# *Database Management II (420-D20-HR)*

# *Lab 6 – Automated Testing, Packages*

Date assigned: Tuesday, March 14, 2017

Date due: **Tuesday, March 14, 2017, 5:50pm**

**Objectives:**

At the end of this lab you will be able to:

* Write a stored procedure that fulfills a specified requirement
* Write an automated test case to test a stored procedure given a test case
* Implement SQL packages with private and public procedures/functions

**To be handed in:**

1. The ***username*\_D20\_L06\_AutomatedTestPackages.docx** should be uploaded to **Moodle**.

**Reference:**

* Slides S09, S10, Notes N10 found on Moodle

**To Start:**

1. Rename this document to ***username*\_D20\_L06\_AutomatedTestPackages*.*docx**.

**Use ANSI/ISO standard joins for all joins.**

**Name all identifiers according to the naming standards shown below.**

**Format all blocks using the SQL Navigator Formatter**

**Provide output from the execution of the procedures and testing.**

|  |  |  |
| --- | --- | --- |
| **Problem** | **Mark** | **Out of** |
| A – Intro to Packages |  | 18 |
| * + Test/automation |  | 24 |
| B – Package – private procedures |  | 10 |
| * + Test/automation |  | 10 |
| Organization - Blocks Formatted and naming standards used, assessment |  | 5 |
| **Total** |  | **67** |

**Naming Standards:**

|  |  |  |
| --- | --- | --- |
| **Identifier type** | **Prefix** | **Suffix** |
| type | **typ\_** |  |
| local variable | **lv\_** |  |
| local record | **lrec\_** |  |
| local cursor | **lcur\_** |  |
| exception | **e\_** |  |
| procedure name |  | **\_sp** |
| IN parameter | **pv\_** | **\_i** |
| OUT parameter | **pv\_** | **\_o** |
| IN OUT parameter | **pv\_** | **\_io** |
| parameter record | **prec\_** | **\_i, \_o or \_io** |
| package |  | **\_pkg** |
| package function |  | **\_pf** |
| package procedure |  | **\_pp** |
|  |  |  |

# Packages

***Objectives*:** Learnto create and use procedures and functions in a package.

***To Do:***

## Create a package called **D20\_L06\_iu\_room\_pkg** that contains a procedure called **room\_capacity\_pp** and a function called **class\_capacity\_pf**:

### The **room\_capacity\_pp** procedure accepts a roomid as an input parameter and returns the building, room number and capacity of the room. If the roomid is invalid, the capacity will be -1 and the other return parameters will be unchanged.

### The **class\_capacity\_pf** function accepts a csid parameter and returns the maximum number of students in the section. If the csid is invalid, it returns a value of -1.

## Add a second **room\_capacity\_pp** procedure to the **D20\_L06\_iu\_room\_pkg**. It should accept the building and room number as input and return the capacity. If the building and/or room number are invalid, the capacity should be -1.

## Test your package on the following test cases. Automate your test cases and include them with your package, call the test procedure **test\_pkg\_sp()**.

## At the end of the test, it should indicate:

How many tests were run

How many tests passed, and % tests passed

How many tests failed, and % tests failed

Make it look nice, so that columns in the report lines up nicely.

**room\_capacity\_pp** Test Cases:

| **Scenario** | **Input** | **Expected Results** | | |
| --- | --- | --- | --- | --- |
| **roomid** | **building** | **room no.** | **capacity** |
| 1. The roomid is valid | 19 | Kennedy | 210 | 30 |
| 2. The roomid is invalid | 55 | - | - | -1 |

**class\_capacity\_pf** Test Cases:

| **Scenario** | **Input** | **Expected Results** |
| --- | --- | --- |
| **csid** | **max count** |
| 1. The csid is valid | 1209 | 40 |
| 2. The csid is invalid | 1415 | -1 |

**room\_capacity\_pp** Test Cases:

| **Scenario** | **Input** | | **Expected Results** |
| --- | --- | --- | --- |
| **building** | **room no.** | **capacity** |
| 1. The building and room number are valid | Kennedy | 210 | 30 |
| 2. The building and room number are invalid | Jasper | 101 | -1 |

