

Introduction to PL/SQL

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Claudia Dinuca (claudia.dinuca@oracle.com) has a non-transferable license to use this Student Guide.

Course Road Map

Lesson 1: Course Overview

Unit 1: Introducing PL/SQL

Unit 2: Programming with PL/SQL

Unit 3: Working with PL/SQL Code

▶ **Lesson 2: PL/SQL Overview**

▶ Lesson 3: Declaring PL/SQL Variables

▶ Lesson 4: Writing Executable Statements

▶ Lesson 5: Using SQL Statements in PLSQL Programs

You are here!

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

You are in lesson 2, which is part of Unit 1: Introducing PL/SQL.

Objectives

After completing this lesson, you should be able to do the following:

- Explain the need for PL/SQL
- Explain the benefits of PL/SQL
- Identify the different types of PL/SQL blocks
- Output messages in PL/SQL



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

This lesson introduces PL/SQL and the PL/SQL programming constructs. You also learn about the benefits of PL/SQL.

Agenda

- Understanding the benefits and structure of PL/SQL
- Understanding PL/SQL blocks
- Generating output messages in PL/SQL



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Limitations of SQL

- Performs one operation at a time on the database
- Lacks the capability of logically grouping multiple database operations



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Structured Query Language (SQL) provides a standard interface to access relational databases. All SQL statements are instructions to relational databases.

A SQL statement can query and manipulate a database, one operation at a time. A single SQL statement can create a table in a database, query data from the database, modify data in the database, or delete data in the database; however each SQL statement can perform any one of these operation only at a time.

Let's look at a scenario in which you transfer funds by using a Bank application. A typical funds transfer will have the following steps to be performed on the database.

1. Deduct funds from the sender's account (an update operation on the corresponding row).
2. Add funds to the receiver's account (an update operation on the receiver's account).

This scenario may also have conditional checks such as:

1. You may want to check whether there are sufficient funds in the sender's account.
2. You may also want to check whether the current transaction will take the balance in the sender's account below minimum balance.

SQL provides for all these conditional checks and updates as independent individual operations. PL/SQL enables you to group such operations which together make a logical unit in the application context.

Why PL/SQL?

- Enables modularization in the application
- Provides better security
- Enables maintainability
- Provides exception handling



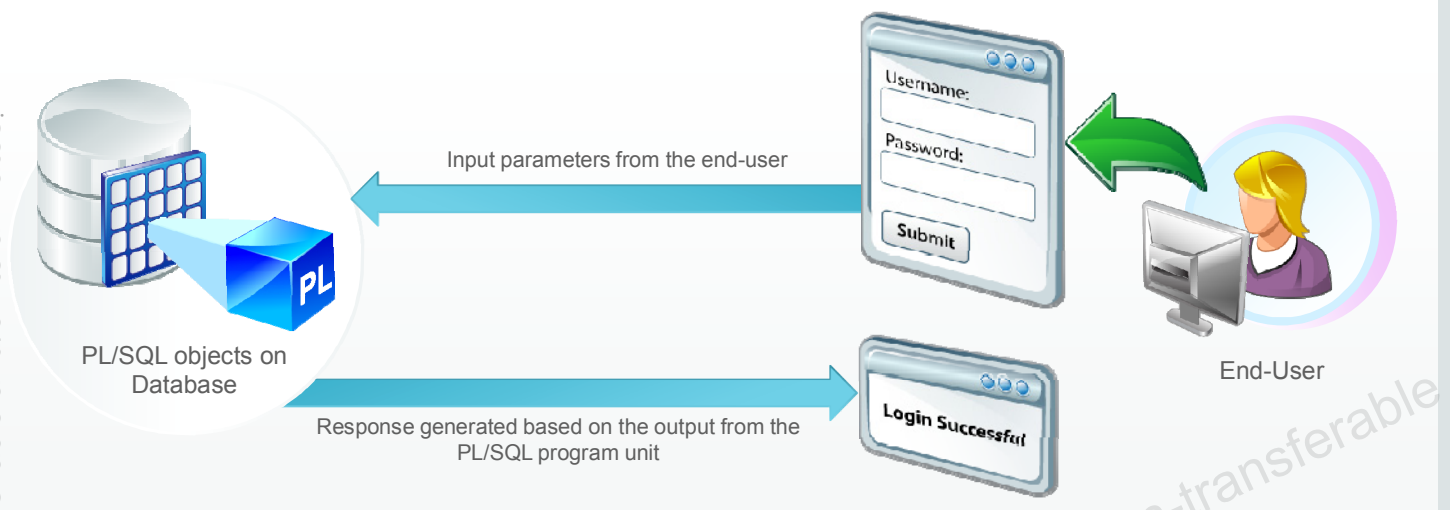
ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

