EXERCISE : 16

PROGRAM-1

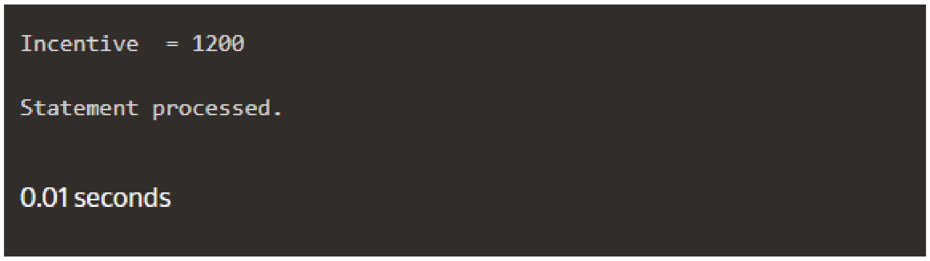
Problem Statement:

Write a PL/SQL block to calculate the incentive of an employee whose ID is 110.

PL/SQL CODE:

```
DECLARE
  incentive   NUMBER(8,2);
BEGIN
  SELECT salary * 0.12 INTO incentive
  FROM employees
  WHERE employee_id = 110;
DBMS_OUTPUT.PUT_LINE('Incentive  = ' || TO_CHAR(incentive));
END;
/
```

OUTPUT:



```
Incentive  = 1200

Statement processed.

0.01 seconds
```

PROGRAM-2

Problem Statement:

Write a PL/SQL block to show an invalid case-insensitive reference to a quoted and without quoted user-defined identifier.

PL/SQL CODE:

Identifier with quotation:

```
DECLARE
  "WELCOME" varchar2(10) := 'welcome'; -- identifier with quotation
BEGIN
  DBMS_Output.Put_Line("Welcome"); --reference to the identifier with quotation
and different case
END;
/
```

OUTPUT:

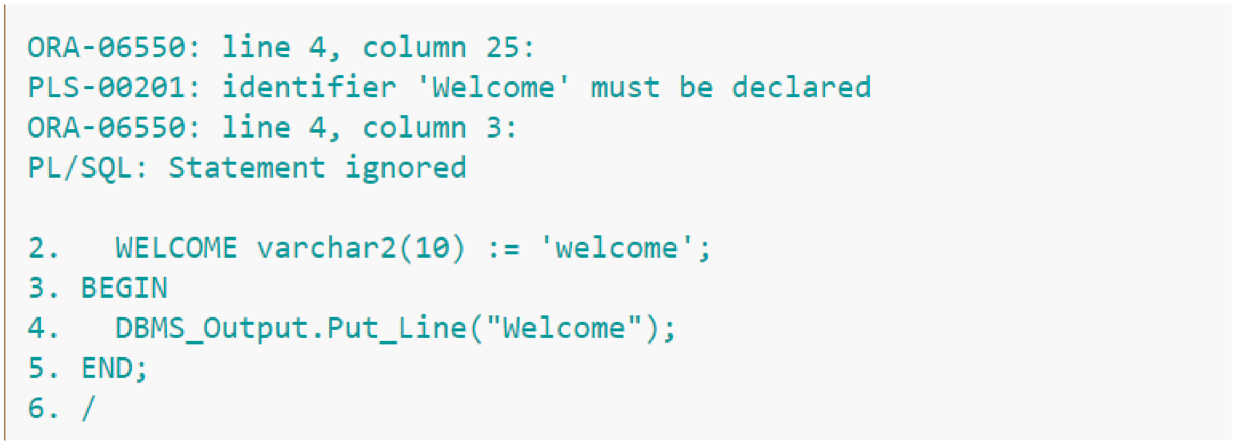
```
ORA-06550: line 4, column 25:
PLS-00201: identifier 'Welcome' must be declared
ORA-06550: line 4, column 3:
PL/SQL: Statement ignored

2.   "WELCOME" varchar2(10) := 'welcome';
3. BEGIN
4.   DBMS_Output.Put_Line("Welcome");
5. END;
6. /
```

