

# Exercises for Chapter 21: Packages

## Try It Yourself

The projects in this section are meant to have you utilize all of the skills that you have acquired throughout this chapter. Here are some exercises that will help you test the depth of your understanding.

- 1) Add a procedure to the `student_api` package called `remove_student`. This procedure accepts a `student_id` and returns nothing. Based on the student ID passed in, it removes the student from the database. If the student does not exist or there is a problem removing the student (such as a foreign key constraint violation), then let the calling program handle it.

Answer: The package should be similar to the following:

```
CREATE OR REPLACE PACKAGE student_api AS
    v_current_date DATE;

    PROCEDURE discount;

    FUNCTION new_instructor_id
    RETURN instructor.instructor_id%TYPE;

    FUNCTION total_cost_for_student
        (p_student_id IN student.student_id%TYPE)
    RETURN course.cost%TYPE;
    PRAGMA RESTRICT_REFERENCES
        (total_cost_for_student, WNDS, WNPS, RNPS);

    PROCEDURE get_student_info
        (p_student_id    IN student.student_id%TYPE,
         p_last_name     OUT student.last_name%TYPE,
         p_first_name    OUT student.first_name%TYPE,
         p_zip           OUT student.zip%TYPE,
         p_return_code   OUT NUMBER);

    PROCEDURE get_student_info
        (p_last_name     IN student.last_name%TYPE,
         p_first_name    IN student.first_name%TYPE,
         p_student_id    OUT student.student_id%TYPE,
```

```

        p_zip          OUT student.zip%TYPE,
        p_return_code OUT NUMBER);

PROCEDURE remove_student
    (p_studid IN student.student_id%TYPE);
END student_api;
/
CREATE OR REPLACE PACKAGE BODY student_api AS

PROCEDURE discount
IS
    CURSOR c_group_discount IS
        SELECT distinct s.course_no, c.description
            FROM section s, enrollment e, course c
           WHERE s.section_id = e.section_id
        GROUP BY s.course_no, c.description,
                e.section_id, s.section_id
        HAVING COUNT(*) >=8;
BEGIN
    FOR r_group_discount IN c_group_discount LOOP
        UPDATE course
           SET cost = cost * .95
          WHERE course_no = r_group_discount.course_no;

        DBMS_OUTPUT.PUT_LINE
            ('A 5% discount has been given to'||
             r_group_discount.course_no||' '||
             r_group_discount.description);
    END LOOP;
END discount;

FUNCTION new_instructor_id
RETURN instructor.instructor_id%TYPE
IS
    v_new_instid instructor.instructor_id%TYPE;
BEGIN
    SELECT INSTRUCTOR_ID_SEQ.NEXTVAL
        INTO v_new_instid
        FROM dual;
    RETURN v_new_instid;
EXCEPTION
    WHEN OTHERS THEN
        DECLARE
            v_sqlerrm VARCHAR2(250) := SUBSTR(SQLERRM,1,250);
        BEGIN
            RAISE_APPLICATION_ERROR
                (-20003, 'Error in instructor_id: '||v_sqlerrm);
        END;
END new_instructor_id;

FUNCTION get_course_descript_private
    (p_course_no course.course_no%TYPE)
RETURN course.description%TYPE
IS

```

```

        v_course_descript course.description%TYPE;
BEGIN
    SELECT description
        INTO v_course_descript
        FROM course
        WHERE course_no = p_course_no;
    RETURN v_course_descript;
EXCEPTION
    WHEN OTHERS THEN
        RETURN NULL;
END get_course_descript_private;

FUNCTION total_cost_for_student
    (p_student_id IN student.student_id%TYPE)
RETURN course.cost%TYPE
IS
    v_cost course.cost%TYPE;
BEGIN
    SELECT sum(cost)
        INTO v_cost
        FROM course c, section s, enrollment e
        WHERE c.course_no = c.course_no
            AND e.section_id = s.section_id
            AND e.student_id = p_student_id;
    RETURN v_cost;
EXCEPTION
    WHEN OTHERS THEN
        RETURN NULL;
END total_cost_for_student;

PROCEDURE get_student_info
    (p_student_id    IN    student.student_id%TYPE,
     p_last_name     OUT student.last_name%TYPE,
     p_first_name    OUT student.first_name%TYPE,
     p_zip           OUT student.zip%TYPE,
     p_return_code   OUT NUMBER)
IS
BEGIN
    SELECT last_name, first_name, zip
        INTO p_last_name, p_first_name, p_zip
        FROM student
        WHERE student.student_id = p_student_id;
    p_return_code := 0;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE ('Student ID is not valid. ');
        p_return_code := -100;
        p_last_name := NULL;
        p_first_name := NULL;
        p_zip := NULL;
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE
            ('Error in procedure get_student_info');
END get_student_info;

