Part A

Create, compile, and execute a PL/SQL procedure from the Oracle SQL * Plus
Create and PL/SQL functions.

Aim:

- Create, compile, and execute a PL/SQL procedure.
- To understand %ROWTYPE Attribute
- Learn and implement PL/SQL functions.

Prerequisite: Oracle, SQL

Outcome: Students will be able to write PL/SQL procedures

Theory:

- A PL/SQL procedure is a named block stored
  - It is a reusable unit
- The basic syntax of creating a procedure in PL/SQL is as follows:
  -
```
CREATE [OR REPLACE ] PROCEDURE procedure_name (parameter_list)
IS
    [declaration statements]
BEGIN
    [execution statements]
    EXCEPTION
        [exception handler]
END [procedure_name ];
```
- Note that OR REPLACE option allows you to overwrite the current procedure with the new code.
- PL/SQL procedures have to parts
  - PL/SQL procedure header
    - It specifies procedure name and an optional parameter list.
  - PL/SQL procedure body
    - **Declarative part**
    - **Executable part**
    - **Exception-handling part**
- A procedure begins with a header that specifies its name and an optional parameter list.
- Parameters can in one of the following modes:
  - In mode
  - Out mode
  - Inout mode
- An IN parameter is read-only.
  - You can reference an IN parameter inside a procedure, but you cannot change its value.
  - Oracle uses IN as the default mode.
    - It means that if you don't specify the mode for a parameter explicitly, Oracle will use the IN mode.
- An OUT parameter is writable.

  - Typically, you set a returned value for the OUT parameter and return it to the calling program.
  - Note that a procedure ignores the value that you supply for an OUT parameter.
- An INOUT parameter is both readable and writable.
  - The procedure can read and modify it.

%ROWTYPE Attribute

- The %ROWTYPE attribute provides a record type that represents a row in a database table.
- The record can store an entire row of data selected from the table or fetched from a cursor or cursor variable.
- Fields in a record and corresponding columns in a row have the same names and datatypes.
- You can use the %ROWTYPE attribute in variable declarations as a datatype specifier.
- Variables declared using %ROWTYPE are treated like those declared using a datatype name.

Creating a PL/SQL procedure example

- **The following procedure accepts a customer id and prints out the customer's contact information including first name, last name, and email:**

```
CREATE OR REPLACE PROCEDURE print_contact(
    in_customer_id NUMBER
)
IS
  r_contact contacts%ROWTYPE;
BEGIN
  -- get contact based on customer id
  SELECT *
  INTO r_contact
  FROM contacts
  WHERE customer_id = p_customer_id;

  -- print out contact's information
  dbms_output.put_line( r_contact.first_name || ' ' ||
  r_contact.last_name || '<' || r_contact.email ||'>' );

EXCEPTION
   WHEN OTHERS THEN
      dbms_output.put_line( SQLERRM );
END;
```

- **Use / (forward slash) to compile procedures**
- **Using the EXECUTE/EXEC keyword with procedure name to execute procedure**
  - **EXECUTE procedure_name(arguments);**
  - **EXEC procedure_name(arguments);**

- **EXEC print_contact(100);**

```
1 Elisha Lloyd<elisha.lloyd@verizon.com>
```

PL/SQL Functions

```
CREATE [OR REPLACE] FUNCTION function_name
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
RETURN return_datatype
{IS | AS}
BEGIN
   < function_body >
END [function_name];
```

Where,

- *function-name* specifies the name of the function.
- [OR REPLACE] option allows the modification of an existing function.
- The optional parameter list contains name, mode and types of the parameters. IN represents the value that will be passed from outside and OUT represents the parameter that will be used to return a value outside of the procedure.
- The function must contain a **return** statement.
- The *RETURN* clause specifies the data type you are going to return from the function.
- *function-body* contains the executable part.
- The AS keyword is used instead of the IS keyword for creating a standalone function.

