2.

DECLARE

num\_small NUMBER := 8;

num\_large NUMBER := 5;

num\_temp NUMBER;

BEGIN

IF num\_small > num\_large THEN

num\_temp := num\_small;

num\_small := num\_large;

num\_large := num\_temp;

END IF;

DBMS\_OUTPUT.PUT\_LINE ('num\_small = '||num\_small);

DBMS\_OUTPUT.PUT\_LINE ('num\_large = '||num\_large);

END;

3.

DECLARE

n1 NUMBER := &num1;

BEGIN

-- test if the number provided by the user is even

IF MOD(n1,2) = 0 THEN

DBMS\_OUTPUT.PUT\_LINE ('The number. '||n1||

' is even number');

ELSE

DBMS\_OUTPUT.PUT\_LINE ('The number '||n1||' is odd number.');

END IF;

DBMS\_OUTPUT.PUT\_LINE ('Done Successfully');

END;

4.

DECLARE

dt1 DATE := TO\_DATE('&new\_dt', 'DD-MON-YYYY');

get\_day VARCHAR2(15);

BEGIN

get\_day := RTRIM(TO\_CHAR(dt1, 'DAY'));

IF get\_day IN ('SATURDAY', 'SUNDAY') THEN

dbms\_output.new\_line;

DBMS\_OUTPUT.PUT\_LINE

('The day of the given date is '||get\_day||' and it falls on weekend');

ELSE

dbms\_output.new\_line;

DBMS\_OUTPUT.PUT\_LINE ('The day of the given date is '||get\_day||' and it does not fall on the weekend');

END IF;

DBMS\_OUTPUT.PUT\_LINE ('Execution done successfully.');

END;

5.

DECLARE

PROCEDURE test1 (sal\_achieve NUMBER)

IS

incentive NUMBER := 0;

BEGIN

IF sal\_achieve > 44000 THEN

incentive := 1800;

ELSIF sal\_achieve > 32000 THEN

incentive := 800;

ELSE

incentive := 500;

END IF;

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.PUT\_LINE (

'Sale achieved : ' || sal\_achieve || ', incentive : ' || incentive || '.'

);

END test1;

BEGIN

test1(45000);

test1(36000);

test1(28000);

END;

6.

SET SERVEROUTPUT ON

DECLARE

tot\_emp NUMBER;

BEGIN

SELECT Count(\*)

INTO tot\_emp

FROM employees e

join departments d

ON e.department\_id = d.department\_id

WHERE e.department\_id = 50;

dbms\_output.Put\_line ('The employees are in the department 50: '

||To\_char(tot\_emp));

IF tot\_emp >= 45 THEN

dbms\_output.Put\_line ('There are no vacancies in the department 50.');

ELSE

dbms\_output.Put\_line ('There are some vacancies in department 50.');

END IF;

END;

7.

DECLARE

grd CHAR(1);

BEGIN

-- Accept value for grade

grd := '&new\_grd';

IF grd = 'A' THEN

dbms\_output.Put\_line('Your Grade is: Outstanding');

ELSIF grd = 'B' THEN

dbms\_output.Put\_line('Your Grade is: Excellent');

ELSIF grd = 'C' THEN

dbms\_output.Put\_line('Your Grade is: Very Good');

ELSIF grd = 'D' THEN

dbms\_output. Put\_line('Your Grade is: Average');

ELSIF grd = 'F' THEN

dbms\_output.Put\_line('Your Grade is: Poor');

ELSE

dbms\_output.Put\_line('No such grade in the list.');

END IF;

END;

DECLARE

grd CHAR(1);

BEGIN

-- Accept value for grade

grd := '&new\_grd';

CASE grd

WHEN 'A' THEN dbms\_output.Put\_line('Your Grade is: Outstanding');

WHEN 'B' THEN dbms\_output.Put\_line('Your Grade is: Excellent');

WHEN 'C' THEN dbms\_output.Put\_line('Your Grade is: Very Good');

WHEN 'D' THEN dbms\_output. Put\_line('Your Grade is: Average');

WHEN 'F' THEN dbms\_output.Put\_line('Your Grade is: Poor');

ELSE dbms\_output.Put\_line('No such grade in the list.');

END CASE;

END;

--- with exception

DECLARE

grd CHAR(1);

BEGIN

-- Accept value for grade

grd := '&new\_grd';

CASE

WHEN grd = 'A' THEN dbms\_output.Put\_line('Your Grade is: Outstanding');

WHEN grd = 'B' THEN dbms\_output.Put\_line('Your Grade is: Excellent');

WHEN grd = 'C' THEN dbms\_output.Put\_line('Your Grade is: Very Good');

WHEN grd = 'D' THEN dbms\_output.Put\_line('Your Grade is: Average');

WHEN grd = 'F' THEN dbms\_output.Put\_line('Your Grade is: Poor');

END CASE;

EXCEPTION

WHEN CASE\_NOT\_FOUND THEN

dbms\_output.Put\_line('No such grade in the list.');

END;

8.

