Oracle MOOC: Introduction to PL/SQL Program Units

Week 3

Homework for Lesson 3: Working with Packages and Triggers

Homework is your chance to put what you've learned in this lesson into practice. This homework is not "graded" and you are encouraged to write additional code beyond what is asked.

Note:

- The solutions to the homework are NOT provided. We encourage you to try it out and discuss in the course forum for further learning.
- The homework is NOT mandatory to get the course completion award.
- Post your questions, comments, or suggestions (if any) in the course forum @ https://community.oracle.com/community/technology_network_community/moocs/plsql-program-units

Prerequisites:

- Before starting this tutorial, you should have:
  - Completed the setup instructions provided on the course page.
  - Solved the assignments given in the homework document for week 1.

Watch out for:

- Reference video that discussed the corresponding concept in this MOOC.
- Hints that can help you solve the assignment.

Assignment 1: In this practice, you create the package specification and package body for a new package named `STUDENT_PKG` containing a copy of the `ADD_STUDENT`, `UPD_STUDENT`, `DEL_STUDENT` procedures and the `GET_EXAM_ELIGIBILITY` function **you created while solving the homework for week 1.** You then invoke the constructs in the package by using sample data.

Oracle MOOC: PL/SQL Program Units

- Create the package specification including the procedures and function headings as public constructs.

Sample output:

```
Script Output
Task completed in 0.51 seconds

Package STUDENT_PKG compiled

No errors.
```

- Create the package body with the implementations for each of the subprograms.

Hint: Reuse the procedures and functions created while solving the homework for week 1 while creating the package specification and the package body.

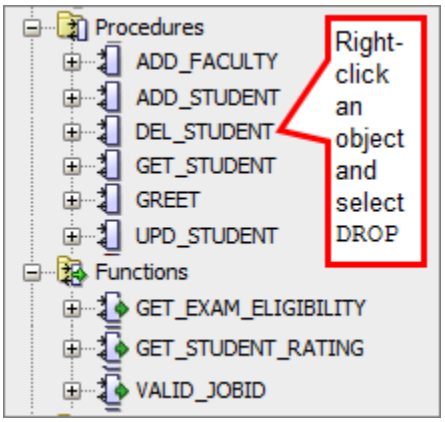
Sample output:

```
Script Output
Task completed in 0.133 seconds

Package Body STUDENT_PKG compiled

No errors.
```

- As you included the procedures `ADD_STUDENT`, `UPD_STUDENT`, `DEL_STUDENT` and the function `GET_EXAM_ELIGIBILITY` in the `STUDENT_PKG` package, delete the standalone versions of them.

Hint: You can delete the standalone procedures and function either from the SQL Developer's object browser or by using the `DROP` statements.

Oracle MOOC: PL/SQL Program Units



- Invoke the `ADD_STUDENT` procedure from the packages, by passing the values `900`, `Ricky`, and `04-JAN-2018` as parameters. Query the `AD_STUDENT_DETAILS` table to see the result.

Sample output:

```
Script Output
Task completed in 345.91 seconds

PL/SQL procedure successfully completed.


STUDENT_ID FIRST_NAME                                          STUDENT_R
---------- -------------------------------------------------- ---------
       900 Ricky                                              04-JAN-18
```

See 3-3: Working with PL/SQL Packages for reference.

Assignment 2: In this practice, you create and invoke a package that contains private and public constructs.

- Create a package specification and a package body called `FACULTY_PKG` that contains the following procedures and function.

Oracle MOOC: PL/SQL Program Units

Construct name	Type	Parameters	Purpose
ADD_FACULTY	**PUBLIC** Procedure	• first_name • last_name • email • jobid: Use 'FA_ST' as the default value. • sal: Use 4500 as the default value. • Use the FACULTY_SEQ sequence to set the faculty_id column. • Set the hire_date column to TRUNC(SYSDATE).	• To add a faculty to the AD_FACULTY_DETAILS table. • The row should be added to the AD_FACULTY_DETAILS table if the VALID_JOBID function returns TRUE; otherwise, alert the user with an appropriate message. **Hint:** Reuse the logic created while solving the homework for week 1.
GET_FACULTY	**PUBLIC** Procedure	• faculty_id – IN parameter • sal- OUT parameter • jobid – OUT parameter	To query the AD_FACULTY_DETAILS table using faculty ID and pass the salary and job ID as OUT parameters.
VALID_JOBID	**PRIVATE** Function	• jobid – input parameter • Returning TRUE or FALSE.	To validate a specified job ID and return a BOOLEAN value of TRUE if the job exists. **Hints:** • Reuse the logic created while solving the homework for week 1. • Don't include it in the package specification.

Oracle MOOC: PL/SQL Program Units

Sample output:

```
Script Output
Task completed in 0.217 seconds

Package FACULTY_PKG compiled

No errors.

Package Body FACULTY_PKG compiled

No errors.
```

- Invoke the `ADD_FACULTY` procedure from the `FACULTY_PKG` package using job ID '`FA_SA`' for faculty '`Jane Harris`' with the email ID `Jane.Harris@xyz.com`. Because job ID '`FA_SA`' does not exist, you should get an error message as per the logic in your procedure.

  Sample output:

```
Invalid Job ID. Try again.


PL/SQL procedure successfully completed.
```

- Invoke the `ADD_FACULTY` procedure from the `FACULTY_PKG` package using job ID '`FA_SF`' for faculty '`Tom Hanry`' with the email ID '`Tom.Hanry@xyz.com`'. Query the `AD_FACULTY_DETAILS` table to verify that the new faculty was added.