PL/SQL provides the following features, which enable efficient application development:

- **Modularization:** Large software systems are developed as modules first, which are later integrated together. The modules communicate through interfaces and work together as a single application.
- **Security:** The implementation of the module is invisible to the end user. That is, the end user knows only the input and output of the module, which is the interface of the module. Security is provided by hiding the implementation details.
- **Maintainability:** When you want to upgrade the performance of a module, you can do it independent of the other modules in the application.
- **Exception handling:** PL/SQL enables you to handle exceptions efficiently. You can define separate blocks for dealing with exceptions. You learn more about exception handling in the lesson titled “Handling Exceptions.”

Why PL/SQL



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Consider a scenario where the HR manager of a company has to make some changes to the database. The HR manager cannot directly access the database. The system will first have to authenticate the HR manager. He/she must provide security credentials such as a username and password. These credentials are then verified against the credentials in the user database to authenticate the HR manager.

The HR application would, therefore, provide a user interface that would accept the required credentials from the end user.

These credentials are provided as parameters to a PL/SQL object on the database.

Note: Practical implementation of this scenario would include other intermediary infrastructure components. For simplicity, we have considered only the user interface and the database components.

The PL/SQL program units reside on the database and hide the implementation from end users. A PL/SQL unit that is defined for user authentication would accept credentials as parameters and perform the authentication process on the database.

In this case, the PL/SQL program unit would accept a username and password from the end user, and verify them against those that exist in the database. The authentication process would require execution of multiple SQL statements. A PL/SQL program unit groups these SQL statements into a single program unit, which can be explicitly invoked.

About PL/SQL



- Stands for “Procedural Language Extension to SQL”
- Integrates procedural constructs with SQL
- Provides a block structure for executable units of code
- Provides procedural constructs such as:
 - Variables, constants, and data types
 - Control structures: Loops, conditional statements
- Enables writing reusable program units

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

PL/SQL is a Procedural Language extension to SQL. PL/SQL integrates SQL statements into procedural language structures such as conditional statements and loop constructs.

Consider a scenario where you want to run an operation such as “Giving 10% bonus to all the employees in the company” on the database. If you execute this operation through SQL, you may have to execute an `UPDATE` statement for each row in the Employees table.

With PL/SQL, you can execute the `UPDATE` statement (SQL statement) on each row of the Employees table by using a loop construct (procedural construct). Thus, PL/SQL makes things simpler.

Maintaining and debugging code is also made easier with such a structure because you can easily understand the flow and execution of the program unit.

Benefits of PL/SQL

- Integration of procedural constructs with SQL
- Improved performance
- Modularized program development
- Integration with Oracle tools
- Portability
- Exception handling
- Support for Object Oriented Programming

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Integration of procedural constructs with SQL: Being a declarative language, SQL does not have procedural constructs such as conditional statements and loop statements. With PL/SQL, you can combine the data manipulating capability of SQL with the processing power of procedural languages.

Improved performance: You can create a program unit in PL/SQL, which can execute multiple SQL statements. When a SQL statement is executed, there is data exchange between the client and the database server. In the case of a PL/SQL program unit, which contains multiple SQL statements, there is exchange of data only once as a single PL/SQL program unit. This, in turn, improves application performance because of reduced data exchange.

