PLSQL:

```
BEGIN DBMS_OUTPUT.PUT_LINE('I REMEMBER THE DATE');  END;

I REMEMBER THE DATE

Statement processed. 0.00 seconds
```

DECLARATION:

```
DECLARE V_DATE DATE:=SYSDATE; BEGIN  DBMS_OUTPUT.PUT_LINE(V_DATE); END;

12-Aug-2024

Statement processed. 0.00 seconds
```

DECLARATION EXECTUTABLE AMD EXCEPTION:

```
1   CREATE TABLE ONEMORE(FIRST_NAME CHAR(30),LAST_NAME CHAR(30));
2   INSERT INTO ONEMORE VALUES('UZHMAA','ADA');
3   INSERT INTO ONEMORE VALUES('ARSH','BAKAR');
4   INSERT INTO ONEMORE VALUES('SRIDHAR','YOGI');
5   DECLARE
6   V_FIRST_NAME CHAR(30);
7   V_LAST_NAME CHAR(30);
8   BEGIN
9   SELECT FIRST_NAME,LAST_NAME
10  INTO V_FIRST_NAME,V_LAST_NAME
11  FROM ONEMORE
12  WHERE LAST_NAME = 'ADA';
13  DBMS_OUTPUT.PUT_LINE ('The employee of the month is: '
14  || V_FIRST_NAME || ' ' || V_LAST_NAME || '.');
15  EXCEPTION
16  WHEN TOO_MANY_ROWS THEN
17  DBMS_OUTPUT.PUT_LINE ('Your select statement retrieved
18  multiple rows. Consider using a cursor or changing
19  the search criteria.');
20  END;
21
```

```
month is: ' || V_FIRST_NAME || ' ' || V_LAST_NAME || '
the search criteria.'); END;

The employee of the month is: UZHMAA ADA .

Statement processed. 0.01 seconds
```

ADDING:

```
1   DECLARE
2   A INTEGER:=11;
3   B INTEGER:=11;
4   C INTEGER;
5   F REAL;
6   BEGIN
7   C:A+B;
8   DBMS_OUTPUT.PUT_LINE('VALUE OF C = '||C);
9
10  F:=60.0/3.0;
11  DBMS_OUTPUT.PUT_LINE('VALUE OF F = '||F);
12  END;
```

```
DECLARE A INTEGER:=11; B INTEGER:=11; C INTEGER; F REAL; BEGIN C:=A+B; DBMS_OUTPUT.PUT_LINE('VALUE OF C = '||C);  F:=60.0/3.0; DBMS_OUTPUT.PUT_LINE('VALUE OF F = '||F); END;

VALUE OF C = 22
VALUE OF F = 20

Statement processed. 0.01 seconds
```

```
DECLARE
    -- constant declaration
    pi constant number := 3.141592654;
    -- other declarations
    radius number(5,2);
    dia number(5,2);
    circumference number(7, 2);
    area number (10, 2);
BEGIN
    -- processing
    radius := 9.5;
    dia := radius * 2;
    circumference := 2.0 * pi * radius;
    area := pi * radius * radius;
    -- output
    dbms_output.put_line('Radius: ' || radius);
    dbms_output.put_line('Diameter: ' || dia);
    dbms_output.put_line('Circumference: ' || circumference);
    dbms_output.put_line('Area: ' || area);
END;
```

```
Radius: 9.5
Diameter: 19
Circumference: 59.69
Area: 283.53

Statement processed. 0.01 seconds
```