Identifier with no quotation:

```
DECLARE
  WELCOME varchar2(10) := 'welcome'; -- identifier without quotation
BEGIN
  DBMS_Output.Put_Line("Welcome"); --reference to the identifier with quotation
and different case
END;
/
```

OUTPUT:



```
ORA-06550: line 4, column 25:
PLS-00201: identifier 'Welcome' must be declared
ORA-06550: line 4, column 3:
PL/SQL: Statement ignored

2.   WELCOME varchar2(10) := 'welcome';
3. BEGIN
4.   DBMS_Output.Put_Line("Welcome");
5. END;
6. /
```

PROGRAM-3

Problem Statement:

Write a PL/SQL block to adjust the salary of the employee whose ID 122. Sample table: employees

PL/SQL CODE:

```
DECLARE
  salary_of_emp  NUMBER(8,2);

  PROCEDURE approx_salary (
    emp         NUMBER,
    empsal IN OUT NUMBER,
    addless      NUMBER
  ) IS
  BEGIN
    empsal := empsal + addless;
  END;

BEGIN
  SELECT salary INTO salary_of_emp
  FROM employees
  WHERE employee_id = 122;

  DBMS_OUTPUT.PUT_LINE
   ('Before invoking procedure, salary_of_emp: ' || salary_of_emp);

  approx_salary (100, salary_of_emp, 1000);

  DBMS_OUTPUT.PUT_LINE
   ('After invoking procedure, salary_of_emp: ' || salary_of_emp);
END;
/
```

OUTPUT:

```
Before invoking procedure, salary_of_emp: 7900
After invoking procedure, salary_of_emp: 8900

Statement processed.

0.02 seconds
```

PROGRAM-4

Problem Statement:

Write a PL/SQL block to create a procedure using the "IS [NOT] NULL Operator" and show AND operator returns TRUE if and only if both operands are TRUE.

PL/SQL CODE:

```
CREATE OR REPLACE PROCEDURE pri_bool(
  boo_name   VARCHAR2,
  boo_val    BOOLEAN
) IS
BEGIN
  IF boo_val IS NULL THEN
    DBMS_OUTPUT.PUT_LINE( boo_name || ' = NULL');
  ELSIF boo_val = TRUE THEN
    DBMS_OUTPUT.PUT_LINE( boo_name || ' = TRUE');
  ELSE
    DBMS_OUTPUT.PUT_LINE( boo_name || ' = FALSE');
  END IF;
END;
/



DECLARE
  PROCEDURE pri_m_and_n (
    m  BOOLEAN,
    n  BOOLEAN
  ) IS
  BEGIN
   pri_bool ('m', m);
   pri_bool ('n', n);
   pri_bool ('m AND n', m AND n);
 END pri_m_and_n;

BEGIN
DBMS_OUTPUT.PUT_LINE('------------- FOR m and n both FALSE ---------------------
');
 pri_m_and_n (FALSE, FALSE);
DBMS_OUTPUT.PUT_LINE('------------- FOR m TRUE AND n FALSE ---------------------
');
 pri_m_and_n (TRUE, FALSE);
DBMS_OUTPUT.PUT_LINE('------------- FOR m FALSE AND n TRUE ---------------------
');
```

```
 pri_m_and_n (FALSE, TRUE);
DBMS_OUTPUT.PUT_LINE('------------- FOR m TRUE AND n TRUE ---------------------
');
 pri_m_and_n (TRUE, TRUE);
DBMS_OUTPUT.PUT_LINE('------------- FOR m TRUE AND n NULL ---------------------
');
 pri_m_and_n (TRUE, NULL);
DBMS_OUTPUT.PUT_LINE('------------- FOR m FALSE AND n NULL---------------------
');
 pri_m_and_n (FALSE, NULL);
DBMS_OUTPUT.PUT_LINE('------------- FOR m NULL AND n TRUE ---------------------
');
 pri_m_and_n (NULL, TRUE);
DBMS_OUTPUT.PUT_LINE('------------- FOR m NULL AND n FALSE ---------------------
');
 pri_m_and_n (NULL, FALSE);
END;
/
```

