Oracle MOOC: Introduction to PL/SQL Program Units

Week 1

Homework for Lesson 1: Working with Stored Procedures and Functions

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:

- Ensure you complete the setup instructions provided on the course page before attempting the homework.
- The solutions to the homework are NOT provided. We encourage you to take the homework as a challenge and provide your own solutions. You can also use the course forum to collaborate on the solution with your fellow students.
- 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

Watch out for:

 - Reference video that discussed the corresponding concept in this MOOC.

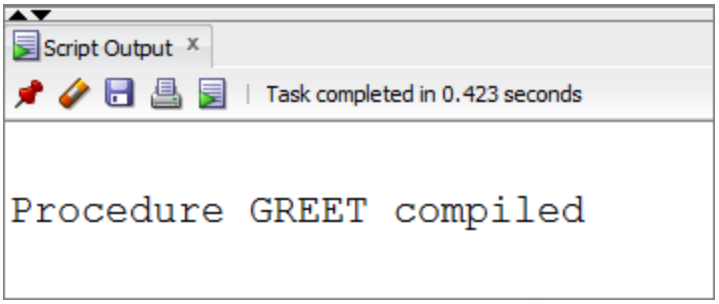
 - Hints that can help you solve the assignment.

Assignment 1: In this practice, you convert an anonymous PL/SQL block into a stored procedure.

- Open the `lab_01_01.sql` file and modify the script to convert the anonymous block to a procedure called `greet`.
- Execute the script to create the procedure.

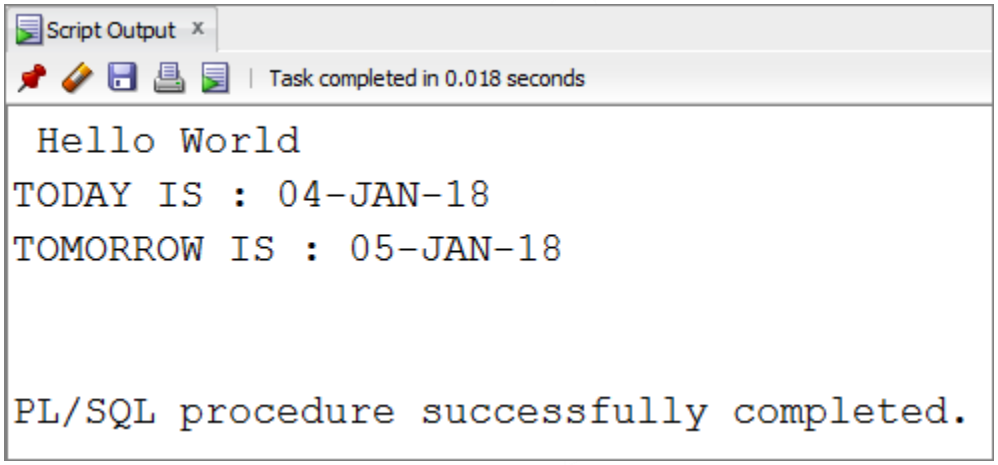
Oracle MOOC: PL/SQL Program Units

Sample output:



- Create and execute an anonymous block to invoke the `greet` procedure.

Sample output:



See 1-3: Creating Stored Procedures for reference.

Hints:

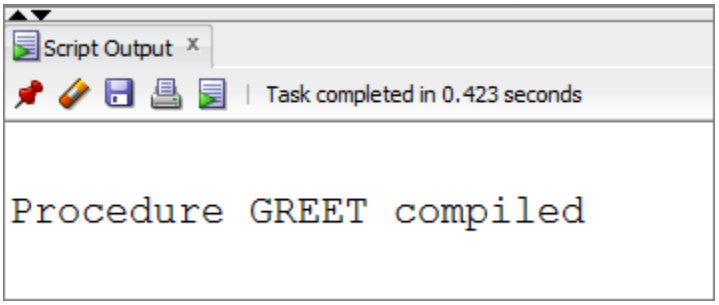
- Don't include the `SET SERVEROUTPUT ON` command in your stored procedure.
- Ensure that you enable `SERVEROUTPUT` at the beginning of the anonymous block that invokes the stored procedure `greet`.

Oracle MOOC: PL/SQL Program Units

Assignment 2: In this practice, you modify the procedure greet to accept a name as an input parameter and print `Hello <name>` instead of printing `Hello World`.

- Modify the procedure `greet` to accept an argument of type `VARCHAR2`.
- Modify the logic in the executable section to print `Hello <name>` instead of printing `Hello World`.
- Execute the script to create the procedure.

