DBMS LAB
Practical-5

Name: Vedant Bhutada
Roll : 69
Batch: A4

Aim:-To write and execute PL/SQL blocks (with exception handling) including PL/SQL subprograms using Oracle 11g.

```
SQL> set serveroutput on;
SQL> --Write a PL-SQL block to find greatest among three given numbers.
SQL> declare
  2  a number:=10;
  3  b number:=12;
  4  c number:=5;
  5  begin
  6  dbms_output.put_line('a='||a||' b='||b||' c='||c);
  7  if a>b AND a>c
  8  then
  9  dbms_output.put_line('a is greatest');
 10  else
 11  if b>a AND b>c
 12  then
 13  dbms_output.put_line('b is greatest');
 14  else
 15  dbms_output.put_line('c is greatest');
 16  end if;
 17  end if;
 18  end;
 19  /
a=10 b=12
c=5
b is
greatest

PL/SQL procedure successfully completed.

SQL> DECLARE
  2     a NUMBER := 10;
  3     b NUMBER := 12;
  4     c NUMBER := 5;
  5  BEGIN
  6     DBMS_OUTPUT.PUT_LINE('a=' || a || ' b=' || b || ' c=' || c);
  7
  8     IF a > b AND a > c THEN
  9        DBMS_OUTPUT.PUT_LINE('a is greatest');
 10     ELSIF b > a AND b > c THEN
 11        DBMS_OUTPUT.PUT_LINE('b is greatest');
 12     ELSE
 13        DBMS_OUTPUT.PUT_LINE('c is greatest');
 14     END IF;
 15  END;
 16  /
a=10 b=12
c=5
b is
greatest

PL/SQL procedure successfully completed.

SQL> --Write a PL-SQL block to find out if a year is a leap year.(A leap year
is divisible by 4 but not by 100,or it
SQL> --is divisible by 400)
SQL> DECLARE
  2     year NUMBER := &year;
  3  BEGIN
  4     IF (MOD(year, 4) = 0 AND MOD(year, 100) != 0) OR MOD(year, 400) = 0
THEN
  5        DBMS_OUTPUT.PUT_LINE(year || ' is a leap year.');
  6     ELSE
  7        DBMS_OUTPUT.PUT_LINE(year || ' is not a leap year.');
  8     END IF;
  9  END;
 10  /
Enter value for year: 2022
old   2:     year NUMBER := &year;
new   2:     year NUMBER := 2022;
2022 is not a leap
year.

PL/SQL procedure successfully completed.

SQL> /
Enter value for year: 2012
old   2:     year NUMBER := &year;
new   2:     year NUMBER := 2012;
2012 is a leap
year.

PL/SQL procedure successfully completed.

SQL> --Input a number with a substitution variable, and then print its
multiplication table using a While loop.
SQL> DECLARE
  2     num NUMBER := &num;
  3     multiplier NUMBER := 1;
  4  BEGIN
  5     WHILE multiplier <= 10 LOOP
  6        DBMS_OUTPUT.PUT_LINE(num || ' * ' || multiplier || ' = ' || num *
multiplier);
  7        multiplier := multiplier + 1;
  8     END LOOP;
  9  END;
 10  /
Enter value for num: 2
old   2:     num NUMBER := &num;
new   2:     num NUMBER := 2;
2 * 1 =
2
2 * 2 =
4
2 * 3 =
6
2 * 4 =
8
2 * 5 =
10
2 * 6 =
12
2 * 7 =
14
2 * 8 =
16
2 * 9 =
18
2 * 10 =
20

PL/SQL procedure successfully completed.

SQL> DECLARE
  2     v_num NUMBER := &num;
  3     v_multiplier NUMBER := 1;
  4  BEGIN
  5     WHILE v_multiplier <= 10 LOOP
  6        DBMS_OUTPUT.PUT_LINE(v_num || ' * ' || v_multiplier || ' = ' ||
v_num * v_multiplier);
  7        v_multiplier := v_multiplier + 1;
  8     END LOOP;
  9  END;
 10  /
Enter value for num: 5
old   2:     v_num NUMBER := &num;
```

```
new   2:     v_num NUMBER := 5;
5 * 1 =
5
5 * 2 =
10
5 * 3 =
15
5 * 4 =
20
5 * 5 =
25
5 * 6 =
30
5 * 7 =
35
5 * 8 =
40
5 * 9 =
45
5 * 10 =
50

PL/SQL procedure successfully completed.

SQL> --Write a PL-SQL block to print all odd numbers between 1 and 10 using a
basic loop.