DECLARE

get\_ctr CHAR(1) := '&input\_a\_character';

BEGIN

IF ( get\_ctr >= 'A'

AND get\_ctr <= 'Z' )

OR ( get\_ctr >= 'a'

AND get\_ctr <= 'z' ) THEN

dbms\_output.Put\_line ('The given character is a letter');

ELSE

dbms\_output.Put\_line ('The given character is not a letter');

IF get\_ctr BETWEEN '0' AND '9' THEN

dbms\_output.Put\_line ('The given character is a number');

ELSE

dbms\_output.Put\_line ('The given character is not a number');

END IF;

END IF;

END;

9.

DECLARE

temp1 NUMBER := &input\_a\_temp;

t\_scale CHAR := '&input\_temp\_scale';

new\_temp NUMBER;

new\_scale CHAR;

BEGIN

IF t\_scale != 'C'

AND

t\_scale != 'F' THEN

dbms\_output.Put\_line ('The scale you input is not a valid scale');

new\_temp := 0;

new\_scale := 'C';

ELSE

IF t\_scale = 'C' THEN

new\_temp := ( ( 9 \* temp1 ) / 5 ) + 32;

new\_scale := 'F';

ELSE

new\_temp := ( ( temp1 - 32 ) \* 5 ) / 9;

new\_scale := 'C';

END IF;

END IF;

dbms\_output.Put\_line ('The new temperature in scale '

||new\_scale

||' is: '

||new\_temp);

END;

10.

SET serveroutput ON

DECLARE

t\_dt DATE := To\_date('&input\_a\_date', 'DD-MON-YYYY');

t\_day VARCHAR2(1);

BEGIN

t\_day := To\_char(t\_dt, 'D');

CASE t\_day

WHEN '1' THEN

dbms\_output.Put\_line ('The date you entered is Sunday.');

WHEN '2' THEN

dbms\_output.Put\_line ('The date you entered is Monday.');

WHEN '3' THEN

dbms\_output.Put\_line ('The date you entered is Tuesday.');

WHEN '4' THEN

dbms\_output.Put\_line ('The date you entered is Wednesday.');

WHEN '5' THEN

dbms\_output.Put\_line ('The date you entered is Thursday.');

WHEN '6' THEN

dbms\_output.Put\_line ('The date you entered is Friday.');

WHEN '7' THEN

dbms\_output.Put\_line ('The date you entered is Saturday.');

END CASE;

END;

11.

DECLARE

emp\_min\_salary NUMBER(6,0);

emp\_max\_salary NUMBER(6,0);

emp\_mid\_salary NUMBER(6,2);

tmp\_salary EMPLOYEES.SALARY%TYPE;

tmp\_emp\_id EMPLOYEES.EMPLOYEE\_ID%TYPE := 167;

tmp\_emp\_name EMPLOYEES.FIRST\_NAME%TYPE;

BEGIN

SELECT min\_salary,

max\_salary

INTO emp\_min\_salary,

emp\_max\_salary

FROM JOBS

WHERE JOB\_ID = (SELECT JOB\_ID

FROM EMPLOYEES

WHERE EMPLOYEE\_ID = tmp\_emp\_id);

-- calculate mid-range

emp\_mid\_salary := (emp\_min\_salary + emp\_max\_salary) / 2;

-- get salary of the given employee

SELECT salary,first\_name

INTO tmp\_salary,tmp\_emp\_name

FROM employees

WHERE employee\_id = tmp\_emp\_id;

-- update salary

IF tmp\_salary < emp\_mid\_salary THEN

UPDATE employees

SET salary = emp\_mid\_salary

WHERE employee\_id = tmp\_emp\_id;

ELSE

UPDATE employees

SET salary = salary + salary \* 8 /100

WHERE employee\_id = tmp\_emp\_id;