  Sample output:

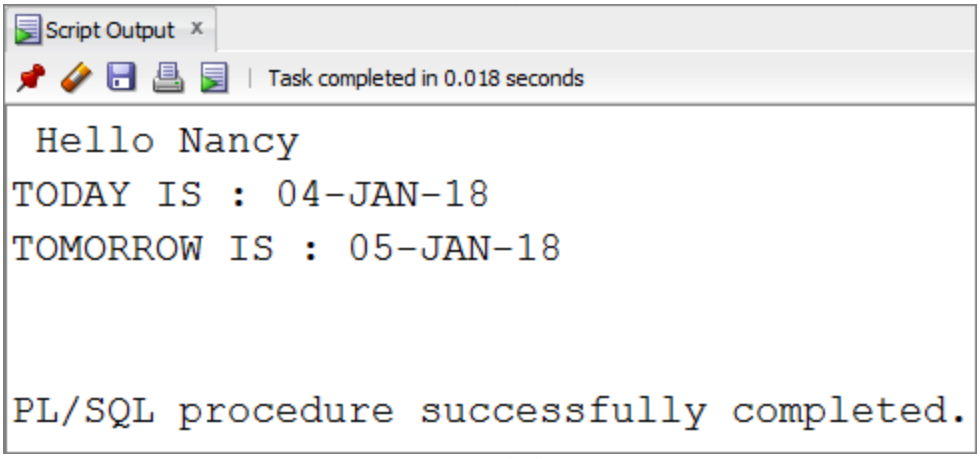


- Create and execute an anonymous block to invoke the `greet` procedure passing `'Nancy'` as the input parameter.

  Hint: Use the `CREATE OR REPLACE PROCEDURE` statement to modify the existing stored procedure `greet`.

Oracle MOOC: PL/SQL Program Units

Sample output:



See 1-3: Creating Stored Procedures for reference.

Assignment 3: In this practice, you create a set of stored procedures to manage the students' data in the Academic database.

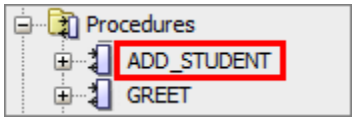
Stored procedure	Purpose	Arguments
ADD_STUDENT	To add a student row into the AD_STUDENT_DETAILS table.	Student ID First name Registration year
UPD_STUDENT	To update a student's email ID in the AD_STUDENT_DETAILS table.	Student ID Email ID
DEL_STUDENT	To delete a student row from the AD_STUDENT_DETAILS table.	Student ID
GET_STUDENT	To query the AD_STUDENT_DETAILS table.	Student ID First name Registration year

1. ADD_STUDENT procedure:

Oracle MOOC: PL/SQL Program Units

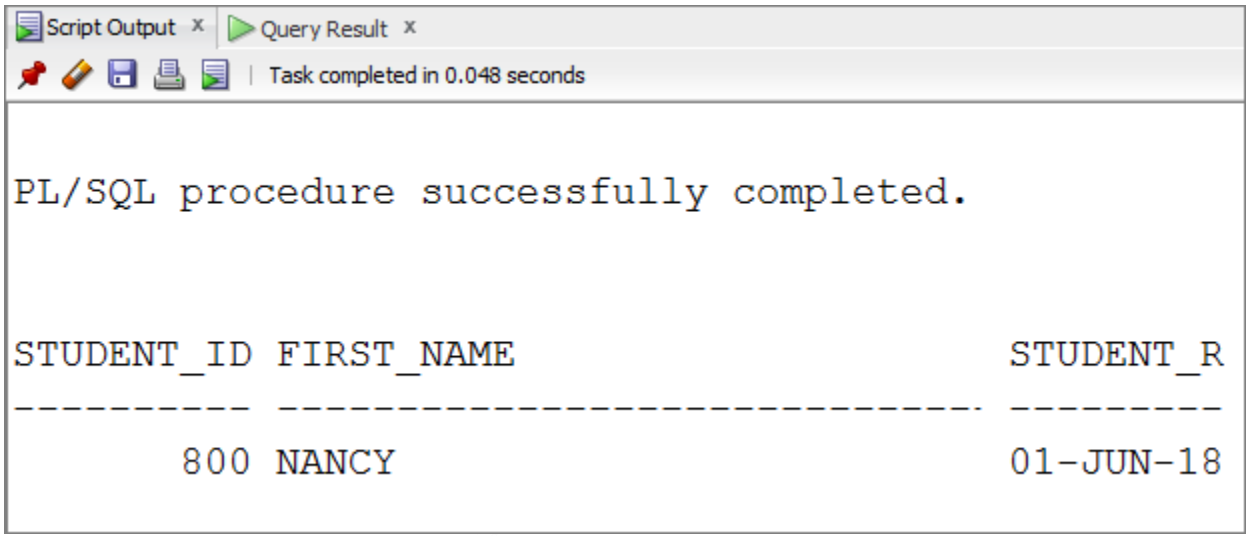
a. Create and compile the `ADD_STUDENT` procedure.

