Assignment No 5:

Create a package for mathematical power like square,cube of given number.

```
Create or replace package math_power_pkg as
   Function square(n in number) return number;
   Function cube(n in number) return number;
   Function power(n in number, exp in number) return number;
End math_power_pkg;
/
```

Create package body

```
Create or replace package body math_power_pkg as

   Function square(n in number) return number is
   Begin
      Return n * n;
   End square;

   Function cube(n in number) return number is
   Begin
      Return n * n * n;
   End cube;

   Function power(n in number, exp in number) return number is
   Begin
      Return power(n, exp);
   End power;

End math_power_pkg;
/
```

OUTPUT

```
Select math_power_pkg.square(4) from dual; --16
Select math_power_pkg.cube(3) from dual; --27
Select math_power_pkg.power(2, 5) from dual; --32
```

Create package which contain procedure, function, and cursor for making student
Marks sheet(use std ,marks table).

Create a package for private object.

Package spec

```
Create or replace package private_pkg as
```

```
   Procedure public_proc;
   Function public_func return number;
End private_pkg;
/
```

Package body

```
Create or replace package body private_pkg as
```

Private variable

```
   Private_var number := 50;
```

Private procedure

```
   Procedure private_proc is
   Begin
      Dbms_output.put_line('private procedure');
   End private_proc;
```

Public procedure

```
   Procedure public_proc is
   Begin
      Private_proc;  -- calling private procedure
      Dbms_output.put_line('public procedure');
   End public_proc;
```

Public function

```
   Function public_func return number is
   Begin
      Return private_var;  -- returning private variable
   End public_func;

End private_pkg;
/
```

OUTPUT

Call public procedure

```
Begin
   Private_pkg.public_proc;
End;
/
```

```
Private procedure
Public procedure
```

Call public function

```
Declare
   Result number;
Begin
   Result := private_pkg.public_func;
   Dbms_output.put_line(‘result: ‘ || result);
End;
/

Result: 50
```

4)Define cursor for display information of Employee.

```
Declare
Declare the cursor to fetch employee details
   Cursor emp_cursor is
       Select employee_id, employee_name, department, salary
       From employees;

Declare variables to hold the fetched data
   V_employee_id employees.employee_id%type;
   V_employee_name employees.employee_name%type;
   V_department employees.department%type;
   V_salary employees.salary%type;
Begin
Open the cursor
   Open emp_cursor;

Fetch and process each row
   Loop
       Fetch emp_cursor into v_employee_id, v_employee_name, v_department,
v_salary;

Exit the loop when no more rows are found
       Exit when emp_cursor%notfound;

Display employee details
       Dbms_output.put_line(‘employee id: ‘ || v_employee_id ||
                            ‘, name: ‘ || v_employee_name ||
                            ‘, department: ‘ || v_department ||
                            ‘, salary: ‘ || v_salary);
   End loop;

Close the cursor after processing all rows
   Close emp_cursor;
End;
/
```

OUTPUT

```
Employee id: 101, name: suhas, department: CSE, salary: 5000
Employee id: 102, name: Pratuuu, department: CSE, salary: 6000
```

5)Define a cursor which can accept deptid as parameter and display department name.

```
Declare
   Cursor dept_cursor(p_deptid number) is
      Select department_name
      From departments
      Where department_id = p_deptid;
   V_department_name departments.department_name%type;
Begin

   Open dept_cursor(10);
   Fetch dept_cursor into v_department_name;
   Dbms_output.put_line('department name: ' || v_department_name);
   Close dept_cursor;
End;
/
```

OUTPUT

```
Department name: SELS
```

6)Define a cursor within cursor which can show student Information(rollno,name,address,stream) and marks(3 subject marks,percentage).

```
DECLARE

   V_rollno     students.rollno%TYPE;
   V_name       students.name%TYPE;
   V_address    students.address%TYPE;
   V_stream     students.stream%TYPE;


   V_subject1   marks.subject1_marks%TYPE;
   V_subject2   marks.subject2_marks%TYPE;
   V_subject3   marks.subject3_marks%TYPE;
   V_percentage NUMBER;


   CURSOR student_cursor IS
      SELECT rollno, name, address, stream
      FROM students;
   CURSOR marks_cursor (p_rollno IN students.rollno%TYPE) IS
      SELECT subject1_marks, subject2_marks, subject3_marks
      FROM marks
      WHERE rollno = p_rollno;

BEGIN

   OPEN student_cursor;


   LOOP
      FETCH student_cursor INTO v_rollno, v_name, v_address, v_stream;
      EXIT WHEN student_cursor%NOTFOUND;
      OPEN marks_cursor(v_rollno);

      FETCH marks_cursor INTO v_subject1, v_subject2, v_subject3;
      IF marks_cursor%FOUND THEN
         V_percentage := (v_subject1 + v_subject2 + v_subject3) / 3;


         DBMS_OUTPUT.PUT_LINE(‘rollno: ‘ || v_rollno);
         DBMS_OUTPUT.PUT_LINE(‘name: ‘ || v_name);
         DBMS_OUTPUT.PUT_LINE(‘address: ‘ || v_address);
         DBMS_OUTPUT.PUT_LINE(‘stream: ‘ || v_stream);
         DBMS_OUTPUT.PUT_LINE(‘subject 1 marks: ‘ || v_subject1);
         DBMS_OUTPUT.PUT_LINE(‘subject 2 marks: ‘ || v_subject2);
         DBMS_OUTPUT.PUT_LINE(‘subject 3 marks: ‘ || v_subject3);
         DBMS_OUTPUT.PUT_LINE(‘percentage: ‘ || v_percentage);
      END IF;


      CLOSE marks_cursor;
   END LOOP;


   CLOSE student_cursor;
END;
/
```

