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
```

2. 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;
/
```

3. 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
```

4. 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;
```