PL/SQL Fundamentals

Additional Practices

PL/SQL Fundamentals

Additional Practices

41023GC13
Production 1.3
July 1999
M08922

ORACLE®

Authors

Ellen Gravina
Priya Nathan

Technical Contributors and Reviewers

Claire Bennet
Christa Miethaner
Tony Hickman
Sherin Nassa
Nancy Greenberg
Hazel Russell
Kenneth Goetz
Piet van Zon
Ulrike Dietrich
Helen Robertson
Thomas Nguyen
Lisa Jansson
Kuljit Jassar

Publisher

Jerry Brosnan

Additional Practices

These exercises can be used for extra practice when discussing the following topics: declaring variables and writing executable statements..

1. Evaluate each of the following declarations. Determine which of them are *not* legal and explain why.

   a.
   ```
   DECLARE
     v_name,v_dept                    VARCHAR2(14);
   ```

   b.
   ```
   DECLARE
     v_test                           NUMBER(5);
   ```

   c.
   ```
   DECLARE
     V_MAXSALARY                      NUMBER(7,2) = 5000;
   ```

   d.
   ```
   DECLARE
     V_JOINDATE              BOOLEAN := SYSDATE;
   ```

2. In each of the following assignments, determine the datatype of the resulting expression.

   a. `v_email := v_firstname || to_char(v_empno);`

   b. `v_confirm := to_date('20-JAN-1999', 'DD-MON-YYYY');`

   c. `v_sal := (1000*12) +  500`

   d. `v_test := FALSE;`

   e. `v_temp := v_temp1 < (v_temp2/ 3);`

   f. `v_var := sysdate;`

3. PL/SQL Block

```
DECLARE
  v_custid   NUMBER(4) := 1600;
  v_custname VARCHAR2(300) := 'Women Sports Club';
  v_new_custid   NUMBER(3) := 500;
BEGIN
  DECLARE
        v_custid   NUMBER(4) := 0;
        v_custname  VARCHAR2(300) := 'Shape up Sports Club';
        v_new_custid  NUMBER(3) := 300;
        v_new_custname  VARCHAR2(300) := 'Jansports Club';
  BEGIN
        v_custid := v_new_custid;
        v_custname := v_custname || ' ' ||  v_new_custname;
  END;
v_custid := (v_custid *12) / 10;
END;
/
```

Evaluate the PL/SQL block above and determine the datatype and value of each of the following variables according to the rules of scoping.

a. The value of V_CUSTID in the subblock is:

b. The value of V_CUSTNAME in the subblock is:

c. The value of V_NEW_CUSTID in the main block is:

d. The value of V_NEW_CUSTNAME in the subblock is:

e. The value of V_CUSTID in the main block is:

f. The value of V_CUSTNAME in the main block is:

These exercises can be used for extra practice when discussing the following topics: declaring variables, writing executable statements, interacting with the Oracle Server and writing control structures.

4. Write a PL/SQL block to accept a year and check if it is a leap year. For example, if the year entered is 1990, the output should be "1990 is not a leap year".

   Hint: The year should be exactly divisible by 4, but not exactly divisible by 100 or should by exactly divisible by 400. Test your solution with the following years:

1990	not a leap year
2000	leap year
1996	leap year
1886	not a leap year
1992	leap year
1824	leap year

```
Please enter the year : 1990
old   2:         V_YEAR NUMBER(4) := &P_YEAR;
new   2:         V_YEAR NUMBER(4) := 1990;
1990 is not a leap year
```

These exercises can be used for extra practice when discussing the following topics: declaring variables, writing executable statements, interacting with the Oracle Server and writing control structures.

5. a. For the exercises below, you will require a temporary table to store the results. Create a table TEMP with the following three columns.

Column Name	NUM_STORE	CHAR_STORE	DATE_STORE
Key Type			
Nulls/Unique			
FK Table			
FK Column			
Datatype	Number	VARCHAR2	Date
Length	7,2	35	

Note: You can either create the table yourself or run the `labA_5.sql` script that will create the table for you.

b. Write a PL/SQL block that contains two variables, MESSAGE and DATE_WRITTEN. Declare MESSAGE as VARCHAR2 datatype with a length of 35 and DATE_WRITTEN of DATE datatype. Assign the following values to the variables:

Variable	**Contents**
MESSAGE	'This is my first PL/SQL program'
DATE_WRITTEN	Today's date

Store the values in appropriate columns of the TEMP table. Verify your results by querying the TEMP table.

6. Write a PL/SQL block to accept a department number from the user and print the number of people working in that department.

   Hint: enable DBMS_OUTPUT in SQL*Plus with SET SERVEROUTPUT ON

```
Enter value for p_deptno: 10
old   3:          V_DEPTNO DEPT.deptno%TYPE  := &P_DEPTNO;
new   3:          V_DEPTNO DEPT.deptno%TYPE  := 10;
3 employee(s) work for department number 10
```

7. Write a PL/SQL block to declare a variable, V_Salary, to store the salary of an employee. In the executable part of the program do the following:
   - Accept an employee name and store his or her salary in the variable, V_Salary.
   - If the salary is less than $3,000, then, give the employee a raise of $500 and print on the screen '<Employee Name>'s salary updated'.
   - If the salary is more than $3,000, then, print the employee's salary in the format, '<Employee Name> earns ……'

   Note: UNDEFINE the variable that stores the employee's name at the end of the script.

