



Bansilal Ramnath Agarwal Charitable Trust's
Vishwakarma Institute of Information Technology

**Department of
Artificial Intelligence and Data Science**

Name: Siddhesh Dilip Khairnar

Class: SY

Division: B

Roll No: 272028

Semester: III

Academic Year: 2022-2023

Subject Name & Code: Database Management System: ADUA21204

Title of Assignment: PL/SQL Stored Procedure and Stored Function: Write a PL/SQL procedure to find the number of students ranging from 100-80%, 79-70% ,69-60%, 59-50 & below 49% in each course from the Student_course table given by the procedure as parameter. Student_course (Roll_no, Course, Couse_code, Semester, Total_ Marks, Percentage).

Assignment No.- 05

Name: Siddhesh Dilip Khairnar
Roll no: 272028 PRN no: 22110398

PAGE NO.:
DATE: / /

Assignment no: 5

PL/SQL :-

- developed by Oracle Corporation in the 1980's
- It's procedure language extension for SQL and Oracle relational database
- PL/SQL is a block structural language that can have multiple block in PL.
- PL/SQL Block Basic syntax.

DECLARE

< declaration section >

BEGIN

< executable command >

EXCEPTION

< exception handling >

END;

- PROCEDURES IN PL/SQL

It's a named block of statement. It may or may not return a value.

Syntax:

CREATE [OR REPLACE] PROCEDURE procedure_name

([parameter_name [IN|OUT|INOUT] type[, ...]])

IS

[declaration]

BEGIN

execution

[EXCEPTION

exception_section]

END [name of procedure];

PL/SQL Function:

- It is similar to PL/SQL Procedure
- PL/SQL function must always return a value whereas procedure may or may not return a value.
- Function must contain RETURN Statement.

Syntax:

```
CREATE [OR REPLACE] function function_name [Parameter]
[ ( Parameter_name [IN|OUT|INOUT] type[, ...] ) ]
```

```
RETURN return_datatype
```

```
{ IS / AS }
```

```
BEGIN
```

```
    <Function body>
```

```
END [function_name];
```

Example: Addition of 2 Number.

```
CREATE OR REPLACE FUNCTION added ( N1 in number, N2 in number)
```

```
return number
```

```
IS
```

```
  N3 number(8);
```

```
  BEGIN
```

```
    N3 := N1 + N2;
```

```
  return N3;
```

```
  end;
```

```
  /
```

- calling function




(I) exec functionname;

(II) begin function name end;








Implementation:

Table used for the following operations

```
1 • use db4;
2
3 • insert into student_course values
4 (1,"CS",101,3,360,90),(2,"CS",101,3,240,60),
5 (3,"CS",101,3,300,75),(4,"CS",101,3,320,80),
6 (5,"CS",101,3,200,50),(6,"CS",101,3,180,45),
7 (7,"CS",101,3,280,70),(8,"CS",101,3,290,73);
```

Result Grid						
Filter Rows: <input type="text"/>						
Edit:    Export/Import						
	Roll_no	Course	Course_code	Semester	Total_Marks	Percentage
▶	1	CS	101	3	360	90
	2	CS	101	3	240	60
	3	CS	101	3	300	75
	4	CS	101	3	320	80
	5	CS	101	3	200	50
	6	CS	101	3	180	45
	7	CS	101	3	280	70
	8	CS	101	3	290	73
*	NULL	NULL	NULL	NULL	NULL	NULL

Stored Procedure

Name:	procedure1
DDL:	<div>      </div> <pre>1 • CREATE DEFINER=`root`@`localhost` PROCEDURE `procedure1`(IN SR INT,IN ER INT,OUT stud INT) 2 BEGIN 3 select count(*) INTO stud from student_course where Percentage between SR and ER; 4 END</pre>

For 80%-100%

```

1  set @stud=0;
2  • call db4.procedure1(80,100,@stud);
3  • select @stud;

```

Result Grid	
	@stud
▶	2

For 70%-79%

```

1  set @stud=0;
2  • call db4.procedure1(70,79,@stud);
3  • select @stud;

```

Result Grid	
	@stud
▶	3

For 60%-69%

```

1  set @stud=0;
2  • call db4.procedure1(60,69,@stud);
3  • select @stud;

```

Result Grid	
	@stud
▶	1

For 50%-59%

```

1  set @stud=0;
2  • call db4.procedure1(50,59,@stud);
3  • select @stud;

```

Result Grid	
	@stud
▶	1

For below 49%

```

1  set @stud=0;
2  • call db4.procedure1(0,49,@stud);
3  • select @stud;

```

Result Grid	
	@stud
▶	1