EXPERIMENT 7

7) Program development using WHILE LOOPS, numeric FOR LOOPS, nested loops using ERROR Handling, BUILT –IN Exceptions, USE defined Exceptions, RAISE-APPLICATION ERROR.

Solution:

To generate first 10 natural numbers using loop, while and for

/* using loop statement */

```
Declare
        I number;
Begin
        I:=1;
        Loop
                Dbms_output.put_line(I);
                I:=I+1;
        Exit when I>10;
        End loop;
End;
```

/* using while */

```
Declare
        I number;
Begin
        I:=1;
        While (I<=10)
        loop
                Dbms_output.put_line(I);
                I:=I+1;
        End loop;
End;
```

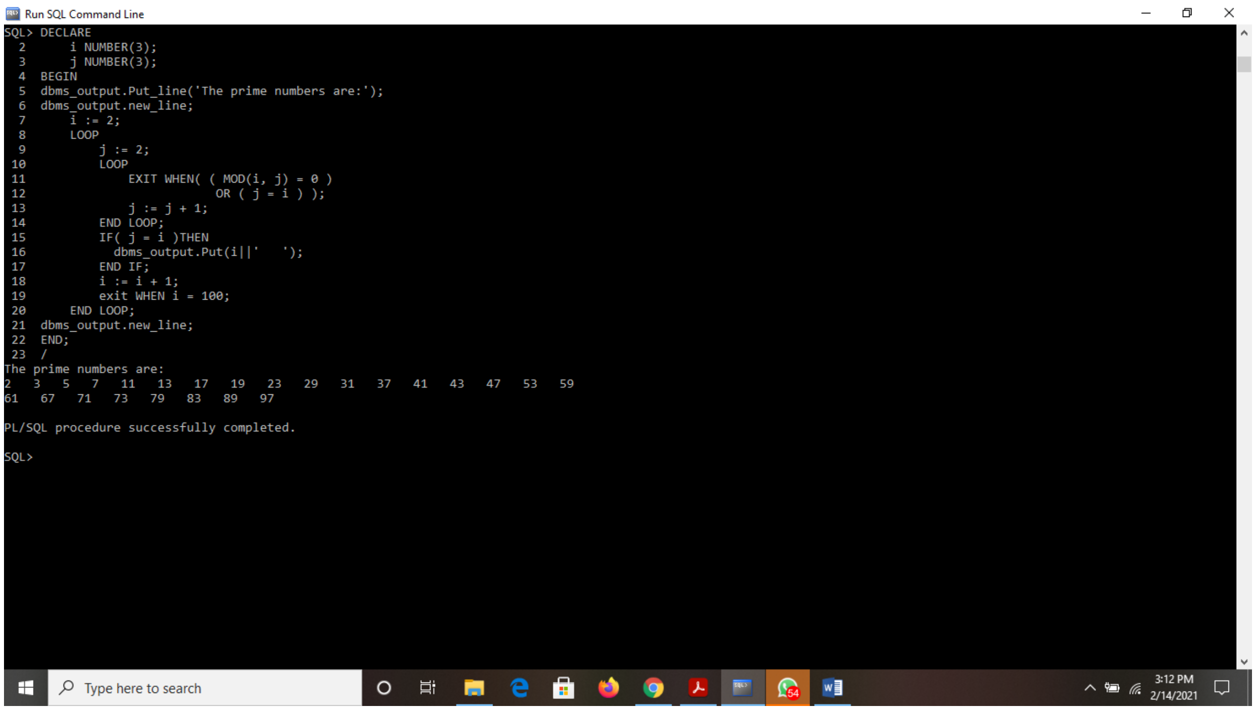
/* using for loop*/

```
Begin
        For I in 1..10
        loop
                Dbms_output.put_line(I);
        End loop;
End;
```

