

# ASSIGNMENT 7

Name: Bhavik Ransubhe

Class: TE (B) COMP

ROLL NO :39055

\*\*\*\*\*

## 1) Stored Procedure without parameters :-

*Creating tables (tables will remain same for all ) and inserting values:*

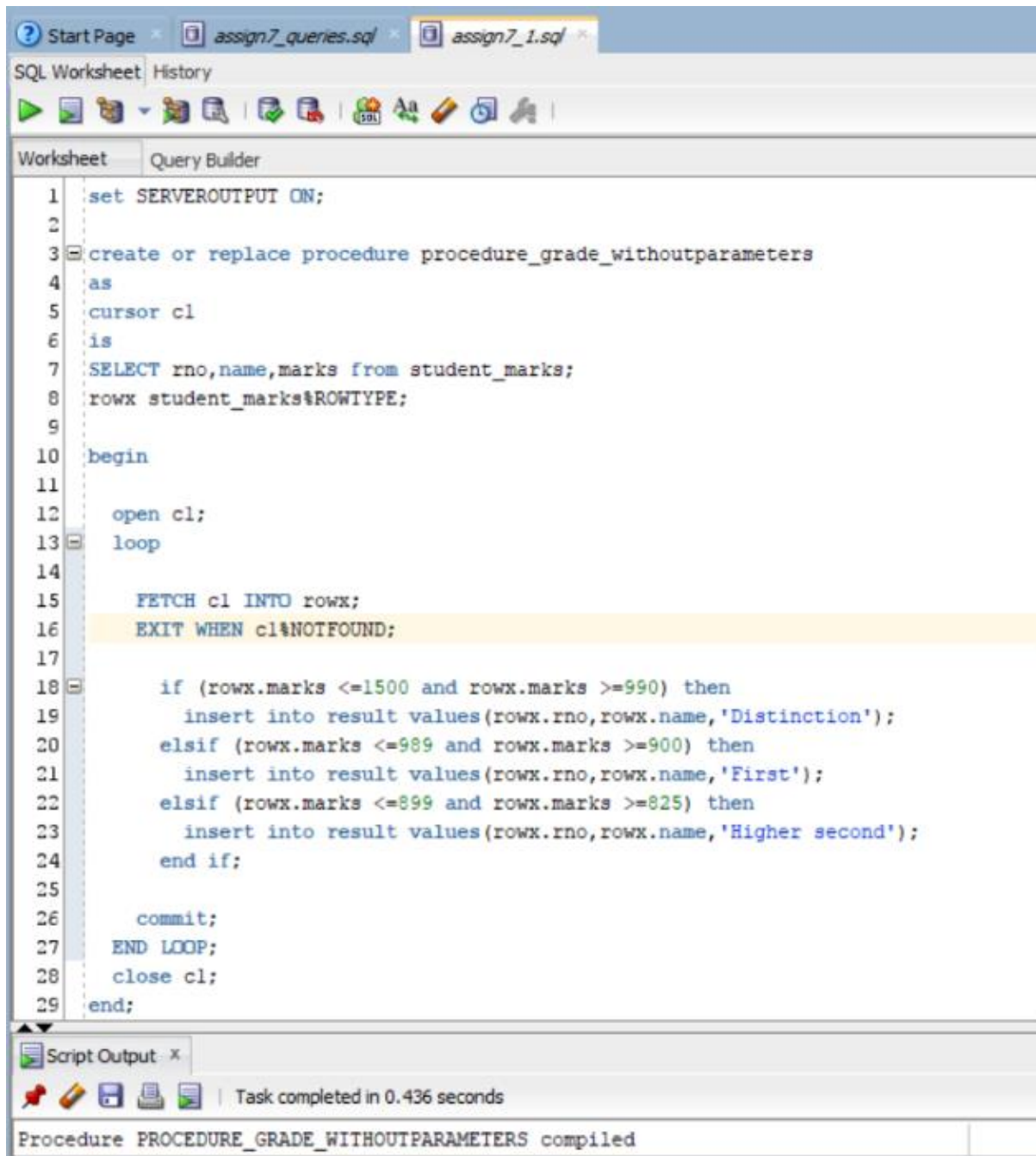
The screenshot shows an SQL IDE with two tabs: 'assign7\_queries.sql' and 'assign7\_1.sql'. The 'Worksheet' tab is active, displaying the following SQL code:

```
1 create table student_marks(  
2   rno number(5),  
3   name varchar2(20),  
4   marks number(6)  
5 );  
6  
7 create table result(  
8   rno number(5),  
9   name varchar2(20),  
10  class varchar2(20)  
11 );  
12  
13 insert into student_marks values(1,'Bhavik',1400);  
14 insert into student_marks values(2,'Kim',850);  
15 insert into student_marks values(3,'Kylie',970);  
16  
17 select * from student_marks;  
18
```

The 'Query Result' tab is also visible, showing the output of the last query. It displays a table with 3 rows and 3 columns: RNO, NAME, and MARKS.

