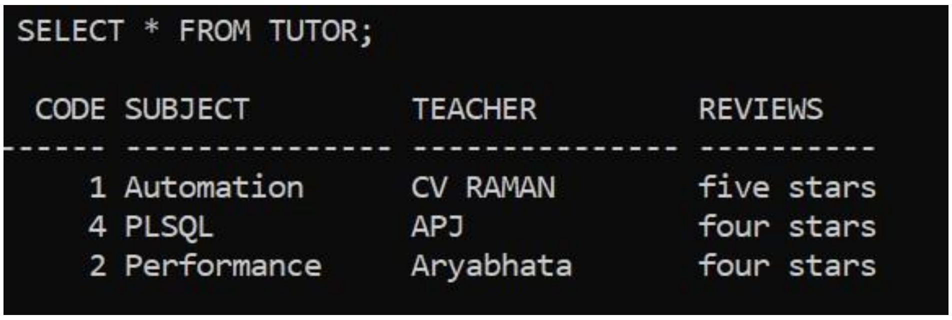
VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Lab sheet – 10 (L9+L10)

Cursor

A cursor in PL/SQL gives a name and acts as a pointer to the area of work called a context area and then uses its information. It keeps the number of rows processed by the SQL statement. These rows are called as an active set. The size of the active set is equal to the count of the rows that meet the condition.

```
CREATE TABLE TUTOR (CODE INT NOT NULL,
                    SUBJECT VARCHAR(15) NOT NULL,
                    TEACHER VARCHAR(15),
                    REVIEWS VARCHAR (10) NOT NULL,
                    PRIMARY KEY (CODE));


INSERT INTO TUTOR (CODE,SUBJECT,TEACHER,REVIEWS) VALUES
(1, 'Automation', 'CV RAMAN', 'five stars');

INSERT INTO TUTOR (CODE,SUBJECT,TEACHER,REVIEWS) VALUES
(4, 'PLSQL', 'APJ', 'four stars');

INSERT INTO TUTOR (CODE,SUBJECT,TEACHER,REVIEWS)
VALUES (2, 'Performance', 'Aryabhata', 'four stars');
```



```
SELECT * FROM TUTOR;

 CODE SUBJECT         TEACHER         REVIEWS
------ --------------- --------------- ----------
     1 Automation      CV RAMAN        five stars
     4 PLSQL           APJ             four stars
     2 Performance     Aryabhata       four stars
```

Implicit Cursors

The implicit cursors are allocated by Oracle by default while executing SQL statements. It holds the affected rows by the DML operations like UPDATE, DELETE and INSERT. Thus, implicit cursors are used when we don’t have an explicit cursor in place.

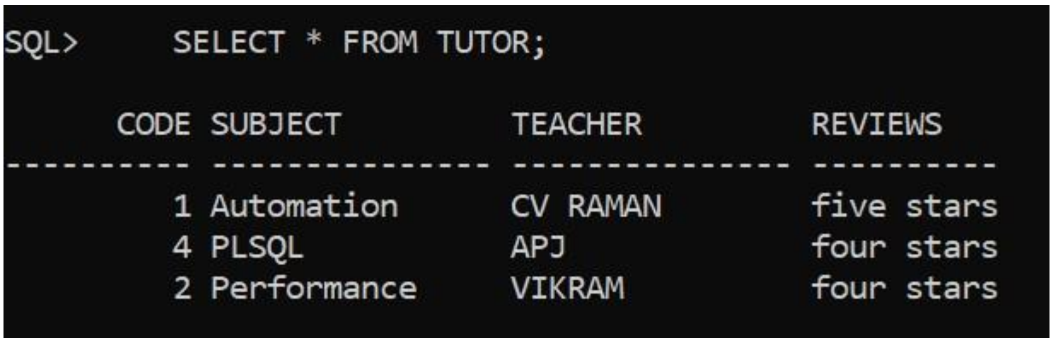
While we are inserting a row, the cursor keeps that particular data. Similarly, for deletion and updating operations, the affected rows are stored by the cursors. The implicit cursors are not given any names and hence cannot be manipulated by the developers and the data contained on it cannot be used anywhere.