8. Write a PL/SQL block to accept the salary of an employee. In the executable part of the program, do the following:
   - Calculate the annual salary as salary * 12.
   - Calculate bonus as indicated in the table below

Annual Salary	Bonus
>= 20,000	2,000
19,999 - 10,000	1,000
<= 9,999	500

   - Print the bonus on the screen is the following format:

     The bonus is $……..

These exercises can be used for extra practice when discussing the following topics: declaring variables, writing executable statements, interacting with the Oracle Server, writing control structures, working with composite datatypes, cursors and handling exceptions.

9. Write a PL/SQL block to accept an employee number, the new department number and the percentage increase in the salary. Update the deptno of the employee with the new department number and the salary with the new salary. Once the update is complete, print on the screen, 'Updation complete'. If no matching records are found, handle the exception accordingly.

10. Create a PL/SQL block to declare a cursor EMP_CUR to select the employee name, salary and hire date from the employee table. Process each row from the cursor and if the salary is greater that $1,500 and the hire date is greater than 22-FEB-1981, print the employee name, salary and hire date on the screen in the format shown in the sample output below.

    JONES earns 2975 and joined the organization on 02-APR-1981

    BLAKE earns 2850 and joined the organization on 01-MAY-1981

    CLARK earns 2450 and joined the organization on 09-JUN-1981

    SCOTT earns 3000 and joined the organization on 19-APR-1987

    KING earns 5000 and joined the organization on 17-NOV-1981

    FORD earns 3000 and joined the organization on 03-DEC-1981

11. Create a PL/SQL block to retrieve the name and salary of each employee from the emp table. From the values retrieved from the emp table, populate two PL/SQL tables, one to store the records of the employee names and the other to store the records of their salaries. Using a loop, retrieve the employee name information and the salary information from the PL/SQL tables and print it to the screen, using DBMS_OUTPUT.PUT_LINE.

    Employee Name: SMITH Salary: 800

    Employee Name: ALLEN Salary: 1600

    Employee Name: WARD Salary: 1250

    Employee Name: JONES Salary: 32725

    Employee Name: MARTIN Salary: 1250

    Employee Name: BLAKE Salary: 2850

    Employee Name: CLARK Salary: 2450

    Employee Name: SCOTT Salary: 3000

    Employee Name: KING Salary: 5500

    Employee Name: TURNER Salary: 1500

    Employee Name: ADAMS Salary: 1210

    Employee Name: JAMES Salary: 950

    Employee Name: FORD Salary: 3000

    Employee Name: MILLER Salary: 1430

12. Create a PL/SQL block that declares a cursor called DATE_CUR. Pass a parameter of date datatype to the cursor and print the details of all employees who have joined after that date.

```
Please enter the hire date: 08-SEP-81
    7839 KING 17-NOV-81
    7654 MARTIN 28-SEP-81
    7900 JAMES 03-DEC-81
    7902 FORD 03-DEC-81
    7788 SCOTT 09-DEC-82
    7876 ADAMS 12-JAN-83
    7934 MILLER 23-JAN-82
```

13. Create a PL/SQL block to promote clerks who earn more than $1000 to 'SR. CLERK' and increase their salary by 10%. Verify the results by querying on the EMP table.

    Hint : Use a Cursor with FOR UPDATE and CURRENT OF syntax.

14. a. For the exercise below, you will require a table to store the results. Create a table ANALYSIS with the following three columns.

Column Name	ENAME	YEARS	SAL
Key Type			
Nulls/Unique			
FK Table			
FK Column			
Datatype	VARCHAR2	Number	Number
Length	10	2	7,2

    Note: You can create the table yourself or run the `labA_14a.sql` script that will create the table for you.

    b. Create a PL/SQL block to populate the ANALYSIS table with the information from the EMP table. Accept an employee number. Query the emp table to find if the number of years that the employee has been with the organization is greater than five and if his or her salary is less than $1500; handle the exception with an appropriate exception handler that inserts the values employee name, number of years of service, and the current salary into the ANALYSIS table. Otherwise print 'Not due for a raise' on the screen. Verify the results by querying the ANALYSIS table.

Additional Practice Solutions

1. Evaluate each of the following declarations. Determine which of them are *not* legal and explain why.

   a.
   ```
   DECLARE
     v_name,v_dept                    VARCHAR2(14);
   ```
   Illegal because only one identifier per declaration is allowed

   b.
   ```
   DECLARE
     v_test                           NUMBER(5);
   ```
   Legal

   c.
   ```
   DECLARE
     V_MAXSALARY                      NUMBER(7,2) = 5000;
   ```
   Illegal because assignment operator is wrong. It should be :=

   d.
   ```
   DECLARE
     V_JOINDATE            BOOLEAN := SYSDATE;
   ```
   Illegal because there is a mismatch in the datatypes. Boolean datatype cannot be assigned a date value. Datatype should be date.

