VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Name: Apurba Koirala

Reg no: 22BCE3799

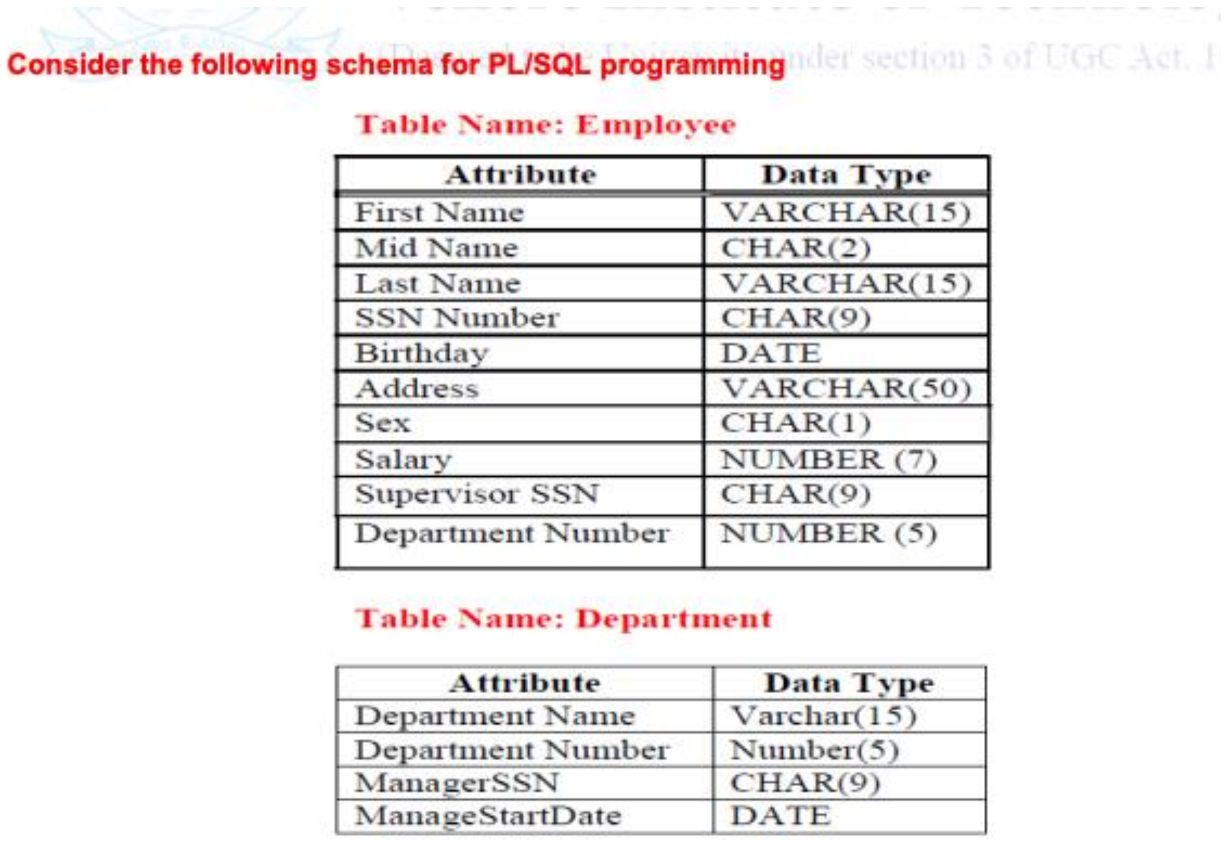
Subject Code: BCSE302P

Course Title: Database Systems

Lab Slot: L33 + L34

Guided by: Dr. Shashank Mouli Satapathy

Exercise - 1



Consider the following schema for PL/SQL programming

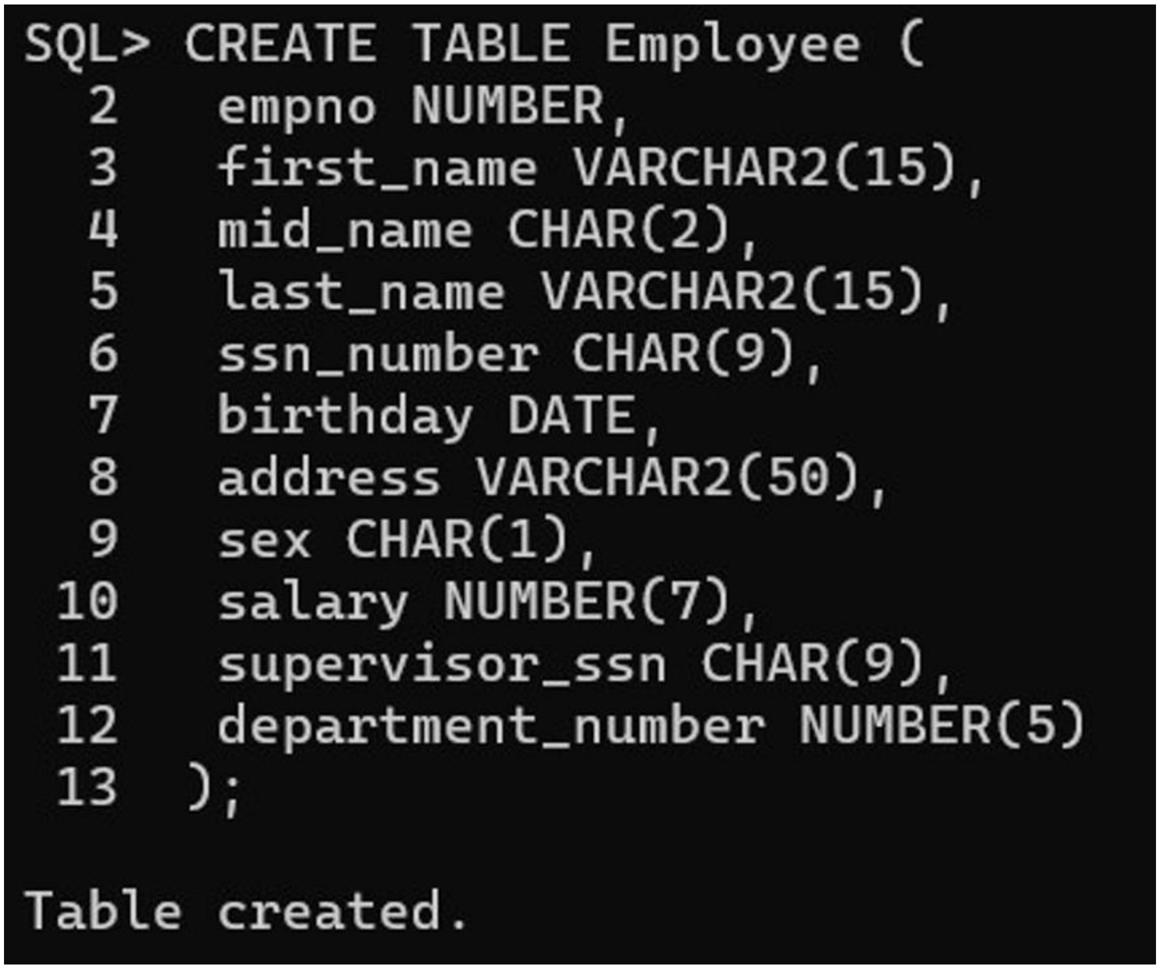
Table Name: Employee

Attribute	Data Type
First Name	VARCHAR(15)
Mid Name	CHAR(2)
Last Name	VARCHAR(15)
SSN Number	CHAR(9)
Birthday	DATE
Address	VARCHAR(50)
Sex	CHAR(1)
Salary	NUMBER (7)
Supervisor SSN	CHAR(9)
Department Number	NUMBER (5)

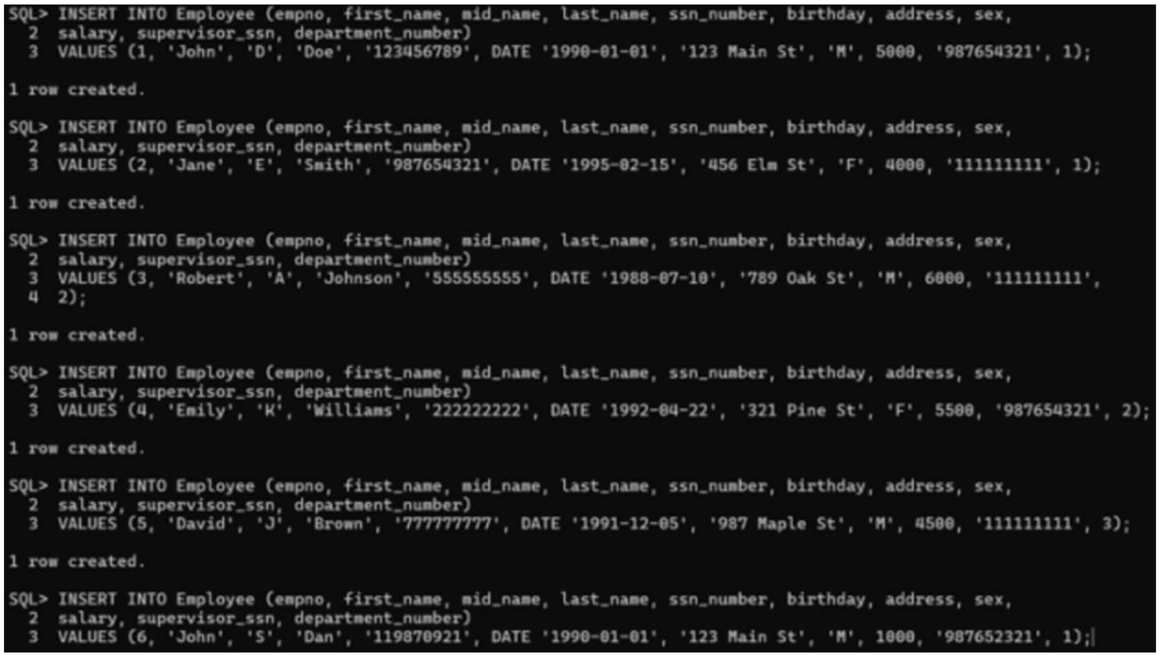
Table Name: Department

Attribute	Data Type
Department Name	Varchar(15)
Department Number	Number(5)
ManagerSSN	CHAR(9)
ManageStartDate	DATE