Write a PL/SQL program to generate all prime numbers below 100.

```
DECLARE
    i NUMBER(3);
    j NUMBER(3);
BEGIN
dbms_output.Put_line('The prime numbers are:');
     dbms_output.new_line;
    i := 2;
    LOOP
        j := 2;
        LOOP
            EXIT WHEN( ( MOD(i, j) = 0 )
                       OR ( j = i ) );
            j := j + 1;
        END LOOP;
        IF( j = i )THEN
          dbms_output.Put(i||'    ');
        END IF;
        i := i + 1;
        exit WHEN i = 100;
    END LOOP;
     dbms_output.new_line;
END;
/
```

OUTPUT:



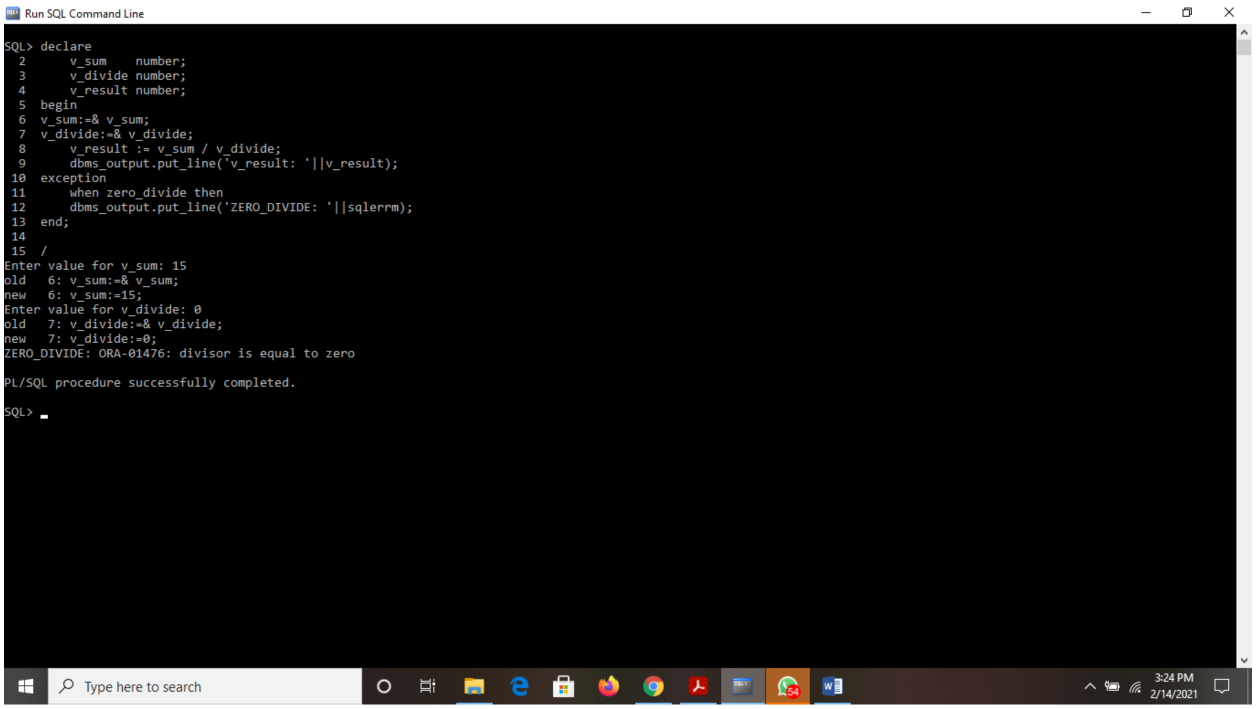
```
SQL> DECLARE
         i NUMBER(3);
         j NUMBER(3);
     BEGIN
     dbms_output.Put_line('The prime numbers are:');
     dbms_output.new_line;
         i := 2;
         LOOP
             j := 2;
             LOOP
                 EXIT WHEN( ( MOD(i, j) = 0 )
                             OR ( j = i ) );
                 j := j + 1;
             END LOOP;
             IF( j = i )THEN
               dbms_output.Put(i||'    ');
             END IF;
             i := i + 1;
             exit WHEN i = 100;
         END LOOP;
     dbms_output.new_line;
     END;
     /
The prime numbers are:
2    3    5    7    11    13    17    19    23    29    31    37    41    43    47    53    59
61    67    71    73    79    83    89    97

PL/SQL procedure successfully completed.

SQL>
```