Count words:

```
DECLARE
      str VARCHAR2(40) := 'Tutorials Point';
      nchars NUMBER(4) := 0;
      nwords NUMBER(4) := 1;
      s CHAR;
BEGIN
   FOR i IN 1..Length(str) LOOP
      s := Substr(str, i, 1);
      nchars:= nchars+ 1;
      IF s = ' ' THEN
      nwords := nwords + 1;
      END IF;
END LOOP;
dbms_output.Put_line('count of characters is:'
   ||nchars);

dbms_output.Put_line('Count of words are: '
   ||nwords);
END;
```

```
DECLARE str VARCHAR2(40) := 'Tutorials Point'; nchars NUMBER(4) := 0; nwords NUMBER(4) := 1; s CHAR; BEGIN FOR i IN 1..Length(str) LOOP s := Substr(str, i, 1);
nchars:= nchars+ 1; IF s = ' ' THEN nwords := nwords + 1; END IF; END LOOP; dbms_output.Put_line('count of characters is:' ||nchars); dbms_output.Put_line('Count of words are:
||nwords); END;

count of characters is:15
Count of words are: 2

Statement processed. 0.00 seconds
```

Sum of numbers:

```plsql
DECLARE
    n NUMBER := 10;
    sum_num NUMBER := 0;
BEGIN
    FOR i IN 1..n LOOP
        sum_num := sum_num +i;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('The sum of numbers from 1 to ' || n || ' is: ' || sum_num);
END;
```

```
DECLARE     n NUMBER := 10;     sum_
|| sum_num); END;

The sum of numbers from 1 to 10 is: 55

Statement processed. 0.01 seconds
```

Array:

```plsql
DECLARE
    type namesarray IS VARRAY(5) OF VARCHAR2(10);
    type grades IS VARRAY(5) OF INTEGER;
    names namesarray;
    marks grades;
    total integer;
BEGIN
    names := namesarray('Kavita', 'Pritam', 'Ayan', 'Rishav', 'Aziz');
    marks:= grades(98, 97, 78, 87, 92);
    total := names.count;
    dbms_output.put_line('Total '|| total || ' Students');
    FOR i in 1 .. total LOOP
      dbms_output.put_line('Student: ' || names(i) || '
      Marks: ' || marks(i));
    END LOOP;
END;
```

```
Total 5 Students
Student: Kavita Marks: 98
Student: Pritam Marks: 97
Student: Ayan Marks: 78
Student: Rishav Marks: 87
Student: Aziz Marks: 92

Statement processed. 0.00 seconds
```

With grade:

```
DECLARE
    type namesarray IS VARRAY(5) OF VARCHAR2(10);
    type grades IS VARRAY(5) OF INTEGER;
    names namesarray;
    marks grades;
    total integer;
     FUNCTION get_grade(mark INTEGER) RETURN VARCHAR2 IS
    BEGIN
      IF mark >= 90 THEN
        RETURN 'A';
      ELSIF mark >= 80 THEN
        RETURN 'B';
      ELSIF mark >= 70 THEN
        RETURN 'C';
      ELSIF mark >= 60 THEN
        RETURN 'D';
      ELSE
        RETURN 'F';
      END IF;
    END;
BEGIN
    names := namesarray('Kavita', 'Pritam', 'Ayan', 'Rishav', 'Aziz');
    marks:= grades(98, 97, 78, 87, 92);
    total := names.count;
    dbms_output.put_line('Total '|| total || ' Students');
    FOR i IN 1 .. total LOOP
      dbms_output.put_line('Student: ' || names(i) ||
                           ' Marks: ' || marks(i) ||
                           ' Grade: ' || get_grade(marks(i)));
    END LOOP;
END;
```

```
Total 5 Students
Student: Kavita Marks: 98 Grade: A
Student: Pritam Marks: 97 Grade: A
Student: Ayan Marks: 78 Grade: C
Student: Rishav Marks: 87 Grade: B
Student: Aziz Marks: 92 Grade: A

Statement processed. 0.01 seconds
```