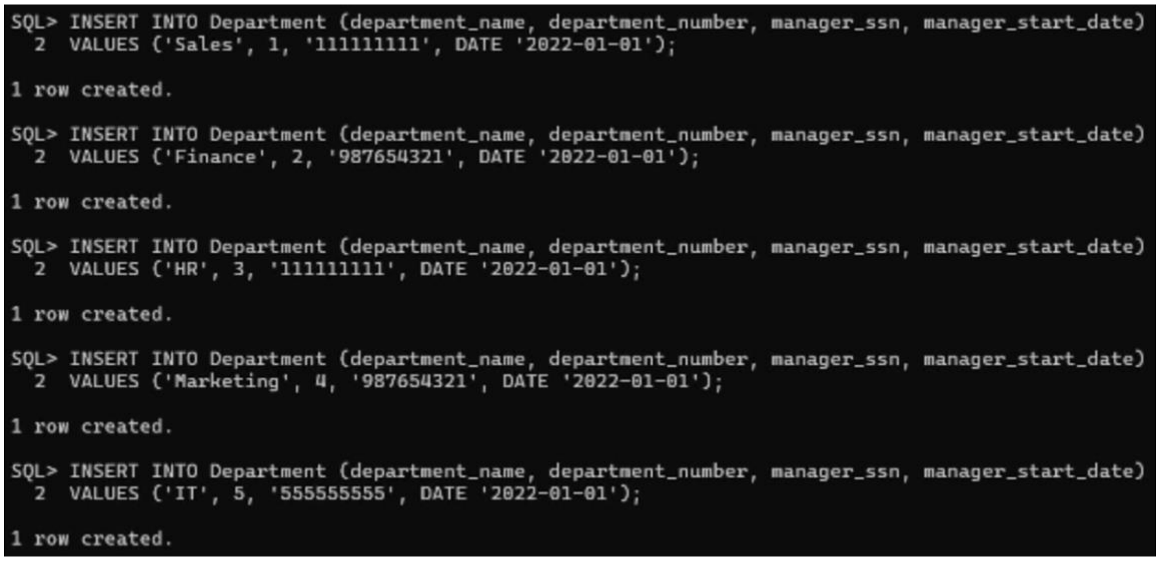
Constructing Schema :



```
SQL> CREATE TABLE Employee (
  empno NUMBER,
  first_name VARCHAR2(15),
  mid_name CHAR(2),
  last_name VARCHAR2(15),
  ssn_number CHAR(9),
  birthday DATE,
  address VARCHAR2(50),
  sex CHAR(1),
  salary NUMBER(7),
  supervisor_ssn CHAR(9),
  department_number NUMBER(5)
 );

Table created.
```



```
SQL> INSERT INTO Employee (empno, first_name, mid_name, last_name, ssn_number, birthday, address, sex,
  2  salary, supervisor_ssn, department_number)
  3  VALUES (1, 'John', 'D', 'Doe', '123456789', DATE '1990-01-01', '123 Main St', 'M', 5000, '987654321', 1);

1 row created.

SQL> INSERT INTO Employee (empno, first_name, mid_name, last_name, ssn_number, birthday, address, sex,
  2  salary, supervisor_ssn, department_number)
  3  VALUES (2, 'Jane', 'E', 'Smith', '987654321', DATE '1995-02-15', '456 Elm St', 'F', 4000, '111111111', 1);

1 row created.

SQL> INSERT INTO Employee (empno, first_name, mid_name, last_name, ssn_number, birthday, address, sex,
  2  salary, supervisor_ssn, department_number)
  3  VALUES (3, 'Robert', 'A', 'Johnson', '555555555', DATE '1988-07-10', '789 Oak St', 'M', 6000, '111111111',
  4  2);

1 row created.

SQL> INSERT INTO Employee (empno, first_name, mid_name, last_name, ssn_number, birthday, address, sex,
  2  salary, supervisor_ssn, department_number)
  3  VALUES (4, 'Emily', 'K', 'Williams', '222222222', DATE '1992-04-22', '321 Pine St', 'F', 5500, '987654321', 2);

1 row created.

SQL> INSERT INTO Employee (empno, first_name, mid_name, last_name, ssn_number, birthday, address, sex,
  2  salary, supervisor_ssn, department_number)
  3  VALUES (5, 'David', 'J', 'Brown', '777777777', DATE '1991-12-05', '987 Maple St', 'M', 4500, '111111111', 3);

1 row created.

SQL> INSERT INTO Employee (empno, first_name, mid_name, last_name, ssn_number, birthday, address, sex,
  2  salary, supervisor_ssn, department_number)
  3  VALUES (6, 'John', 'S', 'Dan', '119870921', DATE '1990-01-01', '123 Main St', 'M', 1000, '987652321', 1);
```



```
SQL> INSERT INTO Department (department_name, department_number, manager_ssn, manager_start_date)
  2  VALUES ('Sales', 1, '111111111', DATE '2022-01-01');

1 row created.

SQL> INSERT INTO Department (department_name, department_number, manager_ssn, manager_start_date)
  2  VALUES ('Finance', 2, '987654321', DATE '2022-01-01');

1 row created.

SQL> INSERT INTO Department (department_name, department_number, manager_ssn, manager_start_date)
  2  VALUES ('HR', 3, '111111111', DATE '2022-01-01');

1 row created.

SQL> INSERT INTO Department (department_name, department_number, manager_ssn, manager_start_date)
  2  VALUES ('Marketing', 4, '987654321', DATE '2022-01-01');

1 row created.

SQL> INSERT INTO Department (department_name, department_number, manager_ssn, manager_start_date)
  2  VALUES ('IT', 5, '555555555', DATE '2022-01-01');

1 row created.
```

Exercise – 1

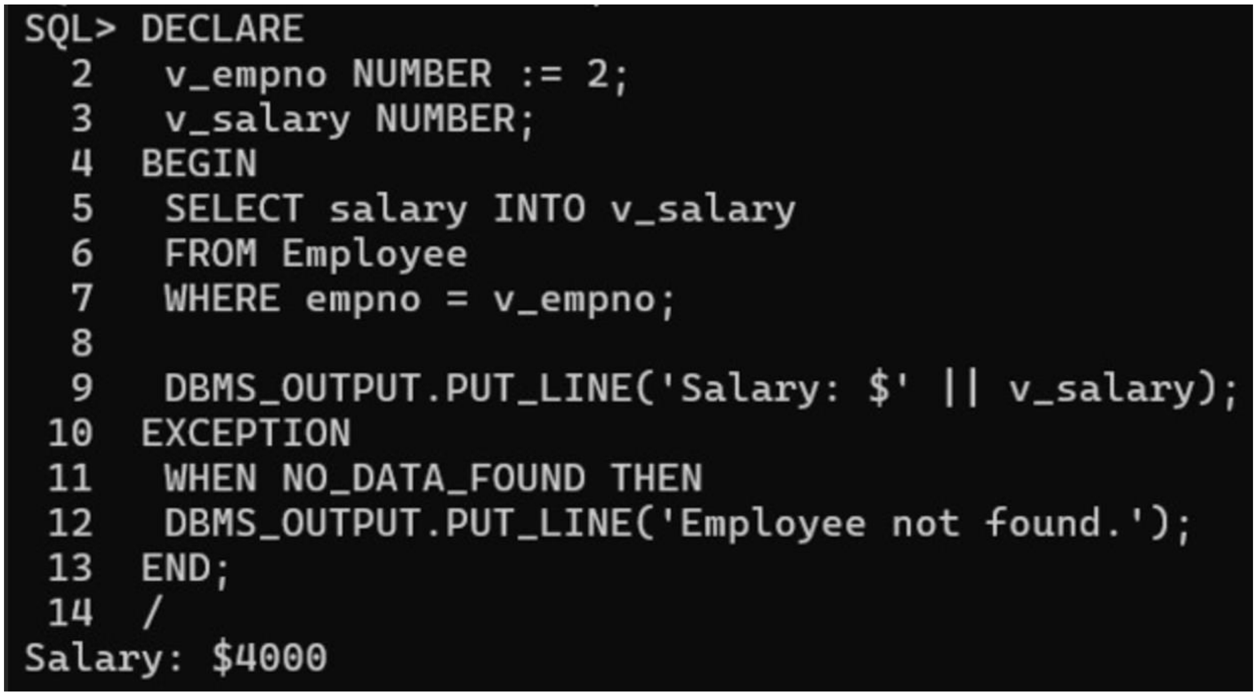
1) **Write a PL/SQL block to accept an empno and display the salary of the person.**
2) **Write a PL/SQL program to delete one record in employee table.**
3) **Write a program to delete employee details who are having age >60.**
4) **Write a PL/SQL block to display employees must make a minimum salary of $1,000.**
5) **Write a PL/SQL to delete a records whose basic salary is <2000 from Emp table.**

1.

Code:

```
DECLARE  v_empno NUMBER := 2;   v_salary NUMBER; BEGIN

 SELECT salary INTO v_salary

 FROM Employee

 WHERE empno = v_empno;


 DBMS_OUTPUT.PUT_LINE('Salary: $' || v_salary);

EXCEPTION

 WHEN NO_DATA_FOUND THEN

 DBMS_OUTPUT.PUT_LINE('Employee not found.');

END;
```

Output:



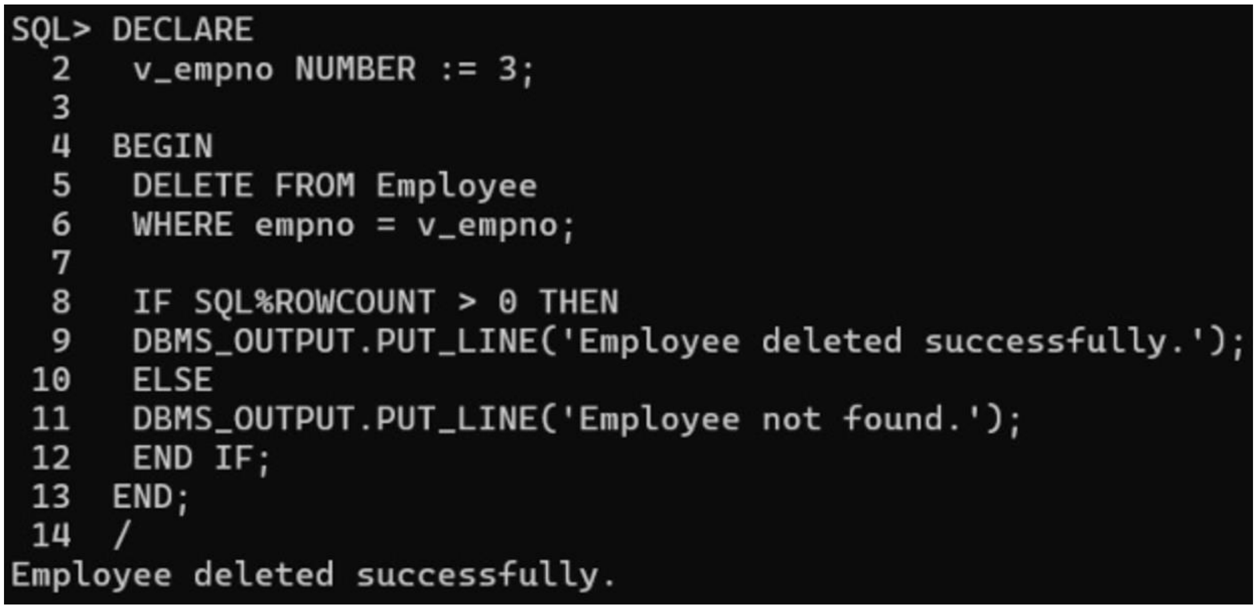
```
SQL> DECLARE
  2   v_empno NUMBER := 2;
  3   v_salary NUMBER;
  4  BEGIN
  5   SELECT salary INTO v_salary
  6   FROM Employee
  7   WHERE empno = v_empno;
  8
  9   DBMS_OUTPUT.PUT_LINE('Salary: $' || v_salary);
 10  EXCEPTION
 11   WHEN NO_DATA_FOUND THEN
 12   DBMS_OUTPUT.PUT_LINE('Employee not found.');
 13  END;
 14  /
Salary: $4000
```

2.

