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

# *Happy Valley Kennels Project*

# *Assignment 3 – Stored Procedures/Function, Packages and Triggers*

Date assigned: Tuesday, March 14, 2017

Date due: **Monday, March 27, 2017, 11:50PM**

**Learning Objectives**

After completing this assignment, the student will be able to:

1. Write PL/SQL stored procedures and functions using input and output parameters;
2. Use default parameters;
3. Use SYS\_REFCURSOR to return a result set to an application program;
4. Write automated test cases.
5. Combine procedures in a package.
6. Write PL/SQL triggers

**To Be Handed in:**

* 1. The ***username***\_**D20\_A03\_StoredProcsPackages.docx** file containing the SQL source code and the test cases for this assignment

**To Start:**

1. Rename this document to ***username*\_D20\_A03\_StoredProcsPackages.docx**. Add your answers and output in the appropriate sections.
2. If you’re unsure of the state of your HVK database, back it up before you start this assignment. You can then rebuild a clean copy by running:
   1. HVK Database Backup\_2017 – this has all the updates you should need. Get the latest copy from Moodle.

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

**Use ISO/ANSI standard joins in all your SELECT statements.**

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

**Include output from program execution**

**Marking:**

|  |  |  |
| --- | --- | --- |
| Section | Question | Mark |
| 1. SYS\_REFCURSOR | a-f – Test case Design (x6) | 48 |
|  | a-f – Test case Automation and execution (x6) | 24 |
|  | a-f – pp code (x6) | 24 |
|  | Test case summary/reporting, private procedures | 10 |
| 1. Triggers | Change Log Table Created | 4 |
|  | Change Log filled by triggers (code/test/output) | 12 |
|  | Validation triggers (code/test/output) | 12 |
| English and handed in properly (coding/naming standards, all output messages are meaningful and clear, effort tracking), assessment |  | 6 |
| Totals |  | 140 |

**Naming Conventions:**

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

# SYS\_REFCURSOR

Purpose: Learn to write PL/SQL procedures that use SYS\_REFCURSOR to return a result set to an application program.

Reference: See [here](https://oracle-base.com/articles/misc/using-ref-cursors-to-return-recordsets) for details, syntax and examples for SYS\_REFCURSOR. It is essentially, a way to return a result set from a Select. [This](https://community.oracle.com/thread/888365) article goes a bit more in depth if you’re curious.

To Do:

## Create a package called **hvk\_table\_lists\_pkg**. It should include the following procedures:

### list\_available\_runs\_pp which returns a SYS\_REFCURSOR through an output parameter. The cursor should return run number and run size as output for a specified input start date and end date range.

Provide Test Case:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case name** | **Start date** | **End date** | **Expected result** | **Actual result** |
| Happy path | 12-Sep-2015 | 19-Sep-2015 | 10 rows returned | 10 |
| All runs available | 10-Mar-2017 | 25-Mar-2017 | 12 rows returned | 12 |
| No runs available | 10-Mar-2014 | 25-Mar-2018 | 0 rows returned | 1 |

### list\_reservations\_starting\_pp which returns a SYS\_REFCURSOR through an output parameter. The cursor should list all reservation ids, start date and end date as output for a specified input start date sorted by reservation id.

Provide Test Case:

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case name** | **Start Date** | **Expected result** | **Actual result** |
| Happy path | 01-Jan-2017 | 39 rows returned | 39 |
| Before first res | 01-Jan-2014 | 65 rows returned | 65 |
| After last res | 01-Jan-2018 | 0 rows returned | 1 |

### list\_reservations\_ending\_pp which returns a SYS\_REFCURSOR through an output parameter. The cursor should list all reservation ids, start date and end date as output for a specified input end date sorted by reservation id.

Provide Test Case:

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case name** | **End Date** | **Expected result** | **Actual result** |
| Happy path | 01-Jan-2017 | 48 rows returned | 48 |
| Before first res | 01-Jan-2014 | 65 rows returned | 65 |
| After last res | 01-Jan-2018 | 0 rows returned | 1 |

### list\_active\_reservations\_pp which has an optional owner number parameter and returns a SYS\_REFCURSOR through an output parameter. The cursor should list of all active reservations if the owner number is not specified and the active reservation for only that owner if it is specified. The cursor should list the reservation number, the owner name, the pet number and name, the run number and the start and end date of the reservation. For test purposes, have an optional date parameter that defaults to “today”, but can be set for your automated test cases.

Provide Test Case:

THESE TEST CASES USE SYSDATE, SO WILL NOT BE ACCURATE WHEN YOU GRADE.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test case name** | **Owner number** | **Start Date** | **Expected result** | **Actual result** |
| Both owner number and date | 10 | 01-Jan-2017 |  |  |
| Only owner number | 10 | null | 0 rows |  |
| Only date | Null | 01-Jan-2018 |  |  |
| No paramters | null | null | 0 rows |  |

### Overload list\_active\_reservations\_pp() above to return the same output format, but provide a specified start and end date range as input (in addition to the optional owner number).

Provide Test Case:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case name** | **Owner number** | **Start Date** | **End Date** | **Expected result** | **Actual result** |
| Both owner number and date | 10 | 01-Jan-2017 |  |  |  |
| Only owner number | 10 | null |  | 0 rows |  |
| Only date | Null | 01-Jan-2018 |  |  |  |
| No paramters | null | null |  | 0 rows |  |

### list\_check\_vaccinations\_pp that returns a SYS\_REFCURSOR of vaccinations to check for a given reservation number and pet number. The output is a list of unvalidated vaccination names (null is all vaccinations are up-to-date and validated, or input in invalid).

Provide Test Case:

Test and Implementation Notes:

The List procedures are just a lightweight wrapper on selects.  It has no error handling. Your caller has to handle any exceptions or failures.

As you’re writing automated tests, you can modify, test, and rollback for your test.  Assume your tests have the same starting point as the HVK database provided from Moodle.

### Write a test function/procedure for each procedure (i.e. test\_list\_available\_runs\_pp() to test list\_available\_runs()) test individually, then make each test function private and combine in one public procedure test\_package\_pp(). Hence, running test\_package\_pp() will run all the test procedures in the package. Besides, test\_package\_pp(), all other test procedures/functions will be private.

As tests are run, it should indicate a test name/description, pass/fail.

## If any tests fail, it should indicate (via DBMS\_OUTPUT):

Test name/number that failed

Expected vs Actual results (only output this on failure!)

## At the end of the test\_package\_pp(), 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.

**PL/SQL Specification:**

**CREATE OR REPLACE PACKAGE hvk\_tables\_list\_package AS**

**--PART A**

**PROCEDURE list\_available\_runs\_pp (**

**pv\_start\_date\_i hvk\_reservation.reservation\_start\_date%TYPE,**

**pv\_end\_date\_i hvk\_reservation.reservation\_end\_date%TYPE,**

**pv\_cursor\_o OUT SYS\_REFCURSOR**

**);**

**--PART B**

**PROCEDURE list\_reservations\_starting\_pp (**

**pv\_start\_date\_i hvk\_reservation.reservation\_start\_date%TYPE,**

**pv\_cursor\_o OUT SYS\_REFCURSOR**

**);**

**--PART C**

**PROCEDURE list\_reservations\_ending\_pp (**

**pv\_end\_date\_i hvk\_reservation.reservation\_end\_date%TYPE,**

**pv\_cursor\_o OUT SYS\_REFCURSOR**

**);**

**--PART D**

**PROCEDURE list\_active\_reservations\_pp (**

**pv\_owner\_number\_i hvk\_owner.owner\_number%TYPE DEFAULT null,**

**pv\_date\_i hvk\_reservation.reservation\_start\_date%TYPE DEFAULT SYSDATE,**

**pv\_cursor\_o OUT SYS\_REFCURSOR**

**);**

**--PART E**

**PROCEDURE list\_active\_reservations\_pp (**

**pv\_owner\_number\_i hvk\_owner.owner\_number%TYPE DEFAULT null,**

**pv\_start\_date\_i hvk\_reservation.reservation\_start\_date%TYPE,**

**pv\_end\_date\_i hvk\_reservation.reservation\_end\_date%TYPE,**

**pv\_cursor\_o OUT SYS\_REFCURSOR**

**);**

**--PART F**

**PROCEDURE list\_check\_vaccinations\_pp (**

**pv\_pet\_number\_i hvk\_pet.pet\_number%TYPE,**

**pv\_res\_number\_i hvk\_reservation.reservation\_number%TYPE,**

**pv\_cursor\_o OUT SYS\_REFCURSOR**

**);**

**--Tests**

**PROCEDURE test\_all\_procedures\_pp (**

**lv\_number\_passed\_o out number**

**);**

**END hvk\_tables\_list\_package;**

**Body:**

**create or replace PACKAGE BODY hvk\_tables\_list\_package AS**

**--PART A**

**PROCEDURE list\_available\_runs\_pp (**

**pv\_start\_date\_i hvk\_reservation.reservation\_start\_date%TYPE,**

**pv\_end\_date\_i hvk\_reservation.reservation\_end\_date%TYPE,**

**pv\_cursor\_o OUT SYS\_REFCURSOR**

**) AS**

**BEGIN**

**OPEN pv\_cursor\_o FOR**

**SELECT run\_number, run\_size FROM hvk\_run**

**MINUS**

**SELECT DISTINCT run\_number, run\_size**

**FROM hvk\_run r, hvk\_pet\_reservation pr**

**LEFT JOIN hvk\_reservation res**

**ON res.RESERVATION\_NUMBER = pr.RES\_RESERVATION\_NUMBER**

**WHERE res.reservation\_start\_date >= pv\_start\_date\_i**

**AND res.reservation\_end\_date <= pv\_end\_date\_i**

**AND pr.RUN\_RUN\_NUMBER = r.RUN\_NUMBER**

**ORDER BY run\_number;**

**END list\_available\_runs\_pp;**

**--TEST PART A**

**PROCEDURE test\_list\_available\_runs (**

**pv\_num\_passed\_o OUT number,**

**pv\_num\_run\_o OUT number**

**) AS**

**lv\_num\_total number := 0;**

**lv\_num\_pass number := 0;**

**type numRuns IS VARRAY(3) OF NUMBER;**

**type sizeRuns IS VARRAY(3) OF CHAR(1 BYTE);**

**type rowCount IS VARRAY(3) OF NUMBER;**

**type testDate IS VARRAY(3) OF DATE;**

**lv\_runNums\_exp numRuns;**

**lv\_runSize\_exp sizeRuns;**

**lv\_numRow\_exp rowCount;**

**lv\_start\_dates testDate;**

**lv\_end\_dates testDate;**

**lcur\_test SYS\_REFCURSOR;**

**lv\_runNum\_act number;**

**lv\_runSize\_act CHAR(1 byte);**

**lv\_numRow\_act number;**

**BEGIN**

**DBMS\_OUTPUT.PUT\_LINE('------TEST LIST\_AVAILABLE\_RUNS-----');**

**lv\_runNums\_exp := numRuns(1, 1, null);**

**lv\_runSize\_exp := sizeRuns('R', 'R', null);**

**lv\_numRow\_exp := rowCount(10, 12, 0);**

**lv\_start\_dates := testDate(to\_date('12-Sep-15', 'dd-MM-yy'), to\_date('10-Mar-17', 'dd-MM-yy'), to\_date('01-Jan-14', 'dd-MM-yy'));**

**lv\_end\_dates := testDate(to\_date('19-Sep-15', 'dd-MM-yy'), to\_date('25-Mar-17', 'dd-MM-yy'), to\_date('01-Jan-18', 'dd-MM-yy'));**