Implementation Code- with the implicit cursor:

```
DECLARE     total_count
number(30);
BEGIN
     --updating a row
    UPDATE TUTOR
    SET TEACHER = 'VIKRAM' where CODE = 2;
     -- result in boolean, true returned if no rows affected
IF sql%notfound THEN
        dbms_output.put_line('no subjects fetched');

         -- result in boolean, true returned if any rows affected
ELSIF sql%found THEN

        --   count   the   number   of   rows   affected   rows   affected
total_count := sql%rowcount;          dbms_output.put_line( total_count
|| ' teacher name updated ');      END IF;
END;
/
```

Let us now verify the changes reflected in the table named TUTOR.

SELECT * FROM TUTOR;



```
SQL>     SELECT * FROM TUTOR;

      CODE SUBJECT            TEACHER            REVIEWS
---------- ------------------ ------------------ ----------
         1 Automation         CV RAMAN           five stars
         4 PLSQL              APJ                four stars
         2 Performance        VIKRAM             four stars
```

Explicit Cursors

The developers can have their own user-defined context area to run DML operations. Thus they can exercise more power over it. The declaration section of the PL/SQL block of code contains explicit cursors. It is normally built on SELECT operations that fetch multiple rows.

Syntax of explicit cursor:

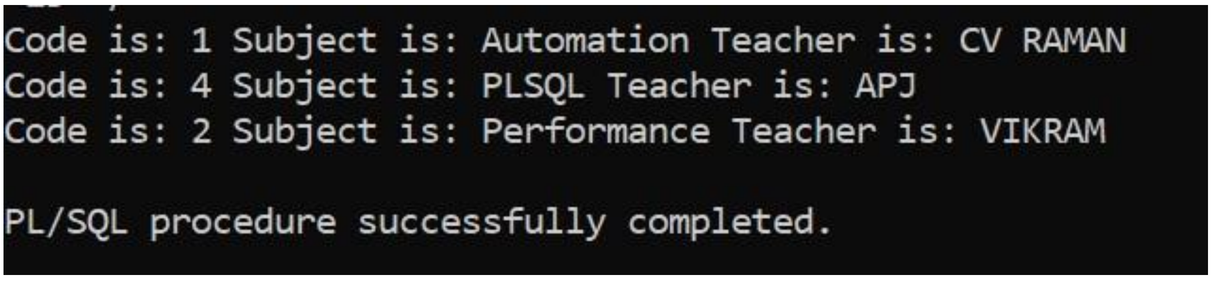
```
DECLARE
CURSOR <<cursor name>> IS <<select statement>>
<<Cursor variable>>
BEGIN
```

```
OPEN <<cursor name>>;
FETCH <<cursor name>> INTO <Cursor variable>;
.
.
CLOSE <cursor name>;
END;
```

Implementation Code: with explicit cursor:

```
SET SERVEROUTPUT ON;
DECLARE
   -- cursor declaration
CURSOR t_tutorials is
SELECT code, subject, teacher FROM Tutor;
t_code Tutor.code%type;  t_subject
Tutor.subject%type;  t_teacher
Tutor.teacher%type;


BEGIN

   -- opening a cursor
   OPEN t_tutorials;
LOOP

    -- fetching values from cursor
    FETCH t_tutorials into t_code, t_subject, t_teacher;
    EXIT WHEN t_tutorials%notfound;

    -- printing in console
    dbms_output.put_line('Code is: ' || t_code || ' ' || 'Subject is: ' ||
t_subject || ' Teacher is: ' || t_teacher);
END LOOP;
CLOSE t_tutorials;
END;
/
```