Code:

```
DECLARE  v_empno NUMBER := 3;


BEGIN
 DELETE FROM Employee
 WHERE empno = v_empno;


 IF SQL%ROWCOUNT > 0 THEN
 DBMS_OUTPUT.PUT_LINE('Employee deleted successfully.');
 ELSE
 DBMS_OUTPUT.PUT_LINE('Employee not found.');
 END IF;
END;
/
```

Output:

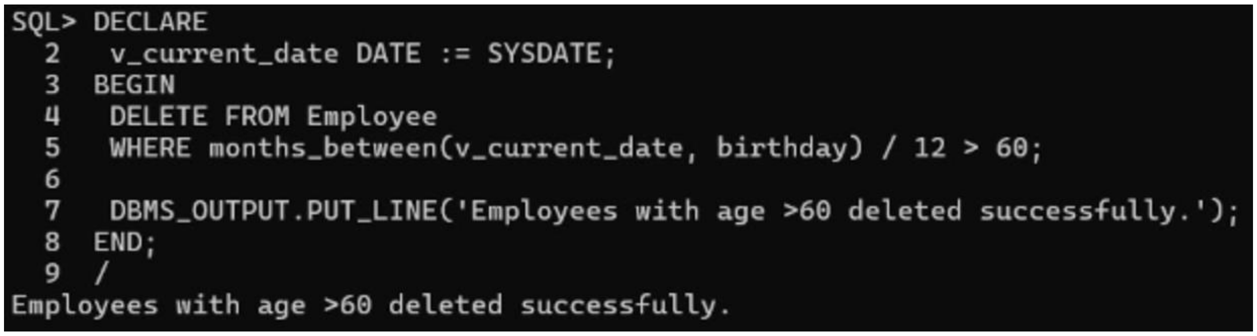


```
SQL> DECLARE
  v_empno NUMBER := 3;

BEGIN
  DELETE FROM Employee
  WHERE empno = v_empno;

  IF SQL%ROWCOUNT > 0 THEN
  DBMS_OUTPUT.PUT_LINE('Employee deleted successfully.');
  ELSE
  DBMS_OUTPUT.PUT_LINE('Employee not found.');
  END IF;
END;
/
Employee deleted successfully.
```

3.

Code:

```
DECLARE
 v_current_date DATE := SYSDATE;
BEGIN
 DELETE FROM Employee
 WHERE months_between(v_current_date, birthday) / 12 > 60;

 DBMS_OUTPUT.PUT_LINE('Employees with age >60 deleted successfully.');
END;
/
```



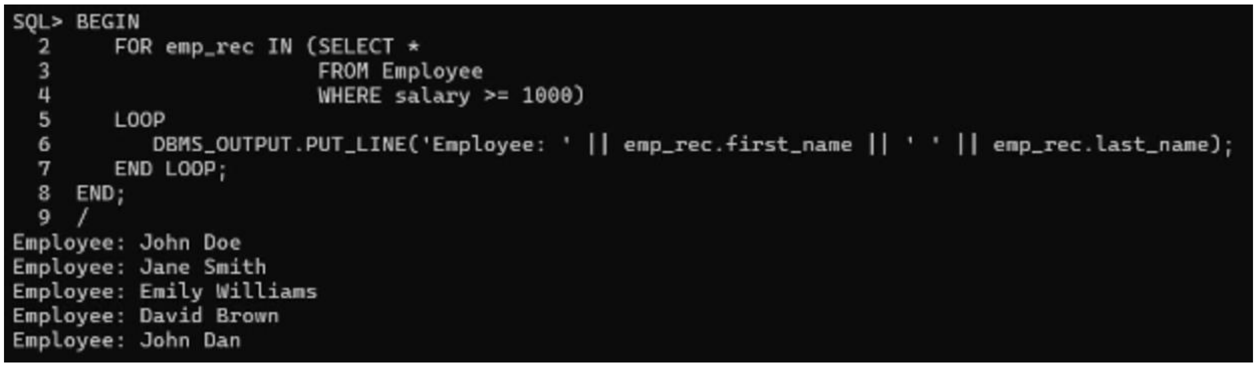
```
SQL> DECLARE
 v_current_date DATE := SYSDATE;
BEGIN
 DELETE FROM Employee
 WHERE months_between(v_current_date, birthday) / 12 > 60;

 DBMS_OUTPUT.PUT_LINE('Employees with age >60 deleted successfully.');
END;
/
Employees with age >60 deleted successfully.
```

4.

Code:

```
BEGIN
 FOR emp_rec IN (SELECT *
 FROM Employee
 WHERE salary < 1000)
 LOOP
 DBMS_OUTPUT.PUT_LINE('Employee: ' || emp_rec.first_name || ' ' || emp_rec.last_name);
 END LOOP;
END;
/
```

Output:



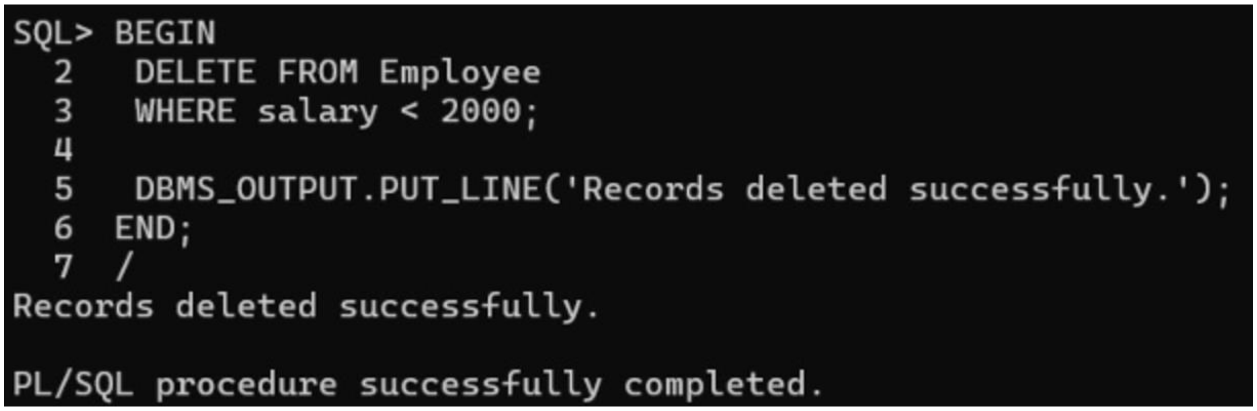
```
SQL> BEGIN
    FOR emp_rec IN (SELECT *
                    FROM Employee
                    WHERE salary >= 1000)
    LOOP
       DBMS_OUTPUT.PUT_LINE('Employee: ' || emp_rec.first_name || ' ' || emp_rec.last_name);
    END LOOP;
 END;
 /
Employee: John Doe
Employee: Jane Smith
Employee: Emily Williams
Employee: David Brown
Employee: John Dan
```

5.

Code:

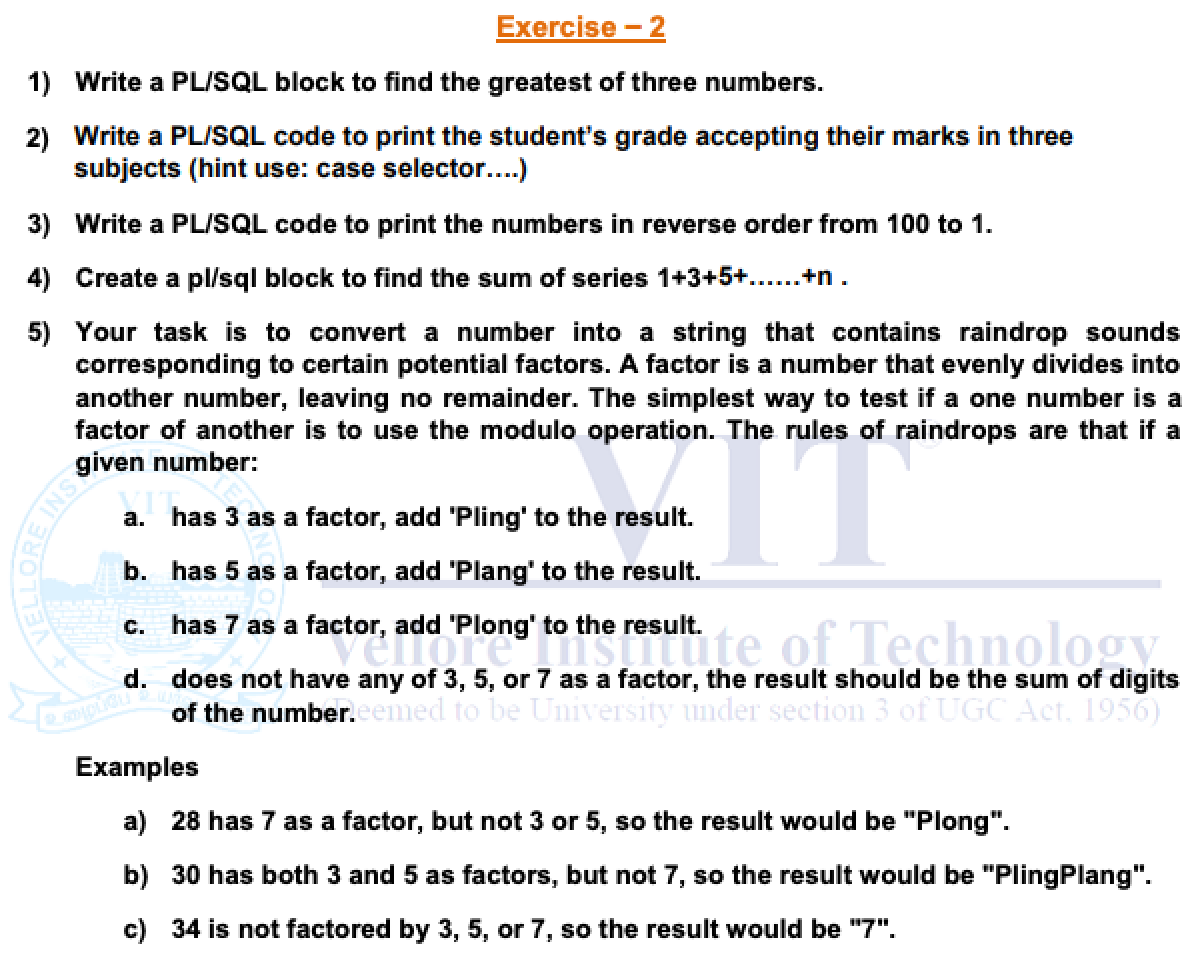
```
BEGIN
 DELETE FROM Employee
 WHERE salary < 2000;


 DBMS_OUTPUT.PUT_LINE('Records deleted successfully.');
END;
/
```

Output:



```
SQL> BEGIN
  DELETE FROM Employee
  WHERE salary < 2000;

  DBMS_OUTPUT.PUT_LINE('Records deleted successfully.');
 END;
 /
Records deleted successfully.

PL/SQL procedure successfully completed.
```

Exercise – 2



Exercise – 2

1) Write a PL/SQL block to find the greatest of three numbers.
2) Write a PL/SQL code to print the student's grade accepting their marks in three subjects (hint use: case selector....)
3) Write a PL/SQL code to print the numbers in reverse order from 100 to 1.
4) Create a pl/sql block to find the sum of series 1+3+5+......+n .
5) Your task is to convert a number into a string that contains raindrop sounds corresponding to certain potential factors. A factor is a number that evenly divides into another number, leaving no remainder. The simplest way to test if a one number is a factor of another is to use the modulo operation. The rules of raindrops are that if a given number:
    a. has 3 as a factor, add 'Pling' to the result.
    b. has 5 as a factor, add 'Plang' to the result.
    c. has 7 as a factor, add 'Plong' to the result.
    d. does not have any of 3, 5, or 7 as a factor, the result should be the sum of digits of the number.

    Examples

    a) 28 has 7 as a factor, but not 3 or 5, so the result would be "Plong".
    b) 30 has both 3 and 5 as factors, but not 7, so the result would be "PlingPlang".
    c) 34 is not factored by 3, 5, or 7, so the result would be "7".