Practice Exercise

1	a) Create a table with following schema: Coustmer_contact(cid number(10),cfname varchar(10),clname varchar(10),cemail varchar(10)) b) Create a PL/SQL procedure accepts a customer id and prints out the customer's contact information including first name, last name, and email:
2	Create a employee table with following schema employee (e_id number(10), e_name varchar(10),e_sales number(8), target number(8), salary number(8,2)) Create a PL/SQL procedure which follows following: I. For a given e_id it checks whether e_sales greater than target or not. If greater then UPDATE employees SET salary = salary + bonus WHERE employee_id = emp_id; Otherwise no update to salary

	II. Bonus is target-sales. III. An exception is raised if no records were found for given e_id Use inbuilt exception NO_DATA_FOUND
3	Create a student table with following schema student (s_id number(10), s_name varchar(10),s_marks number(8)) Create a PL/SQL Program which follows following: I. FOR a given s_id select the s_marks. II. If marks greater than 90 print excellent III. If marks greater than 80 but less than 90 print very good IV. If marks greater than 70 but less than 80 print good V. If marks greater than 60 but less than 70 print fair VI. If marks greater than 50 but less than 60 print poor VII. For all other marks print fail An exception is raised if no records were found
4	Create a procedure which gets the name of the employee when the employee id is passed. Use out parameter to store and employee name and print it.
5	Create a PL/SQL functions to calculate the area and perimeter (perimeter) of circle.
6	Create a procedure and function to get the total number of employees in the employee table.

Instructions:

1. Write and execute the query in Oracle SQL server/ SQL* Plus.
2. Paste the snapshot of the output in input & output section.

Part B

Code and Output:

1.a.

```
1  create table Customer_contact
2  (cid number(10),cfname varchar(10),clname varchar(10),cemail varchar(10));
```

```
Table created.
```

```
1  insert into Customer_contact values(10,'Anil','Loyal','Ani@go.com');
```

```
1 row(s) inserted.
```

1.b.

```
1  create or replace procedure info(in_id number)
2  is contact Customer_contact%rowtype;
3  begin
4      select * into contact from Customer_contact where cid=in_id;
5      dbms_output.put_line(contact.cfname||' '||contact.clname||'<'||contact.cemail||'>');
6  exception
7      when others then
8          dbms_output.put_line(SQLERRM);
9  end info;
10 /
```

```
Procedure created.
```

```
1  exec info(10);
```

```
Statement processed.
Anil Loyal<Ani@go.com>
```

2)
a)

```
1  create table employee(e_id number(10), e_name varchar(10),e_sales number(8),
2  target number(8), salary number(8,2))
```

```
Table created.
```

```
1  desc employee;
```

TABLE EMPLOYEE

Column	Null?	Type
E_ID	-	NUMBER(10,0)
E_NAME	-	VARCHAR2(10)
E_SALES	-	NUMBER(8,0)
TARGET	-	NUMBER(8,0)
SALARY	-	NUMBER(8,2)

Download CSV

5 rows selected.

2

b)

```
1  insert into employee values(10,'Rohan',5000,8000,25000);
2  insert into employee values(11,'Raj',12000,9000,30000);
3  insert into employee values(12,'Ramu',10000,8000,35000);
4  insert into employee values(13,'Suresh',9000,7000,23000);
5  insert into employee values(14,'sohan',6000,5600,20000);
6
7  select * from employee;
```

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

E_ID	E_NAME	E_SALES	TARGET	SALARY
10	Rohan	5000	8000	25000
11	Raj	12000	9000	30000
12	Ramu	10000	8000	35000
13	Suresh	9000	7000	23000
14	sohan	6000	5600	20000

```
1
2  create or replace procedure d_employee(emp_id number)
3  is
4  bonus number;
5  emp employee%rowtype;
6  begin
7  select target into bonus from employee where e_id=emp_id;
8  select * into emp from employee where e_id=emp_id;
9  if(emp.e_sales>emp.target) then
10 update employee set salary=salary+bonus where e_id=emp_id;
11 end if;
12 exception
13 when no_data_found then
14 dbms_output.put_line('No records found');
15 end d_employee;
16 /
```

```
Procedure created.
```

```
1  exec d_employee(10);
2  exec d_employee(11);
3  exec d_employee(12);
4
5
6  select * from employee;
```

```
Statement processed.

Statement processed.
```

E_ID	E_NAME	E_SALES	TARGET	SALARY
10	Rohan	5000	8000	25000
11	Raj	12000	9000	39000
12	Ramu	10000	8000	43000
13	Suresh	9000	7000	23000
14	sohan	6000	5600	20000

Download CSV

5 rows selected.