The output of the above code should be:



```
Code is: 1 Subject is: Automation Teacher is: CV RAMAN
Code is: 4 Subject is: PLSQL Teacher is: APJ
Code is: 2 Subject is: Performance Teacher is: VIKRAM

PL/SQL procedure successfully completed.
```

Notes:

Explicit Cursor works on the processes listed below:

#1) Cursor declaration for memory initialization. Here, a named context area is created which serves as a cursor name.
Syntax:
CURSOR tutorial_s IS

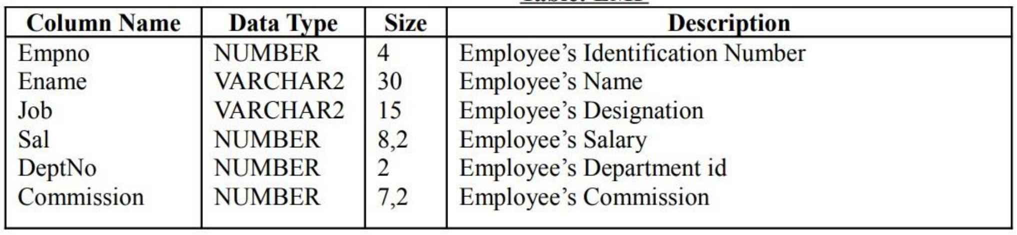
SELECT code FROM TUTORIAL;
#2) Cursor opening for memory allocation. A cursor is now available for fetching the updated rows from the database.
Syntax:
OPEN tutorial_s;

#3) Cursor is fetched for getting the data. After the SELECT operation is done, the rows obtained are put in the memory allocated and these are now considered as active sets. The cursor can access one row at a time.
Syntax:
FETCH tutorial_s INTO c_code;

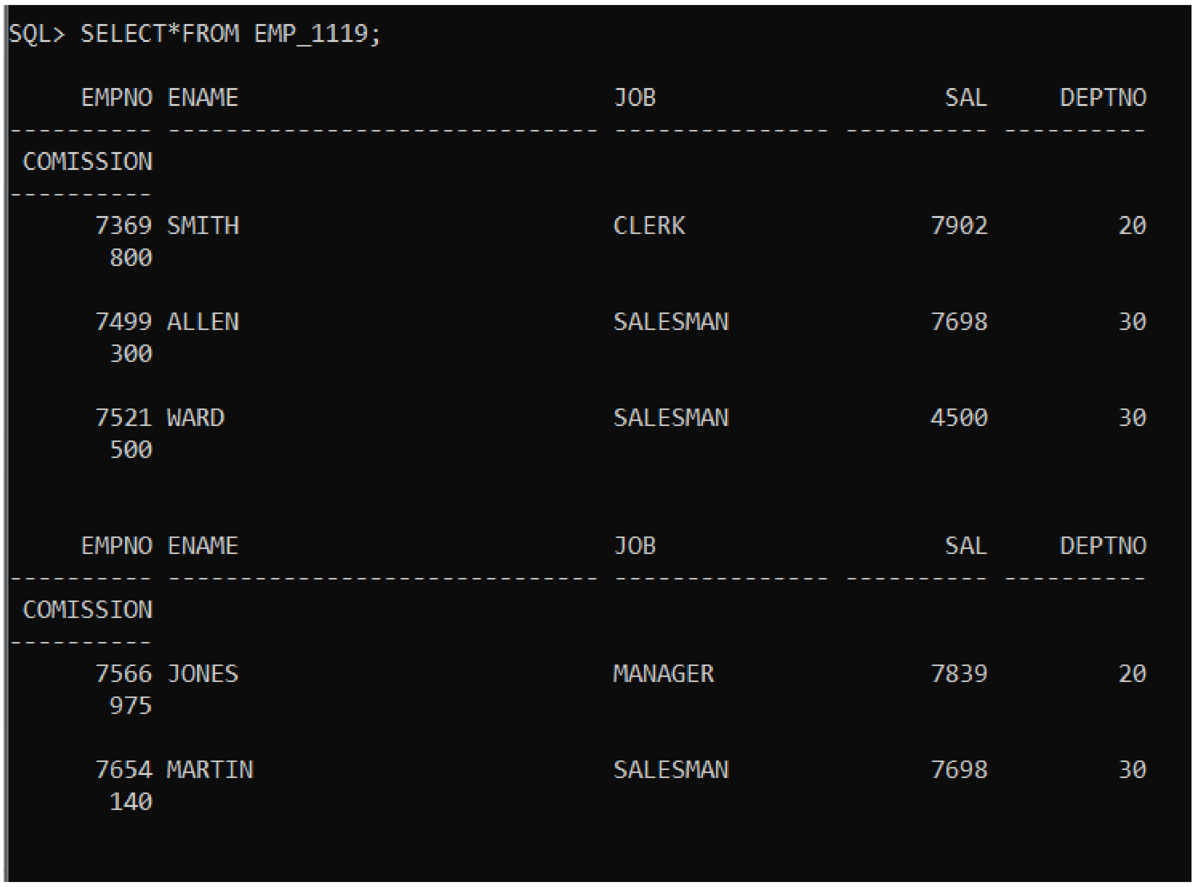
#4) Cursor is finally closed to free the allocated memory. As all the records are obtained one by one, the cursor is closed to release context area memory.
Syntax:
CLOSE tutorial_s;