END IF;

--display message

IF tmp\_salary > emp\_mid\_salary THEN

DBMS\_OUTPUT.PUT\_LINE('The employee '||tmp\_emp\_name||' ID ' || TO\_CHAR(tmp\_emp\_id) ||

' works in salary ' || TO\_CHAR(tmp\_salary) ||

' which is higher than mid-range of salary ' || TO\_CHAR(emp\_mid\_salary));

ELSIF tmp\_salary < emp\_mid\_salary THEN

DBMS\_OUTPUT.PUT\_LINE('The employee '||tmp\_emp\_name||' ID ' || TO\_CHAR(tmp\_emp\_id) ||

' works in salary ' || TO\_CHAR(tmp\_salary) ||

' which is lower than mid-range of salary ' || TO\_CHAR(emp\_mid\_salary));

ELSE

DBMS\_OUTPUT.PUT\_LINE('The employee '||tmp\_emp\_name||' ID ' || TO\_CHAR(tmp\_emp\_id) ||

' works in salary ' || TO\_CHAR(tmp\_salary) ||

' which is equal to the mid-range of salary ' || TO\_CHAR(emp\_mid\_salary));

END IF;

END;

12.

DECLARE

n number:= &first\_n\_number;

BEGIN

DBMS\_OUTPUT.PUT\_LINE ('The first '||n||' numbers are: ');

for i in 1..n loop

dbms\_output.put(i||' ');

END LOOP;

dbms\_output.new\_line;

END;

13.

DECLARE

n number:= &first\_n\_number;

i number:=1;

m number:=1;

BEGIN

DBMS\_OUTPUT.PUT\_LINE ('The first '||n||' numbers are: ');

DBMS\_OUTPUT.PUT (i||' ');

for i in 1..n-1 loop

m:=m+3;

dbms\_output.put(m||' ');

END LOOP;

dbms\_output.new\_line;

END;

14.

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 = 50;

END LOOP;

dbms\_output.new\_line;

END;

15.

DECLARE

msg VARCHAR2(30);

n PLS\_INTEGER := 83;

BEGIN

FOR i in 2..ROUND(SQRT(n)) LOOP

IF n MOD i = 0 THEN

msg := ' is not a prime number';

GOTO when\_prime;

END IF;

END LOOP;

msg := ' is a prime number';

<>

DBMS\_OUTPUT.PUT\_LINE(TO\_CHAR(n) || msg);

END;

16.

DROP TABLE emp\_temp;

CREATE TABLE emp\_temp AS

SELECT employee\_id, first\_name, last\_name,email

FROM employees;

BEGIN

UPDATE emp\_temp

SET email = 'not available'

WHERE first\_name LIKE 'B%';

dbms\_output.Put\_line('Number of record updated: '

||To\_char(SQL%rowcount));

END;

17.

DROP TABLE emp\_temp;

CREATE TABLE emp\_temp AS

SELECT employee\_id, first\_name, last\_name

FROM employees;

CREATE OR REPLACE PROCEDURE test\_proc (

z\_emp\_id NUMBER

) AUTHID DEFINER AS

BEGIN

DELETE FROM emp\_temp

WHERE employee\_id = z\_emp\_id;

IF SQL%FOUND THEN

DBMS\_OUTPUT.PUT\_LINE (

'Delete succeeded for employee\_id: ' || z\_emp\_id

);

ELSE

DBMS\_OUTPUT.PUT\_LINE ('No employee of ID '|| z\_emp\_id||'is found.');

END IF;

END;

/

BEGIN

test\_proc(175);

test\_proc(444);

END;

18.

DROP TABLE emp\_temp;

CREATE TABLE emp\_temp AS

SELECT employee\_id, first\_name, last\_name,email

FROM employees;

DECLARE

z\_emp\_id NUMBER:=&employee\_id;

BEGIN

UPDATE emp\_temp

SET email = 'not available'

WHERE employee\_id = z\_emp\_id;

IF SQL%NOTFOUND THEN

DBMS\_OUTPUT.PUT\_LINE ('No employee of ID '|| z\_emp\_id||' is found.');

ELSE

DBMS\_OUTPUT.PUT\_LINE (

'Update succeeded for employee\_id: ' || z\_emp\_id

);

END IF;

END;

19.