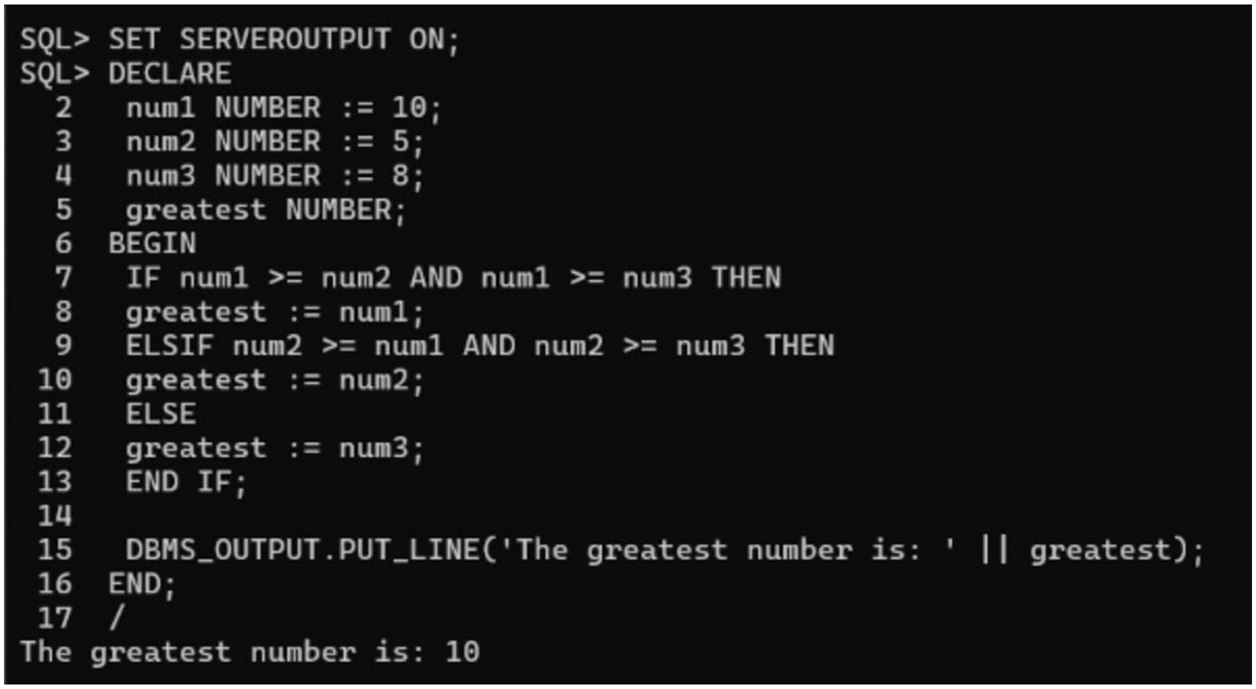
1.

Code:

```
DECLARE  num1
NUMBER := 10;
num2 NUMBER := 5;
num3 NUMBER := 8;
greatest NUMBER;
BEGIN
```

```
 IF num1 >= num2 AND num1 >= num3 THEN
greatest := num1;
 ELSIF num2 >= num1 AND num2 >= num3
THEN  greatest := num2;  ELSE  greatest := num3;
 END IF;


 DBMS_OUTPUT.PUT_LINE('The greatest number is: ' || greatest);
END;
/
```

Output:



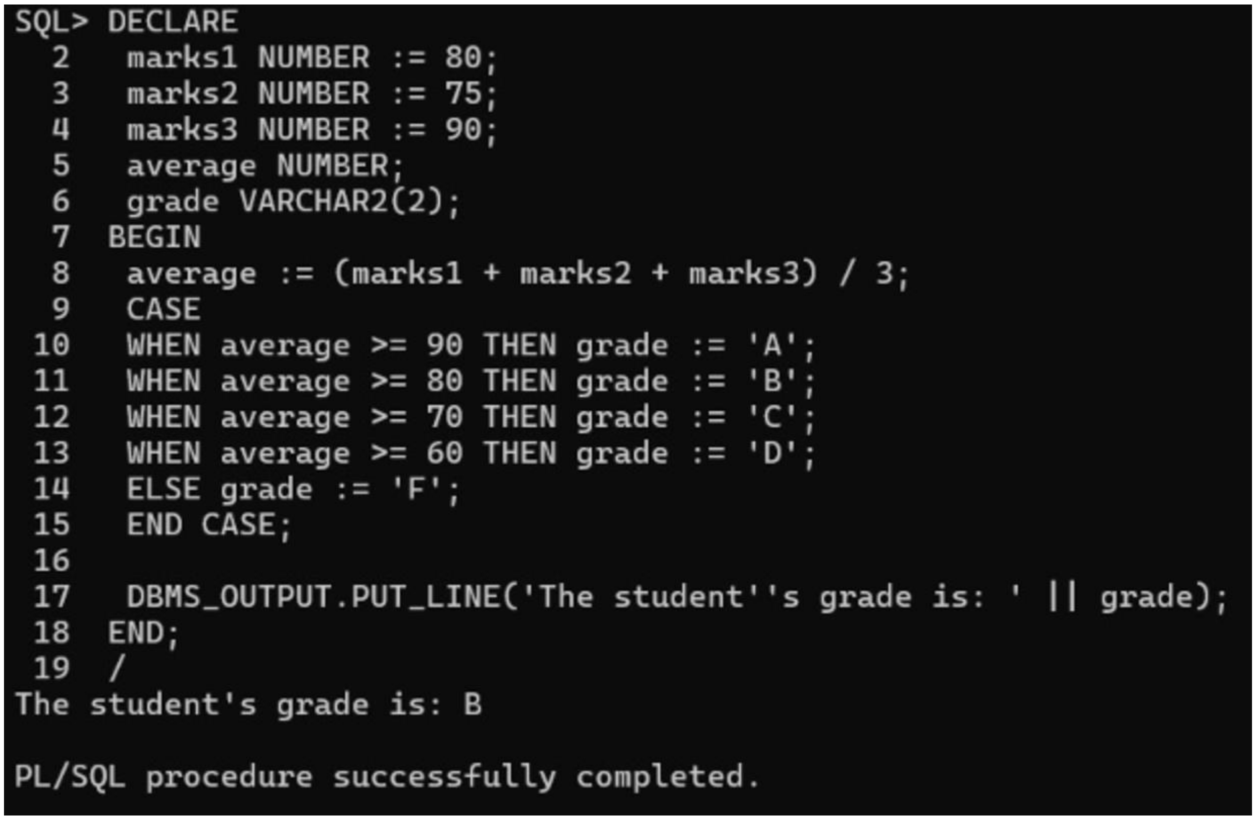
```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
   num1 NUMBER := 10;
   num2 NUMBER := 5;
   num3 NUMBER := 8;
   greatest NUMBER;
  BEGIN
   IF num1 >= num2 AND num1 >= num3 THEN
   greatest := num1;
   ELSIF num2 >= num1 AND num2 >= num3 THEN
   greatest := num2;
   ELSE
   greatest := num3;
   END IF;

   DBMS_OUTPUT.PUT_LINE('The greatest number is: ' || greatest);
  END;
  /
The greatest number is: 10
```

2.

Code:

```
DECLARE  marks1
NUMBER := 80;
marks2 NUMBER :=
75;  marks3 NUMBER
:= 90;  average
NUMBER;  grade
VARCHAR2(2);
BEGIN
 average := (marks1 + marks2 + marks3) / 3;
 CASE
 WHEN average >= 90 THEN grade := 'A';
 WHEN average >= 80 THEN grade := 'B';
WHEN average >= 70 THEN grade := 'C';
 WHEN average >= 60 THEN grade := 'D';
 ELSE grade := 'F';
 END CASE;

 DBMS_OUTPUT.PUT_LINE('The student''s grade is: ' || grade);
END;
/
```

Output:

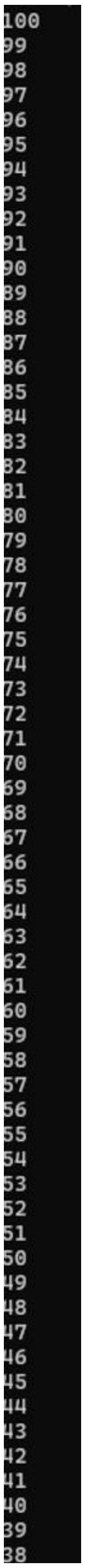


```
SQL> DECLARE
   marks1 NUMBER := 80;
   marks2 NUMBER := 75;
   marks3 NUMBER := 90;
   average NUMBER;
   grade VARCHAR2(2);
  BEGIN
   average := (marks1 + marks2 + marks3) / 3;
   CASE
   WHEN average >= 90 THEN grade := 'A';
   WHEN average >= 80 THEN grade := 'B';
   WHEN average >= 70 THEN grade := 'C';
   WHEN average >= 60 THEN grade := 'D';
   ELSE grade := 'F';
   END CASE;

   DBMS_OUTPUT.PUT_LINE('The student''s grade is: ' || grade);
  END;
  /
The student's grade is: B

PL/SQL procedure successfully completed.
```

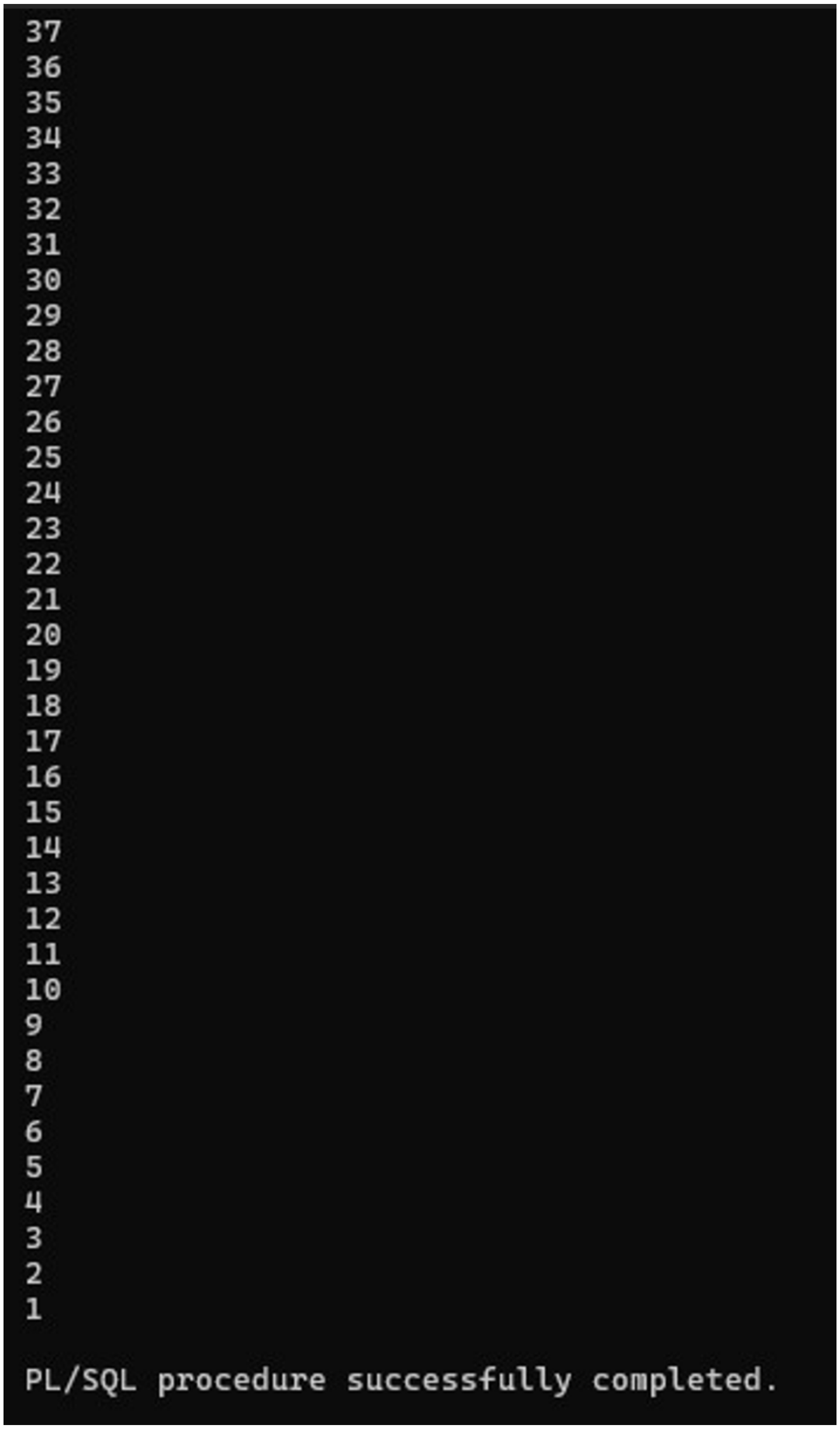
3.