**--Make this change so that my test are accurate. Changes back after.**

**UPDATE hvk\_pet\_reservation**

**SET run\_run\_number = 22**

**WHERE pet\_res\_number = 200;**

**commit;**

**FOR x in 1..lv\_runNums\_exp.count LOOP**

**lv\_runNum\_act := null;**

**lv\_runSize\_act := null;**

**lv\_numRow\_act := 0;**

**list\_available\_runs\_pp(lv\_start\_dates(x), lv\_end\_dates(x), lcur\_test);**

**FETCH lcur\_test into lv\_runNum\_act, lv\_runSize\_act;**

**IF lcur\_test%FOUND THEN lv\_numRow\_act := lv\_numRow\_act + 1; END IF;**

**IF NVL(lv\_runNum\_act, -1) = NVL(lv\_runNums\_exp(x), -1)**

**AND NVL(lv\_runSize\_act, 'Z') = NVL(lv\_runSize\_exp(x), 'Z') THEN**

**lv\_num\_pass := lv\_num\_pass + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Test ' || ((x\*2)-1) || ' passed.');**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('---TEST CASE ' || ((x\*2)-1) || ' FAILED. --');**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN NUMBER: ' || lv\_runNum\_act || ' EXPECTED NUMBER: ' || lv\_runNums\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_runSize\_act || ' EXPECTED SIZE: ' || lv\_runSize\_exp(x));**

**END IF;**

**lv\_num\_total := lv\_num\_total + 1;**

**LOOP**

**FETCH lcur\_test into lv\_runNum\_act, lv\_runSize\_act;**

**IF lcur\_test%FOUND THEN lv\_numRow\_act := lv\_numRow\_act + 1; END IF;**

**EXIT WHEN lcur\_test%NOTFOUND;**

**END LOOP;**

**IF lv\_numRow\_act = lv\_numRow\_exp(x) THEN**

**lv\_num\_pass := lv\_num\_pass + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Test ' || (x\*2) || ' passed.');**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('---TEST CASE ' || (x\*2) || ' FAILED. --');**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL ROW COUNT: ' || lv\_numRow\_act || ' EXPECTED ROW COUNT: ' || lv\_numRow\_exp(x));**

**END IF;**

**lv\_num\_total := lv\_num\_total + 1;**

**END LOOP;**

**--Change back to original data**

**UPDATE hvk\_pet\_reservation**

**SET run\_run\_number = null**

**WHERE pet\_res\_number = 200;**

**commit;**

**pv\_num\_passed\_o := lv\_num\_pass;**

**pv\_num\_run\_o := lv\_num\_total;**

**END;**

**--PART B**

**PROCEDURE list\_reservations\_starting\_pp (**

**pv\_start\_date\_i hvk\_reservation.reservation\_start\_date%TYPE,**

**pv\_cursor\_o OUT SYS\_REFCURSOR**

**) AS**

**BEGIN**

**OPEN pv\_cursor\_o FOR**

**SELECT r.reservation\_number, r.reservation\_start\_date, r.reservation\_end\_date**

**FROM hvk\_reservation r**

**WHERE r.reservation\_start\_date >= pv\_start\_date\_i**

**ORDER BY r.reservation\_number;**

**END list\_reservations\_starting\_pp;**

**--TEST PART B**

**PROCEDURE test\_list\_res\_starting (**

**pv\_num\_passed\_o OUT number,**

**pv\_num\_run\_o OUT number**

**) AS**

**lv\_num\_total number := 0;**

**lv\_num\_pass number := 0;**

**type resNum IS VARRAY(3) OF NUMBER;**

**type resDate IS VARRAY(3) OF DATE;**

**lv\_resNum\_exp resNum;**

**lv\_numRow\_exp resNum;**

**lv\_sDate\_exp resDate;**

**lv\_eDate\_exp resDate;**

**lv\_start\_dates resDate;**

**lcur\_test SYS\_REFCURSOR;**

**lv\_resNum\_act number;**

**lv\_sDate\_act DATE;**

**lv\_eDate\_act DATE;**

**lv\_numRow\_act number;**

**lv\_date date := null;**

**BEGIN**

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

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

**DBMS\_OUTPUT.PUT\_LINE('------TEST LIST\_RESERVATIONS\_STARTING-----');**

**lv\_resNum\_exp := resNum(122, 100, null);**

**lv\_sDate\_exp := resDate(to\_date('01-Jan-17', 'dd-MM-yy'), to\_date('12-Sep-15', 'dd-MM-yy'), null);**

**lv\_eDate\_exp := resDate(to\_date('05-Jan-17', 'dd-MM-yy'), to\_date('19-Sep-15', 'dd-MM-yy'), null);**

**lv\_start\_dates := resDate(to\_date('01-Jan-17', 'dd-MM-yy'), to\_date('01-Jan-14', 'dd-MM-yy'), to\_date('01-Jan-18', 'dd-MM-yy'));**

**lv\_numRow\_exp := resNum(39, 65, 0);**

**FOR x in 1..lv\_resNum\_exp.count LOOP**

**lv\_resNum\_act := null;**

**lv\_sDate\_act := null;**

**lv\_eDate\_act := null;**

**lv\_numRow\_act := 0;**

**lv\_numRow\_act := 0;**

**list\_reservations\_starting\_pp(lv\_start\_dates(x), lcur\_test);**

**FETCH lcur\_test into lv\_resNum\_act, lv\_sDate\_act, lv\_eDate\_act;**

**IF lcur\_test%FOUND THEN lv\_numRow\_act := lv\_numRow\_act + 1; END IF;**

**IF nvl(lv\_resNum\_act, 0) = nvl(lv\_resNum\_exp(x), 0)**

**AND nvl(lv\_sDate\_act, to\_date('01-01-01','dd-mm-yy')) = nvl(lv\_sDate\_exp(x), to\_date('01-01-01','dd-mm-yy'))**

**AND nvl(lv\_eDate\_act, to\_date('01-01-01','dd-mm-yy')) = nvl(lv\_eDate\_exp(x), to\_date('01-01-01','dd-mm-yy'))**

**THEN**

**lv\_num\_pass := lv\_num\_pass + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Test ' || ((x\*2)-1) || ' passed.');**

**ELSE**

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

**DBMS\_OUTPUT.PUT\_LINE('---TEST CASE ' || ((x\*2)-1) || ' FAILED. --');**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN NUMBER: ' || lv\_resNum\_act || ' EXPECTED NUMBER: ' || lv\_resNum\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_sDate\_act || ' EXPECTED NUMBER: ' || lv\_sDate\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_eDate\_act || ' EXPECTED NUMBER: ' || lv\_eDate\_exp(x));**

**END IF;**

**lv\_num\_total := lv\_num\_total + 1;**

**LOOP**

**FETCH lcur\_test into lv\_resNum\_act, lv\_sDate\_act, lv\_eDate\_act;**

**IF lcur\_test%FOUND THEN lv\_numRow\_act := lv\_numRow\_act + 1; END IF;**

**EXIT WHEN lcur\_test%NOTFOUND;**

**END LOOP;**

**IF lv\_numRow\_act = lv\_numRow\_exp(x) THEN**

**lv\_num\_pass := lv\_num\_pass + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Test ' || (x\*2) || ' passed.');**

**ELSE**

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

**DBMS\_OUTPUT.PUT\_LINE('---TEST CASE ' || (x\*2) || ' FAILED. --');**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL ROW COUNT: ' || lv\_numRow\_act || ' EXPECTED ROW COUNT: ' || lv\_numRow\_exp(x));**

**END IF;**

**lv\_num\_total := lv\_num\_total + 1;**

**END LOOP;**

**pv\_num\_passed\_o := lv\_num\_pass;**

**pv\_num\_run\_o := lv\_num\_total;**

**END;**

**--PART C**

**PROCEDURE list\_reservations\_ending\_pp (**

**pv\_end\_date\_i hvk\_reservation.reservation\_end\_date%TYPE,**

**pv\_cursor\_o OUT SYS\_REFCURSOR**

**) AS**

**BEGIN**

**OPEN pv\_cursor\_o FOR**

**SELECT r.reservation\_number, r.reservation\_start\_date, r.reservation\_end\_date**

**FROM hvk\_reservation r**

**WHERE r.reservation\_end\_date >= pv\_end\_date\_i**

**ORDER BY r.reservation\_number;**

**END list\_reservations\_ending\_pp;**

**--TEST PART C**

**PROCEDURE test\_list\_res\_ending (**

**pv\_num\_passed\_o OUT number,**

**pv\_num\_run\_o OUT number**

**) AS**

**lv\_num\_total number := 0;**

**lv\_num\_pass number := 0;**

**type resNum IS VARRAY(3) OF NUMBER;**

**type resDate IS VARRAY(3) OF DATE;**

**lv\_resNum\_exp resNum;**

**lv\_numRow\_exp resNum;**

**lv\_sDate\_exp resDate;**

**lv\_eDate\_exp resDate;**

**lv\_end\_dates resDate;**

**lcur\_test SYS\_REFCURSOR;**

**lv\_resNum\_act number;**

**lv\_sDate\_act DATE;**

**lv\_eDate\_act DATE;**

**lv\_numRow\_act number;**

**lv\_date date := null;**

**BEGIN**

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

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

**DBMS\_OUTPUT.PUT\_LINE('------TEST LIST\_RESERVATIONS\_ENDING-----');**

**lv\_resNum\_exp := resNum(122, 100, null);**

**lv\_sDate\_exp := resDate(to\_date('01-Jan-17', 'dd-MM-yy'), to\_date('12-Sep-15', 'dd-MM-yy'), null);**

**lv\_eDate\_exp := resDate(to\_date('05-Jan-17', 'dd-MM-yy'), to\_date('19-Sep-15', 'dd-MM-yy'), null);**

**lv\_end\_dates := resDate(to\_date('01-Jan-17', 'dd-MM-yy'), to\_date('01-Jan-14', 'dd-MM-yy'), to\_date('01-Jan-18', 'dd-MM-yy'));**

**lv\_numRow\_exp := resNum(48, 65, 0);**

**FOR x in 1..lv\_resNum\_exp.count LOOP**

**lv\_resNum\_act := null;**

**lv\_sDate\_act := null;**

**lv\_eDate\_act := null;**

**lv\_numRow\_act := 0;**

**list\_reservations\_ending\_pp(lv\_end\_dates(x), lcur\_test);**

**FETCH lcur\_test into lv\_resNum\_act, lv\_sDate\_act, lv\_eDate\_act;**

**IF lcur\_test%FOUND THEN lv\_numRow\_act := lv\_numRow\_act + 1; END IF;**

**IF nvl(lv\_resNum\_act, 0) = nvl(lv\_resNum\_exp(x), 0)**

**AND nvl(lv\_sDate\_act, to\_date('01-01-01','dd-mm-yy')) = nvl(lv\_sDate\_exp(x), to\_date('01-01-01','dd-mm-yy'))**

**AND nvl(lv\_eDate\_act, to\_date('01-01-01','dd-mm-yy')) = nvl(lv\_eDate\_exp(x), to\_date('01-01-01','dd-mm-yy'))**

**THEN**

**lv\_num\_pass := lv\_num\_pass + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Test ' || ((x\*2)-1) || ' passed.');**

**ELSE**

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

**DBMS\_OUTPUT.PUT\_LINE('---TEST CASE ' || ((x\*2)-1) || ' FAILED. --');**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN NUMBER: ' || lv\_resNum\_act || ' EXPECTED NUMBER: ' || lv\_resNum\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_sDate\_act || ' EXPECTED NUMBER: ' || lv\_sDate\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_eDate\_act || ' EXPECTED NUMBER: ' || lv\_eDate\_exp(x));**