DECLARE

z\_employee employees%ROWTYPE;

BEGIN

SELECT \*

INTO z\_employee -- INTO clause always notifies only single row can be fetch

FROM employees

WHERE employee\_id = 149;

dbms\_output.Put\_line('Employee Details : ID:'

||z\_employee.employee\_id

||' Name: '

||z\_employee.first\_name

||' '

||z\_employee.last\_name

||' Salary: '

||z\_employee.salary);

END;

20.

DECLARE

CURSOR z\_emp\_info IS

SELECT employee\_id,

first\_name,

last\_name,

salary

FROM employees;

r\_emp\_info z\_emp\_info%ROWTYPE;

BEGIN

OPEN z\_emp\_info;

LOOP

FETCH z\_emp\_info INTO r\_emp\_info;

EXIT WHEN z\_emp\_info%NOTFOUND;

dbms\_output.Put\_line('Employees Information:: '

||' ID: '

||r\_emp\_info.employee\_id

||' Name: '

||r\_emp\_info.first\_name

||' '

||r\_emp\_info.last\_name);

END LOOP;

dbms\_output.Put\_line('Total number of rows : '

||z\_emp\_info%rowcount);

CLOSE z\_emp\_info;

END;

21.

DECLARE

z\_empid employees.employee\_id%TYPE;

z\_empname employees.first\_name%TYPE;

z\_salary employees.salary%TYPE;

CURSOR employee\_cursor IS -- declaring a cursor

SELECT employee\_id,

first\_name,

salary

FROM employees;

BEGIN

OPEN employee\_cursor; -- opening the cursor

LOOP

FETCH employee\_cursor -- fetching records from the cursor

INTO z\_empid,

z\_empname,

z\_salary;

EXIT

WHEN employee\_cursor%NOTFOUND;

IF (z\_salary > 8000) THEN

dbms\_output.Put\_line(z\_empid

|| ' '

|| z\_empname

|| ' '

|| z\_salary);

ELSE

dbms\_output.Put\_line(z\_empname

|| ' salary is less then 8000');

END IF;

END LOOP;

CLOSE employee\_cursor; --closing the cursor

END;

22.

DECLARE

CURSOR cur\_emp\_detail IS

SELECT employee\_id,

first\_name,

last\_name,

salary

FROM employees;

TYPE type\_record\_type IS RECORD (

emp\_id employees.employee\_id%TYPE,

emp\_f\_name employees.first\_name%TYPE,

emp\_l\_name employees.last\_name%TYPE,

emp\_s\_salary employees.salary%TYPE );

emp\_rec\_type type\_record\_type;

BEGIN

OPEN cur\_emp\_detail;

LOOP

FETCH cur\_emp\_detail INTO emp\_rec\_type;

EXIT WHEN cur\_emp\_detail%NOTFOUND;

dbms\_output.Put\_line('Employees Information:: '

||' ID: '

||emp\_rec\_type.emp\_id

||'| Name: '

||emp\_rec\_type.emp\_f\_name

||' '

||emp\_rec\_type.emp\_l\_name

||'| Salary: '

||emp\_rec\_type.emp\_s\_salary);

END LOOP;

dbms\_output.Put\_line('Total number of Employees : '

||cur\_emp\_detail%rowcount);

CLOSE cur\_emp\_detail;

END;

23.

BEGIN

FOR emprec IN(SELECT department\_name,

d.department\_id,

first\_name,

last\_name,

job\_id,

salary

FROM departments d

join employees e

ON e.department\_id = d.department\_id

WHERE job\_id = 'ST\_CLERK'

AND salary > 3000) LOOP

dbms\_output.Put\_line('Name: '

||emprec.first\_name

||' '

||emprec.last\_name||chr(9)

||' Department: '

||emprec.department\_name||chr(9)

||' Department ID: '

||emprec.department\_id||chr(9)

||' Job ID: '

||emprec.job\_id||chr(9)

||' Salary: '

||emprec.salary);

END LOOP;

END;

24. can be done.

25.

DROP TABLE emp\_temp;

CREATE TABLE emp\_temp AS

SELECT employee\_id,

first\_name,

last\_name,

department\_id,

salary

FROM employees;

DECLARE

CURSOR employee\_cur IS

SELECT employee\_id,

salary

FROM emp\_temp

WHERE department\_id = 50

FOR UPDATE;

incr\_sal NUMBER;

BEGIN

FOR employee\_rec IN employee\_cur LOOP

IF employee\_rec.salary < 15000 THEN

incr\_sal := .15;

ELSE

incr\_sal := .10;

END IF;

UPDATE emp\_temp

SET salary = salary + salary \* incr\_sal

WHERE CURRENT OF employee\_cur;

END LOOP;

END;

26. to be done

27.