Code:

```
DECLARE
 counter NUMBER := 100;
BEGIN
 WHILE counter >= 1 LOOP
 DBMS_OUTPUT.PUT_LINE(counter);
counter := counter - 1;
 END LOOP;
END;
/
```



```
100
99
98
97
96
95
94
93
92
91
90
89
88
87
86
85
84
83
82
81
80
79
78
77
76
75
74
73
72
71
70
69
68
67
66
65
64
63
62
61
60
59
58
57
56
55
54
53
52
51
50
49
48
47
46
45
44
43
42
41
40
39
38
```



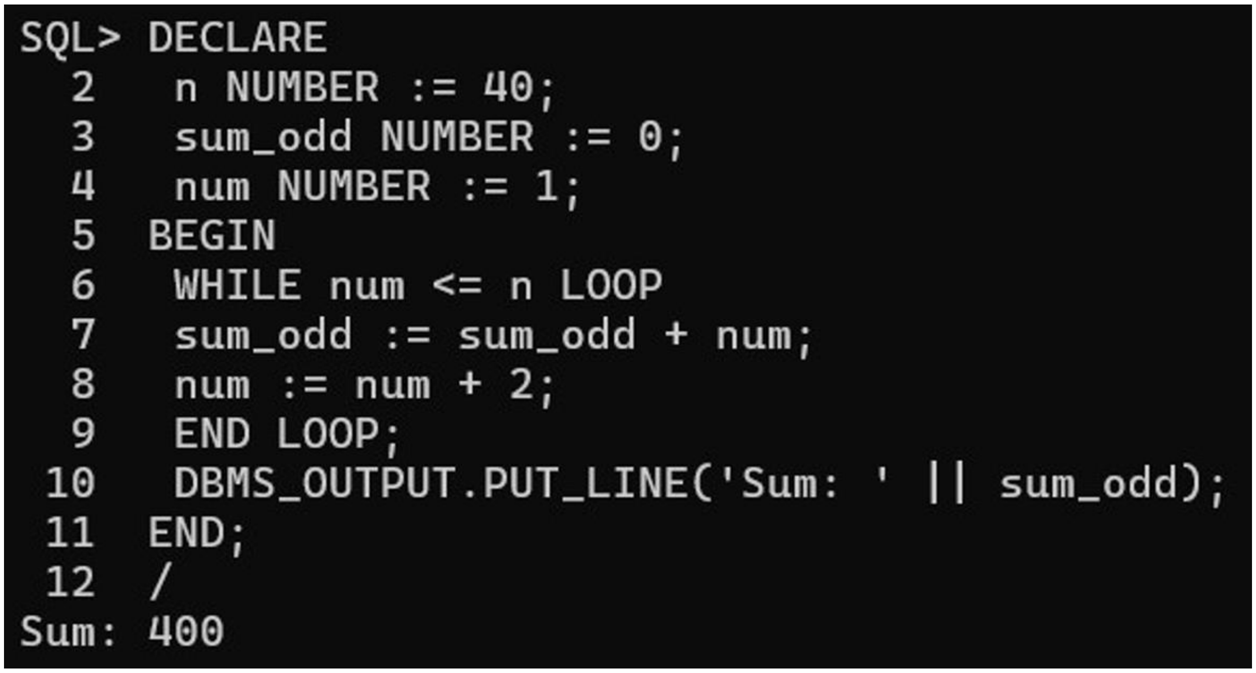
```
37
36
35
34
33
32
31
30
29
28
27
26
25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1

PL/SQL procedure successfully completed.
```

4.

Code:

```
DECLARE
     n NUMBER := 40;
sum_odd NUMBER := 0;
num NUMBER := 1;
BEGIN
     WHILE num <= n LOOP
sum_odd := sum_odd + num;
num := num + 2;
     END LOOP;
    DBMS_OUTPUT.PUT_LINE('Sum: ' || sum_odd);
   END;
  /
```

Output:



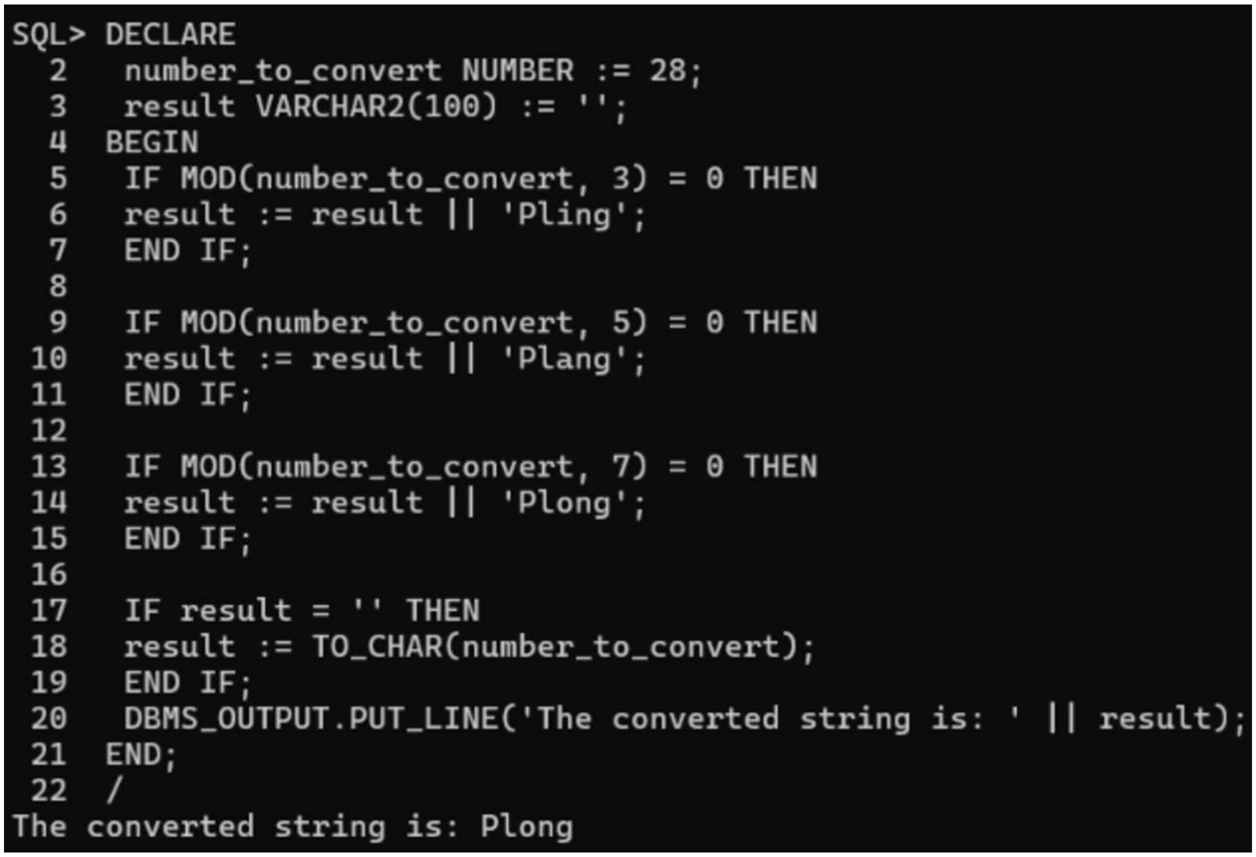
```
SQL> DECLARE
   n NUMBER := 40;
   sum_odd NUMBER := 0;
   num NUMBER := 1;
  BEGIN
   WHILE num <= n LOOP
   sum_odd := sum_odd + num;
   num := num + 2;
   END LOOP;
   DBMS_OUTPUT.PUT_LINE('Sum: ' || sum_odd);
  END;
  /
Sum: 400
```

5.

Code:

```
DECLARE  number_to_convert
NUMBER := 28;  result
VARCHAR2(100) := '';
BEGIN
 IF MOD(number_to_convert, 3) = 0 THEN
 result := result || 'Pling';
 END IF;

 IF MOD(number_to_convert, 5) = 0 THEN
 result := result || 'Plang';
 END IF;

 IF MOD(number_to_convert, 7) = 0 THEN
 result := result || 'Plong';
 END IF;

 IF result = '' THEN
 result := TO_CHAR(number_to_convert);
 END IF;
 DBMS_OUTPUT.PUT_LINE('The converted string is: ' || result);
END;
/
```

Output:



```
SQL> DECLARE
   number_to_convert NUMBER := 28;
   result VARCHAR2(100) := '';
  BEGIN
   IF MOD(number_to_convert, 3) = 0 THEN
   result := result || 'Pling';
   END IF;

   IF MOD(number_to_convert, 5) = 0 THEN
   result := result || 'Plang';
   END IF;

   IF MOD(number_to_convert, 7) = 0 THEN
   result := result || 'Plong';
   END IF;

   IF result = '' THEN
   result := TO_CHAR(number_to_convert);
   END IF;
   DBMS_OUTPUT.PUT_LINE('The converted string is: ' || result);
  END;
  /
The converted string is: Plong
```