Write a PL/SQL program to demonstrate predefined exceptions

```
declare
   v_sum     number;
   v_divide number;
   v_result number;
begin
        v_sum:=& v_sum;
        v_divide:=& v_divide;
        v_result := v_sum / v_divide;
        dbms_output.put_line('v_result: '||v_result);
exception
        when zero_divide then
        dbms_output.put_line('ZERO_DIVIDE: '||sqlerrm);
end;
```



```
SQL> declare
  2      v_sum    number;
  3      v_divide number;
  4      v_result number;
  5  begin
  6  v_sum:=& v_sum;
  7  v_divide:=& v_divide;
  8      v_result := v_sum / v_divide;
  9      dbms_output.put_line('v_result: '||v_result);
 10  exception
 11      when zero_divide then
 12      dbms_output.put_line('ZERO_DIVIDE: '||sqlerrm);
 13  end;
 14
 15  /
Enter value for v_sum: 15
old   6: v_sum:=& v_sum;
new   6: v_sum:=15;
Enter value for v_divide: 0
old   7: v_divide:=& v_divide;
new   7: v_divide:=0;
ZERO_DIVIDE: ORA-01476: divisor is equal to zero

PL/SQL procedure successfully completed.

SQL> _
```

Write a pl/sql program to demonstrate user defined exceptions?

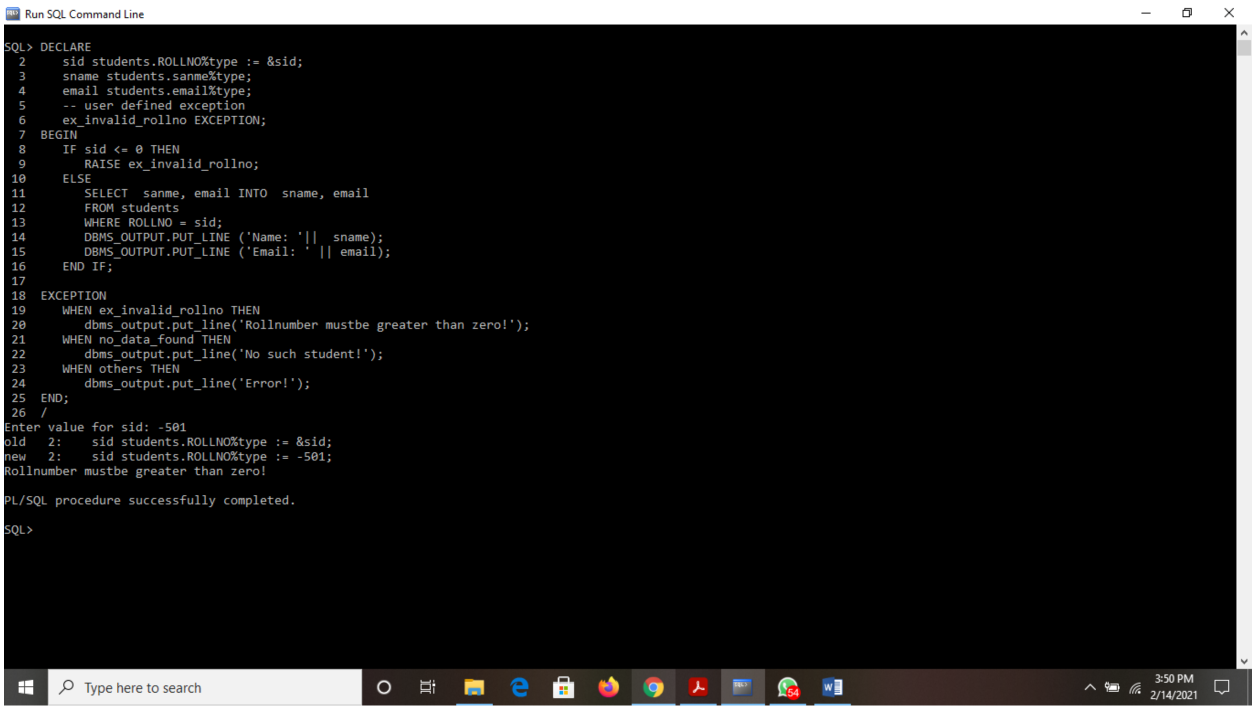
```
DECLARE
   sid students.ROLLNO%type := &sid;
   sname students.sanme%type;
   email students.email%type;
   -- user defined exception
   ex_invalid_rollno EXCEPTION;
BEGIN
   IF sid <= 0 THEN
      RAISE ex_invalid_rollno;
   ELSE
      SELECT  sanme, email INTO  sname, email
      FROM students
      WHERE ROLLNO = sid;
      DBMS_OUTPUT.PUT_LINE ('Name: '||  sname);
      DBMS_OUTPUT.PUT_LINE ('Email: ' || email);
   END IF;

EXCEPTION
   WHEN ex_invalid_rollno THEN
      dbms_output.put_line('Rollnumber mustbe greater than zero!');
   WHEN no_data_found THEN
      dbms_output.put_line('No such student!');
   WHEN others THEN
      dbms_output.put_line('Error!');
END;
/
```