**END IF;**

**lv\_num\_total := lv\_num\_total + 1;**

**LOOP**

**FETCH lcur\_test into lv\_resNum\_act, lv\_sDate\_act, lv\_eDate\_act;**

**IF lcur\_test%FOUND THEN lv\_numRow\_act := lv\_numRow\_act + 1; END IF;**

**EXIT WHEN lcur\_test%NOTFOUND;**

**END LOOP;**

**IF lv\_numRow\_act = lv\_numRow\_exp(x) THEN**

**lv\_num\_pass := lv\_num\_pass + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Test ' || (x\*2) || ' passed.');**

**ELSE**

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

**DBMS\_OUTPUT.PUT\_LINE('---TEST CASE ' || (x\*2) || ' FAILED. --');**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL ROW COUNT: ' || lv\_numRow\_act || ' EXPECTED ROW COUNT: ' || lv\_numRow\_exp(x));**

**END IF;**

**lv\_num\_total := lv\_num\_total + 1;**

**END LOOP;**

**pv\_num\_passed\_o := lv\_num\_pass;**

**pv\_num\_run\_o := lv\_num\_total;**

**END;**

**--PART D**

**PROCEDURE list\_active\_reservations\_pp (**

**pv\_owner\_number\_i hvk\_owner.owner\_number%TYPE DEFAULT null,**

**pv\_date\_i hvk\_reservation.reservation\_start\_date%TYPE DEFAULT SYSDATE,**

**pv\_cursor\_o OUT SYS\_REFCURSOR**

**) AS**

**BEGIN**

**OPEN pv\_cursor\_o FOR**

**SELECT r.reservation\_number, o.owner\_first\_name, o.owner\_last\_name,**

**p.pet\_number, p.pet\_name, pr.run\_run\_number,**

**r.reservation\_start\_date, r.reservation\_end\_date**

**FROM hvk\_owner o, hvk\_pet p, hvk\_pet\_reservation pr, hvk\_reservation r**

**WHERE r.reservation\_start\_date <= pv\_date\_i**

**AND r.reservation\_end\_Date >= pv\_date\_i**

**AND (pv\_owner\_number\_i is null**

**OR pv\_owner\_number\_i = o.owner\_number)**

**AND o.owner\_number = p.own\_owner\_number**

**AND p.pet\_number = pr.pet\_pet\_number**

**AND pr.res\_reservation\_number = r.reservation\_number**

**ORDER BY r.reservation\_number;**

**END list\_active\_reservations\_pp;**

**--TEST PART D**

**PROCEDURE test\_list\_active\_res (**

**pv\_num\_passed\_o OUT number,**

**pv\_num\_run\_o OUT number**

**) AS**

**lv\_num\_total number := 0;**

**lv\_num\_pass number := 0;**

**type pk IS VARRAY(4) OF NUMBER;**

**type names IS VARRAY(4) OF VARCHAR2(25);**

**type resDate IS VARRAY(4) OF DATE;**

**--Expected Data**

**lv\_resNum\_exp pk;**

**lv\_fName\_exp names;**

**lv\_lName\_exp names;**

**lv\_petNum\_exp pk;**

**lv\_pName\_exp names;**

**lv\_runNum\_exp pk;**

**lv\_sDate\_exp resDate;**

**lv\_eDate\_exp resDate;**

**lv\_numRows\_exp pk;**

**--Passed data**

**lv\_ownNums pk;**

**lv\_sDates resDate;**

**lcur\_test SYS\_REFCURSOR;**

**lv\_resNum\_act number;**

**lv\_fName\_act VARCHAR2(25);**

**lv\_lName\_act VARCHAR2(25);**

**lv\_petNum\_act number;**

**lv\_pName\_act VARCHAR2(25);**

**lv\_runNum\_act number;**

**lv\_sDate\_act DATE;**

**lv\_eDate\_act DATE;**

**lv\_numRow\_act number;**

**lv\_date date := null;**

**BEGIN**

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

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

**DBMS\_OUTPUT.PUT\_LINE('------TEST LIST\_ACTIVE\_RESERVATIONS-----');**

**--MAKE UP DATA**

**lv\_resNum\_exp := pk(630, null, 630, null);**

**lv\_fName\_exp := names('Barb B.', null, 'Barb B.', null);**

**lv\_lName\_exp := names('Que', null, 'Que', null);**

**lv\_petNum\_exp := pk(33, null, 33, null);**

**lv\_pName\_exp := names('Willie', null, 'Willie', null);**

**lv\_runNum\_exp := pk(13, null, 13, null);**

**lv\_sDate\_exp := resDate(to\_date('05-Mar-17','dd-MM-yy'), null, to\_date('05-Mar-17','dd-MM-yy'), null);**

**lv\_eDate\_exp := resDate(to\_date('13-Mar-17','dd-MM-yy'), null, to\_date('13-Mar-17','dd-MM-yy'), null);**

**lv\_numRows\_exp := pk(1, 0, 1, 0);**

**lv\_ownNums := pk(18, 18, null, null);**

**lv\_sDates := resDate(to\_date('10-Mar-17','dd-MM-yy'), null, to\_date('10-Mar-17','dd-MM-yy'), null);**

**FOR x in 1..lv\_resNum\_exp.count LOOP**

**lv\_resNum\_act := null;**

**lv\_fName\_act := null;**

**lv\_lName\_act := null;**

**lv\_petNum\_act := null;**

**lv\_pName\_act := null;**

**lv\_runNum\_act := null;**

**lv\_sDate\_act := null;**

**lv\_eDate\_act := null;**

**lv\_numRow\_act := 0;**

**list\_active\_reservations\_pp(lv\_ownNums(x), lv\_sDates(x), lcur\_test);**

**FETCH lcur\_test into lv\_resNum\_act, lv\_fName\_act, lv\_lName\_act, lv\_petNum\_act, lv\_pName\_act, lv\_runNum\_act, lv\_sDate\_act, lv\_eDate\_act;**

**IF lcur\_test%FOUND THEN lv\_numRow\_act := lv\_numRow\_act + 1; END IF;**

**IF nvl(lv\_resNum\_act, -1) = nvl(lv\_resNum\_exp(x), -1) AND nvl(lv\_fName\_act, '~') = nvl(lv\_fName\_exp(x), '~')**

**AND nvl(lv\_lName\_act, '~') = nvl(lv\_lName\_exp(x), '~') AND nvl(lv\_petNum\_act, -1) = nvl(lv\_petNum\_exp(x), -1)**

**AND nvl(lv\_pName\_act, '~') = nvl(lv\_pName\_exp(x), '~') AND nvl(lv\_runNum\_act, -1) = nvl(lv\_runNum\_exp(x), -1)**

**AND nvl(lv\_sDate\_act, to\_date('01-01-01','dd-mm-yy')) = nvl(lv\_sDate\_exp(x), to\_date('01-01-01','dd-mm-yy'))**

**AND nvl(lv\_eDate\_act, to\_date('01-01-01','dd-mm-yy')) = nvl(lv\_eDate\_exp(x), to\_date('01-01-01','dd-mm-yy'))**

**THEN**

**lv\_num\_pass := lv\_num\_pass + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Test ' || ((x\*2)-1) || ' passed.');**

**ELSE**

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

**DBMS\_OUTPUT.PUT\_LINE('---TEST CASE ' || ((x\*2)-1) || ' FAILED. --');**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN NUMBER: ' || lv\_resNum\_act || ' EXPECTED NUMBER: ' || lv\_resNum\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_sDate\_act || ' EXPECTED NUMBER: ' || lv\_sDate\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_eDate\_act || ' EXPECTED NUMBER: ' || lv\_eDate\_exp(x));**

**END IF;**

**lv\_num\_total := lv\_num\_total + 1;**

**LOOP**

**FETCH lcur\_test into lv\_resNum\_act, lv\_fName\_act, lv\_lName\_act, lv\_petNum\_act, lv\_pName\_act, lv\_runNum\_act, lv\_sDate\_act, lv\_eDate\_act;**

**IF lcur\_test%FOUND THEN lv\_numRow\_act := lv\_numRow\_act + 1; END IF;**

**EXIT WHEN lcur\_test%NOTFOUND;**

**END LOOP;**

**IF lv\_numRow\_act = lv\_numRows\_exp(x) THEN**

**lv\_num\_pass := lv\_num\_pass + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Test ' || (x\*2) || ' passed.');**

**ELSE**

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

**DBMS\_OUTPUT.PUT\_LINE('---TEST CASE ' || (x\*2) || ' FAILED. --');**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL ROW COUNT: ' || lv\_numRow\_act || ' EXPECTED ROW COUNT: ' || lv\_numRows\_exp(x));**