b. To view the newly created procedure, click the **Procedures** node in the Object Navigator. If the newly created procedure is not displayed, right-click the **Procedures** node, and then select **Refresh** from the shortcut menu. The new procedure is displayed as follows:



c. Invoke the procedure with `800` as the student ID, `NANCY` as first name, and `01-JUN-2018` as the registration year. Query the `AD_STUDENT_DETAILS` table and view the results.

Sample output:



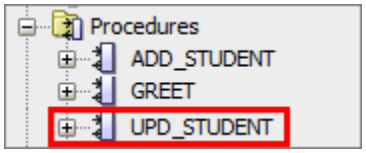
d. Invoke the procedure with `740` as the student ID, `JOANN` as student name, and `01-JAN-16` as the registration year. Query the `AD_STUDENT_DETAILS` table and view the results. What happens and why?

2. `UPD_STUDENT` procedure:

Oracle MOOC: PL/SQL Program Units

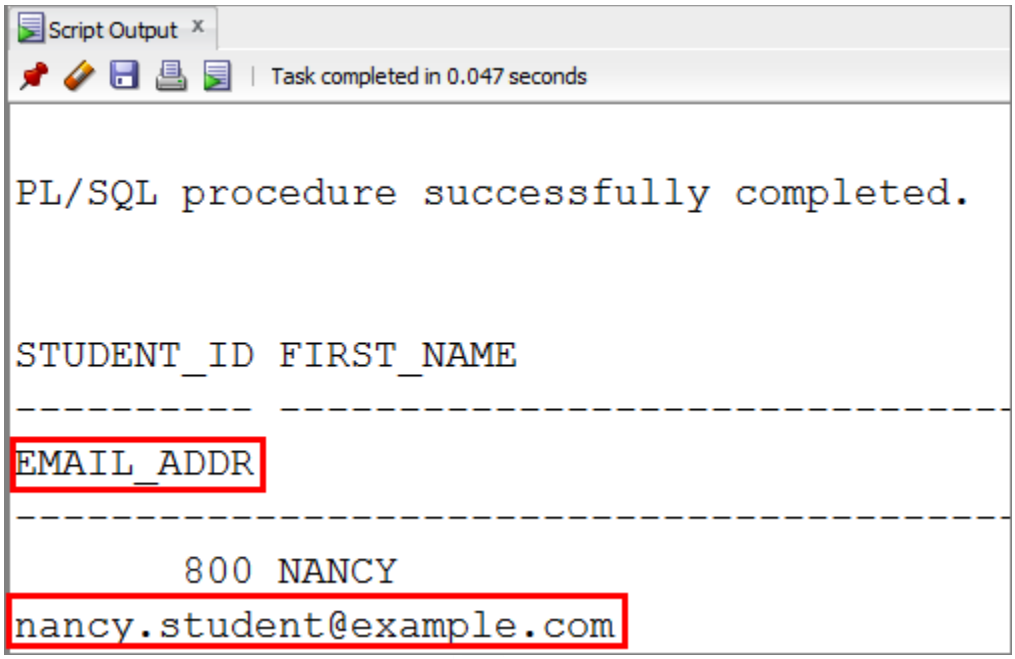
a. Create and compile `UPD_STUDENT` to update the student email ID. Provide the student ID and email ID as input parameters.

b. To view the newly created procedure, click the **Procedures** node in the Object Navigator. The new procedure is displayed as follows:



c. Invoke the procedure to change the student email ID of the student ID `800` to `nancy.student@example.com`. Query the `AD_STUDENT_DETAILS` table and view the results.

Sample output:

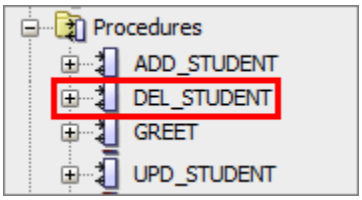


3. `DEL_STUDENT` procedure:

a. Create and compile `DEL_STUDENT` to delete a student row. Provide the student ID as input parameter.