Odd and even count:

```
DECLARE
   number      INTEGER := 23146579;
   digit       INTEGER;
   temp_num    INTEGER := number;
   odd_count   INTEGER := 0;
   even_count  INTEGER := 0;
BEGIN
   WHILE temp_num > 0 LOOP
      digit := temp_num MOD 10;
      temp_num := temp_num / 10;
      IF digit MOD 2 = 0 THEN
         even_count := even_count + 1;
      ELSE
         odd_count := odd_count + 1;
      END IF;
   END LOOP;

   dbms_output.put_line('Count of odd digits: ' || odd_count);
   dbms_output.put_line('Count of even digits: ' || even_count);
END;
/
```

```
Count of odd digits: 4
Count of even digits: 4
```

Minimum number:

```
DECLARE
    a number;
    b number;
    c number;
PROCEDURE findMin(x IN number, y IN number, z OUT number) IS
BEGIN
    IF x < y THEN
        z:= x;
    ELSE
        z:= y;
    END IF;
END;
BEGIN
    a:= 23;
    b:= 45;
    findMin(a, b, c);
    dbms_output.put_line(' Minimum of (23, 45) : ' || c);
END;
```

```
Minimum of (23, 45) : 23
```

Calculator:

```
DECLARE
   a NUMBER;
   b NUMBER;
   addi NUMBER;
   c CHAR(1);
PROCEDURE adding(x IN NUMBER, y IN NUMBER,z IN CHAR,result out NUMBER ) IS
BEGIN
   IF z ='+'  THEN
    result:=x+y;
   ELSIF z='-' THEN
     result:=x-y;
    ELSIF Z='*' THEN
      result:=x*y;
    END IF;
END;
BEGIN
   a:= 23;
   b:= 45;
   c:='-';
   adding(a, b,c,addi);
   dbms_output.put_line('addition of a and b is'|| addi);
END;
```

output

```
addition of a and b is-22
```

```
DECLARE
   a NUMBER;
   b NUMBER;
   addi NUMBER;
   c CHAR(1);
PROCEDURE adding(x IN NUMBER, y IN NUMBER,z IN CHAR,result out NUMBER ) IS
BEGIN
   IF z ='+'  THEN
    result:=x+y;
   ELSIF z='-' THEN
     result:=x-y;
    ELSIF Z='*' THEN
      result:=x*y;
    END IF;
END;
BEGIN
   a:= 23;
   b:= 45;
   c:='-';
   adding(a, b,c,addi);
   dbms_output.put_line('addition of a and b is'|| addi);
END;
```

FACTORIAL OF A NUMBER:

```
DECLARE
   num number;
   factorial number;

FUNCTION fact(x number)
RETURN number
IS
   f number;
BEGIN
   IF x=0 THEN
      f := 1;
   ELSE
      f := x * fact(x-1);
   END IF;
RETURN f;
END;

BEGIN
   num:= 6;
   factorial := fact(num);
   dbms_output.put_line(factorial);
END;
```

```
Factorial 6 is 720
```

Max using function:

```
DECLARE
    a_num NUMBER;
    b_num NUMBER;
    c_num NUMBER;
FUNCTION findmax(x IN number, y IN number)
RETURN number
IS
    z number;
BEGIN
    IF x>y then
    z:=x;
    else
    z:=y;
    end if;
    return z;
END;
begin
a_num:=23;
b_num:=45;
c_num:=findmax(a_num,b_num);
dbms_output.put_line(c_num);
end;
```

```
45
```

Fibonacci series:

```
DECLARE
num number:=10;
function fibo(n IN number) return number is
begin
if n=0 then
return 0;
elsif n=1 then
return 1;
else
return fibo(n-1)+fibo(n-2);
end if;
end;
begin
for i in 0..num loop
dbms_output.put_line(fibo(i));
end loop;
end;
```

```
0
1
1
2
3
5
8
13
21
34
55
```

IMPLICITY:

```
DECLARE
total_rows number(2);
BEGIN
UPDATE EMPLOYEES1
SET salary = salary + 500;
IF sql%notfound THEN
dbms_output.put_line('NO  EMPLOYEES SELECTED');
ELSIF sql%found THEN
total_rows := sql%rowcount;
dbms_output.put_line( total_rows || ' EMPLOYEES SELECTED ');
END IF;
END;
```

```
8 EMPLOYEES SELECTED
```