**END IF;**

**lv\_num\_total := lv\_num\_total + 1;**

**END LOOP;**

**pv\_num\_passed\_o := lv\_num\_pass;**

**pv\_num\_run\_o := lv\_num\_total;**

**END;**

**--PART E**

**PROCEDURE list\_active\_reservations\_pp (**

**pv\_cursor\_o OUT SYS\_REFCURSOR,**

**pv\_owner\_number\_i hvk\_owner.owner\_number%TYPE DEFAULT null,**

**pv\_start\_date\_i hvk\_reservation.reservation\_start\_date%TYPE DEFAULT SYSDATE,**

**pv\_end\_date\_i hvk\_reservation.reservation\_end\_date%TYPE DEFAULT SYSDATE**

**) AS**

**lv\_sDate DATE;**

**lv\_eDate DATE;**

**BEGIN**

**IF pv\_start\_date\_i IS NULL THEN**

**lv\_sDate := sysdate;**

**ELSE**

**lv\_sDate := pv\_start\_date\_i;**

**END IF;**

**IF pv\_end\_date\_i IS NULL THEN**

**lv\_eDate := sysdate;**

**ELSE**

**lv\_eDate := pv\_end\_date\_i;**

**END IF;**

**OPEN pv\_cursor\_o FOR**

**SELECT r.reservation\_number, o.owner\_first\_name, o.owner\_last\_name,**

**p.pet\_number, p.pet\_name, pr.run\_run\_number,**

**r.reservation\_start\_date, r.reservation\_end\_date**

**FROM hvk\_owner o, hvk\_pet p, hvk\_pet\_reservation pr, hvk\_reservation r**

**WHERE r.reservation\_start\_date >= lv\_sDate**

**AND r.reservation\_end\_date <= lv\_eDate**

**AND (pv\_owner\_number\_i is null**

**OR pv\_owner\_number\_i = o.owner\_number)**

**AND o.owner\_number = p.own\_owner\_number**

**AND p.pet\_number = pr.pet\_pet\_number**

**AND pr.res\_reservation\_number = r.reservation\_number**

**ORDER BY r.reservation\_number;**

**END list\_active\_reservations\_pp;**

**--TEST PART E**

**PROCEDURE test\_list\_active\_res\_2 (**

**pv\_num\_passed\_o OUT number,**

**pv\_num\_run\_o OUT number**

**) AS**

**lv\_num\_total number := 0;**

**lv\_num\_pass number := 0;**

**type pk IS VARRAY(8) OF NUMBER;**

**type names IS VARRAY(8) OF VARCHAR2(25);**

**type resDate IS VARRAY(8) OF DATE;**

**--Expected Data**

**lv\_resNum\_exp pk;**

**lv\_fName\_exp names;**

**lv\_lName\_exp names;**

**lv\_petNum\_exp pk;**

**lv\_pName\_exp names;**

**lv\_runNum\_exp pk;**

**lv\_sDate\_exp resDate;**

**lv\_eDate\_exp resDate;**

**lv\_numRows\_exp pk;**

**--Passed data**

**lv\_ownNums pk;**

**lv\_sDates resDate;**

**lv\_eDates resDate;**

**lcur\_test SYS\_REFCURSOR;**

**lv\_resNum\_act number;**

**lv\_fName\_act VARCHAR2(25);**

**lv\_lName\_act VARCHAR2(25);**

**lv\_petNum\_act number;**

**lv\_pName\_act VARCHAR2(25);**

**lv\_runNum\_act number;**

**lv\_sDate\_act DATE;**

**lv\_eDate\_act DATE;**

**lv\_numRow\_act number;**

**lv\_date date := null;**

**BEGIN**

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

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

**DBMS\_OUTPUT.PUT\_LINE('------TEST LIST\_ACTIVE\_RESERVATIONS-----');**

**lv\_resNum\_exp := pk(630, 630, null, null, 625, 625, 721, null);**

**lv\_fName\_exp := names('Barb B.', 'Barb B.', null, null, 'Ella', 'Ella', 'Mike', null);**

**lv\_lName\_exp := names('Que', 'Que', null, null, 'Mentary', 'Mentary', 'O''Phone', null);**

**lv\_petNum\_exp := pk(33, 33, null, null, 20, 20, 3, null);**

**lv\_pName\_exp := names('Willie', 'Willie', null, null, 'Poppy', 'Poppy', 'Jasper', null);**

**lv\_runNum\_exp := pk(13, 13, null, null, null, null, 27, null);**

**lv\_sDate\_exp := resDate(to\_date('05-Mar-17', 'dd-MM-yy'), to\_date('05-Mar-17', 'dd-MM-yy'),**

**null, null, to\_date('15-Mar-17','dd-MM-yy'), to\_date('15-Mar-17','dd-MM-yy'), to\_date('05-Apr-17','dd-MM-yy'), null);**

**lv\_eDate\_exp := resDate(to\_date('13-Mar-17', 'dd-MM-yy'), to\_date('13-Mar-17', 'dd-MM-yy'),**

**null, null, to\_date('20-Mar-17','dd-MM-yy'), to\_date('20-Mar-17','dd-MM-yy'), to\_date('09-Apr-17','dd-MM-yy'), null);**

**lv\_numRows\_exp := pk(2, 2, 0, 0, 6, 5, 1, 0);**

**lv\_ownNums := pk(18, 18, 18, 18, null, null, null, null);**

**lv\_sDates := resDate(to\_date('05-Mar-17','dd-MM-yy'), to\_date('05-Mar-17','dd-MM-yy'), NULL, NULL, to\_date('05-Mar-17','dd-MM-yy'), to\_date('05-Mar-17','dd-MM-yy'), NULL, NULL);**

**lv\_eDates := resDate(to\_date('15-Apr-17','dd-MM-yy'), NULL, to\_date('15-Apr-17','dd-MM-yy'), NULL, to\_date('15-Apr-17','dd-MM-yy'), NULL, to\_date('15-Apr-17','dd-MM-yy'), NULL);**

**FOR x in 1..lv\_resNum\_exp.count LOOP**

**lv\_resNum\_act := null;**

**lv\_fName\_act := null;**

**lv\_lName\_act := null;**

**lv\_petNum\_act := null;**

**lv\_pName\_act := null;**

**lv\_runNum\_act := null;**

**lv\_sDate\_act := null;**

**lv\_eDate\_act := null;**

**lv\_numRow\_act := 0;**

**list\_active\_reservations\_pp(lcur\_test, lv\_ownNums(x), lv\_sDates(x), lv\_eDates(x));**

**FETCH lcur\_test into lv\_resNum\_act, lv\_fName\_act, lv\_lName\_act, lv\_petNum\_act, lv\_pName\_act, lv\_runNum\_act, lv\_sDate\_act, lv\_eDate\_act;**

**IF lcur\_test%FOUND THEN lv\_numRow\_act := lv\_numRow\_act + 1; END IF;**

**IF nvl(lv\_resNum\_act, -1) = nvl(lv\_resNum\_exp(x), -1) AND nvl(lv\_fName\_act, '~') = nvl(lv\_fName\_exp(x), '~')**

**AND nvl(lv\_lName\_act, '~') = nvl(lv\_lName\_exp(x), '~') AND nvl(lv\_petNum\_act, -1) = nvl(lv\_petNum\_exp(x), -1)**

**AND nvl(lv\_pName\_act, '~') = nvl(lv\_pName\_exp(x), '~') AND nvl(lv\_runNum\_act, -1) = nvl(lv\_runNum\_exp(x), -1)**

**AND nvl(lv\_sDate\_act, to\_date('01-01-01','dd-mm-yy')) = nvl(lv\_sDate\_exp(x), to\_date('01-01-01','dd-mm-yy'))**

**AND nvl(lv\_eDate\_act, to\_date('01-01-01','dd-mm-yy')) = nvl(lv\_eDate\_exp(x), to\_date('01-01-01','dd-mm-yy'))**

**THEN**

**lv\_num\_pass := lv\_num\_pass + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Test ' || ((x\*2)-1) || ' passed.');**

**ELSE**

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

**DBMS\_OUTPUT.PUT\_LINE('---TEST CASE ' || ((x\*2)-1) || ' FAILED. --');**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN NUMBER: ' || lv\_resNum\_act || ' EXPECTED NUMBER: ' || lv\_resNum\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_fName\_act || ' EXPECTED NUMBER: ' || lv\_fName\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_lName\_act || ' EXPECTED NUMBER: ' || lv\_lName\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_petNum\_act || ' EXPECTED NUMBER: ' || lv\_petNum\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_pName\_act || ' EXPECTED NUMBER: ' || lv\_pName\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_runNum\_act || ' EXPECTED NUMBER: ' || lv\_runNum\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_sDate\_act || ' EXPECTED NUMBER: ' || lv\_sDate\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_eDate\_act || ' EXPECTED NUMBER: ' || lv\_eDate\_exp(x));**

**END IF;**

**lv\_num\_total := lv\_num\_total + 1;**

**LOOP**

**FETCH lcur\_test into lv\_resNum\_act, lv\_fName\_act, lv\_lName\_act, lv\_petNum\_act, lv\_pName\_act, lv\_runNum\_act, lv\_sDate\_act, lv\_eDate\_act;**

**IF lcur\_test%FOUND THEN lv\_numRow\_act := lv\_numRow\_act + 1; END IF;**

**EXIT WHEN lcur\_test%NOTFOUND;**

**END LOOP;**

**IF lv\_numRow\_act = lv\_numRows\_exp(x) THEN**

**lv\_num\_pass := lv\_num\_pass + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Test ' || (x\*2) || ' passed.');**

**ELSE**

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

**DBMS\_OUTPUT.PUT\_LINE('---TEST CASE ' || (x\*2) || ' FAILED. --');**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL ROW COUNT: ' || lv\_numRow\_act || ' EXPECTED ROW COUNT: ' || lv\_numRows\_exp(x));**

**END IF;**

**lv\_num\_total := lv\_num\_total + 1;**

**END LOOP;**

**pv\_num\_passed\_o := lv\_num\_pass;**

**pv\_num\_run\_o := lv\_num\_total;**

**END;**

**--PART F**

**PROCEDURE list\_check\_vaccinations\_pp (**

**pv\_pet\_number\_i hvk\_pet.pet\_number%TYPE,**

**pv\_res\_number\_i hvk\_reservation.reservation\_number%TYPE,**

**pv\_cursor\_o OUT SYS\_REFCURSOR**

**) AS**

**BEGIN**

**OPEN pv\_cursor\_o FOR**

**SELECT v.vaccination\_name**

**FROM hvk\_pet\_vaccination pv, hvk\_vaccination v, hvk\_reservation r**

**WHERE pv.pet\_pet\_number = pv\_pet\_number\_i**

**AND r.reservation\_number = pv\_res\_number\_i**

**AND r.reservation\_end\_date >= pv.vaccination\_expiry\_date**

**AND v.vaccination\_number = pv.vacc\_vaccination\_number**

**UNION (**

**SELECT vaccination\_name FROM hvk\_vaccination**

**MINUS**

**SELECT v.vaccination\_name**

**FROM hvk\_pet\_vaccination pv, hvk\_vaccination v**

**WHERE pv.pet\_pet\_number = pv\_pet\_number\_i**

**AND v.vaccination\_number = pv.vacc\_vaccination\_number**

**);**

**END list\_check\_vaccinations\_pp;**

**--TEST PART F**

**PROCEDURE test\_list\_check\_vacc (**

**pv\_num\_passed\_o OUT number,**

**pv\_num\_run\_o OUT number**

**) AS**

**lv\_num\_total number := 0;**

**lv\_num\_pass number := 0;**

**type numm IS VARRAY(5) OF NUMBER;**

**type vacc IS VARRAY(5) OF VARCHAR2(25);**

**--Expected Data**

**lv\_vacc\_exp vacc;**

**lv\_numRows\_exp numm;**

**--Passed data**

**lv\_petNums numm;**

**lv\_resNums numm;**

**lcur\_test SYS\_REFCURSOR;**

**lv\_vacc\_act VARCHAR2(25);**

**lv\_numRow\_act number;**

**BEGIN**

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

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

**DBMS\_OUTPUT.PUT\_LINE('------TEST LIST\_CHECK\_VACCINATIONS-----');**

**--MAKE UP DATA**

**lv\_vacc\_exp := vacc(null, 'Bordetella', 'Hepatitis');**

**lv\_numRows\_exp := numm(0, 4, 2);**

**lv\_petNums := numm(1, 6, 7);**

**lv\_resNums := numm(100, 106, 631);**

**FOR x in 1..lv\_vacc\_exp.count LOOP**

**lv\_vacc\_act := null;**

**lv\_numRow\_act := 0;**

**list\_check\_vaccinations\_pp(lv\_petNums(x), lv\_resNums(x), lcur\_test);**

**FETCH lcur\_test into lv\_vacc\_act;**

**IF lcur\_test%FOUND THEN lv\_numRow\_act := lv\_numRow\_act + 1; END IF;**

**IF nvl(lv\_vacc\_act, '~') = nvl(lv\_vacc\_exp(x), '~') THEN**

**lv\_num\_pass := lv\_num\_pass + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Test ' || ((x\*2)-1) || ' passed.');**

**ELSE**

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

**DBMS\_OUTPUT.PUT\_LINE('---TEST CASE ' || ((x\*2)-1) || ' FAILED. --');**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL VACCINATION: ' || lv\_vacc\_act || ' EXPECTED VACCINATION: ' || lv\_vacc\_exp(x));**

**END IF;**

**lv\_num\_total := lv\_num\_total + 1;**

**LOOP**

**FETCH lcur\_test into lv\_vacc\_act;**

**IF lcur\_test%FOUND THEN lv\_numRow\_act := lv\_numRow\_act + 1; END IF;**

**EXIT WHEN lcur\_test%NOTFOUND;**

**END LOOP;**

**IF lv\_numRow\_act = lv\_numRows\_exp(x) THEN**

**lv\_num\_pass := lv\_num\_pass + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Test ' || (x\*2) || ' passed.');**

**ELSE**

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

**DBMS\_OUTPUT.PUT\_LINE('---TEST CASE ' || (x\*2) || ' FAILED. --');**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL ROW COUNT: ' || lv\_numRow\_act || ' EXPECTED ROW COUNT: ' || lv\_numRows\_exp(x));**