SQL> DECLARE
  2     counter NUMBER := 1;
  3  BEGIN
  4     LOOP
  5        EXIT WHEN counter > 10;
  6        IF MOD(counter, 2) != 0 THEN
  7           DBMS_OUTPUT.PUT_LINE('Odd number: ' || counter);
  8        END IF;
  9        counter := counter + 1;
 10     END LOOP;
 11  END;
 12  /
Odd number:
1
Odd number:
3
Odd number:
5
Odd number:
7
Odd number:
9
```

```
PL/SQL procedure successfully completed.

SQL> --Using a for loop, print the value 10 to 1 in reverse order.
SQL> DECLARE
  2  BEGIN
  3     FOR i IN REVERSE 10..1 LOOP
  4        DBMS_OUTPUT.PUT_LINE('Value: ' || i);
  5     END LOOP;
  6  END;
  7  /

PL/SQL procedure successfully completed.

SQL> DECLARE
  2  BEGIN
  3     FOR i IN REVERSE 1..10 LOOP
  4        DBMS_OUTPUT.PUT_LINE('Value: ' || i);
  5     END LOOP;
  6  END;
  7  /
Value:
10
Value:
9
Value:
8
Value:
7
Value:
6
Value:
5
Value:
4
Value:
3
Value:
2
Value:
1

PL/SQL procedure successfully completed.

SQL> --Write a PL-SQL program to swap the values of two variables. Print the
variables before and after
SQL> --swapping.
SQL> DECLARE
```

```
  2     var1 NUMBER := &var1;
  3     var2 NUMBER := &var2;
  4     temp NUMBER;
  5  BEGIN
  6     DBMS_OUTPUT.PUT_LINE('Before swapping - var1: ' || var1 || ', var2: '
|| var2);
  7
  8     -- Swapping logic
  9     temp := var1;
 10     var1 := var2;
 11     var2 := temp;
 12
 13     DBMS_OUTPUT.PUT_LINE('After swapping - var1: ' || var1 || ', var2: '
|| var2);
 14  END;
 15  /
Enter value for var1: 12
old   2:     var1 NUMBER := &var1;
new   2:     var1 NUMBER := 12;
Enter value for var2: 19
old   3:     var2 NUMBER := &var2;
new   3:     var2 NUMBER := 19;
Before swapping - var1: 12, var2:
19
After swapping - var1: 19, var2:
12

PL/SQL procedure successfully completed.

SQL> commit;

Commit complete.

SQL> @"C:\Users\vedan\OneDrive\Desktop\sql\script_scott_schema.sql"
Building demonstration tables.  Please wait.

Table dropped.


Table dropped.


Table dropped.


Table dropped.


Table dropped.
Table created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

1 row created.

Table created.

1 row created.
```

```

1 row created.


1 row created.


1 row created.


Table created.


Table created.


1 row created.


1 row created.


1 row created.


1 row created.


1 row created.


Table created.


1 row created.


Commit complete.

Demonstration table build is complete.

SQL> set linesize 300;
SQL> select * from dept;

    DEPTNO
DNAME          LOC


```

```
---------- -------------- ------------
-



        10 ACCOUNTING     NEW
YORK



        20
RESEARCH       DALLAS



        30
SALES          CHICAGO



        40
OPERATIONS     BOSTON




SQL> select * from emp;

     EMPNO ENAME      JOB             MGR
HIREDATE         SAL       COMM     DEPTNO



---------- ---------- --------- ---------- --------- ---------- ---------- ---
------
-



      7369 SMITH      CLERK          7902 17-DEC-
80        800                    20



      7499 ALLEN      SALESMAN       7698 20-FEB-
81       1600       300          30


```

```
      7521 WARD       SALESMAN        7698 22-FEB-
81       1250        500         30



      7566 JONES      MANAGER         7839 02-APR-
81       2975                    20



      7654 MARTIN     SALESMAN        7698 28-SEP-
81       1250       1400         30



      7698 BLAKE      MANAGER         7839 01-MAY-
81       2850                    30



      7782 CLARK      MANAGER         7839 09-JUN-
81       2450                    10



      7788 SCOTT      ANALYST         7566 09-DEC-
82       3000                    20



      7839 KING       PRESIDENT            17-NOV-
81       5000                    10



      7844 TURNER     SALESMAN        7698 08-SEP-
81       1500          0         30



      7876 ADAMS      CLERK           7788 12-JAN-
83       1100                    20




     EMPNO ENAME      JOB              MGR
HIREDATE          SAL       COMM      DEPTNO
```

```
---------- ---------- --------- ---------- --------- ---------- ---------- ---
------
-

      7900 JAMES      CLERK           7698 03-DEC-
81         950                    30

      7902 FORD       ANALYST         7566 03-DEC-
81        3000                    20

      7934 MILLER     CLERK           7782 23-JAN-
82        1300                    10

14 rows selected.