EXPLICITY:

```
DECLARE
c_emp_id EMPLOYEES1.EMPLOYEE_ID%TYPE;
c_emp_name EMPLOYEES1.EMPLOYEE_NAME%TYPE;
c_dept EMPLOYEES1.DEPARTMENT%TYPE;
c_salary EMPLOYEES1.SALARY%TYPE;
CURSOR c_employee IS
SELECT EMPLOYEE_ID, EMPLOYEE_NAME, DEPARTMENT, SALARY FROM EMPLOYEES1;
BEGIN
OPEN c_employee;
LOOP
FETCH c_employee INTO c_emp_id, c_emp_name, c_dept, c_salary;
EXIT WHEN c_employee%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(c_emp_id   ||   ' '   || c_emp_name ||   ' '   || c_dept ||  ' '   || c_salary);
END LOOP;
CLOSE c_employee;
END;
```

455689 SRIKANTH CSE 11000
465433 SRIDHAR AI 26000
2898975 UDAY AIDS 6000
2565649 SRIPATHI AIML 7000
455689 SRIKANTH CSE 11000
465433 SRIDHAR AI 26000
2898975 UDAY AIDS 6000
2565649 SRIPATHI AIML 7000

From the Cursor:

```sql
CREATE TABLE employees_cursor(
    employee_id NUMBER PRIMARY KEY,
    last_name VARCHAR2(50),
    department_id NUMBER
);

INSERT INTO employees_cursor(employee_id, last_name, department_id) VALUES (1, 'Smith', 30);
INSERT INTO employees_cursor(employee_id, last_name, department_id) VALUES (2, 'Johnson', 30);
INSERT INTO employees_cursor(employee_id, last_name, department_id) VALUES (3, 'Williams', 40);
INSERT INTO employees_cursor(employee_id, last_name, department_id) VALUES (4, 'Jones', 30);
```

```sql
SELECT * FROM employees_cursor
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	Smith	30
2	Johnson	30
3	Williams	40
4	Jones	30

```
DECLARE
    CURSOR c_emp_cursor IS
        SELECT employee_id, last_name FROM employees_cursor
        WHERE department_id = 30;
    v_empno  employees_cursor.employee_id%TYPE;
    v_lname  employees_cursor.last_name%TYPE;
BEGIN
    OPEN c_emp_cursor;
    LOOP
        FETCH c_emp_cursor INTO v_empno, v_lname;
        EXIT WHEN c_emp_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE (v_empno || ' ' || v_lname);
    END LOOP;
    CLOSE c_emp_cursor;
END;
```

Into a record:

```
DECLARE
    CURSOR c_emp_cursor IS
        SELECT employee_id, last_name FROM employees_cursor
        WHERE department_id = 30;

    v_emp_record c_emp_cursor%ROWTYPE;
BEGIN
    OPEN c_emp_cursor;
    LOOP
        FETCH c_emp_cursor INTO v_emp_record;
        EXIT WHEN c_emp_cursor%NOTFOUND; -- Moved this line after FETCH
        DBMS_OUTPUT.PUT_LINE(v_emp_record.employee_id || ' ' || v_emp_record.last_name);
    END LOOP;
    CLOSE c_emp_cursor;
END;
```

```
1 Smith
2 Johnson
4 Jones
```

Using row count and notfound attributes:

```
DECLARE
    CURSOR c_emp_cursor IS
      SELECT employee_id, last_name FROM employees_cursor
      WHERE department_id = 30;
    v_emp_record  c_emp_cursor%ROWTYPE;
BEGIN
OPEN c_emp_cursor;
LOOP
    FETCH c_emp_cursor INTO v_emp_record;
    EXIT WHEN c_emp_cursor%ROWCOUNT > 10 OR c_emp_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE (v_emp_record.employee_id || ' ' ||
      v_emp_record.last_name);
    END LOOP;
    CLOSE c_emp_cursor;
END;
```