RNO	NAME	MARKS
1	1 Bhavik	1400
2	2 Kim	850
3	3 Kylie	970

## Code:



```
1 set SERVEROUTPUT ON;
2
3 create or replace procedure procedure_grade_withoutparameters
4 as
5 cursor c1
6 is
7 SELECT rno,name,marks from student_marks;
8 rowx student_marks%ROWTYPE;
9
10 begin
11
12     open c1;
13     loop
14
15         FETCH c1 INTO rowx;
16         EXIT WHEN c1%NOTFOUND;
17
18         if (rowx.marks <=1500 and rowx.marks >=990) then
19             insert into result values(rowx.rno,rowx.name,'Distinction');
20         elsif (rowx.marks <=989 and rowx.marks >=900) then
21             insert into result values(rowx.rno,rowx.name,'First');
22         elsif (rowx.marks <=899 and rowx.marks >=825) then
23             insert into result values(rowx.rno,rowx.name,'Higher second');
24         end if;
25
26         commit;
27     END LOOP;
28     close c1;
29 end;
```

Script Output x

Task completed in 0.436 seconds

Procedure PROCEDURE\_GRADE\_WITHOUTPARAMETERS compiled

## Output:

```
19 exec procedure_grade_withoutparameters;
20 select * from result;
```

RNO	NAME	CLASS
1	1 Bhavik	Distinction
2	2 Kim	Higher second
3	3 Kylie	First

\*\*\*\*\*

## 2.Stored Procedure with parameters :-

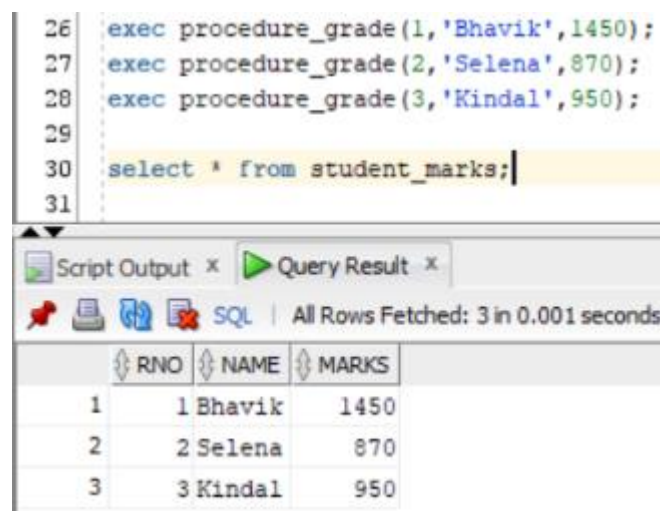
### Code:

Start Page assign7_queries.sql assign7_2.sql	
SQL Worksheet History	
0.147 seconds	
Worksheet Query Builder	
<pre>1 set serveroutput on; 2 3 create or replace procedure procedure_grade(rno number ,name varchar2,marks number) 4 is 5 class varchar2(20); 6 7 begin 8 9     if(marks &lt;=1500 and marks &gt;= 990)then 10         class := 'Disitinction'; 11 12     elsif(marks &lt;=989 and marks &gt;= 900)then 13         class := 'First'; 14 15     elsif(marks &lt;=899 and marks &gt;= 825)then 16         class := 'Highet second'; 17 18     end if; 19 20     insert into student_marks values(rno,name,marks); 21     commit; 22 23     insert into result values(rno,name,class); 24     commit; 25 26 end;</pre>	
Script Output	
Task completed in 0.147 seconds	
Procedure PROCEDURE_GRADE compiled	

## Output:

*Executing procedure with parameters :*

```
26  exec procedure_grade(1,'Bhavik',1450);
27  exec procedure_grade(2,'Selena',870);
28  exec procedure_grade(3,'Kindal',950);
29
30  select * from student_marks;
31
```

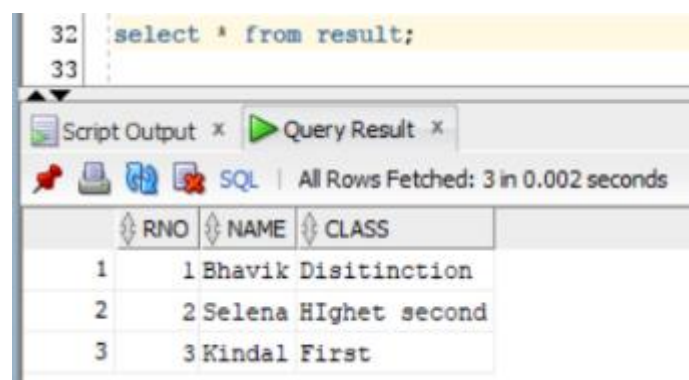


The screenshot shows the SQL Developer interface. The top pane contains the SQL script. The bottom pane shows the 'Query Result' tab with a table of 3 rows and 3 columns: RNO, NAME, and MARKS. The status bar indicates 'All Rows Fetched: 3 in 0.001 seconds'.