Oracle MOOC: PL/SQL Program Units

Sample output:

```
PL/SQL procedure successfully completed.


FIRST_NAME           LAST_NAME                 EMAIL
-------------------- ------------------------- -------------------------
JOB_ID
----------
Tom                  Hanry                     Tom.Hanry@xyz.com
FA_SF
```

See 3-3: Working with PL/SQL Packages for reference.

Assignment 3: In this practice, you create a row level trigger. You also create procedures that are invoked from within this trigger.

- The rows in the `AD_JOBS` table store a minimum and maximum salary allowed for different `JOB_ID` values. You are asked to write code to ensure that faculties' salaries fall in the range allowed for their job type, for insert and update operations.
    a. Create a procedure called `CHECK_SALARY` as follows:
        - The procedure accepts two parameters: Job ID and salary.
        - Query the `AD_JOBS` table using the job ID to determine the minimum and maximum salary for the specified job.
        - If the salary parameter does not fall within the salary range of the job, inclusive of the minimum and maximum, then it should raise an application exception, with the message "`Invalid salary <sal>. Salaries for job <jobid> must be between <min> and <max>.`"

Oracle MOOC: PL/SQL Program Units

Hint: Replace the various items in the message with values supplied by parameters and/or variables populated by queries. Save the file.

Sample output:

```
Procedure CHECK_SALARY compiled

No errors.
```

b. Create a trigger called `CHECK_SALARY_TRG` on the `AD_FACULTY_DETAILS` table that fires before an `INSERT` or `UPDATE` operation on each row:

- The trigger must call the `CHECK_SALARY` procedure to carry out the business logic.
- The trigger should pass the new job ID and salary to the procedure parameters.

Sample output:

```
Trigger CHECK_SALARY_TRG compiled

No errors.
```

c. Test the `CHECK_SALARY_TRG` trigger using the following cases:

- Using your `FACULTY_PKG.ADD_FACULTY` procedure, add faculty `Eleanor Beh` with job ID `FA_AF`. What happens and why?

Oracle MOOC: PL/SQL Program Units

Sample output:

```
Error starting at line : 2 in command -
EXECUTE faculty_pkg.add_faculty('Eleanor','Beh','EBEH','FA_PF')
Error report -
ORA-20100: Invalid salary $4500. Salaries for job FA_PF must be between $15000 and $30000
ORA-06512: at "HR.CHECK_SALARY", line 9
ORA-06512: at "HR.CHECK_SALARY_TRG", line 2
ORA-04088: error during execution of trigger 'HR.CHECK_SALARY_TRG'
ORA-06512: at "HR.FACULTY_PKG", line 26
ORA-06512: at line 1
```

- Update the salary of faculty 109 to $2,000. What happens?

Sample output:

Script Output | Task completed in 0.065 seconds

```
Error starting at line : 1 in command -
UPDATE ad_faculty_details
  SET salary = 2000
WHERE faculty_id = 109
Error report -
ORA-20100: Invalid salary $2000. Salaries for job FA_HOD must be between $20080
and $40000
ORA-06512: at "HR.CHECK_SALARY", line 9
ORA-06512: at
"HR.CHECK_SALARY_TRG", line 2
ORA-04088: error during execution of trigger
'HR.CHECK_SALARY_TRG'
```

Oracle MOOC: PL/SQL Program Units

- Now, change the job ID of faculty 109 to `FA_ST`. What happens and why?

Sample output:

```
Script Output
Task completed in 0.047 seconds

Error starting at line : 1 in command -
UPDATE ad_faculty_details
  SET job_id = 'FA_ST'
WHERE faculty_id = 109
Error report -
ORA-20100: Invalid salary $39000. Salaries for job FA_ST must be between $3000
and $6000
ORA-06512: at "HR.CHECK_SALARY", line 9
ORA-06512: at
"HR.CHECK_SALARY_TRG", line 2
ORA-04088: error during execution of trigger
'HR.CHECK_SALARY_TRG'
```

- Update the salary of faculty 109 to $20,800. What happens?

Sample output:

```
Script Output
Task completed in 0.026 seconds

1 row updated.
```

See 3-5: Working with PL/SQL Triggers for reference.

Assignment 4: In this practice, you create a statement level trigger `DELETE_FACULTY_TRG` that prevents faculties from being deleted during business hours.

Oracle MOOC: PL/SQL Program Units

- Write a statement trigger called `DELETE_FACULTY_TRG` on the `AD_FACULTY_DETAILS` table to prevent rows from being deleted during weekday business hours, which are from 9:00 AM to 6:00 PM.

  Sample output:

```
Trigger DELETE_FACULTY_TRG compiled

No errors.
```

- Attempt to delete faculty with a `JOB_ID` of `FA_ST` either on a weekend (Saturday or Sunday) or during business hours on a weekday. What happens?

  Sample output:

```
Script Output x
Task completed in 0.912 seconds
Error starting at line : 56 in command -
DELETE FROM ad_faculty_details
WHERE job_id = 'FA_ST'
Error report -
SQL Error: ORA-20150: Faculty records cannot be deleted during the business
       hours of 9AM and 6PM
ORA-06512: at "ORA26.DELETE_FACULTY_TRG", line 6
ORA-04088: error during execution of trigger 'ORA26.DELETE_FACULTY_TRG'
```

See 3-5: Working with PL/SQL Triggers for reference.

Congratulations! You successfully practiced the concepts discussed in week 3.