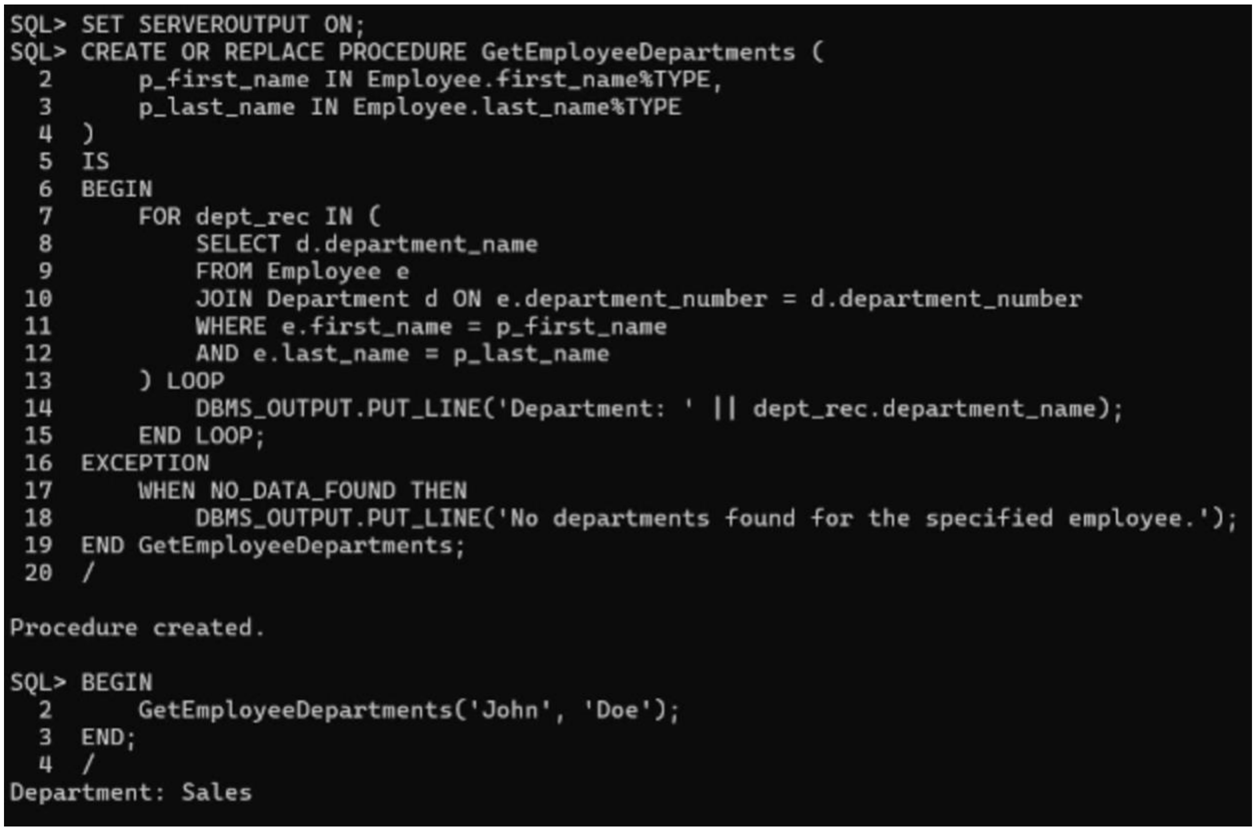
Exercise – 3

Exercise – 3

1) Write a procedure to accept an employee name and display his Department names.
2) Retrieve the employee details using cursors.
3) Write a cursor program to display all the employee and department details
4) Consider the schema discussed during DDL/DML commands lab session. For the Restaurant Database, write a trigger to update the ingredients table whenever a vendor is deleted. For all ingredients supplied by that vendor, set the vendorid to NULL.

1.
Code:
```
CREATE OR REPLACE PROCEDURE

GetEmployeeDepartments (     p_first_name IN

Employee.first_name%TYPE,     p_last_name IN

Employee.last_name%TYPE

)

IS

BEGIN

   FOR dept_rec IN (

      SELECT d.department_name

      FROM Employee e

      JOIN Department d ON e.department_number = d.department_number

      WHERE e.first_name = p_first_name

      AND e.last_name = p_last_name

   ) LOOP

      DBMS_OUTPUT.PUT_LINE('Department: ' || dept_rec.department_name);

   END LOOP;

EXCEPTION

   WHEN NO_DATA_FOUND THEN

      DBMS_OUTPUT.PUT_LINE('No departments found for the specified employee.');

END GetEmployeeDepartments;

/
```

Output:



```
SQL> SET SERVEROUTPUT ON;
SQL> CREATE OR REPLACE PROCEDURE GetEmployeeDepartments (
    p_first_name IN Employee.first_name%TYPE,
    p_last_name IN Employee.last_name%TYPE
)
IS
BEGIN
    FOR dept_rec IN (
        SELECT d.department_name
        FROM Employee e
        JOIN Department d ON e.department_number = d.department_number
        WHERE e.first_name = p_first_name
        AND e.last_name = p_last_name
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('Department: ' || dept_rec.department_name);
    END LOOP;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No departments found for the specified employee.');
END GetEmployeeDepartments;
/

Procedure created.

SQL> BEGIN
    GetEmployeeDepartments('John', 'Doe');
END;
/
Department: Sales
```

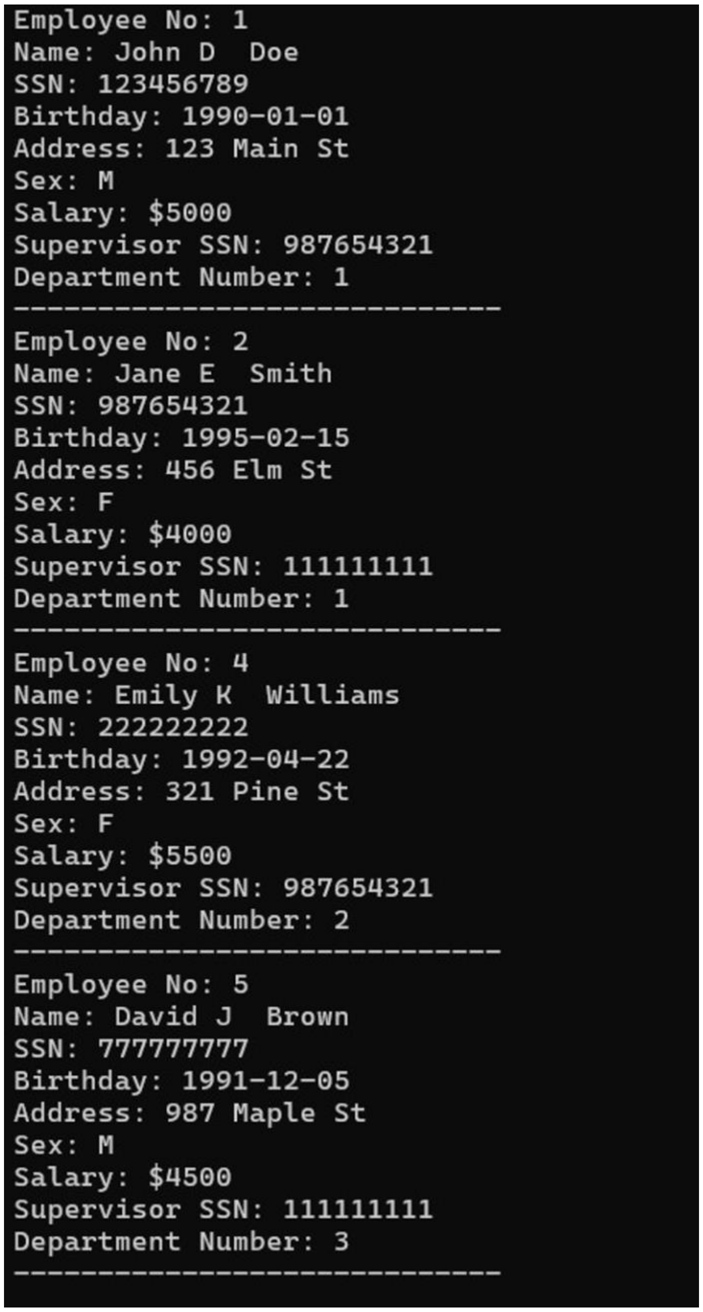
2.

Code:

```
DECLARE

   CURSOR emp_cursor IS

        SELECT empno, first_name, mid_name, last_name, ssn_number, birthday, address, sex,
salary, supervisor_ssn, department_number

        FROM Employee;



   v_empno Employee.empno%TYPE;

v_first_name Employee.first_name%TYPE;

v_mid_name Employee.mid_name%TYPE;

v_last_name Employee.last_name%TYPE;

v_ssn_number Employee.ssn_number%TYPE;

v_birthday Employee.birthday%TYPE;     v_address

Employee.address%TYPE;     v_sex
Employee.sex%TYPE;     v_salary

Employee.salary%TYPE;     v_supervisor_ssn

Employee.supervisor_ssn%TYPE;

v_department_number

Employee.department_number%TYPE;

BEGIN

   OPEN emp_cursor;


   LOOP

      FETCH emp_cursor INTO v_empno, v_first_name, v_mid_name, v_last_name,
v_ssn_number, v_birthday, v_address, v_sex, v_salary, v_supervisor_ssn,
v_department_number;


      EXIT WHEN emp_cursor%NOTFOUND;


      DBMS_OUTPUT.PUT_LINE('Employee No: ' || v_empno);

      DBMS_OUTPUT.PUT_LINE('Name: ' || v_first_name || ' ' || v_mid_name || ' ' ||
v_last_name);

      DBMS_OUTPUT.PUT_LINE('SSN: ' || v_ssn_number);

      DBMS_OUTPUT.PUT_LINE('Birthday: ' || TO_CHAR(v_birthday, 'YYYY-MM-DD'));

      DBMS_OUTPUT.PUT_LINE('Address: ' || v_address);

      DBMS_OUTPUT.PUT_LINE('Sex: ' || v_sex);

      DBMS_OUTPUT.PUT_LINE('Salary: $' || v_salary);

      DBMS_OUTPUT.PUT_LINE('Supervisor SSN: ' || v_supervisor_ssn);
DBMS_OUTPUT.PUT_LINE('Department Number: ' || v_department_number);

      DBMS_OUTPUT.PUT_LINE('-----------------------------');

   END LOOP;
```

```
    CLOSE emp_cursor;

END;

/
```

Output:



```
Employee No: 1
Name: John D  Doe
SSN: 123456789
Birthday: 1990-01-01
Address: 123 Main St
Sex: M
Salary: $5000
Supervisor SSN: 987654321
Department Number: 1
-----------------------------
Employee No: 2
Name: Jane E  Smith
SSN: 987654321
Birthday: 1995-02-15
Address: 456 Elm St
Sex: F
Salary: $4000
Supervisor SSN: 111111111
Department Number: 1
-----------------------------
Employee No: 4
Name: Emily K  Williams
SSN: 222222222
Birthday: 1992-04-22
Address: 321 Pine St
Sex: F
Salary: $5500
Supervisor SSN: 987654321
Department Number: 2
-----------------------------
Employee No: 5
Name: David J  Brown
SSN: 777777777
Birthday: 1991-12-05
Address: 987 Maple St
Sex: M
Salary: $4500
Supervisor SSN: 111111111
Department Number: 3
-----------------------------
```

3.