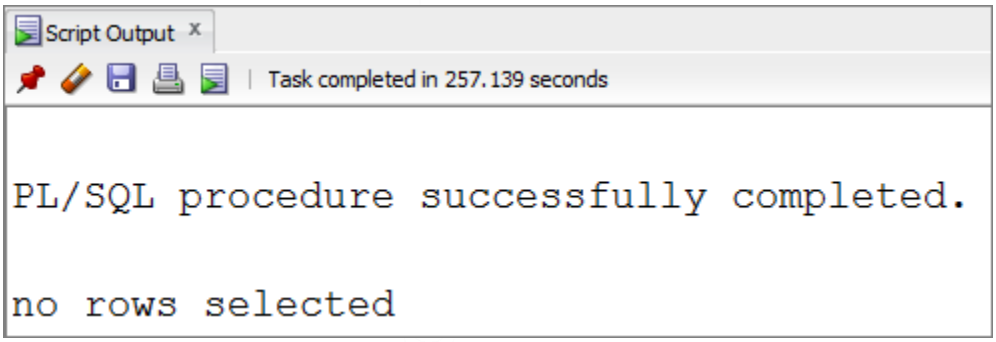
Oracle MOOC: PL/SQL Program Units

b. To view the newly created procedure, click the **Procedures** node in the Object Navigator. The new procedure is displayed as follows:



c. Invoke the procedure using the student ID `800`. Query the `AD_STUDENT_DETAILS` table and view the results.

Sample output:

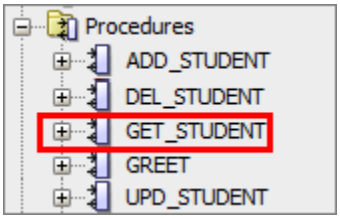


4. `GET_STUDENT` procedure:

a. Create and compile `GET_STUDENT` to query the `AD_STUDENT_DETAILS` table, retrieving the first name and registration year for a student when provided with the student ID.

Hint: Use `IN` parameter for student ID and `OUT` parameters for first name and registration year.

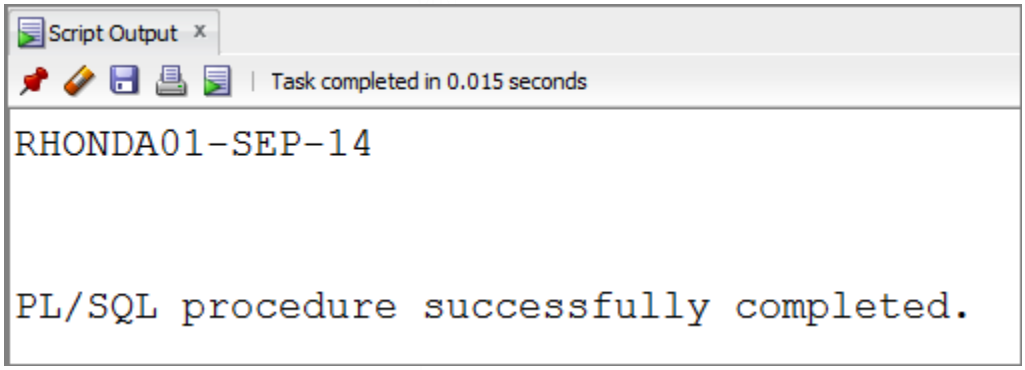
b. To view the newly created procedure, click the **Procedures** node in the Object Navigator. The new procedure is displayed as follows:



Oracle MOOC: PL/SQL Program Units

c. Write an anonymous block to invoke the `GET_STUDENT` procedure using PL/SQL variables for the two `OUT` parameters—one for the first name and the other for the registration year. Display the first name and registration year for student ID `740`.

Sample output:



d. Invoke the procedure again, passing student ID as `600`. What happens and why?

See 1-3: Creating Stored Procedures for reference.

Assignment 4: In this practice, you create and invoke a function that returns the eligibility status of a student to appear for exams.

- Create and compile a function called `GET_EXAM_ELIGIBILITY` to return the exam eligibility status. This function must accept student ID as its input parameter and return the following values based on the value of the `eligibility_for_exams` column in `AD_STUDENT_ATTENDANCE` table.

