ARSHIA A KALKHORANI

106391170

**LAB4**

Q1 CODE:

SET SERVEROUTPUT ON

SET VERIFY OFF

CREATE OR REPLACE PROCEDURE mine(

p\_visa\_exp varchar2,

p\_case\_char char

)

IS

v\_count NUMBER;

e\_invalid\_input EXCEPTION;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Last day of the month ' || p\_visa\_exp || ' is ' || TO\_CHAR(last\_day(to\_date(p\_visa\_exp,'MM/YY')),'day'));

IF (UPPER(p\_case\_char) NOT LIKE 'P' AND UPPER(p\_case\_char) NOT LIKE 'F' and UPPER(p\_case\_char) NOT LIKE 'B')

THEN

RAISE e\_invalid\_input;

ELSIF (UPPER(p\_case\_char) = 'P') THEN

SELECT COUNT(\*) INTO v\_count FROM user\_objects

WHERE object\_type = 'PROCEDURE';

DBMS\_OUTPUT.PUT\_LINE('Number of stored objects of type ' || p\_case\_char || ' is ' || v\_count);

ELSIF (UPPER(p\_case\_char) = 'F') THEN

SELECT COUNT(\*) INTO v\_count FROM user\_objects

WHERE object\_type = 'FUNCTION';

DBMS\_OAUTPUT.PUT\_LINE('Number of stored objects of type ' || p\_case\_char || ' is ' || v\_count);

ELSIF (UPPER(p\_case\_char) = 'B') THEN

SELECT COUNT(\*) INTO v\_count FROM user\_objects

WHERE object\_type = 'PACKAGE';

DBMS\_OUTPUT.PUT\_LINE('Number of stored objects of type ' || p\_case\_char || ' is ' || v\_count);

END IF;

EXCEPTION

WHEN e\_invalid\_input THEN

DBMS\_OUTPUT.PUT\_LINE('You have entered an Invalid letter for the stored object. Try P, F or B.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('You have entered an Invalid FORMAT for the MONTH and YEAR. Try MM/YY');

END mine;

/

EXECUTE MINE('11/09','P');

EXECUTE MINE('12/09','f');

EXECUTE MINE('01/10','T');

EXECUTE MINE('13/09','P');

Q1 OUTPUT:

Procedure created.

Last day of the month 11/09 is monday

Number of stored objects of type P is 3

PL/SQL procedure successfully completed.

Last day of the month 12/09 is thursday

Number of stored objects of type f is 2

PL/SQL procedure successfully completed.

Last day of the month 01/10 is sunday

You have entered an Invalid letter for the stored object. Try P, F or B.

PL/SQL procedure successfully completed.

You have entered an Invalid FORMAT for the MONTH and YEAR. Try MM/YY

PL/SQL procedure successfully completed.

Q2CODE:

SET SERVEROUTPUT ON

SET VERIFY OFF

SET AUTOPRINT ON

CREATE OR REPLACE PROCEDURE add\_zip(

p\_zip zipcode.zip%TYPE,

p\_city zipcode.city%TYPE,

p\_state zipcode.state%TYPE,

p\_rows OUT NUMBER,

p\_success OUT VARCHAR2)

IS

v\_zip zipcode.zip%TYPE;

BEGIN

SELECT zip INTO v\_zip FROM zipcode

WHERE zip = p\_zip;

p\_success := 'FAILURE';

DBMS\_OUTPUT.PUT\_LINE('This ZIPCODE ' || p\_zip ||' is already in the Database. Try again.');

SELECT COUNT(\*) INTO p\_rows FROM zipcode

WHERE state = p\_state;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

p\_success := 'SUCCESS';

INSERT INTO zipcode

VALUES (p\_zip, p\_city, p\_state, USER, SYSDATE, USER, SYSDATE);

SELECT COUNT(\*) INTO p\_rows FROM zipcode

WHERE state = p\_state;

END add\_zip;

/

VARIABLE flag VARCHAR2(10)

VARIABLE zipnum NUMBER

EXECUTE add\_zip('18104', 'Chicago', 'MI', :zipnum, :flag)

SELECT \* FROM zipcode WHERE state = 'MI';

ROLLBACK;

EXECUTE add\_zip('48104', 'Ann Arbor', 'MI', :zipnum, :flag)

SELECT \* FROM zipcode WHERE state = 'MI';

ROLLBACK;

Q2OUTPUT:

Procedure created.

PL/SQL procedure successfully completed.

FLAG

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

SUCCESS

ZIPNUM

----------

2

ZIP CITY ST CREATED\_BY CREATED\_D MODIFIED\_BY MODIFIED\_

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

48104 Ann Arbor MI AMORRISO 03-AUG-99 ARISCHER 24-NOV-99

18104 Chicago MI DBS501\_183A19 14-NOV-18 DBS501\_183A19 14-NOV-18

Rollback complete.

This ZIPCODE 48104 is already in the Database. Try again.

PL/SQL procedure successfully completed.

