	Subject :		Software :	
			Hardware :	
	Branch :	Semester :	Page No. 75	Prog No. 07

PROBLEM STATEMENT: Introduction to PL/SQL & PL/SQL Stored procedures and functions

ALGORITHM & CODE :

What is PL/SQL ?

PL/SQL (Procedural language /Structured Query Language) is a Procedural extension of SQL developed by Oracle Corporation. It is used to write Procedural scripts, functions and triggers that run inside an Oracle database.

PL/SQL combines SQL with procedural features like loops, conditions, and exception handling, making it a powerful and efficient programming languages for database operations.

Feature of PL/SQL

1. Block-Structured language - PL/SQL Programs are written in blocks for better readability and maintainability
2. Procedural Capabilities - Supports loops, conditional Statements (IF, CASE) and exception handling.
3. Supports SQL: You can directly executed SQL Statements within PL/SQL.

INPUT GIVEN		
OUTPUT OBTAINED		
REMARKS		
GRADE :	Signature of Faculty Date :	Signature of Student Date :

	Subject :		Software :	
			Hardware :	
	Branch :	Semester :	Page No. 76	Prog No. 07

PROBLEM STATEMENT

ALGORITHM & CODE :

4. High Performance - Reduces network traffic and improves performance by executing multiple multiple SQL statements at once.

5. Error Handling - Uses EXCEPTION blocks to manage runtime errors.

6. portability - Runs on any system that supports Oracle Database.

7. Security - offers security through privileges and stored procedures.

PL/SQL Block Structures

A PL/SQL Program consists of four Sections:

1. Declaration Section (optional)
variables, constants, cursors, and user-defined types are declared here.

2. Execution Section (mandatory)

The main logic, including SQL statements, loops, and conditional statements.

3. Exception Handling Section (optional)

Handles errors (like division by zero, no data found, etc.).

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :	Signature of Faculty Date :	Signature of Student Date :

Subject :		Software :	
		Hardware :	
Branch :	Semester :	Page No. 77	Prog No. 07

EM STATEMENT

RITHM & CODE :

4. End Statements

Every PL/SQL block ends with END;

Basic PL/SQL Block Example

```
DECLARE
   message VARCHAR2(50);
BEGIN
    message := 'Hello, PL/SQL!';
    DBMS_OUTPUT.PUT_LINE(message);
 END;
 /
```

PL/SQL Control Structures

PL/SQL Supports Conditional and iterative Statements.

Conditional Statements

IF - THEN - ELSE

```
DECLARE
   num NUMBER := 10;
BEGIN
   IF num > 0 THEN
   DBMS_OUTPUT.PUT_LINE ('Positive Number');
```

INPUT GIVEN	
OUTPUT OBTAINED	
REMARKS	
GRADE :	Signature of Faculty Date :	Signature of Student Date :

	Subject :	Software :	
		Hardware :	
	Branch :	Semester :	Page No. 78	Prog No. 07

PROBLEM STATEMENT

ALGORITHM & CODE :

```
ELSE
  DBMS_OUTPUT.PUT_LINE ('Negative Number');
END IF;
END;
/
```

CASE Statement

```
DECLARE
   grade CHAR(1):= 'A';
BEGIN
  CASE grade
     WHEN 'A' THEN DBMS_OUTPUT.PUT_LINE ('Excellent');
     WHEN 'B' THEN DBMS_OUTPUT.PUT_LINE ('Good');
     ELSE DBMS_OUTPUT.PUT_LINE ('Needs Improvement');
  END CASE;
END;
/
```

Loops in PL/SQL

FOR LOOP

```
BEGIN
  FOR i IN 1.5 LOOP
    DBMS_OUTPUT.PUT_LINE ('Iteration :'|| i);
```

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE : | Signature of Faculty Date : | Signature of Student Date :

Subject :

Software :

Hardware :

Branch :

Semester :

Page No. 79

Prog No. 07

PROBLEM STATEMENT

ALGORITHM & CODE :

```
  ENDLOOP;
END;
/
```

WHILE LOOP

```
DECLARE
   num NUMBER := 1;
BEGIN
   WHILE num <= 5 LOOP
     DBMS_OUTPUT.PUT_LINE ('Number: ' || num);
      num := num + 1;
   END LOOP;
END;
/
```

PL/SQL Enception Handling

PL/SQL Provides a way to handle errors using the EXCEPTION block.