2. In each of the following assignments, determine the datatype of the resulting expression.

   a. `v_email := v_firstname || to_char(v_empno);`

   Character string

   b. `v_confirm := to_date('20-JAN-1999', 'DD-MON-YYYY');`

   Date

   c. `v_sal := (1000*12) + 500`

   Number

   d. `v_test := FALSE;`

   Boolean

   e. `v_temp := v_temp1 < (v_temp2/ 3);`

   Boolean

   f. `v_var := sysdate;`

   Date

3. PL/SQL Block

```
DECLARE
   v_custid     NUMBER(4) := 1600;
   v_custname   VARCHAR2(300) := 'Women Sports Club';
   v_new_custid NUMBER(3) := 500;
BEGIN
   DECLARE
     v_custid     NUMBER(4) := 0;
     v_custname   VARCHAR2(300) := 'Shape up Sports Club';
     v_new_custid  NUMBER(3) := 300;
     v_new_custname VARCHAR2(300) := 'Jansports Club';
   BEGIN
     v_custid := v_new_custid;
     v_custname := v_custname || ' ' ||  v_new_custname;
   END;
v_custid := (v_custid *12) / 10;
END;
/
```

Evaluate the PL/SQL block above above and determine the datatype and value of each of the following variables according to the rules of scoping.

a. The value of V_CUSTID in the subblock is:

   300, and the datatype is NUMBER

b. The value of V_CUSTNAME in the subblock is:

   Shape up Sports Club Jansports Club, and the datatype is VARCHAR2

c. The value of V_NEW_CUSTID in the main block is:

   500, and the datatype is NUMBER

d. The value of V_NEW_CUSTNAME in the subblock is:

   Jansports Club, and the datatype is VARCHAR2

e. The value of V_CUSTID in the main block is:

   1920, and the datatype is NUMBER

f. The value of V_CUSTNAME in the main block is:

   Women Sports Club, and the datatype is VARCHAR2

4. Write a PL/SQL block to accept a year and check if it is a leap year. For example, if the year entered is 1990, the output should be "1990 is not a leap year".

   Hint: The year should be exactly divisible by 4, but not exactly divisible by 100 or should by exactly divisible by 400. Test your solution with the following years:

1990	not a leap year
2000	leap year
1996	leap year
1886	not a leap year
1992	leap year
1824	leap year

```
SQL> SET SERVEROUTPUT ON
  2  ACCEPT P_YEAR PROMPT 'Please enter the year : '
  3  DECLARE
  4   V_YEAR NUMBER(4) := &P_YEAR;
  5   V_REMAINDER1 NUMBER(5,2);
  6   V_REMAINDER2 NUMBER(5,2);
  7   V_REMAINDER3 NUMBER(5,2);
  8 BEGIN
  9   V_REMAINDER1 := MOD(V_YEAR,4);
 10   V_REMAINDER2 := MOD(V_YEAR,100);
 11   V_REMAINDER3 := MOD(V_YEAR,400);
 12   IF ((V_REMAINDER1 = 0 AND V_REMAINDER2 <> 0 ) OR
 13       V_REMAINDER3 = 0) THEN
 14      DBMS_OUTPUT.PUT_LINE(V_YEAR || ' is a leap year');
 15 ELSE
 16      DBMS_OUTPUT.PUT_LINE (V_YEAR || ' is not a leap year');
 17 END IF;
 18 END;
 19 /
 20 SET SERVEROUTPUT OFF
```

5. a. For the exercises below, you will require a temporary table to store the results. Create a table TEMP with the following three columns.

Column Name	NUM_STORE	CHAR_STORE	DATE_STORE
Key Type			
Nulls/Unique			
FK Table			
FK Column			
Datatype	Number	VARCHAR2	Date
Length	7,2	35	

```
SQL> CREATE TABLE temp
  2 (num_store number(7,2),
  3 char_store varchar2(35),
  4 date_store date);
```

b. Write a PL/SQL block that contains two variables, MESSAGE and DATE_WRITTEN. Declare MESSAGE as VARCHAR2 datatype with a length of 35 and DATE_WRITTEN of DATE datatype. Assign the following values to the variables:

Variable	Contents
MESSAGE	'This is my first PL/SQL program'
DATE_WRITTEN	Today's date

Store the values in appropriate columns of the TEMP table. Verify your results by querying the TEMP table.

```
SQL> DECLARE
  2    MESSAGE VARCHAR2(35);
  3      DATE_WRITTEN DATE;
  4  BEGIN
  5    MESSAGE := 'This is my first PLSQL Program';
  6    DATE_WRITTEN := SYSDATE;
  7    INSERT INTO temp(CHAR_STORE,DATE_STORE)
  8    VALUES (MESSAGE,DATE_WRITTEN);
  9    END;
 10   /
SQL> SELECT * FROM TEMP;
```

6. Write a PL/SQL block to accept a department number from the user and print the number of people working in that department.