**END IF;**

**lv\_num\_total := lv\_num\_total + 1;**

**END LOOP;**

**pv\_num\_passed\_o := lv\_num\_pass;**

**pv\_num\_run\_o := lv\_num\_total;**

**END;**

**--Test all procedures**

**PROCEDURE test\_package\_pp AS**

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

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

**lv\_temp\_passed number;**

**lv\_temp\_total number;**

**BEGIN**

**test\_list\_available\_runs(lv\_temp\_passed, lv\_temp\_total);**

**lv\_number\_passed := lv\_number\_passed + lv\_temp\_passed;**

**lv\_total := lv\_total + lv\_temp\_total;**

**test\_list\_res\_starting(lv\_temp\_passed, lv\_temp\_total);**

**lv\_number\_passed := lv\_number\_passed + lv\_temp\_passed;**

**lv\_total := lv\_total + lv\_temp\_total;**

**test\_list\_res\_ending(lv\_temp\_passed, lv\_temp\_total);**

**lv\_number\_passed := lv\_number\_passed + lv\_temp\_passed;**

**lv\_total := lv\_total + lv\_temp\_total;**

**test\_list\_active\_res(lv\_temp\_passed, lv\_temp\_total);**

**lv\_number\_passed := lv\_number\_passed + lv\_temp\_passed;**

**lv\_total := lv\_total + lv\_temp\_total;**

**test\_list\_active\_res\_2(lv\_temp\_passed, lv\_temp\_total);**

**lv\_number\_passed := lv\_number\_passed + lv\_temp\_passed;**

**lv\_total := lv\_total + lv\_temp\_total;**

**test\_list\_check\_vacc(lv\_temp\_passed, lv\_temp\_total);**

**lv\_number\_passed := lv\_number\_passed + lv\_temp\_passed;**

**lv\_total := lv\_total + lv\_temp\_total;**

**DBMS\_OUTPUT.PUT\_LINE('Tests run: ' || lv\_total);**

**DBMS\_OUTPUT.PUT\_LINE('Tests passed: ' || lv\_number\_passed || '/' || lv\_total**

**|| ' ' || (lv\_number\_passed/lv\_total \* 100));**

**DBMS\_OUTPUT.PUT\_LINE('Tests failed: ' || (lv\_total-lv\_number\_passed) || '/' || lv\_total**

**|| ' ' || (100-(lv\_number\_passed/lv\_total \* 100)));**

**END test\_package\_pp;**

**END hvk\_tables\_list\_package;**

**Test code:**

**PART A**

**--TEST PART A**

**PROCEDURE test\_list\_available\_runs (**

**pv\_num\_passed\_o OUT number,**

**pv\_num\_run\_o OUT number**

**) AS**

**lv\_num\_total number := 0;**

**lv\_num\_pass number := 0;**

**type numRuns IS VARRAY(3) OF NUMBER;**

**type sizeRuns IS VARRAY(3) OF CHAR(1 BYTE);**

**type rowCount IS VARRAY(3) OF NUMBER;**

**type testDate IS VARRAY(3) OF DATE;**

**lv\_runNums\_exp numRuns;**

**lv\_runSize\_exp sizeRuns;**

**lv\_numRow\_exp rowCount;**

**lv\_start\_dates testDate;**

**lv\_end\_dates testDate;**

**lcur\_test SYS\_REFCURSOR;**

**lv\_runNum\_act number;**

**lv\_runSize\_act CHAR(1 byte);**

**lv\_numRow\_act number;**

**BEGIN**

**DBMS\_OUTPUT.PUT\_LINE('------TEST LIST\_AVAILABLE\_RUNS-----');**

**lv\_runNums\_exp := numRuns(1, 1, null);**

**lv\_runSize\_exp := sizeRuns('R', 'R', null);**

**lv\_numRow\_exp := rowCount(10, 12, 0);**

**lv\_start\_dates := testDate(to\_date('12-Sep-15', 'dd-MM-yy'), to\_date('10-Mar-17', 'dd-MM-yy'), to\_date('01-Jan-14', 'dd-MM-yy'));**

**lv\_end\_dates := testDate(to\_date('19-Sep-15', 'dd-MM-yy'), to\_date('25-Mar-17', 'dd-MM-yy'), to\_date('01-Jan-18', 'dd-MM-yy'));**

**--Make this change so that my test are accurate. Changes back after.**

**UPDATE hvk\_pet\_reservation**

**SET run\_run\_number = 22**

**WHERE pet\_res\_number = 200;**

**commit;**

**FOR x in 1..lv\_runNums\_exp.count LOOP**

**lv\_runNum\_act := null;**

**lv\_runSize\_act := null;**

**lv\_numRow\_act := 0;**

**list\_available\_runs\_pp(lv\_start\_dates(x), lv\_end\_dates(x), lcur\_test);**

**FETCH lcur\_test into lv\_runNum\_act, lv\_runSize\_act;**

**IF lcur\_test%FOUND THEN lv\_numRow\_act := lv\_numRow\_act + 1; END IF;**

**IF NVL(lv\_runNum\_act, -1) = NVL(lv\_runNums\_exp(x), -1)**

**AND NVL(lv\_runSize\_act, 'Z') = NVL(lv\_runSize\_exp(x), 'Z') THEN**

**lv\_num\_pass := lv\_num\_pass + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Test ' || ((x\*2)-1) || ' passed.');**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('---TEST CASE ' || ((x\*2)-1) || ' FAILED. --');**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN NUMBER: ' || lv\_runNum\_act || ' EXPECTED NUMBER: ' || lv\_runNums\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_runSize\_act || ' EXPECTED SIZE: ' || lv\_runSize\_exp(x));**

**END IF;**

**lv\_num\_total := lv\_num\_total + 1;**

**LOOP**

**FETCH lcur\_test into lv\_runNum\_act, lv\_runSize\_act;**

**IF lcur\_test%FOUND THEN lv\_numRow\_act := lv\_numRow\_act + 1; END IF;**

**EXIT WHEN lcur\_test%NOTFOUND;**

**END LOOP;**

**IF lv\_numRow\_act = lv\_numRow\_exp(x) THEN**

**lv\_num\_pass := lv\_num\_pass + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Test ' || (x\*2) || ' passed.');**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('---TEST CASE ' || (x\*2) || ' FAILED. --');**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL ROW COUNT: ' || lv\_numRow\_act || ' EXPECTED ROW COUNT: ' || lv\_numRow\_exp(x));**

**END IF;**

**lv\_num\_total := lv\_num\_total + 1;**

**END LOOP;**

**--Change back to original data**

**UPDATE hvk\_pet\_reservation**

**SET run\_run\_number = null**

**WHERE pet\_res\_number = 200;**

**commit;**

**pv\_num\_passed\_o := lv\_num\_pass;**

**pv\_num\_run\_o := lv\_num\_total;**

**END;**

**PART B**

**--TEST PART B**

**PROCEDURE test\_list\_res\_starting (**

**pv\_num\_passed\_o OUT number,**

**pv\_num\_run\_o OUT number**

**) AS**

**lv\_num\_total number := 0;**

**lv\_num\_pass number := 0;**

**type resNum IS VARRAY(3) OF NUMBER;**

**type resDate IS VARRAY(3) OF DATE;**

**lv\_resNum\_exp resNum;**

**lv\_numRow\_exp resNum;**

**lv\_sDate\_exp resDate;**

**lv\_eDate\_exp resDate;**

**lv\_start\_dates resDate;**

**lcur\_test SYS\_REFCURSOR;**

**lv\_resNum\_act number;**

**lv\_sDate\_act DATE;**

**lv\_eDate\_act DATE;**

**lv\_numRow\_act number;**

**lv\_date date := null;**

**BEGIN**

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

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

**DBMS\_OUTPUT.PUT\_LINE('------TEST LIST\_RESERVATIONS\_STARTING-----');**

**lv\_resNum\_exp := resNum(122, 100, null);**

**lv\_sDate\_exp := resDate(to\_date('01-Jan-17', 'dd-MM-yy'), to\_date('12-Sep-15', 'dd-MM-yy'), null);**

**lv\_eDate\_exp := resDate(to\_date('05-Jan-17', 'dd-MM-yy'), to\_date('19-Sep-15', 'dd-MM-yy'), null);**

**lv\_start\_dates := resDate(to\_date('01-Jan-17', 'dd-MM-yy'), to\_date('01-Jan-14', 'dd-MM-yy'), to\_date('01-Jan-18', 'dd-MM-yy'));**

**lv\_numRow\_exp := resNum(39, 65, 0);**

**FOR x in 1..lv\_resNum\_exp.count LOOP**

**lv\_resNum\_act := null;**

**lv\_sDate\_act := null;**

**lv\_eDate\_act := null;**

**lv\_numRow\_act := 0;**

**lv\_numRow\_act := 0;**

**list\_reservations\_starting\_pp(lv\_start\_dates(x), lcur\_test);**

**FETCH lcur\_test into lv\_resNum\_act, lv\_sDate\_act, lv\_eDate\_act;**

**IF lcur\_test%FOUND THEN lv\_numRow\_act := lv\_numRow\_act + 1; END IF;**

**IF nvl(lv\_resNum\_act, 0) = nvl(lv\_resNum\_exp(x), 0)**

**AND nvl(lv\_sDate\_act, to\_date('01-01-01','dd-mm-yy')) = nvl(lv\_sDate\_exp(x), to\_date('01-01-01','dd-mm-yy'))**

**AND nvl(lv\_eDate\_act, to\_date('01-01-01','dd-mm-yy')) = nvl(lv\_eDate\_exp(x), to\_date('01-01-01','dd-mm-yy'))**

**THEN**

**lv\_num\_pass := lv\_num\_pass + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Test ' || ((x\*2)-1) || ' passed.');**

**ELSE**

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

**DBMS\_OUTPUT.PUT\_LINE('---TEST CASE ' || ((x\*2)-1) || ' FAILED. --');**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN NUMBER: ' || lv\_resNum\_act || ' EXPECTED NUMBER: ' || lv\_resNum\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_sDate\_act || ' EXPECTED NUMBER: ' || lv\_sDate\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_eDate\_act || ' EXPECTED NUMBER: ' || lv\_eDate\_exp(x));**

**END IF;**

**lv\_num\_total := lv\_num\_total + 1;**

**LOOP**

**FETCH lcur\_test into lv\_resNum\_act, lv\_sDate\_act, lv\_eDate\_act;**

**IF lcur\_test%FOUND THEN lv\_numRow\_act := lv\_numRow\_act + 1; END IF;**

**EXIT WHEN lcur\_test%NOTFOUND;**

**END LOOP;**

**IF lv\_numRow\_act = lv\_numRow\_exp(x) THEN**

**lv\_num\_pass := lv\_num\_pass + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Test ' || (x\*2) || ' passed.');**

**ELSE**

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

**DBMS\_OUTPUT.PUT\_LINE('---TEST CASE ' || (x\*2) || ' FAILED. --');**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL ROW COUNT: ' || lv\_numRow\_act || ' EXPECTED ROW COUNT: ' || lv\_numRow\_exp(x));**