OUTPUT:

```
------------- FOR m and n both FALSE ---------------------
m = FALSE
n = FALSE
m AND n = FALSE
------------- FOR m TRUE AND n FALSE ---------------------
m = TRUE
n = FALSE
m AND n = FALSE
------------- FOR m FALSE AND n TRUE ---------------------
m = FALSE
n = TRUE
m AND n = FALSE
------------- FOR m TRUE AND n TRUE ---------------------
m = TRUE
n = TRUE
m AND n = TRUE
------------- FOR m TRUE AND n NULL ---------------------
m = TRUE
n = NULL
m AND n = NULL
------------- FOR m FALSE AND n NULL---------------------
m = FALSE
n = NULL
m AND n = FALSE
------------- FOR m NULL AND n TRUE ---------------------
m = NULL
n = TRUE
m AND n = NULL
------------- FOR m NULL AND n FALSE ---------------------
m = NULL
n = FALSE
m AND n = FALSE

Statement processed.

0.00 seconds
```

PROGRAM-5

Problem Statement:

Write a PL/SQL block to describe the usage of LIKE operator including wildcard characters and escape character.

PL/SQL CODE:

LIKE WITH WILDCARDS:

```
DECLARE
  PROCEDURE pat_match (
    test_string   VARCHAR2,
    pattern       VARCHAR2
  ) IS
  BEGIN
    IF test_string LIKE pattern THEN
      DBMS_OUTPUT.PUT_LINE ('TRUE');
    ELSE
      DBMS_OUTPUT.PUT_LINE ('FALSE');
    END IF;
  END;
BEGIN
  pat_match('Blweate', 'B%a_e');
  pat_match('Blweate', 'B%A_E');
END;
/
```

OUTPUT:

```
TRUE
FALSE

Statement processed.

0.00 seconds
```

LIKE WITH ESCAPE:

```
DECLARE
  PROCEDURE pat_escape (mar_achiv VARCHAR2) IS
  BEGIN
    IF mar_achiv LIKE '70\%  out of 100!' ESCAPE '\' THEN
```

```
      DBMS_OUTPUT.PUT_LINE ('TRUE');
    ELSE
      DBMS_OUTPUT.PUT_LINE ('FALSE');
    END IF;
  END;
BEGIN
  pat_escape('Go and try your best');
  pat_escape('70%  out of 100!');
END;
/
```

OUTPUT:

```
FALSE
TRUE

Statement processed.

0.00 seconds
```

PROGRAM-6

Problem Statement:

Write a PL/SQL program to arrange the number of two variable in such a way that the small number will store in num_small variable and large number will store in num_large variable.

PL/SQL CODE:

```
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;
/
```

OUTPUT:

```
num_small = 5
num_large = 8
```

PROGRAM-7

Problem Statement:

Write a PL/SQL procedure to calculate the incentive on a target achieved and display the message either the record updated or not.

PL/SQL CODE:

```
DECLARE
  PROCEDURE test1 (
    sal_achieve  NUMBER,
    target_qty  NUMBER,
    emp_id NUMBER
  )
  IS
    incentive   NUMBER := 0;
    updated  VARCHAR2(3) := 'No';
  BEGIN
    IF sal_achieve > (target_qty + 200) THEN
      incentive := (sal_achieve - target_qty)/4;

      UPDATE emp
      SET salary = salary + incentive
      WHERE employee_id = emp_id;

      updated := 'Yes';
    END IF;

    DBMS_OUTPUT.PUT_LINE (
      'Table updated?  ' || updated || ', ' ||
      'incentive = ' || incentive || '.'
    );
  END test1;
BEGIN
  test1(2300, 2000, 144);
  test1(3600, 3000, 145);
END;
/
```

OUTPUT:

```
Table updated?  Yes, incentive = 75.
Table updated?  Yes, incentive = 150.

PL/SQL procedure successfully completed.
```

PROGRAM-8

Problem Statement:

Write a PL/SQL procedure to calculate incentive achieved according to the specific sale limit.