## When you are finished, provide your package code:

**PL/SQL:**

**Specifcation:**

**CREATE OR REPLACE PACKAGE D20\_L06\_iu\_room\_pkg IS**

**PROCEDURE room\_capacity\_pp (**

**pv\_roomid\_i iu\_location.roomid%TYPE,**

**pv\_building\_o OUT iu\_location.building%TYPE,**

**pv\_roomid\_o OUT iu\_location.roomid%TYPE,**

**pv\_capacity\_o OUT iu\_location.capacity%TYPE**

**);**

**PROCEDURE room\_capacity\_pp (**

**pv\_roomno\_i iu\_location.roomid%TYPE,**

**pv\_building\_i iu\_location.building%TYPE,**

**pv\_capacity\_o OUT iu\_location.capacity%TYPE**

**);**

**FUNCTION class\_capacity\_pf (**

**pv\_csid\_i iu\_crssection.csid%TYPE**

**) return number;**

**END D20\_L06\_iu\_room\_pkg;**

**Body:**

**CREATE OR REPLACE PACKAGE BODY D20\_L06\_iu\_room\_pkg IS**

**PROCEDURE room\_capacity\_pp (**

**pv\_roomid\_i iu\_location.roomid%TYPE,**

**pv\_building\_o OUT iu\_location.building%TYPE,**

**pv\_roomid\_o OUT iu\_location.roomid%TYPE,**

**pv\_capacity\_o OUT iu\_location.capacity%TYPE**

**)**

**IS**

**BEGIN**

**SELECT building, roomid, capacity**

**INTO pv\_building\_o, pv\_roomid\_o, pv\_capacity\_o**

**FROM iu\_location**

**WHERE roomid = pv\_roomid\_i;**

**EXCEPTION**

**WHEN NO\_DATA\_FOUND THEN**

**pv\_building\_o := '';**

**pv\_roomid\_o := 0;**

**pv\_capacity\_o := -1;**

**END;**

**PROCEDURE room\_capacity\_pp (**

**pv\_roomno\_i iu\_location.roomid%TYPE,**

**pv\_building\_i iu\_location.building%TYPE,**

**pv\_capacity\_o OUT iu\_location.capacity%TYPE**

**) IS**

**BEGIN**

**SELECT capacity INTO pv\_capacity\_o**

**FROM iu\_location**

**WHERE roomno = pv\_roomno\_i**

**AND building = pv\_building\_i;**

**EXCEPTION**

**WHEN NO\_DATA\_FOUND THEN**

**pv\_capacity\_o := -1;**

**END;**

**FUNCTION class\_capacity\_pf (**

**pv\_csid\_i iu\_crssection.csid%TYPE**

**) return number**

**IS**

**lv\_retVal number := -1;**

**BEGIN**

**SELECT l.capacity INTO lv\_retVal**

**FROM iu\_location l, iu\_crssection c**

**WHERE c.csid = pv\_csid\_i**

**AND c.roomid = l.roomid;**

**return lv\_retVal;**

**EXCEPTION**

**WHEN NO\_DATA\_FOUND THEN**

**return -1;**

**END;**

**PROCEDURE test\_pkg\_sp IS**

**BEGIN**

**DECLARE**

**lv\_building iu\_location.building%TYPE;**

**lv\_roomid iu\_location.roomid%TYPE;**

**lv\_capacity iu\_location.capacity%TYPE;**

**lv\_csid iu\_crssection.csid%TYPE;**

**lv\_success number := 0;**

**BEGIN**

**--Test room\_capacity\_pp(roomid)**

**room\_capacity\_pp(19, lv\_building, lv\_roomid, lv\_capacity);**

**lv\_success := CASE**

**WHEN lv\_building = 'Kennedy' and lv\_roomid = 19 and lv\_capacity = 30**

**THEN lv\_success + 1**

**ELSE lv\_success END;**

**DBMS\_OUTPUT.PUT\_LINE('Building - Expected value: Kennedy / Actual value: ' || lv\_building );**

**DBMS\_OUTPUT.PUT\_LINE('Room Id - Expected value: 19 / Actual value: ' || lv\_roomid );**

**DBMS\_OUTPUT.PUT\_LINE('Capacity - Expected value: 30 / Actual value: ' || lv\_capacity );**

**room\_capacity\_pp(55, lv\_building, lv\_roomid, lv\_capacity);**

**lv\_success := CASE**

**WHEN lv\_building IS NULL and lv\_roomid = 0 and lv\_capacity = -1**

**THEN lv\_success + 1**

**ELSE lv\_success END;**

**DBMS\_OUTPUT.PUT\_LINE('Building - Expected value: -- / Actual value: ' || lv\_building );**