```

```

PROCEDURE get_student_info
    (p_last_name      IN student.last_name%TYPE,
     p_first_name     IN student.first_name%TYPE,
     p_student_id     OUT student.student_id%TYPE,
     p_zip            OUT student.zip%TYPE,
     p_return_code    OUT NUMBER)
IS
BEGIN
    SELECT student_id, zip
        INTO p_student_id, p_zip
        FROM student
        WHERE UPPER(last_name) = UPPER(p_last_name)
          AND UPPER(first_name) = UPPER(p_first_name);
    p_return_code := 0;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE ('Student name is not valid. ');
        p_return_code := -100;
        p_student_id := NULL;
        p_zip := NULL;
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE
            ('Error in procedure get_student_info');
END get_student_info;

PROCEDURE remove_student
    (p_studid IN student.student_id%TYPE)
IS
BEGIN
    DELETE
        FROM STUDENT
        WHERE student_id = p_studid;
END;

BEGIN
    SELECT trunc(sysdate, 'DD')
        INTO v_current_date
        FROM dual;
END student_api;

```

---

- 2) Alter remove\_student in the student\_api package body to accept an additional parameter. This new parameter is a VARCHAR2 and is called p\_ri. Make p\_ri default to "R." The new parameter may contain a value of "R" or "C." If "R" is received, it represents DELETE RESTRICT and the procedure acts as it does now. If there are enrollments for the student, the delete is disallowed. If a "C" is received, it represents DELETE CASCADE. This functionally means that the remove\_student procedure locates all records for the student in all of the tables and removes them from the database before attempting to remove the student from the student table. Decide how to handle the situation where the user passes in a code other than "C" or "R."

Answer: The package should look similar to the following:

```

CREATE OR REPLACE PACKAGE student_api AS

```

```

v_current_date DATE;

PROCEDURE discount;

FUNCTION new_instructor_id
    RETURN instructor.instructor_id%TYPE;

FUNCTION total_cost_for_student
    (p_student_id IN student.student_id%TYPE)
RETURN course.cost%TYPE;
PRAGMA RESTRICT_REFERENCES
    (total_cost_for_student, WNDS, WNPS, RNPS);

PROCEDURE get_student_info
    (p_student_id    IN    student.student_id%TYPE,
     p_last_name     OUT student.last_name%TYPE,
     p_first_name    OUT student.first_name%TYPE,
     p_zip           OUT student.zip%TYPE,
     p_return_code   OUT NUMBER);

PROCEDURE get_student_info
    (p_last_name     IN student.last_name%TYPE,
     p_first_name    IN student.first_name%TYPE,
     p_student_id    OUT student.student_id%TYPE,
     p_zip           OUT student.zip%TYPE,
     p_return_code   OUT NUMBER);

PROCEDURE remove_student
    (p_studid IN student.student_id%TYPE,
     p_r1     IN VARCHAR2 DEFAULT 'R');
END student_api;
/

CREATE OR REPLACE PACKAGE BODY student_api AS

PROCEDURE discount
IS
    CURSOR c_group_discount IS
        SELECT distinct s.course_no, c.description
            FROM section s, enrollment e, course c
            WHERE s.section_id = e.section_id
            GROUP BY s.course_no, c.description,
                    e.section_id, s.section_id
            HAVING COUNT(*) >=8;
BEGIN
    FOR r_group_discount IN c_group_discount LOOP
        UPDATE course
            SET cost = cost * .95
            WHERE course_no = r_group_discount.course_no;

        DBMS_OUTPUT.PUT_LINE
            ('A 5% discount has been given to'||
             r_group_discount.course_no||' '||
             r_group_discount.description);
    
```

```

        END LOOP;
    END discount;

    FUNCTION new_instructor_id
    RETURN instructor.instructor_id%TYPE
    IS
        v_new_instid instructor.instructor_id%TYPE;
    BEGIN
        SELECT INSTRUCTOR_ID_SEQ.NEXTVAL
            INTO v_new_instid
            FROM dual;
        RETURN v_new_instid;
    EXCEPTION
        WHEN OTHERS THEN
            DECLARE
                v_sqlerrm VARCHAR2(250) := SUBSTR(SQLERRM,1,250);
            BEGIN
                RAISE_APPLICATION_ERROR
                    (-20003, 'Error in instructor_id: '||v_sqlerrm);
            END;
    END new_instructor_id;

    FUNCTION get_course_descript_private
    (p_course_no course.course_no%TYPE)
    RETURN course.description%TYPE
    IS
        v_course_descript course.description%TYPE;
    BEGIN
        SELECT description
            INTO v_course_descript
            FROM course
            WHERE course_no = p_course_no;
        RETURN v_course_descript;
    EXCEPTION
        WHEN OTHERS THEN
            RETURN NULL;
    END get_course_descript_private;

    FUNCTION total_cost_for_student
    (p_student_id IN student.student_id%TYPE)
    RETURN course.cost%TYPE
    IS
        v_cost course.cost%TYPE;
    BEGIN
        SELECT sum(cost)
            INTO v_cost
            FROM course c, section s, enrollment e
            WHERE c.course_no = c.course_no
              AND e.section_id = s.section_id
              AND e.student_id = p_student_id;
        RETURN v_cost;
    EXCEPTION
        WHEN OTHERS THEN
            RETURN NULL;
    END total_cost_for_student;