3.

```
create table student (s_id number(10), s_name varchar(10),s_marks number(8));
insert into student values(1,'a',45);
insert into student values(2,'b',96);
insert into student values(3,'c',59);
insert into student values(4,'d',62);
insert into student values(5,'e',74);
insert into student values(6,'f',82);
insert into student values(7,'g',60);
```

```
Table created.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

```
select * from student;

create or replace procedure stu_pref(stu_id number)
is
marks student.s_marks%type;
begin
select s_marks into marks from student where s_id=stu_id;
if(marks>=90) then
    dbms_output.put_line('EXCELLENT');
elsif(marks>=80 and marks<90) then
    dbms_output.put_line('VERY GOOD');
elsif(marks>=70 and marks<80) then
    dbms_output.put_line('GOOD');
elsif(marks>=60 and marks<70) then
    dbms_output.put_line('FAIR');
elsif(marks>=50 and marks<60) then
    dbms_output.put_line('POOR');
else
    dbms_output.put_line('FAIL');
end if;
exception
when no_data_found then
dbms_output.put_line('No records found');
end;
/

exec stu_pref(1);
exec stu_pref(2);
exec stu_pref(3);
exec stu_pref(4);
exec stu_pref(5);
exec stu_pref(6);
exec stu_pref(7);
```

S_ID	S_NAME	S_MARKS
1	a	45
2	b	96
3	c	59
4	d	62
5	e	74
6	f	82
7	g	60

Download CSV
7 rows selected.

Procedure created.

Statement processed.
FAIL

Statement processed.
EXCELLENT

Statement processed.
POOR

Statement processed.
FAIR

Statement processed.
GOOD

Statement processed.
VERY GOOD

Statement processed.
FAIR

4.

```
select * from employee;

create or replace procedure prf(emp_id in number,emp_name out employee.e_name%type)
is
begin
select e_name into emp_name from employee where e_id=emp_id;
dbms_output.put_line(emp_name);
end;
/

declare
ename varchar(20);
begin
prf(10,ename);
end;
/
```

E_ID	E_NAME	E_SALES	TARGET	SALARY
10	Rohan	5000	8000	25000
11	Raj	12000	9000	39000
12	Ramu	10000	8000	43000
13	Suresh	9000	7000	23000
14	sohan	6000	5600	20000

Download CSV
5 rows selected.

Procedure created.

Statement processed.
Rohan

5.

```
create or replace function area_of_circle(r number)
return number
is
pi constant number(7,3):=3.141;
area number(7,3);
begin
area:=pi*(r*r);
return area;
end;
/
declare
begin
dbms_output.put_line(area_of_circle(12)||' is the area of circle with radius 12');
end;
/
```

Function created.

Statement processed.
452.304 is the area of circle with radius 12

```
1  create or replace function perimeter_of_circle(r number)
2  return number
3  is
4  pi constant number(7,3):=3.141;
5  p number(7,3);
6  begin
7  p:=2*pi*r;
8  return p;
9  end;
10 /
11 declare
12 begin
13 dbms_output.put_line(perimeter_of_circle(12)||' is the perimeter of circle with radius 12');
14 end;
15 /
```

```
Function created.


Statement processed.
75.384 is the perimeter of circle with radius 12
```

6.

```
1  create or replace procedure total_emp
2  is
3  tot number;
4  begin
5  select count(*) into tot from employee;
6  dbms_output.put_line(tot||' is the total number of employees ');
7  end;
8  /
9
10 exec total_emp;
```

```
Procedure created.


Statement processed.
5 is the total number of employees
```

```
1  create or replace function total_emp_func
2  return number
3  is
4  tot number;
5  begin
6  select count(*) into tot from employee;
7  return tot;
8  end;
9  /
10
11
12 begin
13 dbms_output.put_line(total_emp_func()||' is the total number of employees ');
14 end;
15 /
```

```
Function created.


Statement processed.
5 is the total number of employees
```

Observation & Learning:
Learned how to create a pl/sql procedure , compile it and execute the procedure.
Also understood how to create a pl/sql function and call a function .

Conclusion:
Learned and practiced pl/sql procedures and functions perfectly.