**DBMS\_OUTPUT.PUT\_LINE('Room Id - Expected value: -- / Actual value: ' || lv\_roomid );**

**DBMS\_OUTPUT.PUT\_LINE('Capacity - Expected value: -1 / Actual value: ' || lv\_capacity );**

**--Test room\_capacity\_pp(roomid, building)**

**DBMS\_OUTPUT.PUT\_LINE('');**

**room\_capacity\_pp(210, 'Kennedy', lv\_capacity);**

**lv\_success := CASE WHEN lv\_capacity = 30 THEN lv\_success + 1 ELSE lv\_success END;**

**DBMS\_OUTPUT.PUT\_LINE('Capacity - Expected value: 30 / Actual value: ' || lv\_capacity);**

**room\_capacity\_pp(101, 'Jasper', lv\_capacity);**

**lv\_success := CASE WHEN lv\_capacity = -1 THEN lv\_success + 1 ELSE lv\_success END;**

**DBMS\_OUTPUT.PUT\_LINE('Capacity - Expected value: -1 / Actual value: ' || lv\_capacity);**

**--Test class\_capacity\_pf(csid)**

**DBMS\_OUTPUT.PUT\_LINE('');**

**lv\_csid := class\_capacity\_pf(1209);**

**lv\_success := CASE WHEN lv\_csid = 40 THEN lv\_success + 1 ELSE lv\_success END;**

**DBMS\_OUTPUT.PUT\_LINE('Maxcount - Expected value: 40 / Actual value: ' || lv\_csid);**

**lv\_csid := class\_capacity\_pf(1415);**

**lv\_success := CASE WHEN lv\_csid = -1 THEN lv\_success + 1 ELSE lv\_success END;**

**DBMS\_OUTPUT.PUT\_LINE('Maxcount - Expected value: -1 / Actual value: ' || lv\_csid);**

**--End result**

**DBMS\_OUTPUT.PUT\_LINE('');**

**DBMS\_OUTPUT.PUT\_LINE('Succeeded: ' || lv\_success);**

**DBMS\_OUTPUT.PUT\_LINE('% Passed: ' || (lv\_success / 10) \* 100);**

**DBMS\_OUTPUT.PUT\_LINE('% Failed: ' || ((10-lv\_success) / 10) \* 100);**

**END;**

**END;**

**END D20\_L06\_iu\_room\_pkg;**

**Test code:**

**PROCEDURE test\_pkg\_sp IS**

**BEGIN**

**DECLARE**

**lv\_building iu\_location.building%TYPE;**

**lv\_roomid iu\_location.roomid%TYPE;**

**lv\_capacity iu\_location.capacity%TYPE;**

**lv\_csid iu\_crssection.csid%TYPE;**

**lv\_success number := 0;**

**BEGIN**

**--Test room\_capacity\_pp(roomid)**

**room\_capacity\_pp(19, lv\_building, lv\_roomid, lv\_capacity);**

**lv\_success := CASE**

**WHEN lv\_building = 'Kennedy' and lv\_roomid = 19 and lv\_capacity = 30**

**THEN lv\_success + 1**

**ELSE lv\_success END;**

**DBMS\_OUTPUT.PUT\_LINE('Building - Expected value: Kennedy / Actual value: ' || lv\_building );**

**DBMS\_OUTPUT.PUT\_LINE('Room Id - Expected value: 19 / Actual value: ' || lv\_roomid );**

**DBMS\_OUTPUT.PUT\_LINE('Capacity - Expected value: 30 / Actual value: ' || lv\_capacity );**

**room\_capacity\_pp(55, lv\_building, lv\_roomid, lv\_capacity);**

**lv\_success := CASE**

**WHEN lv\_building IS NULL and lv\_roomid = 0 and lv\_capacity = -1**

**THEN lv\_success + 1**

**ELSE lv\_success END;**

**DBMS\_OUTPUT.PUT\_LINE('Building - Expected value: -- / Actual value: ' || lv\_building );**

**DBMS\_OUTPUT.PUT\_LINE('Room Id - Expected value: -- / Actual value: ' || lv\_roomid );**

**DBMS\_OUTPUT.PUT\_LINE('Capacity - Expected value: -1 / Actual value: ' || lv\_capacity );**

**--Test room\_capacity\_pp(roomid, building)**

**DBMS\_OUTPUT.PUT\_LINE('');**

**room\_capacity\_pp(210, 'Kennedy', lv\_capacity);**

**lv\_success := CASE WHEN lv\_capacity = 30 THEN lv\_success + 1 ELSE lv\_success END;**

**DBMS\_OUTPUT.PUT\_LINE('Capacity - Expected value: 30 / Actual value: ' || lv\_capacity);**

**room\_capacity\_pp(101, 'Jasper', lv\_capacity);**