`eligibility_for_exams` value	Function return value
'Y'	'Eligible for Exams'
'N'	'Not Eligible for Exams'

Oracle MOOC: PL/SQL Program Units

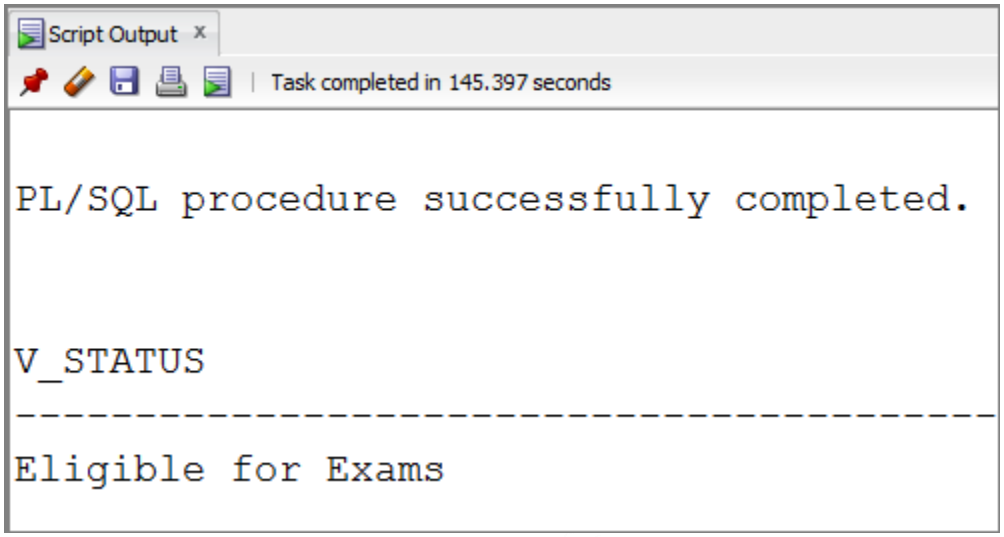
- Create a `VARCHAR2` host variable called `v_status`, allowing a length of 35 characters.

  Hint: Use `VARIABLE <variable name>` to create the host variable.

- Invoke the function with student ID `740` to return the value in the host variable, and then print the host variable to view the result.

  Hint: Use the `EXECUTE` command to invoke the function.

  Sample output:



See 1-4: Creating Stored Functions for reference.

Assignment 5: In this practice, you create a function that returns a rating for a student based on his or her marks in the lab exam for a given course in the `AD_EXAM_RESULTS` table. Later, you will invoke this function directly from a SQL query and print the result.

- Create the `GET_STUDENT_RATING` function, which accepts parameter values for the student ID and course ID. Based on the formula of 1 start for every 20 marks, the total number of stars will be calculated based on the student's total marks and returned by this function.

Oracle MOOC: PL/SQL Program Units

- Use the function in a `SELECT` statement against the `AD_STUDENT_COURSE_DETAILS` table for students in course `192` (Cost Accounting).

Sample output:

Query Result

SQL | All Rows Fetched: 3 in 0.047 seconds

	STUDENT_ID	COURSE_ID	Student Rating
1	750	192	***
2	760	192	****
3	780	192	***

See 1-4: Creating Stored Functions for reference.

Assignment 6: In this practice, you create a procedure `ADD_FACULTY` to insert a new faculty into the `AD_FACULTY_DETAILS` table. The procedure calls a function `VALID_JOBID` to check whether the job ID specified for the new faculty exists in the `AD_JOBS` table.

- Create a function called `VALID_JOBID` to validate a specified job ID and return a `BOOLEAN` value of `TRUE` if the job exists.
- Execute the command in `lab_01_02.sql` file to create `FACULTY_SEQ` sequence object to set the `faculty_id` column of the `AD_FACULTY_DETAILS` table.
- Create the `ADD_FACULTY` procedure 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. Provide the following parameters:
  - `first_name`
  - `last_name`
  - `email`

Oracle MOOC: PL/SQL Program Units

- `job`: 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)`.

- Call `ADD_FACULTY` for the name `'Jane Harris'`, job `'Senior Faculty'`, email `Jane.Harris@xyz.com` leaving other parameters with their default values. What is the result?

  Sample output:

```
Script Output x
Task completed in 201.359 seconds
Invalid Job ID. Try again.


PL/SQL procedure successfully completed.
```

- Add another faculty with the name `'Joe Miller'`, job `'FA_SF'`, email `Joe.Miller@xyz.com` leaving other parameters with their default values. What is the result?

  Sample output:

```
Script Output x
Task completed in 373.248 seconds

PL/SQL procedure successfully completed.
```

- Query the `AD_FACULTY_DETAILS` table and verify if the new faculty is inserted.

Oracle MOOC: PL/SQL Program Units

Sample output:

Script Output | Query Result

SQL | All Rows Fetched: 1 in 0.016 seconds

	FACULTY_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
1	15	Joe	Miller	Joe.Miller@xyz.com	(null)	08-JAN-18	FA SF	4500

Congratulations! You successfully practiced the concepts discussed in week 1.