OUTPUT

```
Rollno: 1
Name: pratuuu
Address: sonand
Stream: sci
Subject 1 marks: 90
Subject 2 marks: 85
Subject 3 marks: 80
```

Percentage: 85

7) Find employee second higest salary of employee using cursor.

```
DECLARE
   CURSOR Emp_Salary_Cursor IS SELECT DISTINCT salary FROM employees
ORDER BY
Salary DESC;
   Emp_salary NUMBER;
   Counter NUMBER := 0;
BEGIN
   OPEN Emp_Salary_Cursor;
   LOOP
      FETCH Emp_Salary_Cursor INTO emp_salary;
      EXIT WHEN Emp_Salary_Cursor%NOTFOUND;
      Counter := counter + 1;
      IF counter = 2 THEN
         DBMS_OUTPUT.PUT_LINE(‘Second Highest Salary: ‘ || emp_salary);
         EXIT;
      END IF;
   END LOOP;
   CLOSE Emp_Salary_Cursor;
END;
/

CURSOR Emp_Salary_Cursor IS
   SELECT DISTINCT salary
   FROM employees
   ORDER BY salary DESC;
```

OUTPUT

8) Create trigger for avoiding inserting the records whose address ‘solapur’ and deleting the records
Whose address ‘satara’.(use emp table with address field).

```
CREATE OR REPLACE TRIGGER prevent_solapur_insert
BEFORE INSERT ON emp
FOR EACH ROW
BEGIN
   IF :NEW.address = ‘solapur’ THEN
      RAISE_APPLICATION_ERROR(-20001, ‘Cannot insert record with address
“solapur”.’);
   END IF;
END;
/
```

OUTPUT

 Cannot insert record with address “Solapur”.

```
CREATE OR REPLACE TRIGGER delete_satara_records
AFTER INSERT OR UPDATE ON emp
FOR EACH ROW
BEGIN
        IF :NEW.address = ‘satara’ THEN
        DELETE FROM emp WHERE employee_id = :NEW.employee_id;
    END IF;
END;
/
```

OUTPUT

9) create Trigger which not allow to insert Negative employee number

```
Create or replace trigger prevent_negative_number
Before insert on employees
For each row
Begin
  If :new.EMPLOYEE_ID < 0 then
    Raise_application_error(-20001, ‘employee number cannot be negative’);
  End if;
End;
/
```

OUTPUT

Employee number cannot be negative

10) create a trigger which can store all ddl operation of databases.

```
Create or replace trigger ddl_trigger
After ddl on SCHEMA
Begin
Insert into ddl_opt values(
Sysdate ,
Sys_context(‘USERENV’,’current_user’),
Ora_dict_obj_type,
Ora_dict_obj_name,
Ora_sysevent);
End;
```

```
/
```

11) Create a trigger audit for emp table

```
Create or replace trigger trig_audit
After INSERT OR DELETE OR UPDATE ON std
For each row
Declare
A varchar2(15);
Begin
If INSERTING THEN
A:='INSERT';
Elsif DELETING THEN
A:='DELETE';
Elsif UPDATING THEN
A:='UPDATE';
End if;
Insert into log_table1 values(
My1.nextval,
Systimestamp,
a);
end;
```

12) Write a PL/SQL block which can handle zero_divide exception.

```
Declare
   N   number := 10;
   D  number := 10;
   R      number;
Begin
   R := n / d;
   Dbms_output.put_line('result: ' || r);
Exception
   When zero_divide then
       Dbms_output.put_line('error: division by zero occurred.');
 End;
/
```

OUTPUT

```
Result: 1
```

13) Write a Pl/SQL block which can accept emp id and show it's name.If that emp id is NOT Exists then
Give appropriate exceptions.

```
Declare
```

```
   Emp_id      number := &emp_id;
   Emp_name    varchar2(100);
Begin
   Select employee_name into emp_name
   From employees
   Where employee_id = emp_id;

   Dbms_output.put_line(‘Employee Name: ‘ || emp_name);

Exception
   When no_data_found then
      Dbms_output.put_line(‘Error: Employee ID ‘ || emp_id || ‘ does not
exist.’);
   When others then
      Dbms_output.put_line(‘An unexpected error occurred: ‘ || sqlerrm);
End;
/
```

OUTPUT

```
Enter value for emp_id: 101
Old   2:     emp_id      number := &emp_id;
New   2:     emp_id      number := 101;
Employee Name: suhas
```

14) Write a PL/SQL block which can use raise_application_error();

```
Declare
   X number := 0;
Begin
   If x = 0 then
      Raise_application_error(-20001, ‘Value cannot be zero’);
   End if;
End;
/
```

OUTPUT

```
Value cannot be zero
```

15) Write a PL/SQL block which can use pragma exception

```
Declare
   E_custom exception;
   Pragma exception_init(e_custom, -20001);
Begin
   Raise e_custom;
```

```
Exception
   When e_custom then
      Dbms_output.put_line('custom error');
End;
/
```

OUTPUT

```
Custom error
```

16) Write a PL/SQL clock which can use defined exception.

```
Declare
   E1 exception;
   Emp_id number := 0;
Begin
   If emp_id = 0 then
      Raise e1;
   End if;
Exception
   When e1 then
      Dbms_output.put_line('Error: ID cannot be zero');
End;
/
```

OUTPUT

```
Error: ID cannot be zero
```