**lv\_success := CASE WHEN lv\_capacity = -1 THEN lv\_success + 1 ELSE lv\_success END;**

**DBMS\_OUTPUT.PUT\_LINE('Capacity - Expected value: -1 / Actual value: ' || lv\_capacity);**

**--Test class\_capacity\_pf(csid)**

**DBMS\_OUTPUT.PUT\_LINE('');**

**lv\_csid := class\_capacity\_pf(1209);**

**lv\_success := CASE WHEN lv\_csid = 40 THEN lv\_success + 1 ELSE lv\_success END;**

**DBMS\_OUTPUT.PUT\_LINE('Maxcount - Expected value: 40 / Actual value: ' || lv\_csid);**

**lv\_csid := class\_capacity\_pf(1415);**

**lv\_success := CASE WHEN lv\_csid = -1 THEN lv\_success + 1 ELSE lv\_success END;**

**DBMS\_OUTPUT.PUT\_LINE('Maxcount - Expected value: -1 / Actual value: ' || lv\_csid);**

**--End result**

**DBMS\_OUTPUT.PUT\_LINE('');**

**DBMS\_OUTPUT.PUT\_LINE('Succeeded: ' || lv\_success);**

**DBMS\_OUTPUT.PUT\_LINE('% Passed: ' || (lv\_success / 10) \* 100);**

**DBMS\_OUTPUT.PUT\_LINE('% Failed: ' || ((10-lv\_success) / 10) \* 100);**

**END;**

**Sample output from Test Code:**

**Building - Expected value: Kennedy / Actual value: Kennedy**

**Room Id - Expected value: 19 / Actual value: 19**

**Capacity - Expected value: 30 / Actual value: 30**

**Building - Expected value: -- / Actual value:**

**Room Id - Expected value: -- / Actual value: 0**

**Capacity - Expected value: -1 / Actual value: -1**

**Capacity - Expected value: 30 / Actual value: 30**

**Capacity - Expected value: -1 / Actual value: -1**

**Maxcount - Expected value: 40 / Actual value: 40**

**Maxcount - Expected value: -1 / Actual value: -1**

**Succeeded: 6**

**% Passed: 60**

**% Failed: 40**

# Private Procedures and Functions

***Objectives*:** Learnto create and use private procedures and functions in a package.

***To Do:***

A private function or procedure is a function or procedure that is contained in a package body, but not the package header.

## A package has been partially created to maintain the faculty table. It contains two public procedures – **add\_faculty\_pp** which adds a new faculty member and assigns an office to him/her and **list\_department\_pp** which returns a list of all the faculty members in a department. Open **D20\_L06\_iu\_faculty\_pkg\_spec.sql** from Moodle. Compile it.

## Open **D20\_L06\_iu\_faculty\_pkg\_body.sql** from Moodle. In addition to the two public procedures, it contains a private procedure –**assign\_office\_pp**. Compile it. It will give you an error because it is missing a function.

## In the package body, create a private function called **get\_available\_office\_pf** afterthe **assign\_office\_pp** procedure:

### There are no parameters

### It returns a roomid.

### It should find the first available office roomid and return it. If there are no available offices, it should return null. An available office is an office with a capacity greater than the number of faculty it has been assigned to.

## Compile the package body. What happens? A function or procedure in a package must be declared before it is called. Move the **get\_available\_office\_pf** function before the **assign\_office\_pp** procedure and compile it again.

## Try to execute the **get\_available\_office\_pf** function from SQL Developer. What happens? You cannot execute a private function or procedure from outside the package.