PL/SQL CODE:

```
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;
/
```

OUTPUT:

```
Sale achieved : 45000, incentive : 1800.

Sale achieved : 36000, incentive : 800.

Sale achieved : 28000, incentive : 500.

PL/SQL procedure successfully completed.
```

PROGRAM-9

Problem Statement:

Write a PL/SQL program to count number of employees in department 50 and check whether this department have any vacancies or not. There are 45 vacancies in this department.

PL/SQL CODE:

```
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;
/
```

OUTPUT:

```
The employees are in the department 50: 45
There are no vacancies in the department 50.

PL/SQL procedure successfully completed.
```

PROGRAM-10

Problem Statement:

Write a PL/SQL program to count number of employees in a specific department and check whether this department have any vacancies or not. If any vacancies,how many vacancies are in that department.

PL/SQL CODE:

```
DECLARE
    tot_emp NUMBER;
      get_dep_id NUMBER;

BEGIN
    get_dep_id := '&new_dep_id';
    SELECT Count(*)
    INTO   tot_emp
    FROM   employees e
           join departments d
             ON e.department_id = d.department_id
    WHERE  e.department_id = get_dep_id;

    dbms_output.Put_line ('The employees are in the department '||get_dep_id||'
is: '
                          ||To_char(tot_emp));

    IF tot_emp >= 45 THEN
      dbms_output.Put_line ('There are no vacancies in the department
'||get_dep_id);
    ELSE
      dbms_output.Put_line ('There are '||to_char(45-tot_emp)||' vacancies in
department '|| get_dep_id );
    END IF;
END;
/
```

OUTPUT:

```
Enter value for new_dep_id: 20
old   6:      get_dep_id := '&new_dep_id';
new   6:      get_dep_id := '20';
The employees are in the department 20 is: 2
There are 43 vacancies in department 20

PL/SQL procedure successfully completed.
```

PROGRAM-11

Problem Statement:

Write a PL/SQL program to display the employee IDs, names, job titles, hire dates,and salaries of all employees.

PL/SQL CODE:

```
DECLARE
v_employee_idemployees.employee_id%TYPE;
v_full_nameemployees.first_name%TYPE;
v_job_idemployees.job_id%TYPE;
v_hire_dateemployees.hire_date%TYPE;
v_salaryemployees.salary%TYPE;
  CURSOR c_employees IS
    SELECT employee_id, first_name || ' ' || last_name AS full_name, job_id,
hire_date, salary
    FROM employees;
BEGIN
  DBMS_OUTPUT.PUT_LINE('Employee ID | Full Name | Job Title | Hire Date |
Salary');
  DBMS_OUTPUT.PUT_LINE('---------------------------------------------------------
----------');
  OPEN c_employees;
  FETCH c_employees INTO v_employee_id, v_full_name, v_job_id, v_hire_date,
v_salary;
  WHILE c_employees%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE(v_employee_id || '        | ' || v_full_name || ' | ' ||
v_job_id || ' | ' || v_hire_date || ' | ' || v_salary);
    FETCH c_employees INTO v_employee_id, v_full_name, v_job_id, v_hire_date,
v_salary;
  END LOOP;
  CLOSE c_employees;
END;
```