**END IF;**

**lv\_num\_total := lv\_num\_total + 1;**

**END LOOP;**

**pv\_num\_passed\_o := lv\_num\_pass;**

**pv\_num\_run\_o := lv\_num\_total;**

**END;**

**PART C**

**PROCEDURE test\_list\_res\_ending (**

**pv\_num\_passed\_o OUT number,**

**pv\_num\_run\_o OUT number**

**) AS**

**lv\_num\_total number := 0;**

**lv\_num\_pass number := 0;**

**type resNum IS VARRAY(3) OF NUMBER;**

**type resDate IS VARRAY(3) OF DATE;**

**lv\_resNum\_exp resNum;**

**lv\_numRow\_exp resNum;**

**lv\_sDate\_exp resDate;**

**lv\_eDate\_exp resDate;**

**lv\_end\_dates resDate;**

**lcur\_test SYS\_REFCURSOR;**

**lv\_resNum\_act number;**

**lv\_sDate\_act DATE;**

**lv\_eDate\_act DATE;**

**lv\_numRow\_act number;**

**lv\_date date := null;**

**BEGIN**

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

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

**DBMS\_OUTPUT.PUT\_LINE('------TEST LIST\_RESERVATIONS\_ENDING-----');**

**lv\_resNum\_exp := resNum(122, 100, null);**

**lv\_sDate\_exp := resDate(to\_date('01-Jan-17', 'dd-MM-yy'), to\_date('12-Sep-15', 'dd-MM-yy'), null);**

**lv\_eDate\_exp := resDate(to\_date('05-Jan-17', 'dd-MM-yy'), to\_date('19-Sep-15', 'dd-MM-yy'), null);**

**lv\_end\_dates := resDate(to\_date('01-Jan-17', 'dd-MM-yy'), to\_date('01-Jan-14', 'dd-MM-yy'), to\_date('01-Jan-18', 'dd-MM-yy'));**

**lv\_numRow\_exp := resNum(48, 65, 0);**

**FOR x in 1..lv\_resNum\_exp.count LOOP**

**lv\_resNum\_act := null;**

**lv\_sDate\_act := null;**

**lv\_eDate\_act := null;**

**lv\_numRow\_act := 0;**

**list\_reservations\_ending\_pp(lv\_end\_dates(x), lcur\_test);**

**FETCH lcur\_test into lv\_resNum\_act, lv\_sDate\_act, lv\_eDate\_act;**

**IF lcur\_test%FOUND THEN lv\_numRow\_act := lv\_numRow\_act + 1; END IF;**

**IF nvl(lv\_resNum\_act, 0) = nvl(lv\_resNum\_exp(x), 0)**

**AND nvl(lv\_sDate\_act, to\_date('01-01-01','dd-mm-yy')) = nvl(lv\_sDate\_exp(x), to\_date('01-01-01','dd-mm-yy'))**

**AND nvl(lv\_eDate\_act, to\_date('01-01-01','dd-mm-yy')) = nvl(lv\_eDate\_exp(x), to\_date('01-01-01','dd-mm-yy'))**

**THEN**

**lv\_num\_pass := lv\_num\_pass + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Test ' || ((x\*2)-1) || ' passed.');**

**ELSE**

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

**DBMS\_OUTPUT.PUT\_LINE('---TEST CASE ' || ((x\*2)-1) || ' FAILED. --');**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN NUMBER: ' || lv\_resNum\_act || ' EXPECTED NUMBER: ' || lv\_resNum\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_sDate\_act || ' EXPECTED NUMBER: ' || lv\_sDate\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_eDate\_act || ' EXPECTED NUMBER: ' || lv\_eDate\_exp(x));**

**END IF;**

**lv\_num\_total := lv\_num\_total + 1;**

**LOOP**

**FETCH lcur\_test into lv\_resNum\_act, lv\_sDate\_act, lv\_eDate\_act;**

**IF lcur\_test%FOUND THEN lv\_numRow\_act := lv\_numRow\_act + 1; END IF;**

**EXIT WHEN lcur\_test%NOTFOUND;**

**END LOOP;**

**IF lv\_numRow\_act = lv\_numRow\_exp(x) THEN**

**lv\_num\_pass := lv\_num\_pass + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Test ' || (x\*2) || ' passed.');**

**ELSE**

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

**DBMS\_OUTPUT.PUT\_LINE('---TEST CASE ' || (x\*2) || ' FAILED. --');**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL ROW COUNT: ' || lv\_numRow\_act || ' EXPECTED ROW COUNT: ' || lv\_numRow\_exp(x));**

**END IF;**

**lv\_num\_total := lv\_num\_total + 1;**

**END LOOP;**

**pv\_num\_passed\_o := lv\_num\_pass;**

**pv\_num\_run\_o := lv\_num\_total;**

**END;**

**PART D**

**--TEST PART D**

**PROCEDURE test\_list\_active\_res (**

**pv\_num\_passed\_o OUT number,**

**pv\_num\_run\_o OUT number**

**) AS**

**lv\_num\_total number := 0;**

**lv\_num\_pass number := 0;**

**type pk IS VARRAY(4) OF NUMBER;**

**type names IS VARRAY(4) OF VARCHAR2(25);**

**type resDate IS VARRAY(4) OF DATE;**

**--Expected Data**

**lv\_resNum\_exp pk;**

**lv\_fName\_exp names;**

**lv\_lName\_exp names;**

**lv\_petNum\_exp pk;**

**lv\_pName\_exp names;**

**lv\_runNum\_exp pk;**

**lv\_sDate\_exp resDate;**

**lv\_eDate\_exp resDate;**

**lv\_numRows\_exp pk;**

**--Passed data**

**lv\_ownNums pk;**

**lv\_sDates resDate;**

**lcur\_test SYS\_REFCURSOR;**

**lv\_resNum\_act number;**

**lv\_fName\_act VARCHAR2(25);**

**lv\_lName\_act VARCHAR2(25);**

**lv\_petNum\_act number;**

**lv\_pName\_act VARCHAR2(25);**

**lv\_runNum\_act number;**

**lv\_sDate\_act DATE;**

**lv\_eDate\_act DATE;**

**lv\_numRow\_act number;**

**lv\_date date := null;**

**BEGIN**

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

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

**DBMS\_OUTPUT.PUT\_LINE('------TEST LIST\_ACTIVE\_RESERVATIONS-----');**

**--MAKE UP DATA**

**lv\_resNum\_exp := pk(630, null, 630, null);**

**lv\_fName\_exp := names('Barb B.', null, 'Barb B.', null);**

**lv\_lName\_exp := names('Que', null, 'Que', null);**

**lv\_petNum\_exp := pk(33, null, 33, null);**

**lv\_pName\_exp := names('Willie', null, 'Willie', null);**

**lv\_runNum\_exp := pk(13, null, 13, null);**

**lv\_sDate\_exp := resDate(to\_date('05-Mar-17','dd-MM-yy'), null, to\_date('05-Mar-17','dd-MM-yy'), null);**

**lv\_eDate\_exp := resDate(to\_date('13-Mar-17','dd-MM-yy'), null, to\_date('13-Mar-17','dd-MM-yy'), null);**

**lv\_numRows\_exp := pk(1, 0, 1, 0);**

**lv\_ownNums := pk(18, 18, null, null);**

**lv\_sDates := resDate(to\_date('10-Mar-17','dd-MM-yy'), null, to\_date('10-Mar-17','dd-MM-yy'), null);**

**FOR x in 1..lv\_resNum\_exp.count LOOP**

**lv\_resNum\_act := null;**

**lv\_fName\_act := null;**

**lv\_lName\_act := null;**

**lv\_petNum\_act := null;**

**lv\_pName\_act := null;**

**lv\_runNum\_act := null;**

**lv\_sDate\_act := null;**

**lv\_eDate\_act := null;**

**lv\_numRow\_act := 0;**

**list\_active\_reservations\_pp(lv\_ownNums(x), lv\_sDates(x), lcur\_test);**

**FETCH lcur\_test into lv\_resNum\_act, lv\_fName\_act, lv\_lName\_act, lv\_petNum\_act, lv\_pName\_act, lv\_runNum\_act, lv\_sDate\_act, lv\_eDate\_act;**

**IF lcur\_test%FOUND THEN lv\_numRow\_act := lv\_numRow\_act + 1; END IF;**

**IF nvl(lv\_resNum\_act, -1) = nvl(lv\_resNum\_exp(x), -1) AND nvl(lv\_fName\_act, '~') = nvl(lv\_fName\_exp(x), '~')**

**AND nvl(lv\_lName\_act, '~') = nvl(lv\_lName\_exp(x), '~') AND nvl(lv\_petNum\_act, -1) = nvl(lv\_petNum\_exp(x), -1)**

**AND nvl(lv\_pName\_act, '~') = nvl(lv\_pName\_exp(x), '~') AND nvl(lv\_runNum\_act, -1) = nvl(lv\_runNum\_exp(x), -1)**

**AND nvl(lv\_sDate\_act, to\_date('01-01-01','dd-mm-yy')) = nvl(lv\_sDate\_exp(x), to\_date('01-01-01','dd-mm-yy'))**

**AND nvl(lv\_eDate\_act, to\_date('01-01-01','dd-mm-yy')) = nvl(lv\_eDate\_exp(x), to\_date('01-01-01','dd-mm-yy'))**

**THEN**

**lv\_num\_pass := lv\_num\_pass + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Test ' || ((x\*2)-1) || ' passed.');**

**ELSE**

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

**DBMS\_OUTPUT.PUT\_LINE('---TEST CASE ' || ((x\*2)-1) || ' FAILED. --');**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN NUMBER: ' || lv\_resNum\_act || ' EXPECTED NUMBER: ' || lv\_resNum\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_sDate\_act || ' EXPECTED NUMBER: ' || lv\_sDate\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_eDate\_act || ' EXPECTED NUMBER: ' || lv\_eDate\_exp(x));**

**END IF;**

**lv\_num\_total := lv\_num\_total + 1;**

**LOOP**

**FETCH lcur\_test into lv\_resNum\_act, lv\_fName\_act, lv\_lName\_act, lv\_petNum\_act, lv\_pName\_act, lv\_runNum\_act, lv\_sDate\_act, lv\_eDate\_act;**

**IF lcur\_test%FOUND THEN lv\_numRow\_act := lv\_numRow\_act + 1; END IF;**

**EXIT WHEN lcur\_test%NOTFOUND;**

**END LOOP;**

**IF lv\_numRow\_act = lv\_numRows\_exp(x) THEN**

**lv\_num\_pass := lv\_num\_pass + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Test ' || (x\*2) || ' passed.');**

**ELSE**

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

**DBMS\_OUTPUT.PUT\_LINE('---TEST CASE ' || (x\*2) || ' FAILED. --');**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL ROW COUNT: ' || lv\_numRow\_act || ' EXPECTED ROW COUNT: ' || lv\_numRows\_exp(x));**