Enample of Enception Handling:

```
DECLARE
   num NUMBER := 10;
   denom NUMBER := 0;
   result NUMBER;
```

NPUT GIVEN

UTPUT OBTAINED

EMARKS

RADE :

Signature of Faculty

Date :

Signature of Student

Date :

Subject :	Software :
	Hardware :
Branch :	Semester :
Page No. 80	Prog No. 07

PROBLEM STATEMENT

ALGORITHM & CODE :

```
BEGIN
    result := num/denom; - Division by zero Error
EXCEPTION
    WHEN ZERO_DIVIDE THEN
        DBMS_OUTPUT.PUT_LINE ('Error: Cannot divide
by zero');
END;
/
```

PL/SQL Stored Procedures & functions

FUNCTION:

A function is a subprogram that computes a value.

Syntax:

```
CREATE [OR REPLACE] FUNCTION function_name (Parameter
list) RETURN return_type
IS / AS
-- Variable declarations (optional)
BEGIN
-- PL/SQL Statements
RETURN return_value;
EXCEPTION
-- Exception handling (optional)
```

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :

Signature of Faculty
Date :

Signature of Student
Date :

Subject :			Software :	
			Hardware :	
Branch :	Semester :		Page No. 81	Prog No. 07

PROBLEM STATEMENT

ALGORITHM & CODE :

WHEN exception-name THEN
-- Exception handling statements
END function-name;

Components

1. CREATE [OR REPLACE] FUNCTION function-name:
   - CREATE : specifies that you are creating a new function.
   - OR REPLACE : optional. If the function already exists, this clause allows you to replace it with a new definition.
   - function-name : The name of the function.

2. (Parameter-list) :
   - A comma-separated list of parameters.
   - Each parameter has a name and a data type. Parameters can be IN, OUT or IN OUT.
   - Example : (P_Param1 IN NUMBER, P_Param2 OUT VARCHAR2)

PUT GIVEN

TPUT OBTAINED

EMARKS

RADE :	Signature of Faculty Date :	Signature of Student Date :

Subject :		Software :	
		Hardware :	
Branch :	Semester :	Page No. 82	Prog No. 07

PROBLEM STATEMENT

ALGORITHM & CODE :

3. RETURN return-type :

• Specifies the data type of the value that the function will return.

• Example: RETURN NUMBER , RETURN VARCHAR2

4. IS or AS:

• Marks the beginning of the function body. Both keywords can be used interchangeably.

5. Variable Declarations (optional):

• Local Variables can be declared here if needed.

6. BEGIN ... END; :

• The executable Part of the function where the main logic is written.

• The RETURN return-value; Statement specifies the value that will be returned by the function.

7. Exception:

• optional section to handle exceptions or errors that occur during execution.

8. END function_name;:

• Ends the function definition and Specifies the function

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :	Signature of Faculty Date :	Signature of Student Date :

Subject :		Software :
		Hardware :
Branch :	Semester :	Page No. 83 | Prog No. 07

PROBLEM STATEMENT

ALGORITHM & CODE :

name again.