```
1 Smith
2 Johnson
4 Jones
```

Use a for curser loop:

```
DECLARE
        CURSOR c_emp_cursor IS
         SELECT employee_id, last_name FROM employees_cursor
         WHERE department_id = 30;
      BEGIN
        FOR emp_record IN c_emp_cursor
         LOOP
           DBMS_OUTPUT.PUT_LINE (emp_record.employee_id || ' ' ||
              emp_record.last_name);
         END LOOP;
      END;
BEGIN
       FOR emp_record IN (SELECT employee_id, last_name FROM
        employees_cursor WHERE department_id = 30)
       LOOP
         DBMS_OUTPUT.PUT_LINE (emp_record.employee_id || ' ' ||
           emp_record.last_name);
        END LOOP;
      END;
```

```
1 Smith
2 Johnson
4 Jones
1 Smith
2 Johnson
4 Jones
```

CREATING A TABLE AND INSERTING 10 VALUES:

```
CREATE TABLE employees_thurs(
    employee_id NUMBER PRIMARY KEY,
    employee_name VARCHAR2(100),
    salary NUMBER
);
BEGIN
    INSERT INTO employees_thurs (employee_id, employee_name, salary) VALUES (1, 'UZHMAA', 50000);
    INSERT INTO employees_thurs(employee_id, employee_name, salary) VALUES (2, 'ARSH', 60000);
    INSERT INTO employees_thurs(employee_id, employee_name, salary) VALUES (3, 'AMEENA', 55000);
    INSERT INTO employees_thurs(employee_id, employee_name, salary) VALUES (4, 'FARA', 70000);
    INSERT INTO employees_thurs(employee_id, employee_name, salary) VALUES (5, 'KHALEEM', 45000);
    INSERT INTO employees_thurs(employee_id, employee_name, salary) VALUES (6, 'SABIRA', 80000);
    INSERT INTO employees_thurs(employee_id, employee_name, salary) VALUES (7, 'SANIYA', 65000);
    INSERT INTO employees_thurs(employee_id, employee_name, salary) VALUES (8, 'HAMZA', 75000);
    INSERT INTO employees_thurs(employee_id, employee_name, salary) VALUES (9, 'YASMEEN', 90000);
    INSERT INTO employees_thurs(employee_id, employee_name, salary) VALUES (10, 'SADIYA', 30000);
    COMMIT;
END;
```

Ascending order:

```
DECLARE
    CURSOR asc_cursor IS
        SELECT * FROM employees_thurs ORDER BY salary ASC;
    emp_record employees_thurs%ROWTYPE;
BEGIN
    OPEN asc_cursor;
    LOOP
        FETCH asc_cursor INTO emp_record;
        EXIT WHEN asc_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(emp_record.employee_id || ' - ' || emp_record.employee_name);
    END LOOP;
    CLOSE asc_cursor;
END;
```

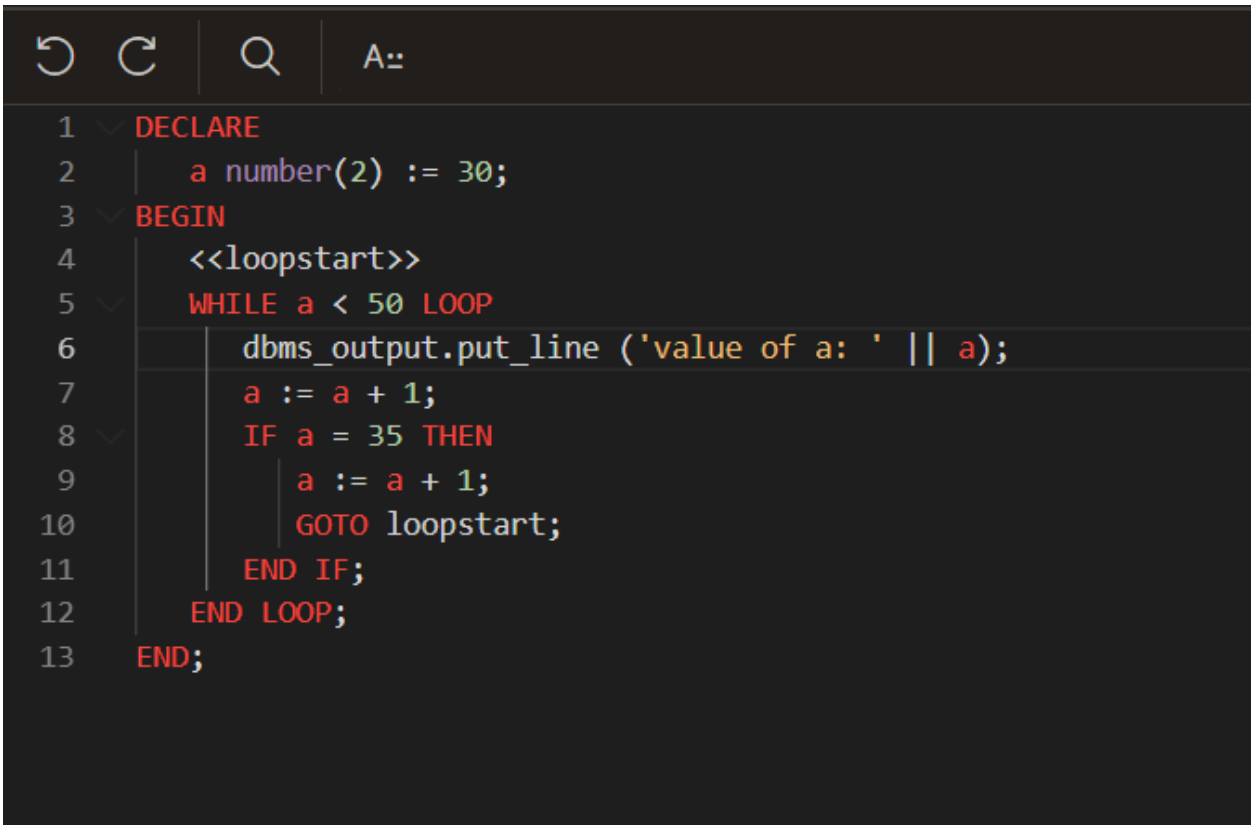
```
10 - SADIYA
5 - KHALEEM
1 - UZHMAA
3 - AMEENA
2 - ARSH
7 - SANIYA
4 - FARA
8 - HAMZA
6 - SABIRA
9 - YASMEEN
```

Descending order:

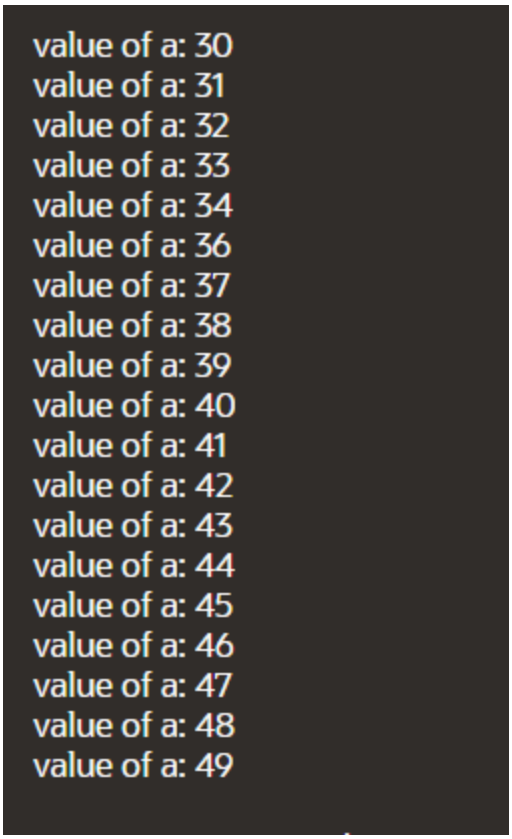
```
DECLARE
    CURSOR desc_cursor IS
        SELECT * FROM employees_thurs ORDER BY salary DESC;
    emp_record employees_thurs%ROWTYPE;
BEGIN
    OPEN desc_cursor;
    LOOP
        FETCH desc_cursor INTO emp_record;
        EXIT WHEN desc_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(emp_record.employee_id || ' - ' || emp_record.employee_name);
    END LOOP;
    CLOSE desc_cursor;
END;
```