/

OUTPUT:

```
Employee ID | Full Name | Job Title | Hire Date | Salary
-------------------------------------------------------------------
100         | Steven      King         | AD_PRES | 17-JUN-87 | 24000
101         | NeenaKochhar       | AD_VP | 18-JUN-87 | 17000
102         | Lex         De Haan      | AD_VP | 19-JUN-87 | 17000
103         | Alexander   Hunold       | IT_PROG | 20-JUN-87 | 9000
104         | Bruce       Ernst        | IT_PROG | 21-JUN-87 | 6000
105         | David       Austin       | IT_PROG | 22-JUN-87 | 4800
106         | ValliPataballa     | IT_PROG | 23-JUN-87 | 4800
107         | Diana       Lorentz      | IT_PROG | 24-JUN-87 | 4200
108         | Nancy       Greenberg    | FI_MGR | 25-JUN-87 | 12000
109         | Daniel      Faviet       | FI_ACCOUNT | 26-JUN-87 | 9000
110         | John        Chen         | FI_ACCOUNT | 27-JUN-87 | 8200
111         | Ismael      Sciarra      | FI_ACCOUNT | 28-JUN-87 | 7700
112         | Jose ManueUrman        | FI_ACCOUNT | 29-JUN-87 | 7800
113         | Luis        Popp         | FI_ACCOUNT | 30-JUN-87 | 6900
.....
```

PROGRAM-12

Problem Statement:

Write a PL/SQL program to display the employee IDs, names, and department names of all employees.

PL/SQL CODE:

```
DECLARE
  CURSOR emp_cursor IS
    SELECT e.employee_id, e.first_name, m.first_name AS manager_name
    FROM employees e
    LEFT JOIN employees m ON e.manager_id = m.employee_id;
emp_recordemp_cursor%ROWTYPE;
BEGIN
  OPEN emp_cursor;
  FETCH emp_cursor INTO emp_record;
  WHILE emp_cursor%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp_record.employee_id);
    DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_record.first_name);
    DBMS_OUTPUT.PUT_LINE('Manager Name: ' || emp_record.manager_name);
    DBMS_OUTPUT.PUT_LINE('-------------------------');
    FETCH emp_cursor INTO emp_record;
  END LOOP;
  CLOSE emp_cursor;
```

```
END;
/
```

OUTPUT:

```
Employee ID: 202
Employee Name: Pat
Manager Name: Michael
-------------------------
Employee ID: 206
Employee Name: William
Manager Name: Shelley
-------------------------
Employee ID: 201
Employee Name: Michael
Manager Name: Steven
-------------------------
Employee ID: 101
Employee Name: Neena
Manager Name: Steven
-------------------------
Employee ID: 102
Employee Name: Lex
Manager Name: Steven
-------------------------
Employee ID: 114
Employee Name: Den
Manager Name: Steven
-------------------------

.....
```

PROGRAM-13

Problem Statement:

Write a PL/SQL program to display the job IDs, titles, and minimum salaries of all jobs.

PL/SQL CODE:

```
DECLARE
  CURSOR job_cursor IS
    SELECT job_id, job_title, min_salary
    FROM jobs;
job_recordjob_cursor%ROWTYPE;
```

```
BEGIN
  OPEN job_cursor;
  FETCH job_cursor INTO job_record;
  WHILE job_cursor%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE('Job ID: ' || job_record.job_id);
    DBMS_OUTPUT.PUT_LINE('Job Title: ' || job_record.job_title);
    DBMS_OUTPUT.PUT_LINE('Minimum Salary: ' || job_record.min_salary);
    DBMS_OUTPUT.PUT_LINE('-------------------------');
    FETCH job_cursor INTO job_record;
  END LOOP;
  CLOSE job_cursor;
END;
/
```

OUTPUT:

```
Job ID: AD_PRES
Job Title: President
Minimum Salary: 20000
-------------------------
Job ID: AD_VP
Job Title: Administration Vice President
Minimum Salary: 15000
-------------------------
Job ID: AD_ASST
Job Title: Administration Assistant
Minimum Salary: 3000
-------------------------
Job ID: FI_MGR
Job Title: Finance Manager
Minimum Salary: 8200
-------------------------
Job ID: FI_ACCOUNT
Job Title: Accountant
Minimum Salary: 4200
-------------------------
Job ID: AC_MGR
Job Title: Accounting Manager
Minimum Salary: 8200
-------------------------
Job ID: AC_ACCOUNT
Job Title: Public Accountant
Minimum Salary: 4200
-------------------------
Job ID: SA_MAN
Job Title: Sales Manager
Minimum Salary: 10000
-------------------------

.....
```

PROGRAM-14

Problem Statement:

Write a PL/SQL program to display the employee IDs, names, and job history start dates of all employees.