   Hint: enable DBMS_OUTPUT in SQL*Plus with SET SERVEROUTPUT ON

```
SQL> SET SERVEROUTPUT ON
  2  ACCEPT P_DEPTNO PROMPT 'Please enter department number: ';
  3  DECLARE
  4  V_COUNT NUMBER(3);
  5  V_DEPTNO DEPT.deptno%TYPE  := &P_DEPTNO;
  6  BEGIN
  7  SELECT COUNT(*) INTO V_COUNT  FROM emp
  8  WHERE deptno = V_DEPTNO;
  9  DBMS_OUTPUT.PUT_LINE (V_COUNT || ' employee(s) work for
 10  department number ' ||V_DEPTNO);
 11  END;
 12  /
 13  SET SERVEROUTPUT OFF
```

7. Write a PL/SQL block to declare a variable, V_Salary, to store the salary of an employee. In the executable part of the program do the following:
   - Accept an employee name and store his or her salary in the variable, V_Salary.
   - If the salary is less than $3,000, then, give the employee a raise of $500 and print on the screen '<Employee Name>'s salary updated.'
   - If the salary is more than $3,000, then, print the employee's salary in the format, '<Employee Name> earns .....'

   Note: UNDEFINE the variable that stores the employee's name at the end of the script.

```
SQL> SET SERVEROUTPUT ON
  2  DECLARE
  3    V_SALARY NUMBER(7,2);
  4    V_LASTNAME EMP.ENAME%TYPE;
  5  BEGIN
  6     SELECT sal INTO V_SALARY
  7     FROM emp WHERE ename = UPPER('&&P_LASTNAME') FOR
  8     UPDATE of  sal;
  9     V_LASTNAME := UPPER('&&P_LASTNAME') ;
```

Practice 7 Solution (continued)

```
10 IF V_SALARY < 3000 THEN
11    UPDATE emp SET sal = sal + 500
12    WHERE ename =  UPPER('&&P_LASTNAME') ;
13    DBMS_OUTPUT.PUT_LINE (V_LASTNAME || '''s salary
14     updated');
15  ELSE
16    DBMS_OUTPUT.PUT_LINE (V_LASTNAME  || ' earns ' ||
17     TO_CHAR(V_SALARY));
18  END IF;
19  END;
20  /
21 SET SERVEROUTPUT OFF
22 UNDEFINE P_LASTNAME
```

8. Write a PL/SQL block to accept the salary of an employee. In the executable part of the program do the following:
   - Calculate the annual salary as salary * 12.
   - Calculate bonus as indicated in the table below

Annual Salary	Bonus
>= 20,000	2,000
19,999 - 10,000	1,000
<= 9,999	500

   - Print the bonus on the screen is the following format:

     The bonus is $.....

```
SQL> SET SERVEROUTPUT ON
  2  ACCEPT P_SALARY PROMPT  'Please enter the salary: ';
  3  DECLARE
  4    V_SALARY  NUMBER(7,2) := &P_SALARY;
  5    V_BONUS   NUMBER(7,2);
  6    V_ANN_SALARY NUMBER(15,2);
  7  BEGIN
```

```
 8       V_ANN_SALARY := V_SALARY * 12;
 9       IF V_ANN_SALARY >= 20000 THEN
10          V_BONUS := 2000;
11       ELSIF V_ANN_SALARY <= 19999 AND V_ANN_SALARY >=10000
12        THEN
13          V_BONUS := 1000;
14       ELSE
15          V_BONUS := 500;
16       END IF;
17  DBMS_OUTPUT.PUT_LINE ('The Bonus is $ ' || TO_CHAR(V_BONUS));
18  END;
19 /
20 SET SERVEROUTPUT OFF
```

9. Write a PL/SQL block to accept an employee number, the new department number and the percentage increase in the salary. Update the deptno of the employee with the new department number and the salary with the new salary. Once the update is complete, print on the screen, 'Updation complete'. If no matching records are found, handle the exception accordingly.

```
SQL> SET SERVEROUTPUT ON
 2  ACCEPT P_EMPNO PROMPT  'Please enter the employee number: ';
 3  ACCEPT P_NEW_DEPTNO PROMPT  'Please enter the new department
 4                                                    number: ';
 5  ACCEPT P_PER_INCREASE PROMPT  'Please enter the percentage
 6                                          raise of salary: ';
 7  DECLARE
 8      V_EMPNO EMP.EMPNO%TYPE := &P_EMPNO;
 9      V_NEW_DEPTNO EMP.DEPTNO%TYPE := &P_NEW_DEPTNO;
10      V_PER_INCREASE NUMBER(7,2) := &P_PER_INCREASE;
11  BEGIN
12      UPDATE emp
13      SET deptno = V_NEW_DEPTNO,
14       sal        = sal + (sal * V_PER_INCREASE/100)
15      WHERE EMPNO = V_EMPNO;
16      DBMS_OUTPUT.PUT_LINE ('Updation Complete');
17  EXCEPTION
18      WHEN NO_DATA_FOUND THEN
19      INSERT INTO MESSAGES VALUES ('No Data Found');
20  END;
21  /
```

