

Experiment-7

Student Name: Gauri Prabhakar

UID: 18BCS6201

Branch: 18AITAIML-2

Section/Group: B

Semester: 7

Date of Performance: 13th October, 2021

Subject Name: Advanced Database Management Lab

Subject Code: CSP - 434

1. Aim/Overview of the practical:

To implement PL/SQL programming using Exception Handling.

2. Task to be done:

To implement PL/SQL programming using Exception Handling.

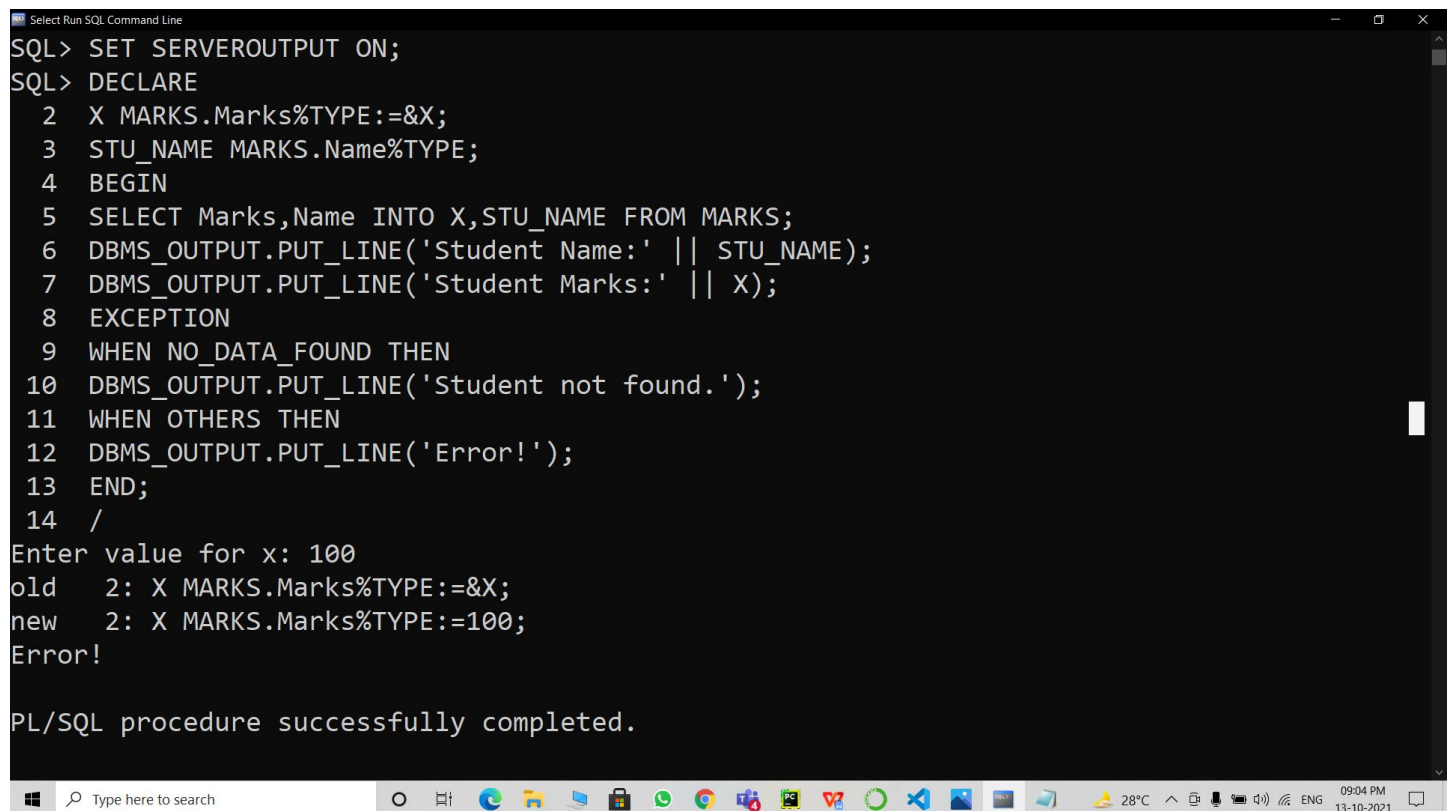
3. Steps to be followed:

Implementing SYSTEM-DEFINED EXCEPTION:

1. SET SERVEROUTPUT ON;

```
DECLARE
X MARKS.Marks%TYPE:=&X;
STU_NAME MARKS.Name%TYPE;
BEGIN
SELECT Marks,Name INTO X,STU_NAME FROM MARKS;
DBMS_OUTPUT.PUT_LINE('Student Name:' || STU_NAME);
DBMS_OUTPUT.PUT_LINE('Student Marks:' || X);
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('Student not found.');
```

WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Error!');
END;
/



The screenshot shows a Windows taskbar at the bottom with various application icons. The main window is titled "Select Run SQL Command Line" and has a black background with white text. The SQL prompt "SQL>" is used to enter the PL/SQL code. The code includes a declaration of variables X and STU_NAME, a SELECT statement, and a block of code to output the student's name and marks. An exception handler is defined for NO_DATA_FOUND, which outputs "Student not found.". The code is then executed, and the prompt "Enter value for x: 100" appears. The output shows the old value of X as "&X;" and the new value as "100". An "Error!" message is displayed, indicating that the exception was raised. The final message "PL/SQL procedure successfully completed." is shown at the bottom of the window.

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
2  X MARKS.Marks%TYPE:=&X;
3  STU_NAME MARKS.Name%TYPE;
4  BEGIN
5  SELECT Marks,Name INTO X,STU_NAME FROM MARKS;
6  DBMS_OUTPUT.PUT_LINE('Student Name:' || STU_NAME);
7  DBMS_OUTPUT.PUT_LINE('Student Marks:' || X);
8  EXCEPTION
9  WHEN NO_DATA_FOUND THEN
10 DBMS_OUTPUT.PUT_LINE('Student not found.');
```

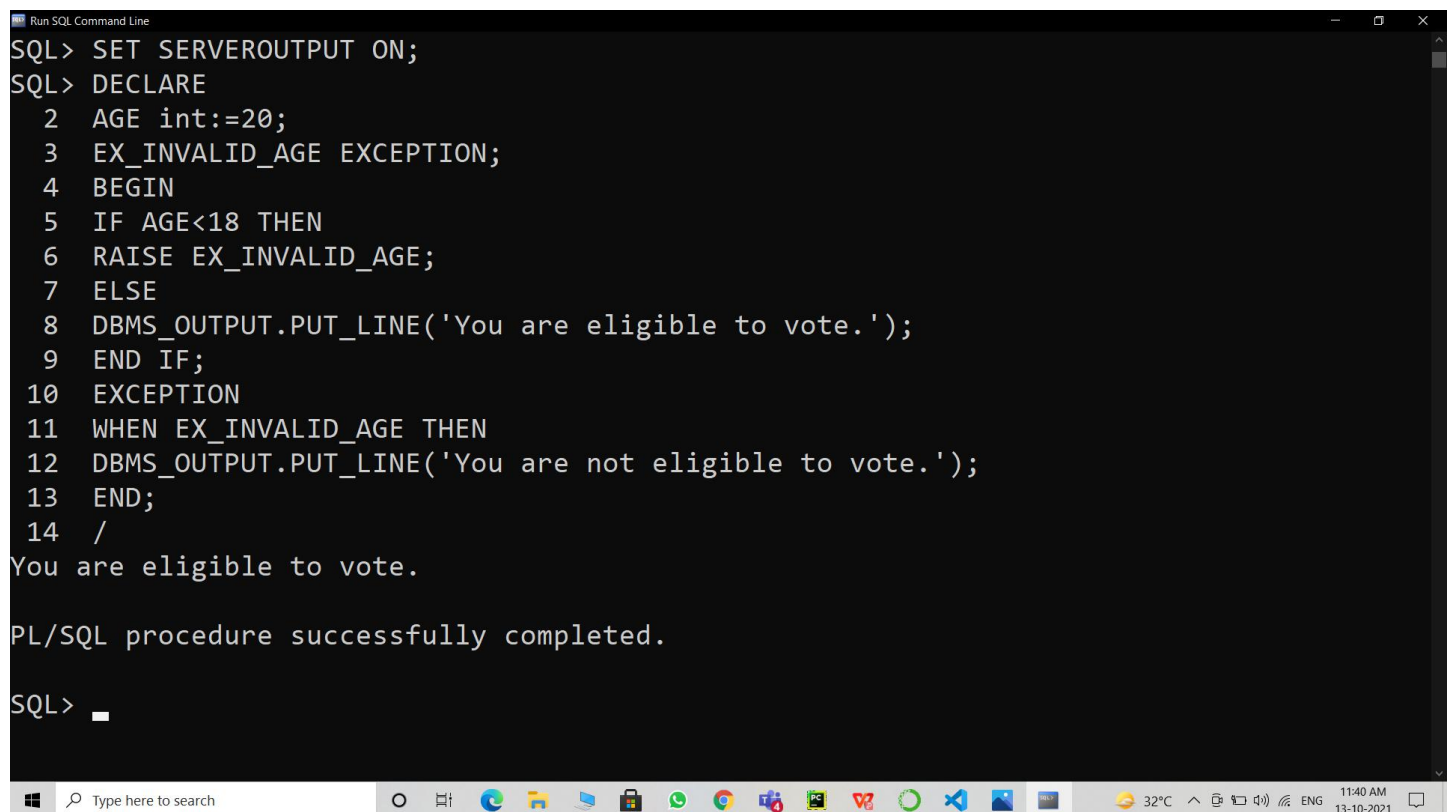
Enter value for x: 100
old 2: X MARKS.Marks%TYPE:=&X;
new 2: X MARKS.Marks%TYPE:=100;
Error!

PL/SQL procedure successfully completed.

Implementing USER-DEFINED EXCEPTION:

2. SET SERVEROUTPUT ON;