PL/SQL CODE:

```
DECLARE
    CURSOR employees_cur IS
      SELECT employee_id,
             first_name,
             job_title,
             hire_date
      FROM   employees
             NATURAL join jobs;
    emp_first_date DATE;
BEGIN
    dbms_output.Put_line(Rpad('Employee ID', 15)
                         ||Rpad('First Name', 25)
                         ||Rpad('Job Title', 35)
                         ||'First Date');

dbms_output.Put_line('-----------------------------------------------------------
------------------------------');

FOR emp_sal_rec IN employees_cur LOOP
    -- find out most recent end_date in job_history
    SELECT Max(end_date) + 1
    INTO   emp_first_date
    FROM   job_history
    WHERE  employee_id = emp_sal_rec.employee_id;

    IF emp_first_date IS NULL THEN
      emp_first_date := emp_sal_rec.hire_date;
    END IF;

    dbms_output.Put_line(Rpad(emp_sal_rec.employee_id, 15)
                         ||Rpad(emp_sal_rec.first_name, 25)
                         || Rpad(emp_sal_rec.job_title, 35)
                         || To_char(emp_first_date, 'dd-mon-yyyy'));
END LOOP;
END;
/
```

OUTPUT:

```
SQL> /
Employee ID    First Name               Job Title                        First Date
-----------------------------------------------------------------------------------------
206            William                  Public Accountant                07-jun-2002
205            Shelley                  Accounting Manager               07-jun-2002
200            Jennifer                 Administration Assistant         01-jan-2007
100            Steven                   President                        17-jun-2003
102            Lex                      Administration Vice President    25-jul-2006
101            Neena                    Administration Vice President    16-mar-2005
110            John                     Accountant                       28-sep-2005
109            Daniel                   Accountant                       16-aug-2002
113            Luis                     Accountant                       07-dec-2007
111            Ismael                   Accountant                       30-sep-2005
112            Jose Manuel              Accountant                       07-mar-2006
108            Nancy                    Finance Manager                  17-aug-2002
203            Susan                    Human Resources Representative   07-jun-2002
...
```

PROGRAM-15

Problem Statement:

Write a PL/SQL program to display the employee IDs, names, and job history end dates of all employees.

PL/SQL CODE:

```
DECLARE
v_employee_idemployees.employee_id%TYPE;
v_first_nameemployees.first_name%TYPE;
v_end_datejob_history.end_date%TYPE;
  CURSOR c_employees IS
    SELECT e.employee_id, e.first_name, jh.end_date
    FROM employees e
    JOIN job_history jh ON e.employee_id = jh.employee_id;
BEGIN
  OPEN c_employees;
  FETCH c_employees INTO v_employee_id, v_first_name, v_end_date;
  WHILE c_employees%FOUND LOOP
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_employee_id);
    DBMS_OUTPUT.PUT_LINE('Employee Name: ' || v_first_name);
    DBMS_OUTPUT.PUT_LINE('End Date: ' || v_end_date);
    DBMS_OUTPUT.PUT_LINE('----------------------');
    FETCH c_employees INTO v_employee_id, v_first_name, v_end_date;
  END LOOP;
  CLOSE c_employees;
END;
```

OUTPUT:

```
Employee ID: 101
Employee Name: Neena
End Date: 27-OCT-93
----------------------
Employee ID: 101
Employee Name: Neena
End Date: 15-MAR-97
----------------------
Employee ID: 102
Employee Name: Lex
End Date: 24-JUL-98
----------------------
Employee ID: 114
Employee Name: Den
End Date: 31-DEC-99
----------------------
Employee ID: 122
Employee Name: Payam
End Date: 31-DEC-99
----------------------
Employee ID: 176
Employee Name: Jonathon
End Date: 31-DEC-98
----------------------
Employee ID: 176
Employee Name: Jonathon
End Date: 31-DEC-99
----------------------

.....
```