Example

1. Write a PL/SQL Program to implement a function that calculates the square of a number:

```
CREATE OR REPLACE FUNCTION Square_number(P_num IN
NUMBER) RETURN NUMBER
IS
V_result NUMBER;
BEGIN
V_result := P_num * P_num;
RETURN V_result;
END square_number;
SELECT square_number(5) AS Square FROM dual;
```

2. Write a Program to add two numbers using function.

```
CREATE OR REPLACE FUNCTION AddNumbers(
    P_num1 IN NUMBER,
    P_num2 IN NUMBER
) RETURN NUMBER
AS
BEGIN
 RETURN P_num1 + P_num2;
END;
```

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :	Signature of Faculty Date :	Signature of Student Date :

	Subject :		Software :	
			Hardware :	
	Branch :	Semester :	Page No. 84	Prog No. 07
PROBLEM STATEMENT				

ALGORITHM & CODE :

```
SELECT AddNumbers (5,10) AS Sum FROM dual;
```

3. Write a Program to find maximum number between two numbers using function.

```
CREATE OR REPLACE FUNCTION maxNumber(
    P_num1 IN NUMBER,
    P_num2 IN NUMBER
) RETURN NUMBER
AS
BEGIN
    IF P_num1 > P_num2 THEN
        RETURN P_num1;
    ELSE
        RETURN P_num2;
    END IF;
END;
SELECT maxNumber(5,10) AS MaxValue FROM dual;
```

This will return 10 as the result.

4. Write a Program to calculate the factorial of a number using function.

```
CREATE OR REPLACE FUNCTION Factorial(
    P_num IN NUMBER
```

INPUT GIVEN	
OUTPUT OBTAINED	
REMARKS	
GRADE :	Signature of Faculty Date : / Signature of Student Date :

Subject :			Software :	
			Hardware :	
Branch :	Semester :		Page No. 85	Prog No. 07

PROBLEM STATEMENT

ALGORITHM & CODE :

```
) RETURN NUMBER
AS
BEGIN
    IF P_num = 0 THEN
        RETURN 1;
    ELSE
      RETURN P_num * Factorial (P_num-1);
    END IF;
END;
SELECT factorial (5) AS factorialResult FROM dual;
```

PROCEDURE

In PL/SQL, a Procedure is a subprogram that performs a specific task but does not return a value. Procedures can be used to execute a set of SQL statements or PL/SQL code that performs operations such as modifying data, managing transactions, or controlling the flow of execution.

Syntax of a Procedure

```
    CREATE [OR REPLACE] PROCEDURE Procedure_name
(Parameter_list)
IS
```

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :

Signature of Faculty	Signature of Student
Date :	Date :

Subject :			Software :	
			Hardware :	
Branch :	Semester :		Page No. 86	Prog No. 07
PROBLEM STATEMENT				

ALGORITHM & CODE :

```
    -- Variable declarations (optional)
BEGIN
    -- PL/SQL Statements
EXCEPTION
    -- Exception handling (optional)
  WHEN exception_name THEN
    -- Exception handling statements
END procedure_name;
```

Components

1. CREATE [OR REPLACE] PROCEDURE procedure_name :
   - CREATE : Indicates that you are creating a new procedure.
   - OR REPLACE : optional. If the procedure already exists, this clause allows you to replace it with a new definition.
   - Procedure_name : The name of the procedure.

2. (Parameter_list):
   - A comma-separated list of parameter.
   - Each parameter has a name, data type, and mode (IN, OUT, or IN OUT):
     - IN : The parameter is used for input to the procedure.

INPUT GIVEN	
OUTPUT OBTAINED	
REMARKS	
GRADE :	Signature of Faculty Date :
	Signature of Student Date :

Subject :		Software :	
		Hardware :	
Branch :	Semester :	Page No. 87	Prog No. 07
PROBLEM STATEMENT			

ALGORITHM & CODE :

~~OUT~~

• OUT: The Parameter is used to return a value from the Procedure.

• IN OUT : The Parameter is used for both input and output.

3. IS or AS :

• marks the beginning of the Procedure body. Both keywords can be used interchangebly.

4. Variable Declarations (optional):

• local variables can be declared here if needed.

5. BEGIN ... END; :

• The executable Part of the Procedure where the main logic is written.

6. EXCEPTION:

• optional section to handle exceptions or errors that occur during execution.