```
DECLARE
AGE int:=20;
EX_INVALID_AGE EXCEPTION;
BEGIN
IF AGE<18 THEN
RAISE EX_INVALID_AGE;
ELSE
DBMS_OUTPUT.PUT_LINE('You are eligible to vote. ');
END IF;
EXCEPTION
WHEN EX_INVALID_AGE THEN
DBMS_OUTPUT.PUT_LINE('You are not eligible to vote. ');
END;
/
```



```
Run SQL Command Line
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
  2 AGE int:=20;
  3 EX_INVALID_AGE EXCEPTION;
  4 BEGIN
  5 IF AGE<18 THEN
  6 RAISE EX_INVALID_AGE;
  7 ELSE
  8 DBMS_OUTPUT.PUT_LINE('You are eligible to vote. ');
  9 END IF;
 10 EXCEPTION
 11 WHEN EX_INVALID_AGE THEN
 12 DBMS_OUTPUT.PUT_LINE('You are not eligible to vote. ');
 13 END;
 14 /
You are eligible to vote.

PL/SQL procedure successfully completed.

SQL> _
```

4. Result/Output/Writing Summary:

- Successfully implemented EXCEPTIONS.
- Successfully implemented SYSTEM-DEFINED EXCEPTIONS.
- Successfully implemented USER-DEFINED EXCEPTIONS.
- Successfully understood the functioning and importance of the above mentioned.

5. Learning outcomes (What I have learnt):

- How to implement EXCEPTIONS on SQL Command Line.
- How to implement SYSTEM-DEFINED EXCEPTIONS on SQL Command Line.
- How to implement USER-DEFINED EXCEPTIONS on SQL Command Line.

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			