ITSE 1345 PL/SQL Lab 3, Competency 3

Elizabeth Arlene Waghalter

663b6c58-34dd-42cc-9f34-24ddc9b988d9

1. Create a procedure to find the data for a given employee using the Empno.

Pre: Empno should exist, if not use an exception handler with message.

Post: Print every column per employee row leaving employee table unchanged.

Use %ROWTYPE to define the structure that will store column values.

Write a driver to call and test procedure.

Test one valid employee (Empno = 7654) and one invalid employee (Empno = 7888).

CREATE OR REPLACE PROCEDURE find\_emp

is

CURSOR emp\_cur IS SELECT EMPNO,ENAME,JOB,MGR ,HIREDATE ,SAL ,COMM ,DEPTNO FROM EMP;

emp\_var emp\_cur%ROWTYPE;

BEGIN

OPEN emp\_cur;

LOOP

FETCH emp\_cur INTO emp\_var;

EXIT WHEN emp\_cur%NOTFOUND;

dbms\_output.put\_line(emp\_var.EMPNO|| emp\_var.ename || emp\_var.job || emp\_var.mgr || emp\_var.hiredate || emp\_var.sal ||emp\_var.comm || emp\_var.deptno);

END LOOP;

CLOSE emp\_cur;

END find\_emp;

/

Procedure FIND\_EMP compiled

call find\_emp ();

Call completed.

7839KINGPRESIDENT17-NOV-95500010

7566JONESMANAGER783902-APR-95297520

7698BLAKEMANAGER783901-MAY-95285030

7782CLARKMANAGER783909-JUN-95245010

7902FORDANALYST756603-DEC-95300020

7369SMITHCLERK790217-DEC-9880020

7499ALLENSALESMAN769820-FEB-99160030030

7521WARDSALESMAN769822-FEB-98125050030

7654MARTINSALESMAN769828-SEP-991250140030

7788SCOTTANALYST756609-DEC-96300020

7844TURNERSALESMAN769808-SEP-951500030

7876ADAMSCLERK778812-JAN-00110020

7900JAMESCLERK769803-DEC-9595030

7934MILLERCLERK778223-JAN-96130010

SELECT \* FROM EMP;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

---------- ---------- ---------- ---------- --------- ---------- ---------- ----------

7839 KING PRESIDENT 17-NOV-95 5000 10

7566 JONES MANAGER 7839 02-APR-95 2975 20

7698 BLAKE MANAGER 7839 01-MAY-95 2850 30

7782 CLARK MANAGER 7839 09-JUN-95 2450 10

7902 FORD ANALYST 7566 03-DEC-95 3000 20

7369 SMITH CLERK 7902 17-DEC-98 800 20

7499 ALLEN SALESMAN 7698 20-FEB-99 1600 300 30

7521 WARD SALESMAN 7698 22-FEB-98 1250 500 30

7654 MARTIN SALESMAN 7698 28-SEP-99 1250 1400 30

7788 SCOTT ANALYST 7566 09-DEC-96 3000 20

7844 TURNER SALESMAN 7698 08-SEP-95 1500 0 30

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

---------- ---------- ---------- ---------- --------- ---------- ---------- ----------

7876 ADAMS CLERK 7788 12-JAN-00 1100 20

7900 JAMES CLERK 7698 03-DEC-95 950 30

7934 MILLER CLERK 7782 23-JAN-96 1300 10

14 rows selected

desc emp;

Name Null Type

-------- -------- ------------

EMPNO NOT NULL NUMBER(4)

ENAME NOT NULL VARCHAR2(10)

JOB VARCHAR2(10)

MGR NUMBER(4)

HIREDATE DATE

SAL NUMBER(6)

COMM NUMBER(6)

DEPTNO NUMBER(3)

2. Write a function which inputs two dates and compares the two dates. The function returns -1, 0, or +1 depending on whether the 1st date is less than, equal to, or greater than the 2nd date. (The time component is not to be considered).

