# DBMS Practical 10

## Aim : Study of PL/SQL

### Theory

PL/SQL (Procedural Language/Structured Query Language) is a powerful procedural extension to the SQL (Structured Query Language) used for database management systems. It combines the capabilities of SQL with procedural programming constructs such as loops, conditionals, variables, and exception handling, allowing developers to write complex and modular code to manipulate and control data within a database. It is often used to create stored procedures, functions, triggers, and packages that can be executed within the database.

Here are some key features and concepts of PL/SQL:

1. **Blocks:** PL/SQL programs are organized into blocks, which are made up of declarations, executable statements, and exception handlers. Blocks can be nested within each other, and they serve as the basic unit of code in PL/SQL.
2. **Variables:** PL/SQL allows the declaration of variables of different data types, such as numbers, strings, dates, and Boolean values. Variables are used to store and manipulate data within the program.
3. **Control structures:** PL/SQL provides various control structures like loops (e.g., `FOR`, `WHILE`), conditionals (e.g., `IF-THEN-ELSE`), and branching statements (e.g., `GOTO`). These structures enable developers to control the flow of execution based on conditions and perform repetitive tasks.
4. **Cursors:** Cursors in PL/SQL are used to fetch and process multiple rows returned by a SQL query. They allow developers to retrieve and manipulate data row by row from result sets.
5. **Exception handling:** PL/SQL has robust exception handling mechanisms to catch and handle runtime errors. Developers can define exception handlers to handle specific exceptions or use the built-in exception handlers to handle common errors.
6. **Procedures and functions:** PL/SQL allows the creation of reusable program units called procedures and functions. Procedures are blocks of code that can be called and executed, while functions return a value and can be used within SQL statements.
7. **Triggers:** PL/SQL triggers are special program units associated with database tables. They are automatically executed in response to specific events such as data manipulation (e.g., `INSERT`, `UPDATE`, `DELETE`) or database operations (e.g., database startup, shutdown).
8. **Packages:** Packages are a way to organize and encapsulate related PL/SQL objects, such as procedures, functions, types, and variables, into a single unit. They provide modularity, reusability, and better code management.

PL/SQL offers a robust and efficient means of writing powerful database applications. It allows developers to leverage SQL for data manipulation while providing the flexibility and control of a procedural programming language, making it well-suited for developing complex business logic and data-centric applications.

```
DECLARE
  total_salary NUMBER := 0;
  employee_count NUMBER := 0;
  average_salary NUMBER := 0;
BEGIN
  -- Calculate total salary and count employees
  FOR employee IN (SELECT salary FROM employees) LOOP
    total_salary := total_salary + employee.salary;
    employee_count := employee_count + 1;
  END LOOP;

  -- Calculate average salary
  IF employee_count > 0 THEN
    average_salary := total_salary / employee_count;
  END IF;

  -- Display the result
  DBMS_OUTPUT.PUT_LINE('Total employees: ' || employee_count);
  DBMS_OUTPUT.PUT_LINE('Average salary: ' || average_salary);
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/
```

In this example, we declare three variables: `total_salary`, `employee_count`, and `average_salary`.

The code block uses a `FOR` loop to iterate over the result set of a SQL query that retrieves the salaries from the "employees" table. Inside the loop, it accumulates the total salary and increments the employee count.

After the loop, it calculates the average salary by dividing the total salary by the number of employees, but only if the employee count is greater than zero.

Finally, the result is displayed using the `DBMS_OUTPUT.PUT_LINE` procedure. Exception handling is included to catch any errors that may occur during execution.

Note that to run this code, you would typically execute it in a PL/SQL environment such as Oracle SQL*Plus or Oracle SQL Developer.