SQL> select * from salgrade;

     GRADE      LOSAL      HISAL

---------- ---------- ---------
-

         1        700       1200

         2       1201       1400

         3       1401       2000
         4       2001       3000



         5       3001       9999



SQL> DECLARE
  2      v_empno emp.empno%TYPE := &empno;
  3      v_ename emp.ename%TYPE;
  4      v_deptno emp.deptno%TYPE;
  5      v_sal emp.sal%TYPE;
  6  BEGIN
  7      SELECT ename, deptno, sal INTO v_ename, v_deptno, v_sal
  8      FROM emp
  9      WHERE empno = v_empno;
 10
 11      DBMS_OUTPUT.PUT_LINE('Employee Details - EmpNo: ' || v_empno || ',
Ename: ' || v_ename || ', DeptNo: ' || v_deptno || ', Sal: ' || v_sal);
 12  EXCEPTION
 13      WHEN NO_DATA_FOUND THEN
 14          DBMS_OUTPUT.PUT_LINE('Employee with EmpNo ' || v_empno || ' not
found.');
 15  END;
 16  /
Enter value for empno: 7499
old   2:     v_empno emp.empno%TYPE := &empno;
new   2:     v_empno emp.empno%TYPE := 7499;

PL/SQL procedure successfully completed.

SQL> set serveroutput on;
SQL> DECLARE
  2      v_empno emp.empno%TYPE := &empno;
  3      v_ename emp.ename%TYPE;
  4      v_deptno emp.deptno%TYPE;
  5      v_sal emp.sal%TYPE;
  6  BEGIN
  7      SELECT ename, deptno, sal INTO v_ename, v_deptno, v_sal
  8      FROM emp
  9      WHERE empno = v_empno;
 10
 11      DBMS_OUTPUT.PUT_LINE('Employee Details - EmpNo: ' || v_empno || ',
Ename: ' || v_ename || ', DeptNo: ' || v_deptno || ', Sal: ' || v_sal);
```

```
 12  EXCEPTION
 13     WHEN NO_DATA_FOUND THEN
 14        DBMS_OUTPUT.PUT_LINE('Employee with EmpNo ' || v_empno || ' not
found.');
 15  END;
 16  /
Enter value for empno: 7499
old   2:     v_empno emp.empno%TYPE := &empno;
new   2:     v_empno emp.empno%TYPE := 7499;
Employee Details - EmpNo: 7499, Ename: ALLEN, DeptNo: 30, Sal:
1600




PL/SQL procedure successfully completed.

SQL> /
Enter value for empno: 1111
old   2:     v_empno emp.empno%TYPE := &empno;
new   2:     v_empno emp.empno%TYPE := 1111;
Employee with EmpNo 1111 not
found.




PL/SQL procedure successfully completed.

SQL> --2
SQL> DECLARE
  2     v_empno emp.empno%TYPE := &empno;
  3     v_sal emp.sal%TYPE;
  4     v_grade VARCHAR2(1);
  5  BEGIN
  6     SELECT sal INTO v_sal
  7     FROM emp
  8     WHERE empno = v_empno;
  9
 10     IF v_sal > 3000 THEN
 11        v_grade := 'A';
 12     ELSIF v_sal > 2000 THEN
 13        v_grade := 'B';
 14     ELSIF v_sal > 1000 THEN
 15        v_grade := 'C';
 16     ELSE
 17        v_grade := 'D';
 18     END IF;
 19
 20     DBMS_OUTPUT.PUT_LINE('Employee Grade: ' || v_grade);
 21  EXCEPTION
 22     WHEN NO_DATA_FOUND THEN
 23        DBMS_OUTPUT.PUT_LINE('Employee with EmpNo ' || v_empno || ' not
found.');
 24  END;
 25  /
Enter value for empno: 7499
old   2:     v_empno emp.empno%TYPE := &empno;
new   2:     v_empno emp.empno%TYPE := 7499;
Employee Grade:
C





PL/SQL procedure successfully completed.

SQL> /
Enter value for empno: 7654
old   2:     v_empno emp.empno%TYPE := &empno;
new   2:     v_empno emp.empno%TYPE := 7654;
Employee Grade:
C





PL/SQL procedure successfully completed.