## Write test\_pkg() to test the add\_faculty\_pp() procedure. Roll back after each test.

| **Test cases** | **Input Parameters** | | | **Expected Output** | | | **Faculty table** |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **facultyid** | **name** | **department** | **building** | **roomno** | **success** |
| 1. Faculty doesn't exist. Office is available. | 654 | May | Computer Science | Heritage | 217 | S | Row added to iu\_faculty table for May |
| 2. Faculty doesn't exist. No available office. | 655 | Trudeau | Computer Science | null | null | U | Row added to iu\_faculty table for Trudeau – null roomid |
| 3. Faculty already exists. | 111 | Jones | Computer Science | null | null | F | No row added |
| 4. Invalid department. | 656 | Mulcair | Information Science | null | null | F | No row added |

**PL/SQL:**

**Specifcation:**

**CREATE OR REPLACE PACKAGE d20\_l06\_iu\_faculty\_pkg**

**IS**

**PROCEDURE add\_faculty\_pp (**

**pv\_facultyid\_i iu\_faculty.facultyid%TYPE,**

**pv\_name\_i IN iu\_faculty.NAME%TYPE,**

**pv\_deptname\_i IN iu\_department.deptname%TYPE,**

**pv\_building\_o OUT iu\_location.building%TYPE,**

**pv\_roomno\_o OUT iu\_location.roomno%TYPE,**

**pv\_success\_o OUT CHAR**

**);**

**PROCEDURE list\_department\_pp (**

**pv\_deptname\_i iu\_department.deptname%TYPE,**

**pcur\_faculty\_o OUT sys\_refcursor**

**);**

**PROCEDURE test\_pkg;**

**END d20\_l06\_iu\_faculty\_pkg;**

**Body:**

**CREATE OR REPLACE PACKAGE BODY d20\_L06\_iu\_faculty\_pkg**

**IS**

**-- get an office that is currently available**

**FUNCTION get\_available\_office\_pf**

**return iu\_location.roomid%TYPE IS**

**lrec\_first number;**

**BEGIN**

**SELECT DISTINCT l.roomid into lrec\_first**

**FROM iu\_location l**

**WHERE l.capacity > (**

**SELECT COUNT(roomid)**

**FROM iu\_faculty f**

**WHERE f.roomid = l.roomid**

**)**

**AND l.roomtype = 'O'**

**AND rownum = 1;**

**return lrec\_first;**

**EXCEPTION**

**WHEN NO\_DATA\_FOUND THEN**

**return null;**

**END get\_available\_office\_pf;**

**-- assign a faculty member to an available office**

**PROCEDURE assign\_office\_pp (**

**pv\_facultyid\_i iu\_faculty.facultyid%TYPE,**

**pv\_building\_o OUT iu\_location.building%TYPE,**

**pv\_roomno\_o OUT iu\_location.roomno%TYPE**

**)**

**IS**

**lv\_roomid iu\_location.roomid%TYPE;**

**lv\_current\_roomid iu\_faculty.roomid%TYPE;**

**BEGIN**

**lv\_roomid := get\_available\_office\_pf;**

**IF lv\_roomid IS NOT NULL**

**THEN**

**SELECT building, roomno**

**INTO pv\_building\_o, pv\_roomno\_o**

**FROM iu\_location**

**WHERE roomid = lv\_roomid;**

**UPDATE iu\_faculty**

**SET roomid = lv\_roomid**

**WHERE facultyid = pv\_facultyid\_i;**

**ELSE**

**pv\_building\_o := NULL; -- no office available**

**END IF;**

**END assign\_office\_pp;**

**-- add a new faculty member**

**PROCEDURE add\_faculty\_pp (**

**pv\_facultyid\_i iu\_faculty.facultyid%TYPE,**

**pv\_name\_i IN iu\_faculty.NAME%TYPE,**

**pv\_deptname\_i IN iu\_department.deptname%TYPE,**

**pv\_building\_o OUT iu\_location.building%TYPE,**

**pv\_roomno\_o OUT iu\_location.roomno%TYPE,**

**pv\_success\_o OUT CHAR**

**)**

**IS**

**lv\_deptid iu\_department.deptid%TYPE;**

**BEGIN**

**SELECT deptid**

**INTO lv\_deptid**

**FROM iu\_department**

**WHERE UPPER (deptname) = UPPER (pv\_deptname\_i);**

**INSERT INTO iu\_faculty**

**(facultyid, NAME, deptid**

**)**

**VALUES (pv\_facultyid\_i, pv\_name\_i, lv\_deptid**

**);**

**assign\_office\_pp (pv\_facultyid\_i, pv\_building\_o, pv\_roomno\_o);**

**IF pv\_building\_o IS NULL**

**THEN**

**pv\_success\_o := 'U';**

**ELSE**

**pv\_success\_o := 'S';**

**END IF;**

**EXCEPTION**

**WHEN OTHERS**

**THEN**

**pv\_success\_o := 'F';**

**END;**

**-- list all the faculty in a department**

**PROCEDURE list\_department\_pp (**

**pv\_deptname\_i iu\_department.deptname%TYPE,**

**pcur\_faculty\_o OUT sys\_refcursor**

**)**

**IS**

**BEGIN**

**OPEN pcur\_faculty\_o**

**FOR**

**SELECT NAME, building || ' Room ' || roomno office, phone**

**FROM iu\_faculty JOIN iu\_department USING (deptid) JOIN iu\_location USING (roomid)**

**WHERE UPPER (deptname) = UPPER (pv\_deptname\_i);**

**END;**

**PROCEDURE test\_pkg IS**

**lv\_passed number := 0;**

**lv\_building iu\_location.building%TYPE;**

**lv\_roomno iu\_location.roomno%TYPE;**

**lv\_success CHAR;**

**BEGIN**

**add\_faculty\_pp(654, 'May', 'Computer Science', lv\_building, lv\_roomno, lv\_success);**

**lv\_passed := CASE**

**WHEN lv\_building = 'Heritage' AND lv\_roomno = 217 AND lv\_success = 'S' THEN**

**lv\_passed + 1**

**ELSE lv\_passed END;**

**DBMS\_OUTPUT.PUT\_LINE('Building - Expected value: Heritage / Actual value: ' || lv\_building );**

**DBMS\_OUTPUT.PUT\_LINE('Room - Expected value: 217 / Actual value: ' || lv\_roomno );**

**DBMS\_OUTPUT.PUT\_LINE('Success - Expected value: S / Actual value: ' || lv\_success );**

**add\_faculty\_pp(655, 'Trudeau', 'Computer Science', lv\_building, lv\_roomno, lv\_success);**

**lv\_passed := CASE**

**WHEN lv\_building is null AND lv\_roomno is null AND lv\_success = 'U' THEN**

**lv\_passed + 1**

**ELSE lv\_passed END;**

**DBMS\_OUTPUT.PUT\_LINE('Building - Expected value: -- / Actual value: ' || lv\_building );**

**DBMS\_OUTPUT.PUT\_LINE('Room - Expected value: -- / Actual value: ' || lv\_roomno );**

**DBMS\_OUTPUT.PUT\_LINE('Success - Expected value: U / Actual value: ' || lv\_success );**

**add\_faculty\_pp(111, 'Jones', 'Computer Science', lv\_building, lv\_roomno, lv\_success);**

**lv\_passed := CASE**

**WHEN lv\_building is null AND lv\_roomno is null AND lv\_success = 'F' THEN**

**lv\_passed + 1**

**ELSE lv\_passed END;**

**DBMS\_OUTPUT.PUT\_LINE('Building - Expected value: -- / Actual value: ' || lv\_building );**

**DBMS\_OUTPUT.PUT\_LINE('Room - Expected value: -- / Actual value: ' || lv\_roomno );**

**DBMS\_OUTPUT.PUT\_LINE('Success - Expected value: F / Actual value: ' || lv\_success );**

**add\_faculty\_pp(656, 'Mulcair', 'Information Science', lv\_building, lv\_roomno, lv\_success);**

**lv\_passed := CASE**

**WHEN lv\_building is null AND lv\_roomno is null AND lv\_success = 'F' THEN**

**lv\_passed + 1**

**ELSE lv\_passed END;**

**DBMS\_OUTPUT.PUT\_LINE('Building - Expected value: -- / Actual value: ' || lv\_building );**

**DBMS\_OUTPUT.PUT\_LINE('Room - Expected value: -- / Actual value: ' || lv\_roomno );**

**DBMS\_OUTPUT.PUT\_LINE('Success - Expected value: F / Actual value: ' || lv\_success );**

**DBMS\_OUTPUT.PUT\_LINE('');**

**DBMS\_OUTPUT.PUT\_LINE('Succeeded: ' || lv\_passed);**

**DBMS\_OUTPUT.PUT\_LINE('% Passed: ' || (lv\_passed / 4) \* 100);**

**DBMS\_OUTPUT.PUT\_LINE('% Failed: ' || ((4-lv\_passed) / 4) \* 100);**

**END;**

**END d20\_L06\_iu\_faculty\_pkg;**

**Test code:**

**PROCEDURE test\_pkg IS**

**lv\_passed number := 0;**

**lv\_building iu\_location.building%TYPE;**