10. Create a PL/SQL block to declare a cursor EMP_CUR to select the employee name, salary and hire date from the employee table. Process each row from the cursor, and if the salary is greater that $1,500 and the hiredate is greater than 22-FEB-1981, print the employee name, salary and hire date on the screen in the format shown in the sample output below:

    JONES earns 2975 and joined the organization on 02-APR-1981

    BLAKE earns 2850 and joined the organization on 01-MAY-1981

    CLARK earns 2450 and joined the organization on 09-JUN-1981

    SCOTT earns 3000 and joined the organization on 19-APR-1987

    KING earns 5000 and joined the organization on 17-NOV-1981

    FORD earns 3000 and joined the organization on 03-DEC-1981

```
SQL> SET SERVEROUTPUT ON
  2  DECLARE
  3 CURSOR EMP_CUR IS
  4 SELECT ename,sal,hiredate FROM EMP;
  5       V_ENAME VARCHAR2(10);
  6       V_SAL         NUMBER(7,2);
  7       V_HIREDATE DATE;
  8  BEGIN
  9  OPEN EMP_CUR;
 10 FETCH EMP_CUR INTO V_ENAME,V_SAL,V_HIREDATE;
 11 WHILE EMP_CUR%FOUND
 12 LOOP
 13 IF V_SAL > 1500 AND V_HIREDATE >=
 14         TO_DATE('22-FEB-1981','DD-MON-YYYY') THEN
 15        DBMS_OUTPUT.PUT_LINE (V_ENAME ||  ' earns ' ||
 16   TO_CHAR(V_SAL)||  ' and joined the organization on '
 17         || TO_DATE(V_HIREDATE,'DD-Mon-YYYY');
 18 END IF;
 19               FETCH EMP_CUR INTO V_ENAME,V_SAL,V_HIREDATE;
 20 END LOOP;
 21 END;
 22 /
 23 SET SERVEROUTPUT OFF
```

11. Create a PL/SQL block to retrieve the name and salary of each employee from the emp table. From the values retrieved from the emp table, populate two PL/SQL tables, one to store the records of the employee names and the other to store the records of their salaries. Using a loop, retrieve the employee name information and the salary information from the PL/SQL tables and print it to the screen, using DBMS_OUTPUT.PUT_LINE.

Employee Name: SMITH Salary: 800

Employee Name: ALLEN Salary: 1600

Employee Name: WARD Salary: 1250

Employee Name: JONES Salary: 32725

Employee Name: MARTIN Salary: 1250

Employee Name: BLAKE Salary: 2850

Employee Name: CLARK Salary: 2450

Employee Name: SCOTT Salary: 3000

Employee Name: KING Salary: 5500

Employee Name: TURNER Salary: 1500

Employee Name: ADAMS Salary: 1210

Employee Name: JAMES Salary: 950

Employee Name: FORD Salary: 3000

Employee Name: MILLER Salary: 1430

```
SQL> SET SERVEROUTPUT ON
  2  DECLARE
  3     TYPE Table_Ename is table of emp.ename%TYPE
  4     INDEX  BY BINARY_INTEGER;
  5     TYPE Table_Sal  is table of emp.sal%TYPE
  6     INDEX  BY BINARY_INTEGER;
  7     V_Tename    Table_Ename;
  8     V_Tsal     Table_sal;
  9     i  BINARY_INTEGER :=0;
 10     CURSOR C_Namesal IS SELECT ename,sal from emp;
 11     V_COUNT  NUMBER (2);
 12  BEGIN
 13     SELECT COUNT(*)    INTO  v_count
 14                 FROM  emp;
 15     FOR emprec in C_Namesal
 16     LOOP
 17                i := i +1;
 18                          V_Tename(i) := emprec.ename;
 19                          V_Tsal(i) := emprec.sal;
```

Practice 11 Solution (continued)

```
20 END LOOP;
21 FOR i IN 1..v_count
22 LOOP
23       DBMS_OUTPUT.PUT_LINE ('Employee Name: '
24       ||V_Tename(i)|| 'Salary:' || V_Tsal(i));
25 END LOOP;
26 END;
27 /
28 SET SERVEROUTPUT OFF
```