7. END Procedure-name;:

• Ends the Procedure definition and specifies the Procedure name again.

INPUT GIVEN		
OUTPUT OBTAINED		
REMARKS		
GRADE :	Signature of Faculty Date :	Signature of Student Date :

Subject :	Software :
	Hardware :
Branch :	Semester :
Page No. 88	Prog No. 07

PROBLEM STATEMENT

ALGORITHM & CODE :

1. Create a Procedure to find whether a given number is odd or even

```
  CREATES OR REPLACE PROCEDURE CheckOddOrEven
(P_num IN NUMBER) AS
BEGIN
    IF P_num  IS NULL THEN
      DBMS_OUTPUT.PUT_LINE ('The number is NULL.');
   ELSEIF MOD (P_num,2) = 0 THEN
      DBMS_OUTPUT.PUT_LINE ('The number' || P_num ||
'is even.');
     ELSE
       DBMS_OUTPUT.PUT_LINE ('The number' || P_num ||
'is odd.');
       END IF;
END;
```

To Execute the Procedure, you can use the following ~~bl~~ block:

```
BEGIN
   CheckOddOrEven (5);
END;
```

Output: The number is Odd.

INPUT GIVEN	
OUTPUT OBTAINED	
REMARKS	
GRADE :	Signature of Faculty Date :
	Signature of Student Date :

Subject :		Software :	
		Hardware :	
Branch :	Semester :	Page No. 89	Prog No. 07

PROBLEM STATEMENT

ALGORITHM & CODE :

2) Create a Procedure to display 1-10 using While loop

```
CREATE OR REPLACE PROCEDURE DisplayNumbers AS
 V_num NUMBER := 1; -- Initialize a Variable to store
the current number
BEGIN
    WHILE V_num <= 10
    LOOP
       DBMS.OUTPUT.PUT_LINE (V_num); -- Output the
Current numberto
       V.num := V_num + 1; -- Increment the number
 END LOOP;
END;
```

How to execute the Procedure, you can use the following command:

```
BEGIN
   DisplayNumbers;
END
```

INPUT GIVEN	
OUTPUT OBTAINED	
REMARKS	
GRADE :	Signature of Faculty Date :
	Signature of Student Date :

Subject :	Software :
	Hardware :

Branch :	Semester :	Page No. 90	Prog No. 07

PROBLEM STATEMENT

ALGORITHM & CODE : 3/ Write a Program to create a table and insert the records to a table by using Procedure.

First, you create the table using a simple CREATE TABLE statement:

```
CREATE TABLE Employees(
    EmployeeID NUMBER(5) PRIMARY KEY,
    FirstName VARCHAR2(50),
     LastName VARCHAR2(50),
     HireDate DATE
);
```

Next, you create a stored Procedure to insert records into the Employees table.

```
CREATE OR REPLACE PROCEDURE InsertEmployee(
    P_EmployeeID IN NUMBER,
     P_FirstName IN VARCHAR2,
     P_LastName IN VARCHAR2,
      P_HireDate IN DATE
)
AS
BEGIN
```

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :

Signature of Faculty

Date :

Signature of Student

Date :

	Subject :		Software :	
			Hardware :	
	Branch :	Semester :	Page No.	Prog No. 07

PROBLEM STATEMENT

ALGORITHM & CODE :

```
      INSERT INTO Employees (EmployeeID, FirstName,
LastName, HireDate)
      VALUES (P_EmployeeID, P_FirstName, P_LastName, P_HireDate);
COMMIT; -- commit the transaction
END;

Execute the Procedure

BEGIN
   Insert Employee (
        P_EmployeeID => 1,
        P_FirstName  => 'John',
        P_lastName   => 'Doo',
        P_HireDate   => TO_DATE ('2024 - 07-21',
 'YYYY-MM-DD')
 );
END;
```

INPUT GIVEN

OUTPUT OBTAINED

REMARKS

GRADE :	Signature of Faculty Date :	Signature of Student Date :