RNO	NAME	MARKS
1	1 Bhavik	1450
2	2 Selena	870
3	3 Kindal	950

*Final output :*

```
32  select * from result;
33
```



The screenshot shows the SQL Developer interface. The top pane contains the SQL script. The bottom pane shows the 'Query Result' tab with a table of 3 rows and 3 columns: RNO, NAME, and CLASS. The status bar indicates 'All Rows Fetched: 3 in 0.002 seconds'.

RNO	NAME	CLASS
1	1 Bhavik	Disitinction
2	2 Selena	Hihet second
3	3 Kindal	First

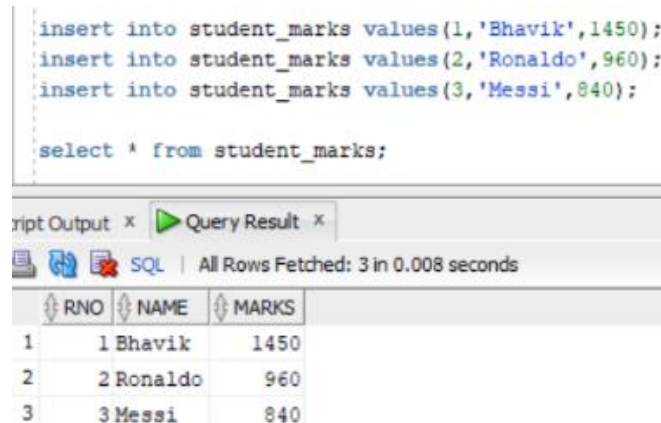
\*\*\*\*\*

## 3.Stored function:-

*Inserting values in table :*

```
insert into student_marks values(1,'Bhavik',1450);
insert into student_marks values(2,'Ronaldo',960);
insert into student_marks values(3,'Messi',840);

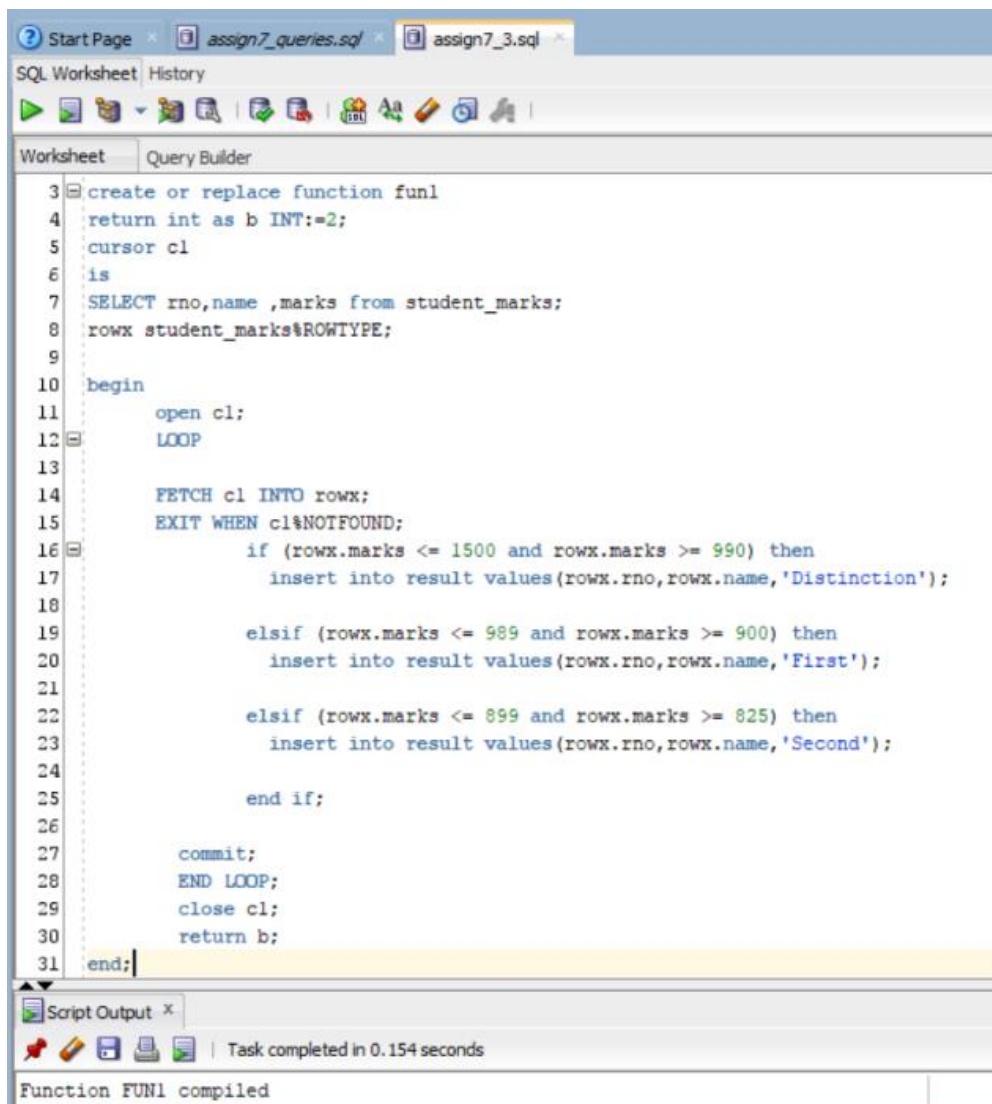
select * from student_marks;
```