Exercise Questions

Table: EMP

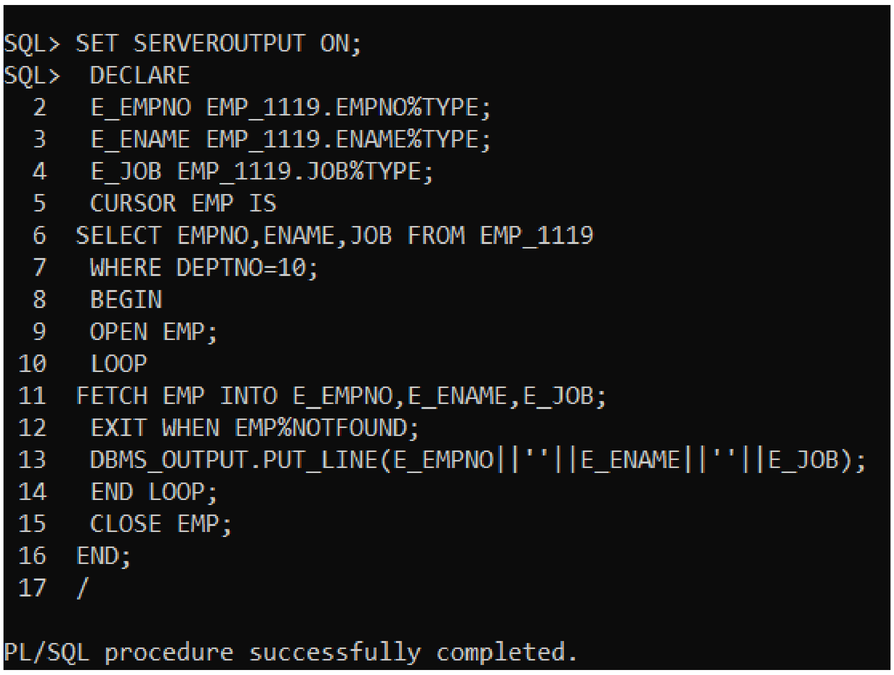


Column Name	Data Type	Size	Description
Empno	NUMBER	4	Employee's Identification Number
Ename	VARCHAR2	30	Employee's Name
Job	VARCHAR2	15	Employee's Designation
Sal	NUMBER	8,2	Employee's Salary
DeptNo	NUMBER	2	Employee's Department id
Commission	NUMBER	7,2	Employee's Commission



```
SQL> SELECT*FROM EMP_1119;

     EMPNO ENAME                          JOB                   SAL     DEPTNO
---------- ------------------------------ --------------- ---------- ----------
 COMISSION
----------
      7369 SMITH                          CLERK                7902         20
       800

      7499 ALLEN                          SALESMAN             7698         30
       300

      7521 WARD                           SALESMAN             4500         30
       500


     EMPNO ENAME                          JOB                   SAL     DEPTNO
---------- ------------------------------ --------------- ---------- ----------
 COMISSION
----------
      7566 JONES                          MANAGER              7839         20
       975

      7654 MARTIN                         SALESMAN             7698         30
       140
```