**lv\_roomno iu\_location.roomno%TYPE;**

**lv\_success CHAR;**

**BEGIN**

**add\_faculty\_pp(654, 'May', 'Computer Science', lv\_building, lv\_roomno, lv\_success);**

**lv\_passed := CASE**

**WHEN lv\_building = 'Heritage' AND lv\_roomno = 217 AND lv\_success = 'S' THEN**

**lv\_passed + 1**

**ELSE lv\_passed END;**

**DBMS\_OUTPUT.PUT\_LINE('Building - Expected value: Heritage / Actual value: ' || lv\_building );**

**DBMS\_OUTPUT.PUT\_LINE('Room - Expected value: 217 / Actual value: ' || lv\_roomno );**

**DBMS\_OUTPUT.PUT\_LINE('Success - Expected value: S / Actual value: ' || lv\_success );**

**add\_faculty\_pp(655, 'Trudeau', 'Computer Science', lv\_building, lv\_roomno, lv\_success);**

**lv\_passed := CASE**

**WHEN lv\_building is null AND lv\_roomno is null AND lv\_success = 'U' THEN**

**lv\_passed + 1**

**ELSE lv\_passed END;**

**DBMS\_OUTPUT.PUT\_LINE('Building - Expected value: -- / Actual value: ' || lv\_building );**

**DBMS\_OUTPUT.PUT\_LINE('Room - Expected value: -- / Actual value: ' || lv\_roomno );**

**DBMS\_OUTPUT.PUT\_LINE('Success - Expected value: U / Actual value: ' || lv\_success );**

**add\_faculty\_pp(111, 'Jones', 'Computer Science', lv\_building, lv\_roomno, lv\_success);**

**lv\_passed := CASE**

**WHEN lv\_building is null AND lv\_roomno is null AND lv\_success = 'F' THEN**

**lv\_passed + 1**

**ELSE lv\_passed END;**

**DBMS\_OUTPUT.PUT\_LINE('Building - Expected value: -- / Actual value: ' || lv\_building );**

**DBMS\_OUTPUT.PUT\_LINE('Room - Expected value: -- / Actual value: ' || lv\_roomno );**

**DBMS\_OUTPUT.PUT\_LINE('Success - Expected value: F / Actual value: ' || lv\_success );**

**add\_faculty\_pp(656, 'Mulcair', 'Information Science', lv\_building, lv\_roomno, lv\_success);**

**lv\_passed := CASE**

**WHEN lv\_building is null AND lv\_roomno is null AND lv\_success = 'F' THEN**

**lv\_passed + 1**

**ELSE lv\_passed END;**

**DBMS\_OUTPUT.PUT\_LINE('Building - Expected value: -- / Actual value: ' || lv\_building );**

**DBMS\_OUTPUT.PUT\_LINE('Room - Expected value: -- / Actual value: ' || lv\_roomno );**

**DBMS\_OUTPUT.PUT\_LINE('Success - Expected value: F / Actual value: ' || lv\_success );**

**DBMS\_OUTPUT.PUT\_LINE('');**

**DBMS\_OUTPUT.PUT\_LINE('Succeeded: ' || lv\_passed);**

**DBMS\_OUTPUT.PUT\_LINE('% Passed: ' || (lv\_passed / 4) \* 100);**

**DBMS\_OUTPUT.PUT\_LINE('% Failed: ' || ((4-lv\_passed) / 4) \* 100);**

**END;**

**Sample output from Test Code:**

**Building - Expected value: Heritage / Actual value: Heritage**

**Room - Expected value: 217 / Actual value: 217**

**Success - Expected value: S / Actual value: S**

**Building - Expected value: -- / Actual value:**

**Room - Expected value: -- / Actual value:**

**Success - Expected value: U / Actual value: U**

**Building - Expected value: -- / Actual value:**

**Room - Expected value: -- / Actual value:**

**Success - Expected value: F / Actual value: F**

**Building - Expected value: -- / Actual value:**

**Room - Expected value: -- / Actual value:**

**Success - Expected value: F / Actual value: F**

**Succeeded: 4**

**% Passed: 100**

**% Failed: 0**

# Assessment

1. What did you learn in completing this lab?

Packages

1. What did you have difficulty with?

Focusing a bit

1. What did you do well?

Part a

1. How many hours did you spend in completing this lab?

2 1/2

1. What took you the most time?

Getting part b working with the broken test case