12. Create a PL/SQL block that declares a cursor called DATE_CUR. Pass a parameter of date datatype to the cursor and print the details of all employees who have joined after that date.

```
SQL> SET SERVEROUTPUT ON
 2  ACCEPT P_HIREDATE PROMPT  'Please enter the hire date: ';
 3  DECLARE
 4      CURSOR DATE_CURSOR(JOIN_DATE DATE) IS
 5      SELECT empno,ename,hiredate FROM emp
 6      WHERE HIREDATE > JOIN_DATE ;
 7      V_EMPNO    emp.empno%TYPE;
 8      V_ENAME    emp.ename%TYPE;
 9      V_HIREDATE emp.hiredate%TYPE;
10      V_DATE emp.hiredate%TYPE :=    '&P_HIREDATE';
11  BEGIN
12      OPEN DATE_CURSOR(V_DATE);
13      LOOP
14          FETCH DATE_CURSOR INTO V_EMPNO,V_ENAME,V_HIREDATE;
15          EXIT WHEN DATE_CURSOR%NOTFOUND;
16          DBMS_OUTPUT.PUT_LINE (V_EMPNO || ' ' || V_ENAME || '
17          ' ||  V_HIREDATE);
18      END LOOP;
19  END;
20  /
21  SET SERVEROUTPUT OFF;
```

13. Create a PL/SQL block to promote clerks who earn more than $,1000 to 'SR. CLERK' and increase their salary by 10%. Verify the results by querying on the EMP table.
    Hint: Use a Cursor with FOR UPDATE and CURRENT OF syntax.

```
SQL> DECLARE
  2       CURSOR C_Senior_Clerk IS
  3       SELECT empno,job FROM emp
  4       WHERE job = 'CLERK' AND sal > 1000
  5  FOR UPDATE OF job;
  6   BEGIN
  7  FOR V_Emrec IN C_Senior_Clerk
  8  LOOP
  9       UPDATE emp
 10       SET job = 'SR. CLERK',
 11       sal = 1.1 * sal
 12       WHERE CURRENT OF C_Senior_Clerk;
 13  END LOOP;
 14  COMMIT;
 15   END;
 16   /
SQL> SELECT * FROM emp;
```

14. a. For the exercise below, you will require a table to store the results. Create a table ANALYSIS with the following three columns:

Column Name	ENAME	YEARS	SAL
Key Type			
Nulls/Unique			
FK Table			
FK Column			
Datatype	VARCHAR2	Number	Number
Length	10	2	7,2

```
SQL> CREATE TABLE analysis
  2 (ename Varchar2(10),
  3 years Number(2),
  4 sal Number(7,2);
```

b. Create a PL/SQL block to populate the ANALYSIS table with the information from the EMP table. Accept an employee number. Query the emp table to find if the number of years that the employee has been with the organization is greater than five and if his or her salary is less than $1,500; handle the exception with an appropriate exception handler that inserts the values employee name, number of years of service, and the current salary into the ANALYSIS table. Otherwise print 'Not due for a raise' on the screen. Verify the results by querying the ANALYSIS table.

```
SQL> SET SERVEROUTPUT ON
  2  ACCEPT P_ENAME PROMPT  'Please enter the employee name: ';
  3  DECLARE
  4  DUE_FOR_RAISE EXCEPTION;
  5       V_HIREDATE EMP.HIREDATE%TYPE;
  6       V_ENAME EMP.ENAME%TYPE :=    '&P_ENAME';
  7  V_SAL EMP.SAL%TYPE;
  8  V_YEARS NUMBER(2);
  9  BEGIN
 10  SELECT ENAME,SAL,HIREDATE
 11  INTO  V_ENAME,V_SAL,V_HIREDATE
 12  FROM emp WHERE ename =  V_ENAME;
 13  V_YEARS := ROUND((SYSDATE-V_HIREDATE)/365);
```

Practice 14 Solution (continued)

```
14      IF V_SAL < 1500 AND V_YEARS > 5  THEN
15                RAISE DUE_FOR_RAISE;
16      ELSE
17           DBMS_OUTPUT.PUT_LINE ('Not due for a raise');
18      END IF;
19  EXCEPTION
20      WHEN DUE_FOR_RAISE THEN
21                INSERT INTO ANALYSIS(ENAME,YEARS,SAL)
22                VALUES (V_ENAME,V_YEARS,V_SAL);
23  END;
24  /
```

Table Descriptions and Data

EMP Table

```
SQL> DESCRIBE emp
```

```
Name                                      Null?    Type
----------------------------------------- -------- ----
EMPNO                                     NOT NULL NUMBER(4)
ENAME                                              VARCHAR2(10)
JOB                                                VARCHAR2(9)
MGR                                                NUMBER(4)
HIREDATE                                           DATE
SAL                                                NUMBER(7,2)
COMM                                               NUMBER(7,2)
DEPTNO                                    NOT NULL NUMBER(2)
```

```
SQL> SELECT * FROM emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		17-NOV-81	5000		10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

DEPT Table

```
SQL> DESCRIBE dept

Name                           Null?    Type
------------------------------ -------- ----
DEPTNO                         NOT NULL NUMBER(2)
DNAME                                   VARCHAR2(14)
LOC                                     VARCHAR2(13)
```