FLAG

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

FAILURE

ZIPNUM

----------

1

ZIP CITY ST CREATED\_BY CREATED\_D MODIFIED\_BY MODIFIED\_

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

48104 Ann Arbor MI AMORRISO 03-AUG-99 ARISCHER 24-NOV-99

Rollback complete.

Q3CODE:

SET SERVEROUTPUT ON

SET AUTOPRINT ON

SET VERIFY OFF

CREATE OR REPLACE function exist\_zip(

v\_zip zipcode.zip%TYPE

)

RETURN BOOLEAN

IS

V\_INPUT NUMBER;

BEGIN

SELECT count(\*) INTO V\_INPUT from ZIPCODE

where ZIP = v\_zip;

IF V\_INPUT > 0 THEN

RETURN TRUE;

ELSE

RETURN FALSE;

END IF;

EXCEPTION

WHEN OTHERS THEN

RETURN FALSE;

END exist\_zip;

/

CREATE OR REPLACE PROCEDURE add\_zip2(

p\_zip zipcode.zip%TYPE,

p\_city zipcode.city%TYPE,

p\_state zipcode.state%TYPE,

p\_rows OUT NUMBER,

p\_success OUT VARCHAR2)

IS

v\_zip zipcode.zip%TYPE;

FALSE\_DATA EXCEPTION;

BEGIN

IF exist\_zip(p\_zip) THEN

p\_success := 'FAILURE';

ELSE

RAISE FALSE\_DATA;

END IF;

DBMS\_OUTPUT.PUT\_LINE('This ZIPCODE ' || p\_zip ||' is already in the Database. Try again.');

SELECT COUNT(\*) INTO p\_rows FROM zipcode

WHERE state = p\_state;

EXCEPTION

WHEN FALSE\_DATA THEN

p\_success := 'SUCCESS';

INSERT INTO zipcode

VALUES (p\_zip, p\_city, p\_state, USER, SYSDATE, USER, SYSDATE);

SELECT COUNT(\*) INTO p\_rows FROM zipcode

WHERE state = p\_state;

END add\_zip2;

/

VARIABLE flag VARCHAR2(10)

VARIABLE zipnum NUMBER

EXECUTE add\_zip2('18104', 'Chicago', 'MI', :zipnum, :flag)

SELECT \* FROM zipcode WHERE state = 'MI';

ROLLBACK;

EXECUTE add\_zip2('48104', 'Ann Arbor', 'MI', :zipnum, :flag)

SELECT \* FROM zipcode WHERE state = 'MI';

ROLLBACK;

Q3OUTPUT:SAME OUTPUT

Q4CODE:

SET SERVEROUTPUT ON

SET AUTOPRINT ON

SET VERIFY OFF

CREATE OR REPLACE function instruct\_status(

v\_first instructor.first\_name%TYPE,

v\_last instructor.last\_name%TYPE

)

RETURN VARCHAR2

IS

v\_instructor\_id instructor.instructor\_id%TYPE;

v\_sections NUMBER;

V\_message varchar2(100);

BEGIN

SELECT instructor\_id into v\_instructor\_id from instructor

where first\_name = v\_first

and last\_name = v\_last;

SELECT COUNT(\*) into v\_sections from section

where instructor\_id = v\_instructor\_id;

IF (v\_sections > 9) THEN

v\_message := 'This Instructor will teach 10 course and needs a vacation';

ELSIF (v\_sections > 0 AND v\_sections <= 9) THEN

v\_message := 'This Instructor will teach ' || v\_sections || ' courses';

ELSE

v\_message := 'This Instructor is NOT scheduled to teach ';

END IF;

RETURN v\_message;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

v\_message := 'There is NO such instructor';

RETURN v\_message;

WHEN OTHERS THEN

v\_message := 'Error';

RETURN v\_message;

END instruct\_status;

/

VARIABLE message VARCHAR2(100);

SELECT last\_name, INSTRUCT\_STATUS(FIRST\_NAME, LAST\_NAME) "Instructor\_Status"

FROM INSTRUCTOR

ORDER by LAST\_NAME;

EXECUTE :message := instruct\_status('Peter', 'Pan');

EXECUTE :message := instruct\_status('Irene', 'Willig');

Q4OUTPUT:

Function created.

LAST\_NAME

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

Instructor\_Status

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

Chow

This Instructor is NOT scheduled to teach

Frantzen

This Instructor will teach 10 course and needs a vacation

Hanks

This Instructor will teach 9 courses

Lowry

This Instructor will teach 9 courses

Morris

This Instructor will teach 10 course and needs a vacation

Pertez

This Instructor will teach 10 course and needs a vacation

Schorin

This Instructor will teach 10 course and needs a vacation

Smythe

This Instructor will teach 10 course and needs a vacation

Willig

This Instructor is NOT scheduled to teach

Wojick

This Instructor will teach 10 course and needs a vacation

10 rows selected.

PL/SQL procedure successfully completed.

MESSAGE

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

There is NO such instructor

PL/SQL procedure successfully completed.

MESSAGE

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

This Instructor is NOT scheduled to teach

SQL> spool off;

Lab 3 DBS 501 By Le, Sean Teacher: Nebojsa

END