CREATE OR REPLACE FUNCTION comp\_dat

(dat\_1 DATE,

dat\_2 DATE)

RETURN NUMBER

IS

lv\_dat\_1 DATE;

lv\_dat\_2 DATE;

BEGIN

lv\_dat\_1 := TO\_DATE('&lv\_dat\_1', 'DD-MON-YYYY');

lv\_dat\_2 := TO\_DATE('&lv\_dat\_2', 'DD-MON-YYYY');

IF lv\_dat\_1 < lv\_dat\_2 THEN

dbms\_output.put\_line(-1);

ELSIF lv\_dat\_1 = lv\_dat\_2 THEN

dbms\_output.put\_line(0);

ELSIF lv\_dat\_1 > lv\_dat\_2 THEN

dbms\_output.put\_line(1);

ELSE

dbms\_output.put\_line('INVALID ENTRY.');

END IF;

END;

/

Function COMP\_DAT compiled

CREATE OR REPLACE FUNCTION comp\_dat

(dat\_1 DATE,

dat\_2 DATE)

RETURN NUMBER

IS

ret\_val NUMBER;

lv\_dat\_1 DATE;

lv\_dat\_2 DATE;

BEGIN

lv\_dat\_1 := TO\_DATE('&lv\_dat\_1', 'DD-MON-YYYY');

lv\_dat\_2 := TO\_DATE('&lv\_dat\_2', 'DD-MON-YYYY');

IF lv\_dat\_1 < lv\_dat\_2 THEN

ret\_val := -1;

ELSIF lv\_dat\_1 = lv\_dat\_2 THEN

ret\_val := 0;

ELSIF lv\_dat\_1 > lv\_dat\_2 THEN

ret\_val := 1;

ELSE

dbms\_output.put\_line('INVALID ENTRY.');

END IF;

dbms\_output.put\_line(ret\_val);

END;

/

Function COMP\_DAT compiled

3. Write a procedure which inputs an account ID and returns the customer information for that account.

Account Id IN IS VALID

Row of customer table with account ID OUT

IF FOUND, row returned

IF NOT FOUND%, return ‘Cust\_ID not valid’

call print field of returned row

CREATE OR REPLACE PROCEDURE cust\_info

(acc\_id IN NUMBER)

IS

CURSOR cus\_cur IS

SELECT \* FROM CUSTOMER

WHERE CUSTOMER\_NUM = acc\_id;

ret\_val CUS\_CUR%ROWTYPE;

BEGIN

OPEN cus\_cur;

LOOP

FETCH cus\_cur INTO ret\_val;

EXIT WHEN cus\_cur%NOTFOUND;

IF ret\_val.customer\_num = acc\_id THEN

dbms\_output.put\_line(ret\_val.customer\_num);

dbms\_output.put\_line(ret\_val.fname || ret\_val.lname);

dbms\_output.put\_line(ret\_val.company);

dbms\_output.put\_line(ret\_val.address1 || ret\_val.address2);

dbms\_output.put\_line(ret\_val.city || ret\_val.state || ret\_val.zipcode);

dbms\_output.put\_line(ret\_val.phone);

ELSE

dbms\_output.put\_line('Customer ID not valid');

END IF;

END LOOP;

CLOSE cus\_cur;

END;

/

Procedure CUST\_INFO compiled

call cust\_info(101);

Call completed.

101

LudwigPauli

All Sports Supplies

213 Erstwild Court

SunnyvaleCA94086

408-789-8075

DESCRIBE CUSTOMER;

Name Null Type

------------ -------- ------------

CUSTOMER\_NUM NOT NULL NUMBER(3)

FNAME VARCHAR2(15)

LNAME VARCHAR2(15)

COMPANY VARCHAR2(20)

ADDRESS1 VARCHAR2(20)

ADDRESS2 VARCHAR2(20)

CITY VARCHAR2(15)

STATE CHAR(2)

ZIPCODE CHAR(5)

PHONE VARCHAR2(18)

SELECT \* FROM CUSTOMER;

CUSTOMER\_NUM FNAME LNAME COMPANY ADDRESS1 ADDRESS2 CITY ST ZIPCO PHONE

------------ --------------- --------------- -------------------- -------------------- -------------------- --------------- -- ----- ------------------

101 Ludwig Pauli All Sports Supplies 213 Erstwild Court Sunnyvale CA 94086 408-789-8075

102 Carole Sadler Sports Spots 785 Geary St San Francisco CA 94117 415-822-1289

103 Phillip Currie Phil's Sports 654 Poplar P.O. Box 3498 Palo Alto CA 94303 415-328-4543

104 Anthony Higgins Play Ball! East Shopping Cntr. 422 Bay Road Redwood City CA 94026 415-368-1100

105 Raymond Vector Los Altos Sports 1899 Le Loma Drive Los Altos CA 94022 415-776-3249

106 George Watson Watson and Son 1143 Carver Place Mountain View CA 94063 415-389-8789

107 Charles Ream Athletic Supplies 41 Jordan Avenue Palo Alto CA 94304 415-356-9876

108 Donald Quinn Quinn's Sports 587 Alvarado Redwood City CA 94063 415-544-8729

109 Jane Miller Sport Stuff Mayfair Mart 7345 Ross Blvd. Sunnyvale CA 94086 408-723-8789

110 Roy Jaeger AA athletics 520 Topaz Way Redwood City CA 94062 415-743-3611

111 Frances Keyes Sports Center 3199 Sterling Court Sunnyvale CA 94085 408-277-7245

CUSTOMER\_NUM FNAME LNAME COMPANY ADDRESS1 ADDRESS2 CITY ST ZIPCO PHONE

------------ --------------- --------------- -------------------- -------------------- -------------------- --------------- -- ----- ------------------

