. Error Handling**

PL/SQL provides robust error handling using the **EXCEPTION** block. Common exceptions include NO\_DATA\_FOUND, TOO\_MANY\_ROWS, and OTHERS

**4. Stored Procedures and Functions**

Reusable subprograms that encapsulate logic.

* **Procedure**: Performs an action but does not return a value.

CREATE OR REPLACE PROCEDURE greet\_user(name IN VARCHAR2) IS

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Hello, ' || name || '!');

END;

* **Function**: Returns a value.

CREATE OR REPLACE FUNCTION calculate\_bonus(salary IN NUMBER) RETURN NUMBER IS

BEGIN

RETURN salary \* 0.1;

END;

**5. Triggers**

A trigger automatically executes in response to certain events (e.g., INSERT, UPDATE, DELETE) on a table.

CREATE OR REPLACE TRIGGER log\_employee\_change

AFTER INSERT OR UPDATE ON employees

FOR EACH ROW

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Employee data changed.');

END;

**6. Cursors**

Used for processing query results row-by-row.

* **Implicit Cursor**: Automatically managed for single queries.
* **Explicit Cursor**: Used for more control over query processing.

DECLARE

CURSOR emp\_cursor IS SELECT name FROM employees;

v\_name employees.name%TYPE;

BEGIN

OPEN emp\_cursor;

LOOP

FETCH emp\_cursor INTO v\_name;

EXIT WHEN emp\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(v\_name);

END LOOP;

CLOSE emp\_cursor;

END;

**7. Modular Programming**

Supports the creation of packages containing related procedures, functions, and variables.

CREATE OR REPLACE PACKAGE employee\_pkg IS

PROCEDURE add\_employee(name VARCHAR2, salary NUMBER);

FUNCTION get\_employee\_salary(id NUMBER) RETURN NUMBER;

END employee\_pkg;

**Stored Procedures and Procedural Constructs in MySQL**

**1. Stored Procedures**

A **stored procedure** is a precompiled collection of SQL statements and procedural logic that can be executed as a single unit.

**Features:**

* Encapsulates business logic for reusability.
* Improves performance by reducing client-server communication.
* Accepts parameters to make procedures dynamic.

DELIMITER //

CREATE PROCEDURE add\_numbers (IN num1 INT, IN num2 INT, OUT result INT)

BEGIN

SET result = num1 + num2;

END //

DELIMITER ;

-- Calling the procedure:

CALL add\_numbers(10, 20, @output);

SELECT @output; -- Displays 30

**2. Procedural Constructs in MySQL**

MySQL provides procedural constructs to handle control flow and logic within stored procedures or functions:

**a. Variables**: Store data for use within a stored procedure or function.

DECLARE total INT;

SET total = 100;

**b. Conditional Statements**

* **IF...THEN...ELSE**

**c. Loops**

MySQL supports several types of loops:

* **LOOP**
* **WHILE**
* **REPEAT UNTIL**

**d. Cursors:** Cursors allow row-by-row processing of query results.

DECLARE cursor\_name CURSOR FOR SELECT column\_name FROM table\_name;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cursor\_name;

REPEAT

FETCH cursor\_name INTO variable\_name;

-- Process each row

UNTIL done END REPEAT;

CLOSE cursor\_name;

**T-SQL (Transact-SQL)**

Microsoft's proprietary extension of SQL (Structured Query Language) used with Microsoft SQL Server and Azure SQL Database. It adds procedural programming, error handling, and transaction control capabilities to standard SQL.

**Key Features of T-SQL**

1. **Procedural Programming:**
   * Adds constructs like variables, loops, and conditionals.
   * Enables creating complex and reusable logic within the database.
2. **Error Handling:**
   * Provides structured error handling with TRY...CATCH.
3. **Transaction Control:**
   * Offers commands like BEGIN TRANSACTION, COMMIT, and ROLLBACK for managing transactions.
4. **Built-in Functions:**
   * Includes system functions for string manipulation, date handling, and more (e.g., GETDATE(), LEN()).
5. **Stored Procedures and Functions:**
   * Enables encapsulating logic in reusable stored procedures and functions.
6. **Triggers:**
   * Automatically execute code in response to INSERT, UPDATE, or DELETE operations.
7. **Cursors:**
   * Supports row-by-row processing of query results.

**Use Cases**

* Writing complex database logic.
* Automating tasks like data validation or report generation.
* Building triggers for real-time data monitoring.
* Optimizing database performance by reducing client-server interactions.

T-SQL is integral to building robust, efficient, and scalable applications with Microsoft SQL Server.

**Angular**

Angular is a TypeScript-based, open-source web application framework developed by Google. It is used for building dynamic, single-page applications (SPAs) with a focus on scalability, maintainability, and modularity.

Features of Angular

1. Two-Way Data Binding: Synchronizes data between the model and view automatically.

2. Component-Based Architecture: Applications are built using reusable, self-contained components.

3. Dependency Injection (DI): Manages services efficiently, promoting code modularity.

4. Directives: Extend HTML functionality (e.g., \*ngIf, \*ngFor, etc.).

5. Routing: Enables navigation between views in single-page applications.

6. RxJS Integration: Facilitates reactive programming and handling asynchronous data streams.

7. Ahead-of-Time (AOT) Compilation: Improves performance by compiling HTML and TypeScript at build time.

8. Modular Structure: Applications are divided into modules, making them scalable and maintainable.

9. CLI Support: Angular CLI simplifies project setup, builds, and testing.

10. Cross-Platform: Supports progressive web apps, native apps, and desktop apps.

How Angular Works

1. Bootstrapping: The Angular app initializes with a main.ts file, bootstrapping the root module (e.g., AppModule). Components are rendered within the root component (AppComponent).

2. Modules: Modules group components, directives, and services (e.g., NgModule).

3. Components: Defined using a TypeScript class, an HTML template, and CSS styles.

Components control specific parts of the UI.

4. Templates and Directives: Templates define the structure of the UI.

Directives add behavior to the DOM (e.g., structural and attribute directives).

5. Data Binding: Synchronizes the model and view through property binding, event binding, and two-way binding.

6. Dependency Injection: Provides instances of services to components.

7. Routing: The RouterModule handles navigation, enabling dynamic view rendering based on URLs.

8. Change Detection: Detects changes in the application state and updates the DOM automatically.

9. Build Process: The Angular CLI compiles TypeScript, processes templates, and generates optimized JavaScript bundles.