Students Table:

```
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
ROLLNO                                    NOT NULL VARCHAR2(10)
SANME                                     NOT NULL VARCHAR2(20)
EMAIL                                              VARCHAR2(20)
OS                                                 NUMBER(3)
DBMS                                      NOT NULL NUMBER(3)
CD                                        NOT NULL NUMBER(3)
```

OUTPUT:



```
SQL> DECLARE
  2     sid students.ROLLNO%type := &sid;
  3     sname students.sanme%type;
  4     email students.email%type;
  5     -- user defined exception
  6     ex_invalid_rollno EXCEPTION;
  7  BEGIN
  8     IF sid <= 0 THEN
  9        RAISE ex_invalid_rollno;
 10     ELSE
 11        SELECT  sanme, email INTO  sname, email
 12        FROM students
 13        WHERE ROLLNO = sid;
 14        DBMS_OUTPUT.PUT_LINE ('Name: '||  sname);
 15        DBMS_OUTPUT.PUT_LINE ('Email: ' || email);
 16     END IF;
 17
 18  EXCEPTION
 19     WHEN ex_invalid_rollno THEN
 20        dbms_output.put_line('Rollnumber mustbe greater than zero!');
 21     WHEN no_data_found THEN
 22        dbms_output.put_line('No such student!');
 23     WHEN others THEN
 24        dbms_output.put_line('Error!');
 25  END;
 26  /
Enter value for sid: -501
old   2:    sid students.ROLLNO%type := &sid;
new   2:    sid students.ROLLNO%type := -501;
Rollnumber mustbe greater than zero!

PL/SQL procedure successfully completed.

SQL>
```

RAISE_APPLICATION_ERROR:

The procedure raise_application_error allows you to issue a user-defined error from a code block or stored program.

By using this procedure, you can report errors to the callers instead of returning unhandled exceptions.

The raise_application_error has the following syntax:

```
raise_application_error(
    error_number,
    message
    [, {TRUE | FALSE}]
);
```

In this syntax:

- The error_number is a negative integer with the range from -20999 to -20000.
- The message is a character string that represents the error message. Its length is up to 2048 bytes.
- If the third parameter is FALSE, the error replaces all previous errors. If it is TRUE, the error is added to the stack of previous errors.