112 Margaret Lawson Runners and Others 234 Wyandotte Way Los Altos Hills CA 94022 415-887-7235

113 Lana Beatty Sportstown 654 Oak Grove Menlo Park CA 94025 415-356-9982

114 Frank Albertson Sporting Place 947 Waverly Place Redwood City CA 94062 415-886-6677

115 Alfred Grant Gold Medal Sports 776 Gary Avenue Menlo Park CA 94025 415-356-1123

116 Jean Parmelee Olympic City 1104 Spinosa Drive Mountain View CA 94040 415-534-8822

117 Arnold Sipes Kids Korner 850 Lytton Court Redwood City CA 94063 415-245-4578

118 Dick Baxter Blue Ribbon Sports 5427 College Oakland CA 94609 415-655-0011

18 rows selected

4. Create a table named Physician with columns: Phys\_ID, Phys\_name, Phys\_phone, and Phys\_specialty. Write an anonymous block that updates the Physician table if the value for Phys\_ID is in the table. Update the values of the remaining columns. If the Phys\_ID is not in the Physician table, insert a row into the Physician table with values. (Use variables for each of the columns and assign a named parameter to each; use the variable in the block.)

Write driver to test the block with a Phys\_ID in the table and one that is not in the table.

CREATE TABLE Physician

(Phys\_ID NUMBER,

Phys\_name VARCHAR2(50),

Phys\_phone VARCHAR2(15),

Phys\_specialty VARCHAR2(25));

Table PHYSICIAN created.

DECLARE

CURSOR phys\_cur IS

SELECT \* FROM PHYSICIAN;

col\_id Physician.Phys\_ID%TYPE;

col\_name PHYSICIAN.PHYS\_NAME%TYPE;

col\_phone PHYSICIAN.PHYS\_PHONE%TYPE;

col\_spec PHYSICIAN.PHYS\_SPECIALTY%TYPE;

BEGIN

FOR phys IN phys\_cur LOOP

col\_id := PHYS.PHYS\_ID;

IF PHYS.PHYS\_ID = col\_id then

UPDATE PHYSICIAN

SET PHYS\_NAME = 'Roger Kiser',

phys\_phone = '903-555-5552',

phys\_specIALTY = 'Geriatrics';

ELSE

INSERT INTO PHYSICIAN (PHYS\_ID, PHYS\_NAME, pHYS\_PHONE, PHYS\_SPECIALTY)

VALUES(101, 'Roger Kiser', '903-555-5552', 'Geriatrics');

END IF;

END LOOP;

END;

/

PL/SQL procedure successfully completed.

5. Create a function that adjusts a string to a specific length.

Delete leading spaces. If specified length is greater than the actual length of the string, pad it on the right with spaces. If the specified length is less than the actual length of the string, truncate it on the right to the specified length.

Input: string, specified length

Return VARCHAR2 (adjusted string)

Display the original string, adjusted string, and length of the adjusted string.

CREATE OR REPLACE FUNCTION str\_play

(pv\_str\_1 IN VARCHAR2,

pv\_len\_str IN NUMBER)

return varchar2

IS

str\_1 VARCHAR2(13) := 'Violin string';

len\_str NUMBER;

ret\_vc VARCHAR2(24) := RPAD(str\_1, 24, 'c');

ret\_vc2 VARCHAR2(10) := RTRIM(str\_1, 'ing');

BEGIN

IF str\_1 < ret\_vc THEN

len\_str := LENGTH(ret\_vc);

dbms\_output.put\_line(str\_1 ||' ' ||ret\_vc|| ' '||len\_str);

ELSIF str\_1 > ret\_vc2 THEN

len\_str := LENGTH(ret\_vc2);

dbms\_output.put\_line(str\_1 || ' '||ret\_vc2|| ' '||len\_str);

ELSE

dbms\_output.put\_line('The soul is like a violin string: it makes music only when stretched. ~Neal Maxwell');

END IF;

END;

/

Function STR\_PLAY compiled