**END IF;**

**lv\_num\_total := lv\_num\_total + 1;**

**END LOOP;**

**pv\_num\_passed\_o := lv\_num\_pass;**

**pv\_num\_run\_o := lv\_num\_total;**

**END;**

**PART E**

**--TEST PART E**

**PROCEDURE test\_list\_active\_res\_2 (**

**pv\_num\_passed\_o OUT number,**

**pv\_num\_run\_o OUT number**

**) AS**

**lv\_num\_total number := 0;**

**lv\_num\_pass number := 0;**

**type pk IS VARRAY(8) OF NUMBER;**

**type names IS VARRAY(8) OF VARCHAR2(25);**

**type resDate IS VARRAY(8) OF DATE;**

**--Expected Data**

**lv\_resNum\_exp pk;**

**lv\_fName\_exp names;**

**lv\_lName\_exp names;**

**lv\_petNum\_exp pk;**

**lv\_pName\_exp names;**

**lv\_runNum\_exp pk;**

**lv\_sDate\_exp resDate;**

**lv\_eDate\_exp resDate;**

**lv\_numRows\_exp pk;**

**--Passed data**

**lv\_ownNums pk;**

**lv\_sDates resDate;**

**lv\_eDates resDate;**

**lcur\_test SYS\_REFCURSOR;**

**lv\_resNum\_act number;**

**lv\_fName\_act VARCHAR2(25);**

**lv\_lName\_act VARCHAR2(25);**

**lv\_petNum\_act number;**

**lv\_pName\_act VARCHAR2(25);**

**lv\_runNum\_act number;**

**lv\_sDate\_act DATE;**

**lv\_eDate\_act DATE;**

**lv\_numRow\_act number;**

**lv\_date date := null;**

**BEGIN**

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

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

**DBMS\_OUTPUT.PUT\_LINE('------TEST LIST\_ACTIVE\_RESERVATIONS-----');**

**lv\_resNum\_exp := pk(630, 630, null, null, 625, 625, 721, null);**

**lv\_fName\_exp := names('Barb B.', 'Barb B.', null, null, 'Ella', 'Ella', 'Mike', null);**

**lv\_lName\_exp := names('Que', 'Que', null, null, 'Mentary', 'Mentary', 'O''Phone', null);**

**lv\_petNum\_exp := pk(33, 33, null, null, 20, 20, 3, null);**

**lv\_pName\_exp := names('Willie', 'Willie', null, null, 'Poppy', 'Poppy', 'Jasper', null);**

**lv\_runNum\_exp := pk(13, 13, null, null, null, null, 27, null);**

**lv\_sDate\_exp := resDate(to\_date('05-Mar-17', 'dd-MM-yy'), to\_date('05-Mar-17', 'dd-MM-yy'),**

**null, null, to\_date('15-Mar-17','dd-MM-yy'), to\_date('15-Mar-17','dd-MM-yy'), to\_date('05-Apr-17','dd-MM-yy'), null);**

**lv\_eDate\_exp := resDate(to\_date('13-Mar-17', 'dd-MM-yy'), to\_date('13-Mar-17', 'dd-MM-yy'),**

**null, null, to\_date('20-Mar-17','dd-MM-yy'), to\_date('20-Mar-17','dd-MM-yy'), to\_date('09-Apr-17','dd-MM-yy'), null);**

**lv\_numRows\_exp := pk(2, 2, 0, 0, 6, 5, 1, 0);**

**lv\_ownNums := pk(18, 18, 18, 18, null, null, null, null);**

**lv\_sDates := resDate(to\_date('05-Mar-17','dd-MM-yy'), to\_date('05-Mar-17','dd-MM-yy'), NULL, NULL, to\_date('05-Mar-17','dd-MM-yy'), to\_date('05-Mar-17','dd-MM-yy'), NULL, NULL);**

**lv\_eDates := resDate(to\_date('15-Apr-17','dd-MM-yy'), NULL, to\_date('15-Apr-17','dd-MM-yy'), NULL, to\_date('15-Apr-17','dd-MM-yy'), NULL, to\_date('15-Apr-17','dd-MM-yy'), NULL);**

**FOR x in 1..lv\_resNum\_exp.count LOOP**

**lv\_resNum\_act := null;**

**lv\_fName\_act := null;**

**lv\_lName\_act := null;**

**lv\_petNum\_act := null;**

**lv\_pName\_act := null;**

**lv\_runNum\_act := null;**

**lv\_sDate\_act := null;**

**lv\_eDate\_act := null;**

**lv\_numRow\_act := 0;**

**list\_active\_reservations\_pp(lcur\_test, lv\_ownNums(x), lv\_sDates(x), lv\_eDates(x));**

**FETCH lcur\_test into lv\_resNum\_act, lv\_fName\_act, lv\_lName\_act, lv\_petNum\_act, lv\_pName\_act, lv\_runNum\_act, lv\_sDate\_act, lv\_eDate\_act;**

**IF lcur\_test%FOUND THEN lv\_numRow\_act := lv\_numRow\_act + 1; END IF;**

**IF nvl(lv\_resNum\_act, -1) = nvl(lv\_resNum\_exp(x), -1) AND nvl(lv\_fName\_act, '~') = nvl(lv\_fName\_exp(x), '~')**

**AND nvl(lv\_lName\_act, '~') = nvl(lv\_lName\_exp(x), '~') AND nvl(lv\_petNum\_act, -1) = nvl(lv\_petNum\_exp(x), -1)**

**AND nvl(lv\_pName\_act, '~') = nvl(lv\_pName\_exp(x), '~') AND nvl(lv\_runNum\_act, -1) = nvl(lv\_runNum\_exp(x), -1)**

**AND nvl(lv\_sDate\_act, to\_date('01-01-01','dd-mm-yy')) = nvl(lv\_sDate\_exp(x), to\_date('01-01-01','dd-mm-yy'))**

**AND nvl(lv\_eDate\_act, to\_date('01-01-01','dd-mm-yy')) = nvl(lv\_eDate\_exp(x), to\_date('01-01-01','dd-mm-yy'))**

**THEN**

**lv\_num\_pass := lv\_num\_pass + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Test ' || ((x\*2)-1) || ' passed.');**

**ELSE**

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

**DBMS\_OUTPUT.PUT\_LINE('---TEST CASE ' || ((x\*2)-1) || ' FAILED. --');**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN NUMBER: ' || lv\_resNum\_act || ' EXPECTED NUMBER: ' || lv\_resNum\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_fName\_act || ' EXPECTED NUMBER: ' || lv\_fName\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_lName\_act || ' EXPECTED NUMBER: ' || lv\_lName\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_petNum\_act || ' EXPECTED NUMBER: ' || lv\_petNum\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_pName\_act || ' EXPECTED NUMBER: ' || lv\_pName\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_runNum\_act || ' EXPECTED NUMBER: ' || lv\_runNum\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_sDate\_act || ' EXPECTED NUMBER: ' || lv\_sDate\_exp(x));**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL RUN SIZE: ' || lv\_eDate\_act || ' EXPECTED NUMBER: ' || lv\_eDate\_exp(x));**

**END IF;**

**lv\_num\_total := lv\_num\_total + 1;**

**LOOP**

**FETCH lcur\_test into lv\_resNum\_act, lv\_fName\_act, lv\_lName\_act, lv\_petNum\_act, lv\_pName\_act, lv\_runNum\_act, lv\_sDate\_act, lv\_eDate\_act;**

**IF lcur\_test%FOUND THEN lv\_numRow\_act := lv\_numRow\_act + 1; END IF;**

**EXIT WHEN lcur\_test%NOTFOUND;**

**END LOOP;**

**IF lv\_numRow\_act = lv\_numRows\_exp(x) THEN**

**lv\_num\_pass := lv\_num\_pass + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Test ' || (x\*2) || ' passed.');**

**ELSE**

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

**DBMS\_OUTPUT.PUT\_LINE('---TEST CASE ' || (x\*2) || ' FAILED. --');**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL ROW COUNT: ' || lv\_numRow\_act || ' EXPECTED ROW COUNT: ' || lv\_numRows\_exp(x));**

**END IF;**

**lv\_num\_total := lv\_num\_total + 1;**

**END LOOP;**

**pv\_num\_passed\_o := lv\_num\_pass;**

**pv\_num\_run\_o := lv\_num\_total;**

**END;**

**PART E**

**PROCEDURE test\_list\_check\_vacc (**

**pv\_num\_passed\_o OUT number,**

**pv\_num\_run\_o OUT number**

**) AS**

**lv\_num\_total number := 0;**

**lv\_num\_pass number := 0;**

**type numm IS VARRAY(5) OF NUMBER;**

**type vacc IS VARRAY(5) OF VARCHAR2(25);**

**--Expected Data**

**lv\_vacc\_exp vacc;**

**lv\_numRows\_exp numm;**

**--Passed data**

**lv\_petNums numm;**

**lv\_resNums numm;**

**lcur\_test SYS\_REFCURSOR;**

**lv\_vacc\_act VARCHAR2(25);**

**lv\_numRow\_act number;**

**BEGIN**

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

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

**DBMS\_OUTPUT.PUT\_LINE('------TEST LIST\_CHECK\_VACCINATIONS-----');**

**--MAKE UP DATA**

**lv\_vacc\_exp := vacc(null, 'Bordetella', 'Hepatitis');**

**lv\_numRows\_exp := numm(0, 4, 2);**

**lv\_petNums := numm(1, 6, 7);**

**lv\_resNums := numm(100, 106, 631);**

**FOR x in 1..lv\_vacc\_exp.count LOOP**

**lv\_vacc\_act := null;**

**lv\_numRow\_act := 0;**

**list\_check\_vaccinations\_pp(lv\_petNums(x), lv\_resNums(x), lcur\_test);**

**FETCH lcur\_test into lv\_vacc\_act;**

**IF lcur\_test%FOUND THEN lv\_numRow\_act := lv\_numRow\_act + 1; END IF;**

**IF nvl(lv\_vacc\_act, '~') = nvl(lv\_vacc\_exp(x), '~') THEN**

**lv\_num\_pass := lv\_num\_pass + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Test ' || ((x\*2)-1) || ' passed.');**

**ELSE**

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

**DBMS\_OUTPUT.PUT\_LINE('---TEST CASE ' || ((x\*2)-1) || ' FAILED. --');**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL VACCINATION: ' || lv\_vacc\_act || ' EXPECTED VACCINATION: ' || lv\_vacc\_exp(x));**

**END IF;**

**lv\_num\_total := lv\_num\_total + 1;**

**LOOP**

**FETCH lcur\_test into lv\_vacc\_act;**

**IF lcur\_test%FOUND THEN lv\_numRow\_act := lv\_numRow\_act + 1; END IF;**

**EXIT WHEN lcur\_test%NOTFOUND;**

**END LOOP;**

**IF lv\_numRow\_act = lv\_numRows\_exp(x) THEN**

**lv\_num\_pass := lv\_num\_pass + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Test ' || (x\*2) || ' passed.');**

**ELSE**

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

**DBMS\_OUTPUT.PUT\_LINE('---TEST CASE ' || (x\*2) || ' FAILED. --');**

**DBMS\_OUTPUT.PUT\_LINE('ACTUAL ROW COUNT: ' || lv\_numRow\_act || ' EXPECTED ROW COUNT: ' || lv\_numRows\_exp(x));**

**END IF;**

**lv\_num\_total := lv\_num\_total + 1;**

**END LOOP;**

**pv\_num\_passed\_o := lv\_num\_pass;**

**pv\_num\_run\_o := lv\_num\_total;**

**END;**

**RUN TESTS**

**PROCEDURE test\_package\_pp AS**

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

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

**lv\_temp\_passed number;**

**lv\_temp\_total number;**

**BEGIN**

**test\_list\_available\_runs(lv\_temp\_passed, lv\_temp\_total);**

**lv\_number\_passed := lv\_number\_passed + lv\_temp\_passed;**

**lv\_total := lv\_total + lv\_temp\_total;**

**test\_list\_res\_starting(lv\_temp\_passed, lv\_temp\_total);**

**lv\_number\_passed := lv\_number\_passed + lv\_temp\_passed;**

**lv\_total := lv\_total + lv\_temp\_total;**

**test\_list\_res\_ending(lv\_temp\_passed, lv\_temp\_total);**

**lv\_number\_passed := lv\_number\_passed + lv\_temp\_passed;**

**lv\_total := lv\_total + lv\_temp\_total;**

**test\_list\_active\_res(lv\_temp\_passed, lv\_temp\_total);**

**lv\_number\_passed := lv\_number\_passed + lv\_temp\_passed;**

**lv\_total := lv\_total + lv\_temp\_total;**

**test\_list\_active\_res\_2(lv\_temp\_passed, lv\_temp\_total);**

**lv\_number\_passed := lv\_number\_passed + lv\_temp\_passed;**

**lv\_total := lv\_total + lv\_temp\_total;**

**test\_list\_check\_vacc(lv\_temp\_passed, lv\_temp\_total);**

**lv\_number\_passed := lv\_number\_passed + lv\_temp\_passed;**

**lv\_total := lv\_total + lv\_temp\_total;**

**DBMS\_OUTPUT.PUT\_LINE('Tests run: ' || lv\_total);**

**DBMS\_OUTPUT.PUT\_LINE('Tests passed: ' || lv\_number\_passed || '/' || lv\_total**

**|| ' ' || (lv\_number\_passed/lv\_total \* 100));**

**DBMS\_OUTPUT.PUT\_LINE('Tests failed: ' || (lv\_total-lv\_number\_passed) || '/' || lv\_total**

**|| ' ' || (100-(lv\_number\_passed/lv\_total \* 100)));**

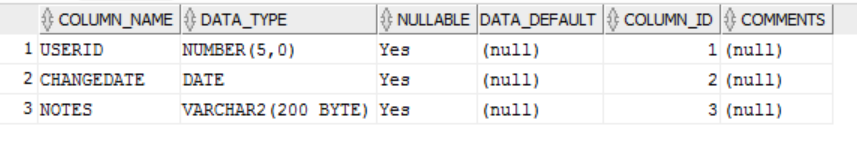
**END test\_package\_pp;**

**Sample output from Test Code:**

# Triggers

## Reservation log: Create a new table called HVK\_CHANGE\_LOG with the columns: userId, date, and a string of 200 characters.

