1. Write a procedure with no parameters. The procedure should say whether the current day is a weekend or weekday. Additionally, it should tell you the user’s name and the current time. It also should specify how many valid and invalid procedures are in the database.

SOLUTION:

SET SERVEROUTPUT ON

CREATE OR REPLACE PROCEDURE my\_procedure AS

v\_date DATE := TO\_DATE(SYSDATE);

v\_hour VARCHAR2(10);

v\_day VARCHAR2(15);

the\_user VARCHAR2(30);

BEGIN

v\_day := RTRIM(TO\_CHAR(v\_date),'DAY');

v\_hour := TO\_CHAR(v\_date,'HH:MI');

IF v\_day IN ('SATURDAY','SUNDAY') THEN

DBMS\_OUTPUT.PUT\_LINE('It is a weekend day!');

ELSE

DBMS\_OUTPUT.PUT\_LINE('It is a week day!');

END IF;

SELECT user

INTO the\_user

FROM dual;

DBMS\_OUTPUT.PUT\_LINE('The current user is '||the\_user||' and the current time is '||v\_hour);

END;

/

EXEC my\_procedure;

/

OUTPUT:

It is a week day!

The current user is SYS and the current time is 12:00

1. Write a procedure that takes in a zip code, city, and state and inserts the values into the zip code table. It should check to see if the zip code is already in the database. If it is, an exception should be raised, and an error message should be displayed. Write an anonymous block that uses the procedure and inserts your zip code.

SOLUTION:

SET SERVEROUTPUT ON

CREATE OR REPLACE PROCEDURE

my\_procedure2(the\_zip\_code IN zipcode.zip%TYPE,

the\_city IN zipcode.city%TYPE,

the\_state IN zipcode.state%TYPE)

AS

zip\_code\_var zipcode.zip%TYPE;

check\_zipcode\_var zipcode.zip%TYPE;

city\_var zipcode.zip%TYPE;

state\_var zipcode.zip%TYPE;

BEGIN

zip\_code\_var := the\_zip\_code;

city\_var := the\_city;

state\_var := the\_state;

SELECT zip

INTO check\_zipcode\_var

FROM zipcode

WHERE zip = zip\_code\_var;

DBMS\_OUTPUT.PUT\_LINE('Zip code exists in the tabel!');

EXCEPTION WHEN NO\_DATA\_FOUND THEN

INSERT INTO ZIPCODE

VALUES(zip\_code\_var, city\_var, state\_var, 'Emanuel Kokovics', SYSDATE, 'Emanuel Kokovics', SYSDATE);

END my\_procedure2;

/

BEGIN

my\_procedure2(30025, 'Sibiu', 'RO');

END;

BEGIN

my\_procedure2(30000, 'Sibiu', 'RO');

END;

/

1. Write a stored function called new\_student\_id that takes in no parameters and returns a student.student\_id%TYPE. The value returned will be used when inserting a new student into the application. It will be derived by using the formula student\_id\_seq.NEXTVAL.

SOLUTION:

SET SERVEROUTPUT ON

CREATE OR REPLACE FUNCTION new\_student\_id RETURN student.student\_id%TYPE IS

last\_student\_id student.student\_id%TYPE;

BEGIN

SELECT student\_id\_seq.NEXTVAL

INTO last\_student\_id

FROM student;

RETURN(last\_student\_id);

END;

OUTPUT:

LINE/COL ERROR

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

4/5 PL/SQL: SQL Statement ignored

4/12 PL/SQL: ORA-02289: sequence does not exist

Errors: check compiler log

1. Write a stored function called zip\_does\_not\_exist that takes in a zipcode.zip%TYPE and returns a Boolean. The function will return TRUE if the zip code passed into it does not exist. It will return a FALSE if the zip code does exist.

SOLUTION:

SET SERVEROUTPUT ON

CREATE OR REPLACE FUNCTION zip\_does\_not\_exist(the\_zip\_code zipcode.zip%TYPE) RETURN BOOLEAN AS

boolean\_var CHAR(10);

BEGIN

SELECT NULL

INTO boolean\_var

FROM zipcode

WHERE zip = the\_zip\_code;

RETURN FALSE;

EXCEPTION WHEN NO\_DATA\_FOUND THEN

RETURN TRUE;

END;