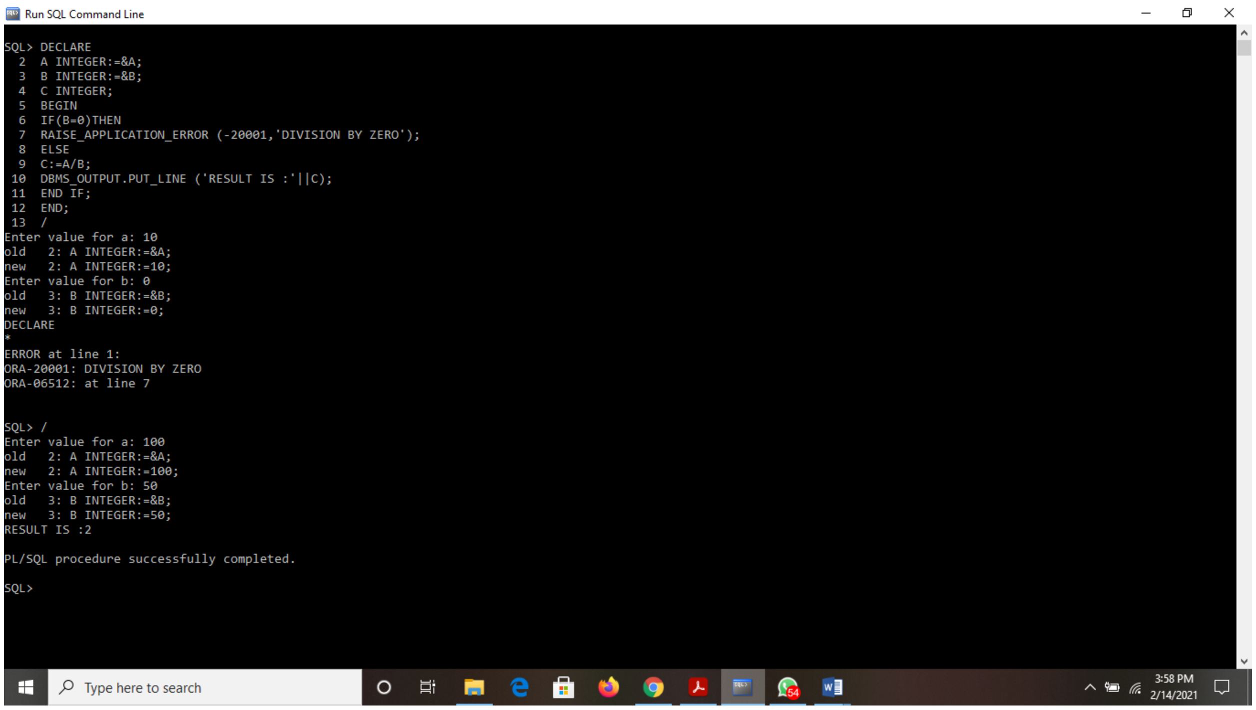
The raise_application_error belongs to the package DBMS_STANDARD, therefore, you do not need to qualify references to it.

When the procedure raise_application_error executes, Oracle halts the execution of the current block immediately. It also reverses all changes made to the OUT or IN OUT parameters.

Example Program:

```
DECLARE
        A INTEGER:=&A;
        B INTEGER:=&B;
        C INTEGER;
BEGIN
        IF(B=0)THEN
                RAISE_APPLICATION_ERROR (-20001,'DIVISION BY ZERO');
        ELSE
                C:=A/B;
                DBMS_OUTPUT.PUT_LINE ('RESULT IS :'||C);
        END IF;
END;
/
```

OUTPUT:



```
SQL> DECLARE
  2  A INTEGER:=&A;
  3  B INTEGER:=&B;
  4  C INTEGER;
  5  BEGIN
  6  IF(B=0)THEN
  7  RAISE_APPLICATION_ERROR (-20001,'DIVISION BY ZERO');
  8  ELSE
  9  C:=A/B;
 10  DBMS_OUTPUT.PUT_LINE ('RESULT IS :'||C);
 11  END IF;
 12  END;
 13  /
Enter value for a: 10
old   2: A INTEGER:=&A;
new   2: A INTEGER:=10;
Enter value for b: 0
old   3: B INTEGER:=&B;
new   3: B INTEGER:=0;
DECLARE
*
ERROR at line 1:
ORA-20001: DIVISION BY ZERO
ORA-06512: at line 7


SQL> /
Enter value for a: 100
old   2: A INTEGER:=&A;
new   2: A INTEGER:=100;
Enter value for b: 50
old   3: B INTEGER:=&B;
new   3: B INTEGER:=50;
RESULT IS :2

PL/SQL procedure successfully completed.

SQL>
```