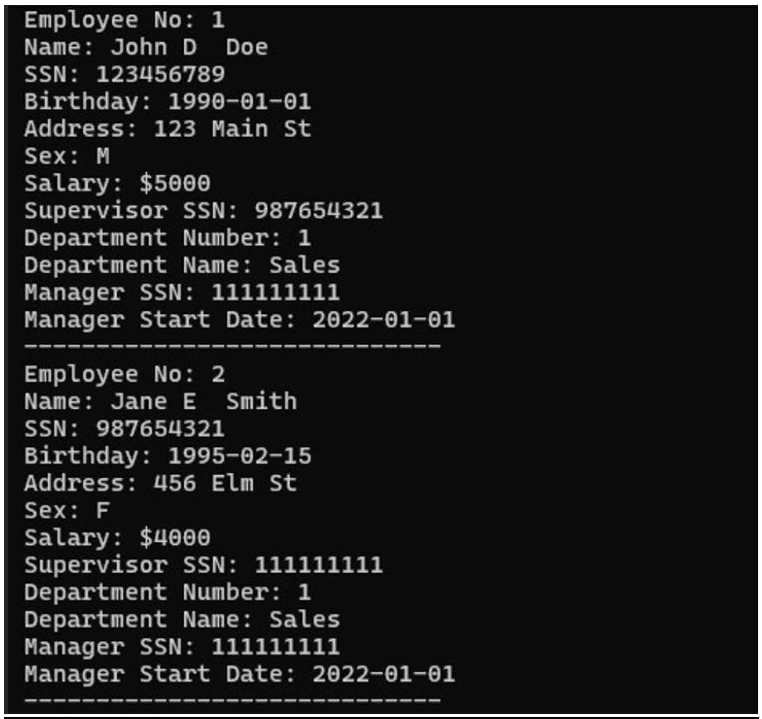
Code:

```
DECLARE
```

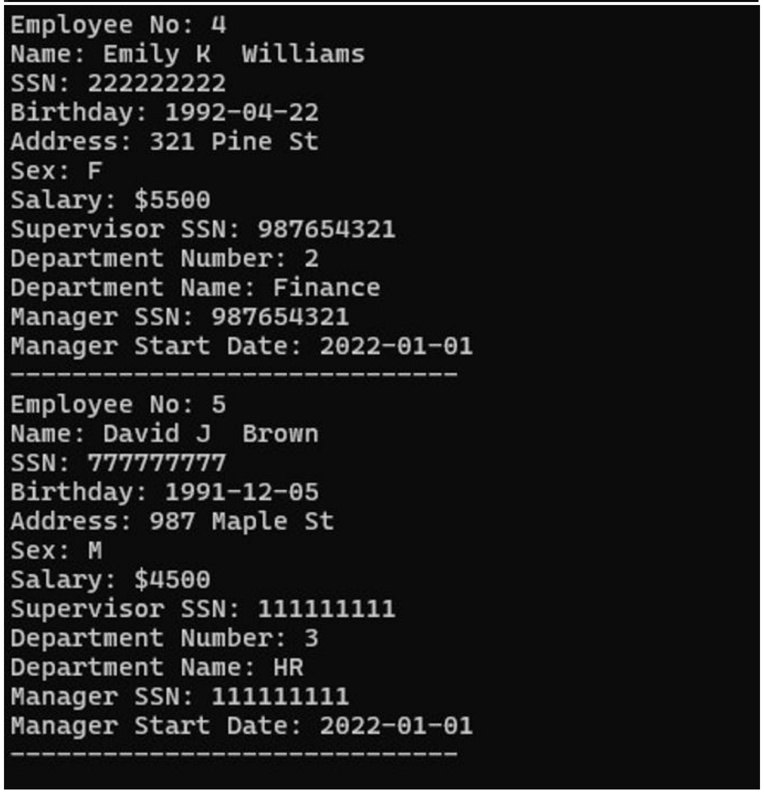
```
    CURSOR emp_dept_cursor IS
        SELECT e.empno, e.first_name, e.mid_name, e.last_name, e.ssn_number, e.birthday,
e.address, e.sex, e.salary,
               e.supervisor_ssn, e.department_number, d.department_name, d.manager_ssn,
d.manager_start_date         FROM Employee e
        JOIN Department d ON e.department_number = d.department_number;

    v_empno Employee.empno%TYPE;     v_first_name
Employee.first_name%TYPE;     v_mid_name
Employee.mid_name%TYPE;     v_last_name
Employee.last_name%TYPE;     v_ssn_number
Employee.ssn_number%TYPE;     v_birthday
Employee.birthday%TYPE;     v_address Employee.address%TYPE;
v_sex Employee.sex%TYPE;     v_salary Employee.salary%TYPE;
v_supervisor_ssn Employee.supervisor_ssn%TYPE;
v_department_number Employee.department_number%TYPE;
v_department_name Department.department_name%TYPE;
v_manager_ssn Department.manager_ssn%TYPE;
v_manager_start_date Department.manager_start_date%TYPE;
BEGIN
    OPEN emp_dept_cursor;

    LOOP
        FETCH emp_dept_cursor INTO v_empno, v_first_name, v_mid_name, v_last_name,
v_ssn_number, v_birthday,                              v_address, v_sex, v_salary,
v_supervisor_ssn, v_department_number,                              v_department_name,
v_manager_ssn, v_manager_start_date;
```

```
        EXIT WHEN emp_dept_cursor%NOTFOUND;


        DBMS_OUTPUT.PUT_LINE('Employee No: ' || v_empno);
        DBMS_OUTPUT.PUT_LINE('Name: ' || v_first_name || ' ' || v_mid_name || ' ' ||
v_last_name);
        DBMS_OUTPUT.PUT_LINE('SSN: ' || v_ssn_number);
        DBMS_OUTPUT.PUT_LINE('Birthday: ' || TO_CHAR(v_birthday, 'YYYY-MM-DD'));
        DBMS_OUTPUT.PUT_LINE('Address: ' || v_address);
        DBMS_OUTPUT.PUT_LINE('Sex: ' || v_sex);
        DBMS_OUTPUT.PUT_LINE('Salary: $' || v_salary);
        DBMS_OUTPUT.PUT_LINE('Supervisor SSN: ' || v_supervisor_ssn);
        DBMS_OUTPUT.PUT_LINE('Department Number: ' || v_department_number);
        DBMS_OUTPUT.PUT_LINE('Department Name: ' || v_department_name);
        DBMS_OUTPUT.PUT_LINE('Manager SSN: ' || v_manager_ssn);
        DBMS_OUTPUT.PUT_LINE('Manager Start Date: ' || TO_CHAR(v_manager_start_date,
'YYYY-MM-DD'));
        DBMS_OUTPUT.PUT_LINE('-----------------------------');
    END LOOP;

    CLOSE emp_dept_cursor;
END;
/
```

Output:



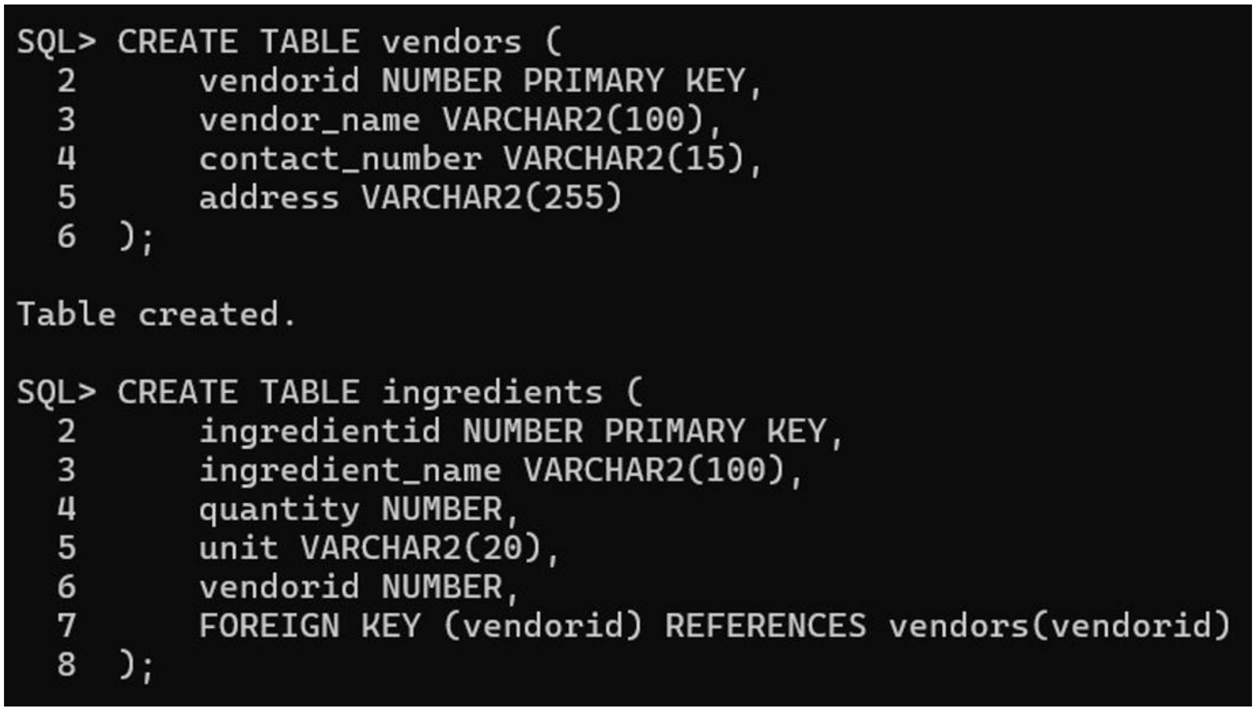
```
Employee No: 1
Name: John D  Doe
SSN: 123456789
Birthday: 1990-01-01
Address: 123 Main St
Sex: M
Salary: $5000
Supervisor SSN: 987654321
Department Number: 1
Department Name: Sales
Manager SSN: 111111111
Manager Start Date: 2022-01-01
-----------------------------
Employee No: 2
Name: Jane E  Smith
SSN: 987654321
Birthday: 1995-02-15
Address: 456 Elm St
Sex: F
Salary: $4000
Supervisor SSN: 111111111
Department Number: 1
Department Name: Sales
Manager SSN: 111111111
Manager Start Date: 2022-01-01
-----------------------------
```



```
Employee No: 4
Name: Emily K  Williams
SSN: 222222222
Birthday: 1992-04-22
Address: 321 Pine St
Sex: F
Salary: $5500
Supervisor SSN: 987654321
Department Number: 2
Department Name: Finance
Manager SSN: 987654321
Manager Start Date: 2022-01-01
-----------------------------
Employee No: 5
Name: David J  Brown
SSN: 777777777
Birthday: 1991-12-05
Address: 987 Maple St
Sex: M
Salary: $4500
Supervisor SSN: 111111111
Department Number: 3
Department Name: HR
Manager SSN: 111111111
Manager Start Date: 2022-01-01
-----------------------------
```

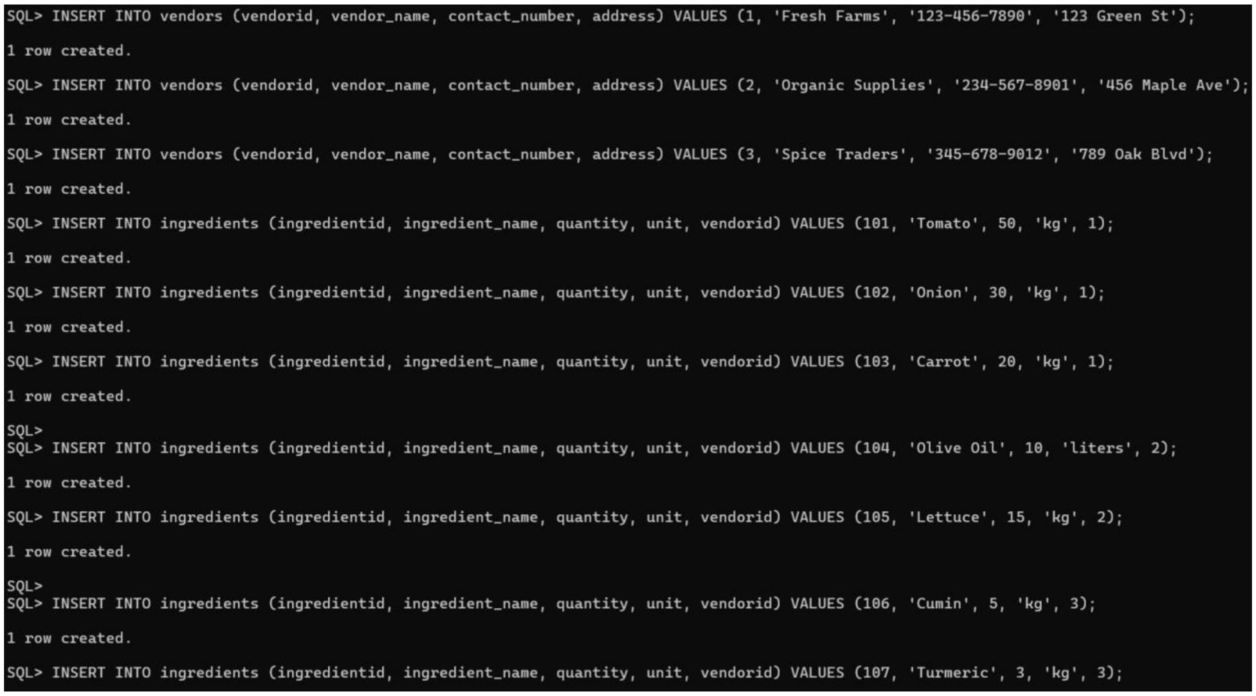
4.