```
SQL> SELECT * FROM dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

SALGRADE Table

```
SQL> DESCRIBE salgrade
```

```
Name                           Null?    Type
------------------------------ -------- ----
GRADE                                   NUMBER
LOSAL                                   NUMBER
HISAL                                   NUMBER
```

```
SQL> SELECT * FROM salgrade;
```

```
    GRADE      LOSAL      HISAL
--------- ---------- ----------
        1        700       1200
        2       1201       1400
        3       1401       2000
        4       2001       3000
        5       3001       9999
```

ORD Table

```
SQL> DESCRIBE ord
```

```
Name                            Null?    Type
------------------------------- -------- ----
ORDID                           NOT NULL NUMBER(4)
ORDERDATE                                DATE
COMMPLAN                                 VARCHAR2(1)
CUSTID                          NOT NULL NUMBER(6)
SHIPDATE                                 DATE
TOTAL                                    NUMBER(8,2)
```

```
SQL> SELECT * FROM ord;
```

ORDID	ORDERDATE	C	CUSTID	SHIPDATE	TOTAL
610	07-JAN-87	A	101	08-JAN-87	101.4
611	11-JAN-87	B	102	11-JAN-87	45
612	15-JAN-87	C	104	20-JAN-87	5860
601	01-MAY-86	A	106	30-MAY-86	2.4
602	05-JUN-86	B	102	20-JUN-86	56
604	15-JUN-86	A	106	30-JUN-86	698
605	14-JUL-86	A	106	30-JUL-86	8324
606	14-JUL-86	A	100	30-JUL-86	3.4
609	01-AUG-86	B	100	15-AUG-86	97.5
607	18-JUL-86	C	104	18-JUL-86	5.6
608	25-JUL-86	C	104	25-JUL-86	35.2
603	05-JUN-86		102	05-JUN-86	224
620	12-MAR-87		100	12-MAR-87	4450
613	01-FEB-87		108	01-FEB-87	6400
614	01-FEB-87		102	05-FEB-87	23940
616	03-FEB-87		103	10-FEB-87	764
619	22-FEB-87		104	04-FEB-87	1260
617	05-FEB-87		105	03-MAR-87	46370
615	01-FEB-87		107	06-FEB-87	710
618	15-FEB-87	A	102	06-MAR-87	3510.5
621	15-MAR-87	A	100	01-JAN-87	730

PRODUCT Table

```
SQL> DESCRIBE product

Name                           Null?    Type
------------------------------ -------- ----
PRODID                         NOT NULL NUMBER(6)
DESCRIP                                 VARCHAR2(30)
```

```
SQL> SELECT * FROM product;

   PRODID DESCRIP
--------- ------------------------------
   100860 ACE TENNIS RACKET I
   100861 ACE TENNIS RACKET II
   100870 ACE TENNIS BALLS-3 PACK
   100871 ACE TENNIS BALLS-6 PACK
   100890 ACE TENNIS NET
   101860 SP TENNIS RACKET
   101863 SP JUNIOR RACKET
   102130 RH: "GUIDE TO TENNIS"
   200376 SB ENERGY BAR-6 PACK
   200380 SB VITA SNACK-6 PACK
```

ITEM Table

```
SQL> DESCRIBE item
```

```
 Name                           Null?    Type
------------------------------- -------- -----------
ORDID                           NOT NULL NUMBER(4)
ITEMID                          NOT NULL NUMBER(4)
PRODID                                   NUMBER(6)
ACTUALPRICE                              NUMBER(8,2)
QTY                                      NUMBER(8)
ITEMTOT                                  NUMBER(8,2)
```

```
SQL> SELECT * FROM item;
```

ORDID	ITEMID	PRODID	ACTUALPRICE	QTY	ITEMTOT
610	3	100890	58	1	58
611	1	100861	45	1	45
612	1	100860	30	100	3000
601	1	200376	2.4	1	2.4
602	1	100870	2.8	20	56
604	1	100890	58	3	174
604	2	100861	42	2	84
604	3	100860	44	10	440
603	2	100860	56	4	224
610	1	100860	35	1	35
610	2	100870	2.8	3	8.4
613	4	200376	2.2	200	440
614	1	100860	35	444	15540
614	2	100870	2.8	1000	2800
612	2	100861	40.5	20	810
612	3	101863	10	150	1500
620	1	100860	35	10	350
620	2	200376	2.4	1000	2400
620	3	102130	3.4	500	1700
613	1	100871	5.6	100	560
613	2	101860	24	200	4800
613	3	200380	4	150	600
619	3	102130	3.4	100	340
617	1	100860	35	50	1750
617	2	100861	45	100	4500
614	3	100871	5.6	1000	5600

Continued on next page

ITEM Table (continued)