(a) Write a PL/SQL code to display the EMP_1119no, Ename and Job of EMP_1119loyees of DeptNo 10 with CURSOR FOR LOOP Statement.



```
SQL> SET SERVEROUTPUT ON;
SQL>  DECLARE
      E_EMPNO EMP_1119.EMPNO%TYPE;
      E_ENAME EMP_1119.ENAME%TYPE;
      E_JOB EMP_1119.JOB%TYPE;
      CURSOR EMP IS
     SELECT EMPNO,ENAME,JOB FROM EMP_1119
      WHERE DEPTNO=10;
      BEGIN
      OPEN EMP;
      LOOP
     FETCH EMP INTO E_EMPNO,E_ENAME,E_JOB;
      EXIT WHEN EMP%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE(E_EMPNO||''||E_ENAME||''||E_JOB);
      END LOOP;
      CLOSE EMP;
     END;
     /

PL/SQL procedure successfully completed.
```

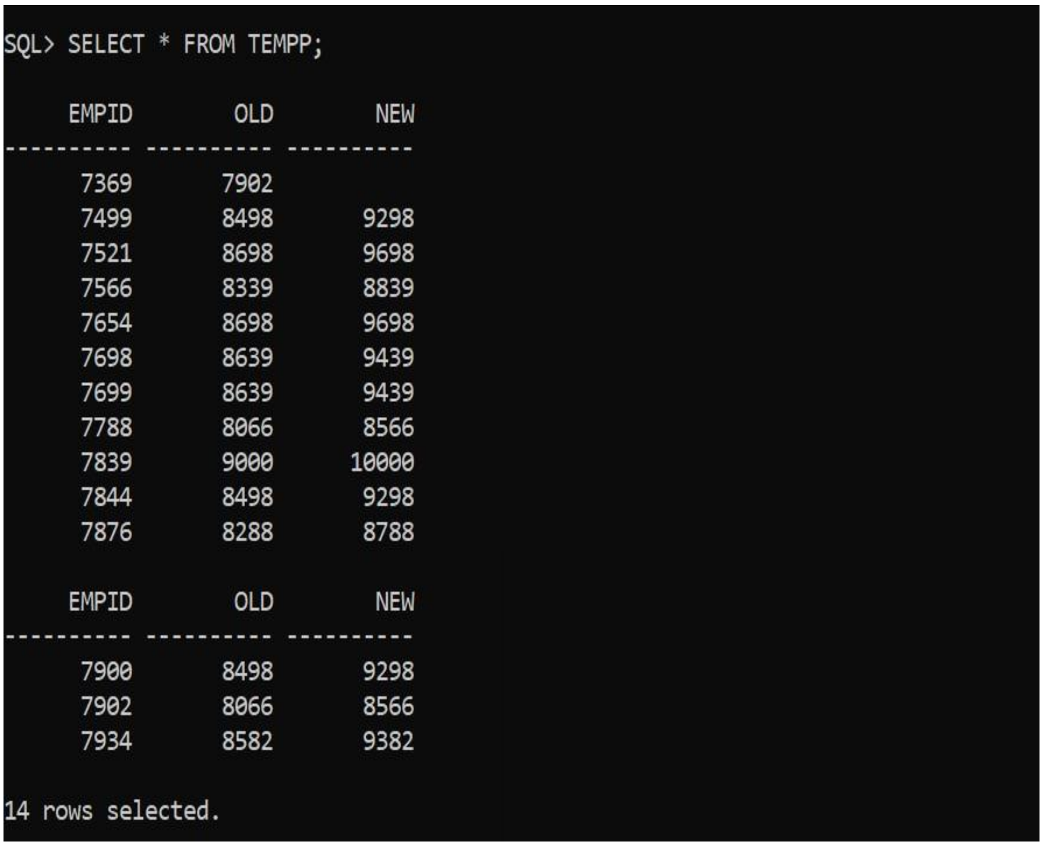
(b) Create a Cursor to increase the salary of EMP_1119loyees according to the following conditions:

Salary of DeptNo 10 EMP_1119loyees increased by 1000.

Salary of DeptNo 20 EMP_1119loyees increased by 500.

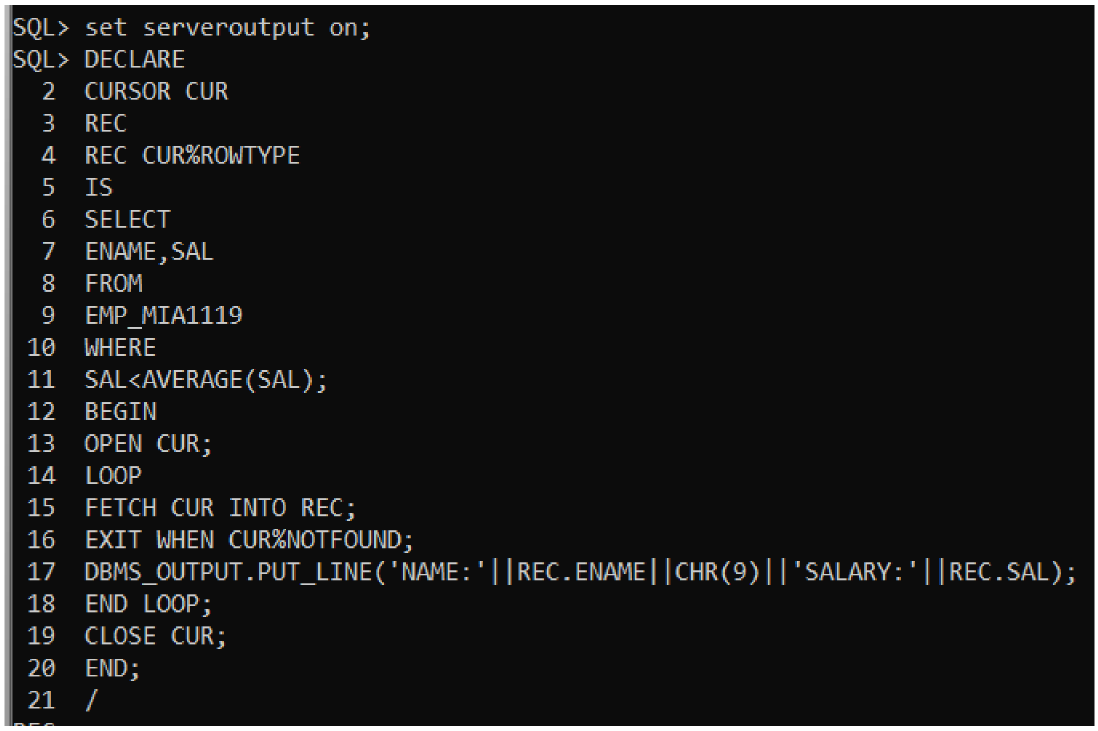
Salary of DeptNo 30 EMP_1119loyees increased by 800.