```

```
END total_cost_for_student;
```

```
PROCEDURE get_student_info
```

```
(p_student_id   IN  student.student_id%TYPE,  
 p_last_name    OUT student.last_name%TYPE,  
 p_first_name   OUT student.first_name%TYPE,  
 p_zip          OUT student.zip%TYPE,  
 p_return_code  OUT NUMBER)
```

```
IS
```

```
BEGIN
```

```
  SELECT last_name, first_name, zip  
         INTO p_last_name, p_first_name, p_zip  
         FROM student  
         WHERE student.student_id = p_student_id;  
  p_return_code := 0;
```

```
EXCEPTION
```

```
  WHEN NO_DATA_FOUND THEN
```

```
    DBMS_OUTPUT.PUT_LINE ('Student ID is not valid.');
```

```
    p_return_code := -100;  
    p_last_name := NULL;  
    p_first_name := NULL;  
    p_zip := NULL;
```

```
  WHEN OTHERS THEN
```

```
    DBMS_OUTPUT.PUT_LINE  
      ('Error in procedure get_student_info');
```

```
END get_student_info;
```

```
PROCEDURE get_student_info
```

```
(p_last_name    IN student.last_name%TYPE,  
 p_first_name   IN student.first_name%TYPE,  
 p_student_id   OUT student.student_id%TYPE,  
 p_zip          OUT student.zip%TYPE,  
 p_return_code  OUT NUMBER)
```

```
IS
```

```
BEGIN
```

```
  SELECT student_id, zip  
         INTO p_student_id, p_zip  
         FROM student  
         WHERE UPPER(last_name) = UPPER(p_last_name)  
               AND UPPER(first_name) = UPPER(p_first_name);  
  p_return_code := 0;
```

```
EXCEPTION
```

```
  WHEN NO_DATA_FOUND THEN
```

```
    DBMS_OUTPUT.PUT_LINE  
      ('Student name is not valid.');
```

```
    p_return_code := -100;  
    p_student_id := NULL;  
    p_zip := NULL;
```

```
  WHEN OTHERS THEN
```

```
    DBMS_OUTPUT.PUT_LINE  
      ('Error in procedure get_student_info');
```

```
END get_student_info;
```

```
PROCEDURE remove_student
```

```

-- the parameters student_id and p_pri give user an
-- option of cascade delete or restrict delete for
-- the given students records
(p_studid IN student.student_id%TYPE,
 p_pri      IN VARCHAR2 DEFAULT 'R')
IS
-- declare exceptions for use in procedure
enrollment_present EXCEPTION;
bad_pri EXCEPTION;
BEGIN
-- R value is for restrict delete option
IF p_pri = 'R' THEN
    DECLARE
        -- a variable is needed to test if the student
        -- is in the enrollment table
        v_dummy CHAR(1);
    BEGIN
        -- This is a standard existence check
        -- If v_dummy is assigned a value via the
        -- SELECT INTO, the exception
        -- enrollment_present will be raised
        -- If the v_dummy is not assigned a value, the
        -- exception no_data_found will be raised
        SELECT NULL
            INTO v_dummy
            FROM enrollment e
            WHERE e.student_id = p_studid
            AND ROWNUM = 1;

        -- The rownum set to 1 prevents the SELECT
        -- INTO statement raise to_many_rows
        -- exception
        -- If there is at least one row in enrollment
        -- table with corresponding student_id, the
        -- restrict delete parameter will disallow the
        -- deletion of the student by raising
        -- the enrollment_present exception
        RAISE enrollment_present;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            -- The no_data_found exception is raised
            -- when there are no students found in the
            -- enrollment table Since the p_pri indicates
            -- a restrict delete user choice the delete
            -- operation is permitted
            DELETE FROM student
            WHERE student_id = p_studid;
    END;
-- when the user enter "C" for the p_pri
-- he/she indicates a cascade delete choice
ELSIF p_pri = 'C' THEN
    -- delete the student form the enrollment and
    -- grade tables
    DELETE FROM enrollment

```



```

        WHERE student_id = p_studid;

DELETE FROM grade
    WHERE student_id = p_studid;

-- delete from student table only after corresponding
-- records have been removed from the other tables
-- because the student table is the parent table
DELETE FROM student
    WHERE student_id = p_studid;
ELSE
    RAISE bad_pri;
END IF;
EXCEPTION
    WHEN bad_pri THEN
        RAISE_APPLICATION_ERROR
            (-20231, 'An incorrect p_pri value was '||
                'entered. The remove_student procedure can '||
                'only accept a C or R for the p_pri parameter.');
```

```

    WHEN enrollment_present THEN
        RAISE_APPLICATION_ERROR
            (-20239, 'The student with ID' || p_studid ||
                ' exists in the enrollment table thus records' ||
                ' will not be removed.');
```

```

END remove_student;

BEGIN
    SELECT trunc(sysdate, 'DD')
        INTO v_current_date
        FROM dual;
END student_api;
```

---