ORDID	ITEMID	PRODID	ACTUALPRICE	QTY	ITEMTOT
616	1	100861	45	10	450
616	2	100870	2.8	50	140
616	3	100890	58	2	116
616	4	102130	3.4	10	34
616	5	200376	2.4	10	24
619	1	200380	4	100	400
619	2	200376	2.4	100	240
615	1	100861	45	4	180
607	1	100871	5.6	1	5.6
615	2	100870	2.8	100	280
617	3	100870	2.8	500	1400
617	4	100871	5.6	500	2800
617	5	100890	58	500	29000
617	6	101860	24	100	2400
617	7	101863	12.5	200	2500
617	8	102130	3.4	100	340
617	9	200376	2.4	200	480
617	10	200380	4	300	1200
609	2	100870	2.5	5	12.5
609	3	100890	50	1	50
618	1	100860	35	23	805
618	2	100861	45.11	50	2255.5
618	3	100870	45	10	450
621	1	100861	45	10	450
621	2	100870	2.8	100	280
615	3	100871	5	50	250
608	1	101860	24	1	24
608	2	100871	5.6	2	11.2
609	1	100861	35	1	35
606	1	102130	3.4	1	3.4
605	1	100861	45	100	4500
605	2	100870	2.8	500	1400
605	3	100890	58	5	290
605	4	101860	24	50	1200
605	5	101863	9	100	900
605	6	102130	3.4	10	34
612	4	100871	5.5	100	550
619	4	100871	5.6	50	280

CUSTOMER Table

```
SQL> DESCRIBE customer

Name                 Null?                Type
-----------------    -----------------    --------------
CUSTID               NOT NULL NUMBER(6)
NAME                                      VARCHAR2(45)
ADDRESS                                   VARCHAR2(40)
CITY                                      VARCHAR2(30)
STATE                                     VARCHAR2(2)
ZIP                                       VARCHAR2(9)
AREA                                      NUMBER(3)
PHONE                                     VARCHAR2(9)
REPID                NOT NULL             NUMBER(4)
CREDITLIMIT                               NUMBER(9,2)
COMMENTS                                  LONG
```

CUSTOMER Table (continued)

```
SQL> SELECT * FROM customer;
```

CUSTID	NAME	ADDRESS
100	JOCKSPORTS	345 VIEWRIDGE
101	TKB SPORT SHOP	490 BOLI RD.
102	VOLLYRITE	9722 HAMILTON
103	JUST TENNIS	HILLVIEW MALL
104	EVERY MOUNTAIN	574 SURRY RD.
105	K + T SPORTS	3476 EL PASEO
106	SHAPE UP	908 SEQUOIA
107	WOMENS SPORTS	VALCO VILLAGE
108	NORTH WOODS HEALTH AND FITNESS SUPPLY CENTER	98 LONE PINE WAY

CITY	ST	ZIP	AREA	PHONE	REPID	CREDITLIMIT
BELMONT	CA	96711	415	598-6609	7844	5000
REDWOOD CITY	CA	94061	415	368-1223	7521	10000
BURLINGAME	CA	95133	415	644-3341	7654	7000
BURLINGAME	CA	97544	415	677-9312	7521	3000
CUPERTINO	CA	93301	408	996-2323	7499	10000
SANTA CLARA	CA	91003	408	376-9966	7844	5000
PALO ALTO	CA	94301	415	364-9777	7521	6000
SUNNYVALE	CA	93301	408	967-4398	7499	10000
HIBBING	MN	55649	612	566-9123	7844	8000

COMMENTS
Very friendly people to work with -- sales rep likes to be called Mike.
Rep called 5/8 about change in order - contact shipping.
Company doing heavy promotion beginning 10/89. Prepare for large orders during orders during winter
Contact rep about new line of tennis rackets.
Customer with high market share (23%) due to aggressive advertising.
Tends to order large amounts of merchandise at once. Accounting is considering raising their credit limit
Support intensive. Orders small amounts (< 800) of merchandise at a time.
First sporting goods store geared exclusively towards women. Unusual promotional style

PRICE Table

```
SQL> DESCRIBE price

Name                                      Null?    Type
----------------------------------------- -------- ----
PRODID                                    NOT NULL NUMBER(6)
STDPRICE                                           NUMBER(8,2)
MINPRICE                                           NUMBER(8,2)
STARTDATE                                          DATE
ENDDATE                                            DATE

SQL> SELECT * FROM price;

   PRODID  STDPRICE  MINPRICE STARTDATE ENDDATE
--------- --------- --------- --------- ---------
   100871       4.8       3.2 01-JAN-85 01-DEC-85
   100890        58      46.4 01-JAN-85
   100890        54      40.5 01-JUN-84 31-MAY-84
   100860        35        28 01-JUN-86
   100860        32      25.6 01-JAN-86 31-MAY-86
   100860        30        24 01-JAN-85 31-DEC-85
   100861        45        36 01-JUN-86
   100861        42      33.6 01-JAN-86 31-MAY-86
   100861        39      31.2 01-JAN-85 31-DEC-85
   100870       2.8       2.4 01-JAN-86
   100870       2.4       1.9 01-JAN-85 01-DEC-85
   100871       5.6       4.8 01-JAN-86
   101860        24        18 15-FEB-85
   101863      12.5       9.4 15-FEB-85
   102130       3.4       2.8 18-AUG-85
   200376       2.4      1.75 15-NOV-86
   200380         4       3.2 15-NOV-86
```