Also, store the EMP_1119No, old salary and new salary in a Table TEMP_1119 having three columns EMP_1119id, Old and New.



```
SQL> SELECT * FROM TEMPP;

     EMPID        OLD        NEW
---------- ---------- ----------
      7369       7902
      7499       8498       9298
      7521       8698       9698
      7566       8339       8839
      7654       8698       9698
      7698       8639       9439
      7699       8639       9439
      7788       8066       8566
      7839       9000      10000
      7844       8498       9298
      7876       8288       8788

     EMPID        OLD        NEW
---------- ---------- ----------
      7900       8498       9298
      7902       8066       8566
      7934       8582       9382

14 rows selected.
```

c) Write a program in PL/SQL to create a cursor displays the name and salary of each EMP_1119loyee in the EMP_1119LOYEES table whose salary is less than average salary of all EMP_1119loyee.



```
SQL> set serveroutput on;
SQL> DECLARE
CURSOR CUR
REC
REC CUR%ROWTYPE
IS
SELECT
ENAME,SAL
FROM
EMP_MIA1119
WHERE
SAL<AVERAGE(SAL);
BEGIN
OPEN CUR;
LOOP
FETCH CUR INTO REC;
EXIT WHEN CUR%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('NAME:'||REC.ENAME||CHR(9)||'SALARY:'||REC.SAL);
END LOOP;
CLOSE CUR;
END;
/
```

TABLE

```
CREATE TABLE EMP_1119 (
    EMP_1119no           NUMBER(4) NOT NULL CONSTRAINT EMP_1119_pk
PRIMARY KEY,     ename           VARCHAR2(10),     job
VARCHAR2(9),
    sal             NUMBER(7,2) CONSTRAINT EMP_1119_sal_ck CHECK (sal >
0),     deptno          NUMBER(2)
    comm            NUMBER(7,2),
);
```

****

```
CREATE TABLE EMP_1119loyee1 (
    EMP_1119no           NUMBER(4) NOT NULL CONSTRAINT EMP_1119_pk
PRIMARY KEY,     ename           VARCHAR2(10),     job
VARCHAR2(9),     mgr             NUMBER(4),     hiredate        DATE,
    sal             NUMBER(7,2) CONSTRAINT EMP_1119_sal_ck CHECK (sal >
0),     comm            NUMBER(7,2),     deptno          NUMBER(2)));
```

Modify the below values as per the EMP_1119 table domain requirements:

```
INSERT INTO EMP_1119 VALUES (7369,'SMITH','CLERK',7902, 20, 800);
INSERT INTO EMP_1119 VALUES (7499,'ALLEN','SALESMAN',7698, 30, 300);
INSERT INTO EMP_1119 VALUES (7521,'WARD','SALESMAN', 30, 500);
INSERT INTO EMP_1119 VALUES (7566,'JONES','MANAGER',7839, 20, 975);
INSERT INTO EMP_1119 VALUES (7654,'MARTIN','SALESMAN',7698, 30, 140);
INSERT INTO EMP_1119 VALUES (7698,'BLAKE','MANAGER',7839,'01-MAY-
81',2850,NULL,30);
INSERT INTO EMP_1119 VALUES (7782,'CLARK','MANAGER',7839,'09-JUN-
81',2450,NULL,10);
INSERT INTO EMP_1119 VALUES (7788,'SCOTT','ANALYST',7566,'19-APR-
87',3000,NULL,20);
INSERT INTO EMP_1119 VALUES (7839,'KING','PRESIDENT',NULL,'17-NOV-
81',5000,NULL,10);
INSERT INTO EMP_1119 VALUES (7844,'TURNER','SALESMAN',7698,'08-SEP-
81',1500,0,30);
INSERT INTO EMP_1119 VALUES (7876,'ADAMS','CLERK',7788,'23-MAY-
87',1100,NULL,20);
INSERT INTO EMP_1119 VALUES (7900,'JAMES','CLERK',7698,'03-DEC-
81',950,NULL,30);
INSERT INTO EMP_1119 VALUES (7902,'FORD','ANALYST',7566,'03-
DEC81',3000,NULL,20);
INSERT INTO EMP_1119 VALUES (7934,'MILLER','CLERK',7782,'23-JAN-
82',1300,NULL,10);
```
****