The screenshot shows the SQL Developer interface. The top pane contains the SQL script. The bottom pane shows the 'Query Result' tab with a table of 3 rows and 3 columns: RNO, NAME, and MARKS. The status bar indicates 'All Rows Fetched: 3 in 0.008 seconds'.

RNO	NAME	MARKS
1	1 Bhavik	1450
2	2 Ronaldo	960
3	3 Messi	840

## Code:

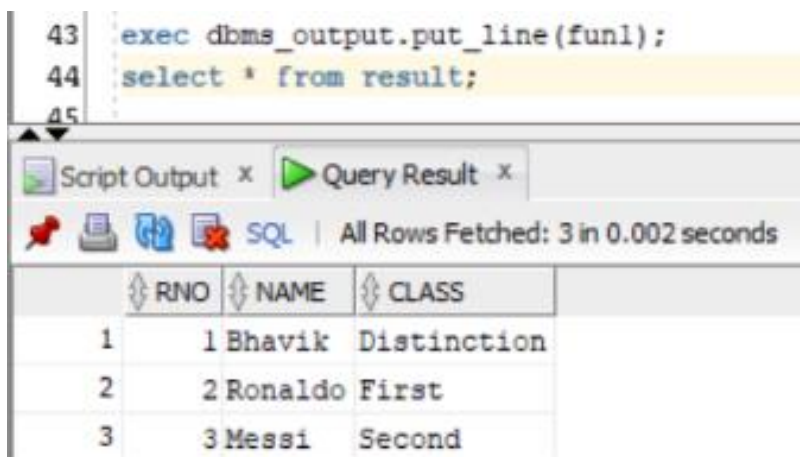


The screenshot shows a SQL Worksheet interface with a tab for 'assign7\_queries.sql'. The 'Worksheet' tab is active, displaying a PL/SQL function named 'fun1'. The function is designed to iterate through a table named 'student\_marks' and categorize students based on their marks into three classes: 'Distinction', 'First', and 'Second'. The function uses a cursor to fetch rows and an if-elsif-else loop to assign the class based on the 'marks' column. The function returns an integer value 'b'.

```
3 create or replace function fun1
4 return int as b INT:=2;
5 cursor c1
6 is
7 SELECT rno,name ,marks from student_marks;
8 rowx student_marks%ROWTYPE;
9
10 begin
11     open c1;
12     LOOP
13
14         FETCH c1 INTO rowx;
15         EXIT WHEN c1%NOTFOUND;
16         if (rowx.marks <= 1500 and rowx.marks >= 990) then
17             insert into result values(rowx.rno,rowx.name,'Distinction');
18
19         elsif (rowx.marks <= 989 and rowx.marks >= 900) then
20             insert into result values(rowx.rno,rowx.name,'First');
21
22         elsif (rowx.marks <= 899 and rowx.marks >= 825) then
23             insert into result values(rowx.rno,rowx.name,'Second');
24
25         end if;
26
27     commit;
28     END LOOP;
29     close c1;
30     return b;
31 end;
```

Below the code editor, the 'Script Output' pane shows the message: 'Function FUN1 compiled'.

## Output:



The screenshot shows the same SQL Worksheet interface, but now the 'Query Result' tab is active. It displays the execution of a query that selects all rows from the 'result' table. The results are shown in a table with three columns: 'RNO', 'NAME', and 'CLASS'. The output shows three rows of data, corresponding to the students Bhavik, Ronaldo, and Messi, with their respective class assignments.

```
43 exec dbms_output.put_line(fun1);
44 select * from result;
```

	RNO	NAME	CLASS
1	1	Bhavik	Distinction
2	2	Ronaldo	First
3	3	Messi	Second

\*\*\*\*\*  
END  
\*\*\*\*\*