DECLARE

CURSOR cur\_mgr IS

SELECT first\_name,

last\_name,

department\_name

FROM employees e

INNER JOIN departments d ON d.manager\_id = e.employee\_id;

v\_mgr cur\_mgr%ROWTYPE;

BEGIN

OPEN cur\_mgr;

LOOP

-- fetch information from cursor into record

FETCH cur\_mgr INTO v\_mgr;

EXIT WHEN cur\_mgr%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(v\_mgr.department\_name || ' :: ' ||

v\_mgr.first\_name || ' ' ||

v\_mgr.last\_name);

END LOOP;

CLOSE cur\_mgr;

END;

28.

DECLARE

CURSOR department\_cur IS

SELECT department\_id,

department\_name,

Max(salary) maxsalary

FROM employees

join departments USING (department\_id)

GROUP BY department\_id,

department\_name;

emp\_first\_name employees.first\_name%TYPE;

BEGIN

dbms\_output.Put\_line('------------------------------------------------------------');

dbms\_output.Put\_line(Rpad('Name of the Department', 35)

||Rpad('First Name', 25));

dbms\_output.Put\_line('------------------------------------------------------------');

FOR emp\_dept\_cur IN department\_cur LOOP

BEGIN

SELECT first\_name

INTO emp\_first\_name

FROM employees

WHERE department\_id = emp\_dept\_cur.department\_id

AND salary = emp\_dept\_cur.maxsalary;

dbms\_output.Put\_line(Rpad(emp\_dept\_cur.department\_name, 35)

|| Rpad(emp\_first\_name, 25));

EXCEPTION

WHEN too\_many\_rows THEN

dbms\_output.Put\_line(Rpad(emp\_dept\_cur.department\_name, 35)

|| ' - More than one employee');

END;

END LOOP;

END;

29.

DECLARE

CURSOR emp\_cur (emp\_job\_nm VARCHAR2, job\_max\_sal NUMBER) IS

SELECT last\_name, first\_name, (salary - job\_max\_sal) overpayment

FROM employees

WHERE job\_id = emp\_job\_nm

AND salary > job\_max\_sal

ORDER BY salary;

PROCEDURE emp\_excesspaid IS

last\_name\_ employees.last\_name%TYPE;

first\_name\_ employees.first\_name%TYPE;

paid\_excess employees.salary%TYPE;

BEGIN

LOOP

FETCH emp\_cur INTO last\_name\_, first\_name\_, paid\_excess;

EXIT WHEN emp\_cur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(last\_name\_ || ', ' || first\_name\_ ||

' (by ' || paid\_excess || ')');

END LOOP;

END emp\_excesspaid;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('---------------------------------');

DBMS\_OUTPUT.PUT\_LINE('Extra Salary paid to Programmers:');

DBMS\_OUTPUT.PUT\_LINE('---------------------------------');

OPEN emp\_cur('IT\_PROG', 6000);

emp\_excesspaid;

CLOSE emp\_cur;

DBMS\_OUTPUT.PUT\_LINE('-----------------------------------');

DBMS\_OUTPUT.PUT\_LINE('Extra Salary paid to Stock Manager:');

DBMS\_OUTPUT.PUT\_LINE('-----------------------------------');

OPEN emp\_cur('ST\_MAN', 5000);

emp\_excesspaid;

CLOSE emp\_cur;

END;

30.

DECLARE

st\_month NUMBER(2) := 1;

no\_o\_emp NUMBER(3);

BEGIN

dbms\_output.Put\_line(Rpad('Month No', 20)

||Rpad('Month Name', 20)

|| 'Number of Employees');

dbms\_output.Put\_line('-------------------------------------------------------------');

FOR month IN 1 .. 12 LOOP

SELECT Count(\*)

INTO no\_o\_emp

FROM employees

WHERE To\_char(hire\_date, 'mm') = month;

dbms\_output.Put\_line(Rpad(To\_char(month, '00'), 20)

||Rpad(To\_char(To\_date(month, 'MM'), 'MONTH'), 20)

|| To\_char(no\_o\_emp, '999'));

END LOOP;

END;