SQL> --3
SQL> DECLARE
  2     v_empname emp.ename%TYPE;
  3  BEGIN
  4     SELECT ename INTO v_empname
  5     FROM (
  6        SELECT ename, RANK() OVER (ORDER BY sal DESC) r
  7        FROM emp
  8     )
  9     WHERE r = 4;
 10
 11     DBMS_OUTPUT.PUT_LINE('Employee with the fourth largest salary: ' ||
v_empname);
 12  EXCEPTION
 13     WHEN NO_DATA_FOUND THEN
 14        DBMS_OUTPUT.PUT_LINE('Not enough employees for the query.');
 15  END;
 16  /
Employee with the fourth largest salary:
JONES




PL/SQL procedure successfully completed.

SQL> --5
SQL> DECLARE
  2      v_empid emp.empno%TYPE := &empid;
  3      v_ename emp.ename%TYPE;
  4      v_sal emp.sal%TYPE;
  5      v_comm emp.comm%TYPE;
  6  BEGIN
  7      SELECT ename, sal, comm INTO v_ename, v_sal, v_comm
  8      FROM emp
  9      WHERE empno = v_empid;
 10
 11      DBMS_OUTPUT.PUT_LINE('Employee Details - Name: ' || v_ename || ',
Total Salary: ' || (v_sal + NVL(v_comm, 0)));
 12  EXCEPTION
 13      WHEN NO_DATA_FOUND THEN
 14         DBMS_OUTPUT.PUT_LINE('Employee with EmpNo ' || v_empid || ' not
found.');
 15  END;
 16  /
Enter value for empid: 7499
old   2:     v_empid emp.empno%TYPE := &empid;
new   2:     v_empid emp.empno%TYPE := 7499;
Employee Details - Name: ALLEN, Total Salary:
1900




PL/SQL procedure successfully completed.

SQL> --6
SQL> DECLARE
  2      v_max_sal emp.sal%TYPE;
  3      v_empname emp.ename%TYPE;
  4  BEGIN
  5      SELECT sal, ename INTO v_max_sal, v_empname
  6      FROM emp
  7      WHERE sal = (SELECT MAX(sal) FROM emp);
  8
  9     DBMS_OUTPUT.PUT_LINE('Highest Salary: ' || v_max_sal || ' earned by Employee: ' || v_empname);
 10  EXCEPTION
 11     WHEN NO_DATA_FOUND THEN
 12        DBMS_OUTPUT.PUT_LINE('No employees found.');
 13  END;
 14  /
Highest Salary: 5000 earned by Employee:
KING




PL/SQL procedure successfully completed.

SQL> --4
SQL> DECLARE
  2     v_rental_date DATE := TO_DATE('&rental_date', 'DD-MON-YYYY');
  3     v_rental_month NUMBER := TO_NUMBER('&rental_month');
  4     v_rental_year NUMBER := TO_NUMBER('&rental_year');
  5     v_due_days NUMBER := 3;
  6     v_return_date DATE;
  7
  8  BEGIN
  9     -- Calculate return date
 10     v_return_date := v_rental_date + v_due_days;
 11
 12     -- Print results
 13     DBMS_OUTPUT.PUT_LINE('Rental Date: ' || TO_CHAR(v_rental_date, 'DD-MON-YYYY'));
 14     DBMS_OUTPUT.PUT_LINE('Return Date: ' || TO_CHAR(v_return_date, 'DD-MON-YYYY'));
 15     DBMS_OUTPUT.PUT_LINE('Return Month: ' || TO_CHAR(v_return_date, 'MM'));
 16     DBMS_OUTPUT.PUT_LINE('Return Year: ' || TO_CHAR(v_return_date, 'YYYY'));
 17  END;
 18  /

Enter value for rental_date: 10-NOV-2023
old   2:     v_rental_date DATE := TO_DATE('&rental_date', 'DD-MON-YYYY');
new   2:     v_rental_date DATE := TO_DATE('10-NOV-2023', 'DD-MON-YYYY');
Enter value for rental_month: 11
old   3:     v_rental_month NUMBER := TO_NUMBER('&rental_month');
new   3:     v_rental_month NUMBER := TO_NUMBER('11');
Enter value for rental_year: 2023
old   4:     v_rental_year NUMBER := TO_NUMBER('&rental_year');
new   4:     v_rental_year NUMBER := TO_NUMBER('2023');
```

```
Rental Date: 10-NOV-
2023



Return Date: 13-NOV-
2023



Return Month:
11



Return Year:
2023




PL/SQL procedure successfully completed.

SQL> commit;

Commit complete.

SQL> spool off
```