**SQL:** **CREATE TABLE hvk\_change\_log (userId number(5), changeDate date, notes varchar2(200));**



## Create a trigger that is fired after inserts, updates or deletes on the HVK\_RESERVATION table, that logs the userId, date, and a string copy of the HVK\_RESERVATION details (print out all fields). Hint, you can get the userid from (SELECT ora\_login\_user FROM DUAL;), see [here](https://docs.oracle.com/cloud/latest/db112/LNPLS/triggers.htm#LNPLS79) for details from Oracle.

**SQL:**

**DROP SEQUENCE change\_log\_seq;**

**CREATE SEQUENCE change\_log\_seq**

**START WITH 100**

**INCREMENT BY 1;**

**CREATE OR REPLACE TRIGGER update\_change\_log**

**AFTER INSERT OR UPDATE OR DELETE ON hvk\_reservation**

**FOR EACH ROW**

**DECLARE**

**lv\_log\_num number;**

**lv\_user VARCHAR(50);**

**BEGIN**

**SELECT change\_log\_seq.NEXTVAL, ora\_login\_user**

**INTO lv\_log\_num, lv\_user**

**FROM DUAL;**

**INSERT INTO hvk\_change\_log (**

**userId, changeDate, notes**

**) values (**

**lv\_log\_num, SYSDATE, :old.reservation\_number || '/'**

**|| :old.reservation\_start\_date || '/'**

**|| :old.reservation\_end\_date**

**);**

**END;**

## Provide an automated test to validate your Logging trigger.

**SQL:**

**SET SERVEROUTPUT ON;**

**DECLARE**

**lv\_user varchar2(50);**

**lv\_date date;**

**lv\_note varchar2(200);**

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

**BEGIN**

**--insert: Trigger fires after**

**INSERT INTO hvk\_reservation (**

**reservation\_number, reservation\_start\_date, reservation\_end\_date**

**) values (**

**999, to\_date('05-Mar-17', 'dd-MM-yy'), to\_date('25-Mar-17','dd-MM-yy')**

**);**

**--Grab the new data from the change log**

**SELECT userID, changedate, notes INTO lv\_user, lv\_date, lv\_note**

**FROM hvk\_change\_log**

**WHERE Rownum = 1;**

**--Check if the data is correct**

**IF lv\_note != 999 || '/' || to\_date('05-Mar-17', 'dd-MM-yy') || '/' || to\_date('25-Mar-17','dd-MM-yy') THEN**

**DBMS\_OUTPUT.PUT\_LINE('Test Case 1 failed.');**

**DBMS\_OUTPUT.PUT\_LINE('Expected: ' || 999 || '/' || to\_date('05-Mar-17', 'dd-MM-yy') || '/' || to\_date('25-Mar-17','dd-MM-yy'));**

**DBMS\_OUTPUT.PUT\_LINE('Received: ' || lv\_note);**

**ELSE lv\_pass := lv\_pass + 1;**

**END IF;**

**--Wipe the changelog**

**DELETE FROM hvk\_change\_log;**

**--Update the reservation: Changelog will update fire after it**

**UPDATE hvk\_reservation**

**SET reservation\_start\_date = to\_date('10-Mar-17', 'dd-MM-yy')**

**WHERE reservation\_number = 999;**

**--Grab the new data from the changelog**

**SELECT userID, changedate, notes INTO lv\_user, lv\_date, lv\_note**

**FROM hvk\_change\_log**

**WHERE Rownum = 1;**

**--check if the data is correct**

**IF lv\_note != 999 || '/' || to\_date('10-Mar-17', 'dd-MM-yy') || '/' || to\_date('25-Mar-17','dd-MM-yy') THEN**

**DBMS\_OUTPUT.PUT\_LINE('Test Case 2 failed.');**

**DBMS\_OUTPUT.PUT\_LINE('Expected: ' || 999 || '/' || to\_date('10-Mar-17', 'dd-MM-yy') || '/' || to\_date('25-Mar-17','dd-MM-yy'));**

**DBMS\_OUTPUT.PUT\_LINE('Received: ' || lv\_note);**

**ELSE lv\_pass := lv\_pass + 1;**

**END IF;**

**--Wipe the changelog**

**DELETE FROM hvk\_change\_log;**

**--Delete: Trigger fires after row is deleted**

**DELETE FROM hvk\_reservation**

**WHERE reservation\_number = 999;**

**--Grab the new data from the changelog**

**SELECT userID, changedate, notes INTO lv\_user, lv\_date, lv\_note**

**FROM hvk\_change\_log**

**WHERE Rownum = 1;**

**--check if the data is correct**

**IF lv\_note != '/' || '/' THEN**

**DBMS\_OUTPUT.PUT\_LINE('Test Case 3 failed.');**

**DBMS\_OUTPUT.PUT\_LINE('Expected: ' || '/' || '/');**

**DBMS\_OUTPUT.PUT\_LINE('Received: ' || lv\_note);**

**ELSE lv\_pass := lv\_pass + 1;**

**END IF;**

**--Wipe the changelog**

**DELETE FROM hvk\_change\_log;**

**IF lv\_pass = 3 THEN**

**DBMS\_OUTPUT.PUT\_LINE('All test cases passed!');**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE(lv\_pass || '/3 test cases passed.');**

**END IF;**

**END;**

**Output:**

**All test cases passed!**

## Create a trigger that is fired before inserts or updates on the HVK\_RESERVATION table that validates that the end date can not preceed the start date. The end date can not be in the past. If this condition is not met, the insert/update must fail.

**SQL:**

**Create or replace TRIGGER validate\_reservation**

**BEFORE INSERT OR UPDATE ON hvk\_reservation**

**FOR EACH ROW**

**BEGIN**

**IF INSERTING THEN**

**IF :new.reservation\_start\_date <= sysdate THEN**

**DBMS\_OUTPUT.PUT\_LINE('The start date cannot be before today.');**

**RAISE\_APPLICATION\_ERROR(-20001, 'The start date cannot be before today.');**

**ELSIF :new.reservation\_start\_date >= :new.reservation\_end\_date THEN**

**DBMS\_OUTPUT.PUT\_LINE('The start date cannot be after the end date.');**

**RAISE\_APPLICATION\_ERROR(-20002, 'The start date cannot be after the end date.');**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('The reservation was added successfully.');**

**END IF;**

**END IF;**

**IF UPDATING THEN**

**IF :new.reservation\_start\_date <= sysdate THEN**

**DBMS\_OUTPUT.PUT\_LINE('The start date cannot be before today. The update was canceled.');**

**:new.reservation\_start\_date := :old.reservation\_start\_date;**

**ELSIF :new.reservation\_start\_date >= :new.reservation\_end\_date THEN**

**DBMS\_OUTPUT.PUT\_LINE('The start date cannot be after the end date. The update was canceled.');**

**:new.reservation\_start\_date := :old.reservation\_start\_date;**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('The reservation was added successfully.');**

**END IF;**

**END IF;**

**END;**

**/\***

**Couldn't find a way to prevent the transaction from occuring entirely without**

**just raising an application error. RAISE\_APPLICATION\_ERROR ruins the idea of**

**automated test cases, so I was trying to avoid using it, but just couldn't**

**seem to find another way to do it. Would be very interested to know if there**

**is a decent way to cancel a transaction that's about to happen. I tried using**

**customer exceptions, but they just let the insert happen anyways, tried**

**rollback, but the transaction hasn't actually happened yet, etc. Best way I**

**can think of doing it is to create a save point at the before insert, then**

**check the conditions after the insert and chose whether to revert to the save**

**created in the before insert.**

**\*/**

## Provide an automated test to test your validation trigger:

**SQL:**

**SET SERVEROUTPUT ON;**

**BEGIN**

**--Valid**

**INSERT INTO hvk\_reservation (**

**reservation\_number, reservation\_start\_date, reservation\_end\_date**

**) values (**

**999, to\_date('05-Mar-18', 'dd-MM-yy'), to\_date('25-Mar-18','dd-MM-yy')**

**);**

**rollback;**

**--Start date before end date**

**UPDATE hvk\_reservation**

**SET RESERVATION\_END\_DATE = to\_date('02-Mar-18','dd-MM-yy')**

**WHERE RESERVATION\_NUMBER = 999;**

**--Start date before today**

**INSERT INTO hvk\_reservation (**

**reservation\_number, reservation\_start\_date, reservation\_end\_date**

**) values (**

**999, to\_date('05-Mar-17', 'dd-MM-yy'), to\_date('25-Mar-17','dd-MM-yy')**

**);**

**--Start date after end date**

**INSERT INTO hvk\_reservation (**

**reservation\_number, reservation\_start\_date, reservation\_end\_date**

**) values (**

**999, to\_date('25-Mar-18', 'dd-MM-yy'), to\_date('05-Mar-18','dd-MM-yy')**

**);**

**END;**

**Output:**

Error report -

ORA-20001: The start date cannot be before today.

ORA-06512: at "PDUMARESQ.VALIDATE\_RESERVATION", line 5

ORA-04088: error during execution of trigger 'PDUMARESQ.VALIDATE\_RESERVATION'

ORA-06512: at line 16

The reservation was added successfully.

The start date cannot be before today.

# Assessment

1. What did you learn in completing this assignment?

This assignment really taught me more about the three big components to it – packages, sys\_refcurors and tiggers.

1. What did you have difficulty with?

I had a lot of difficulty testing part a of the assignment. There was a lot to do and I just struggled wrapping my mind a bit around how to test the entire table.

I also couldn’t get number 5 working in part B because I couldn’t get the test cases to continue running through an application error, which was the only way to prevent the transaction from occurring. It was really frustrating, but I couldn’t find a better way.

1. What did you do well?

I think that I overall did most of the assignment pretty well. Triggers weren’t too difficult, and packages make sense, it’s really just the level of protection on the methods, same thing as declaring them public/private/protected

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

8

1. What took you the most time?

Writing the test cases for part A took the most time.