For Creating Tables:



```
SQL> CREATE TABLE vendors (
        vendorid NUMBER PRIMARY KEY,
        vendor_name VARCHAR2(100),
        contact_number VARCHAR2(15),
        address VARCHAR2(255)
    );

Table created.

SQL> CREATE TABLE ingredients (
        ingredientid NUMBER PRIMARY KEY,
        ingredient_name VARCHAR2(100),
        quantity NUMBER,
        unit VARCHAR2(20),
        vendorid NUMBER,
        FOREIGN KEY (vendorid) REFERENCES vendors(vendorid)
    );
```

Inserting data:



```
SQL> INSERT INTO vendors (vendorid, vendor_name, contact_number, address) VALUES (1, 'Fresh Farms', '123-456-7890', '123 Green St');

1 row created.

SQL> INSERT INTO vendors (vendorid, vendor_name, contact_number, address) VALUES (2, 'Organic Supplies', '234-567-8901', '456 Maple Ave');

1 row created.

SQL> INSERT INTO vendors (vendorid, vendor_name, contact_number, address) VALUES (3, 'Spice Traders', '345-678-9012', '789 Oak Blvd');

1 row created.

SQL> INSERT INTO ingredients (ingredientid, ingredient_name, quantity, unit, vendorid) VALUES (101, 'Tomato', 50, 'kg', 1);

1 row created.

SQL> INSERT INTO ingredients (ingredientid, ingredient_name, quantity, unit, vendorid) VALUES (102, 'Onion', 30, 'kg', 1);

1 row created.

SQL> INSERT INTO ingredients (ingredientid, ingredient_name, quantity, unit, vendorid) VALUES (103, 'Carrot', 20, 'kg', 1);

1 row created.

SQL>
SQL> INSERT INTO ingredients (ingredientid, ingredient_name, quantity, unit, vendorid) VALUES (104, 'Olive Oil', 10, 'liters', 2);

1 row created.

SQL> INSERT INTO ingredients (ingredientid, ingredient_name, quantity, unit, vendorid) VALUES (105, 'Lettuce', 15, 'kg', 2);

1 row created.

SQL>
SQL> INSERT INTO ingredients (ingredientid, ingredient_name, quantity, unit, vendorid) VALUES (106, 'Cumin', 5, 'kg', 3);

1 row created.

SQL> INSERT INTO ingredients (ingredientid, ingredient_name, quantity, unit, vendorid) VALUES (107, 'Turmeric', 3, 'kg', 3);
```

Creating Trigger:

CREATE OR REPLACE TRIGGER update_ingredients_on_vendor_delete

AFTER DELETE ON vendors

FOR EACH ROW

BEGIN

  UPDATE ingredients

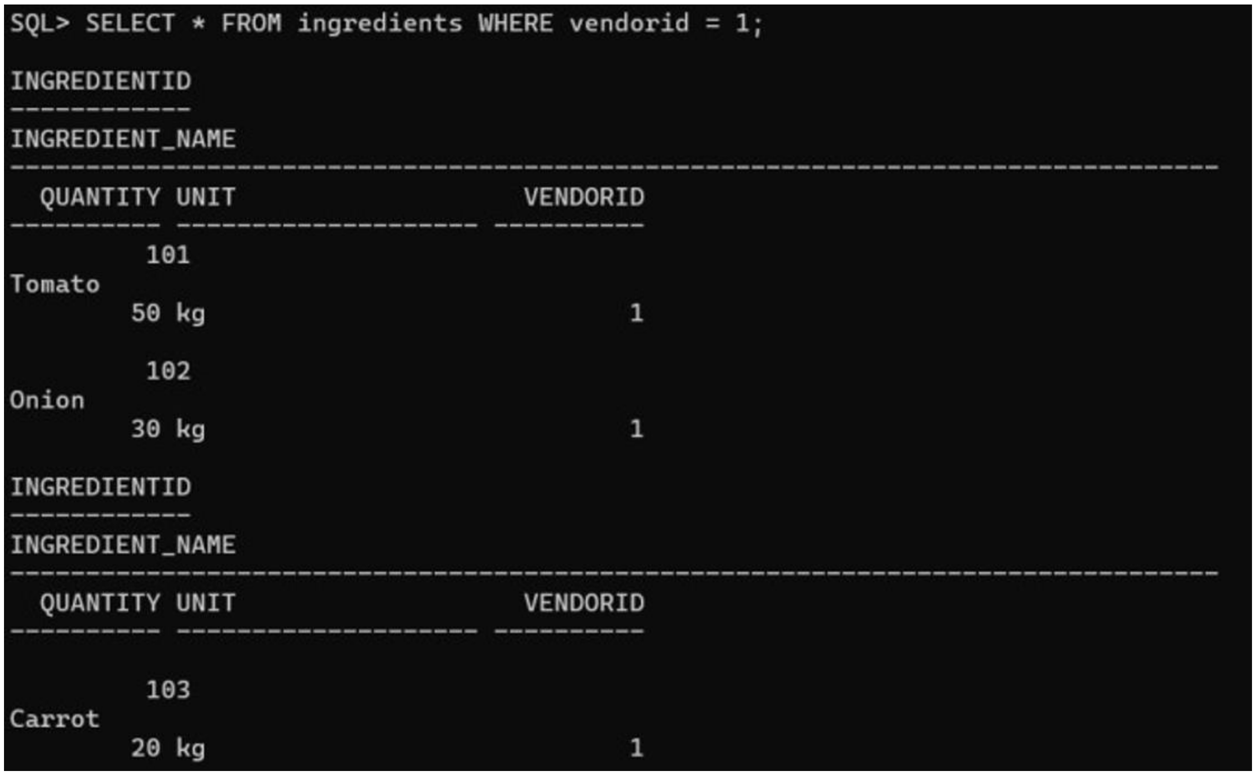
  SET vendorid = NULL

  WHERE vendorid = :OLD.vendorid;

END;

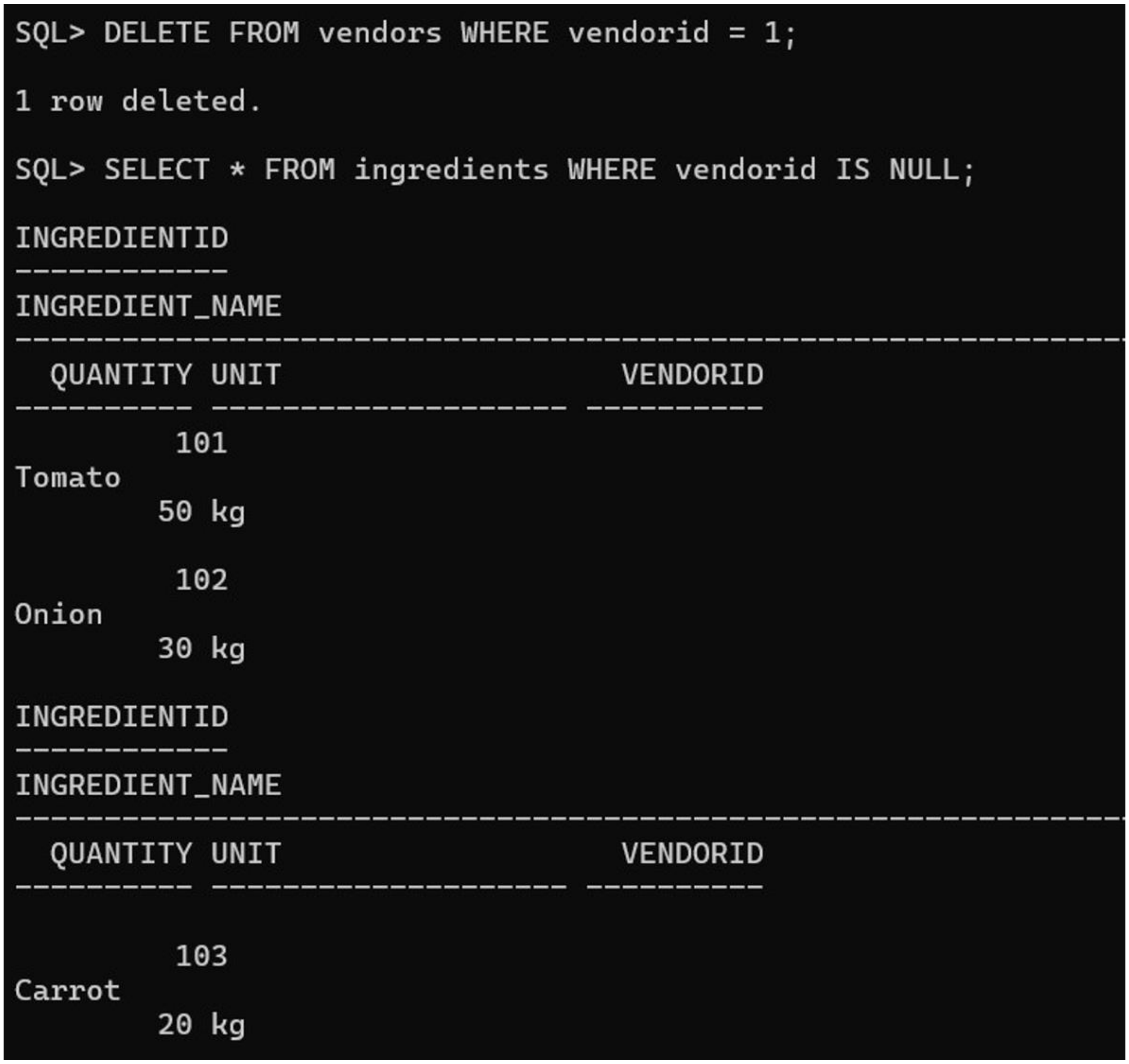
/

Before deleting vendorid = 1 contents:



```
SQL> SELECT * FROM ingredients WHERE vendorid = 1;

INGREDIENTID
------------
INGREDIENT_NAME
--------------------------------------------------------------------------------
  QUANTITY UNIT                   VENDORID
---------- -------------------- ----------
         101
Tomato
        50 kg                            1

         102
Onion
        30 kg                            1

INGREDIENTID
------------
INGREDIENT_NAME
--------------------------------------------------------------------------------
  QUANTITY UNIT                   VENDORID
---------- -------------------- ----------

         103
Carrot
        20 kg                            1
```

After deleting vendorid = 1 contents:



```
SQL> DELETE FROM vendors WHERE vendorid = 1;

1 row deleted.

SQL> SELECT * FROM ingredients WHERE vendorid IS NULL;

INGREDIENTID
------------
INGREDIENT_NAME
----------------------------------------------------------------------------
  QUANTITY UNIT                   VENDORID
---------- -------------------- ----------
         101
Tomato
        50 kg

         102
Onion
        30 kg

INGREDIENTID
------------
INGREDIENT_NAME
----------------------------------------------------------------------------
  QUANTITY UNIT                   VENDORID
---------- -------------------- ----------


         103
Carrot
        20 kg
```