```
9 - YASMEEN
6 - SABIRA
8 - HAMZA
4 - FARA
7 - SANIYA
2 - ARSH
3 - AMEENA
1 - UZHMAA
5 - KHALEEM
10 - SADIYA
```

Go to:



```
DECLARE
   a number(2) := 30;
BEGIN
   <<loopstart>>
   WHILE a < 50 LOOP
      dbms_output.put_line ('value of a: ' || a);
      a := a + 1;
      IF a = 35 THEN
         a := a + 1;
         GOTO loopstart;
      END IF;
   END LOOP;
END;
```


value of a: 30
value of a: 31
value of a: 32
value of a: 33
value of a: 34
value of a: 36
value of a: 37
value of a: 38
value of a: 39
value of a: 40
value of a: 41
value of a: 42
value of a: 43
value of a: 44
value of a: 45
value of a: 46
value of a: 47
value of a: 48
value of a: 49

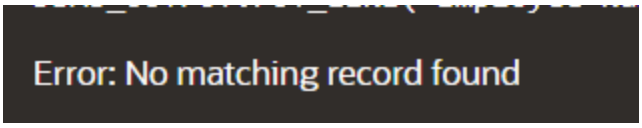
EXCEPTION:

```
1   DECLARE
2   divident NUMBER:=10;
3   divisor NUMBER:=0;
4   result NUMBER;
5   BEGIN
6   BEGIN
7   result:=divident/divisor;
8   DBMS_OUTPUT.PUT_LINE('result : '||result);
9   EXCEPTION
10  WHEN ZERO_DIVIDE THEN
11  DBMS_OUTPUT.PUT_LINE('errror cannot deivide with zero');
12  END;
13  END;
```

```
errror cannot deivide with zero
```

No matching data found while retrieving the data:

```
1   DECLARE
2      emp_name   VARCHAR2(100);
3   emp_id     NUMBER := 225;
4   BEGIN
5      BEGIN
6         SELECT employee_name INTO emp_name FROM employees_thurs WHERE employee_id = emp_id;
7         DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_name);
8      EXCEPTION
9         WHEN NO_DATA_FOUND THEN
10           DBMS_OUTPUT.PUT_LINE('Error: No matching record found');
11     END;
12  END;
```



Multiple rows exception:

```plsql
DECLARE
  v_emp_id employees_thurs.employee_id%TYPE;
  v_emp_name employees_thurs.employee_name%TYPE;
  v_emp_salary employees_thurs.salary%TYPE;
  v_exception_msg VARCHAR2(200);
BEGIN
  SELECT employee_name, employee_id, salary
  INTO v_emp_name,v_dep_id, v_emp_salary
  FROM employees_thurs
  WHERE employee_id = v_emp_id;
  EXCEPTION
    WHEN TOO_MANY_ROWS THEN
      v_exception_msg := 'Multiple rows found for the given department ID: ' || v_dep_id;
      DBMS_OUTPUT.PUT_LINE(v_exception_msg);
    WHEN OTHERS THEN
      v_exception_msg := 'An error occurred: ' || SQLERRM;
      DBMS_OUTPUT.PUT_LINE(v_exception_msg);
END;
```