Modularized program development: The basic unit in all PL/SQL programs is the block. Blocks can be in a sequence or they can be nested in other blocks. Modularized program development has the following advantages:

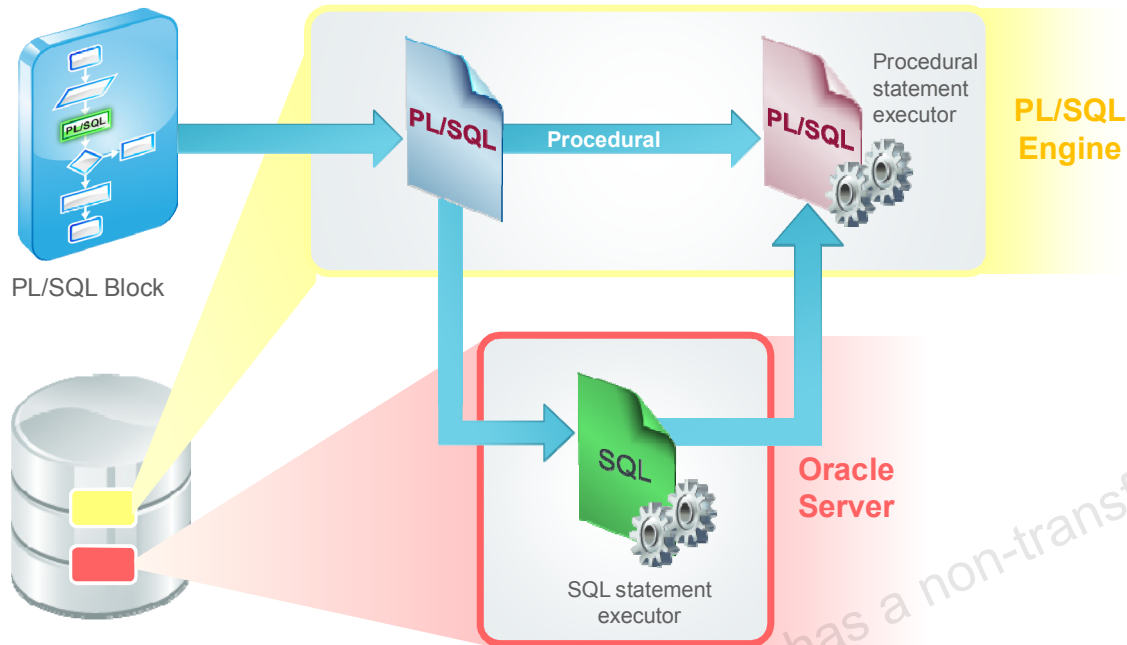
- You can group logically related statements within blocks.
- You can nest blocks inside larger blocks to build powerful programs.
- You can break down your application into smaller modules. If you are designing a complex application, PL/SQL allows you to break down the application into smaller, manageable, and logically related modules.
- You can easily maintain and debug code.

Integration with tools: The PL/SQL engine is integrated in Oracle tools such as Oracle Forms and Oracle Reports. When you use these tools, the locally available PL/SQL engine processes the procedural statements; only the SQL statements are passed to the database.

Portability: PL/SQL programs can run anywhere that an Oracle Server runs, irrespective of the operating system and platform. You do not need to customize them for each new environment. You can write portable program packages and create libraries that can be reused in different environments.

Exception handling: PL/SQL enables you to handle exceptions efficiently. You can define separate blocks for dealing with exceptions. You learn more about exception handling in the lesson titled “Handling Exceptions.”

PL/SQL Runtime Architecture



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The execution of a PL/SQL block has two parts:

1. Execution of the procedural statements
2. Execution of the SQL statements

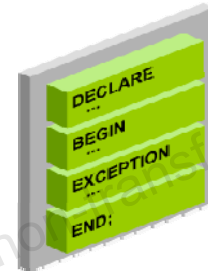
The procedural logic is executed on a PL/SQL engine and the SQL logic is executed on the Oracle Server.

The PL/SQL engine is a virtual machine that resides in memory and processes the PL/SQL m-code instructions. You can install the PL/SQL engine in the database or in an application development tool such as Oracle Forms. It processes all the procedural statements and passes the SQL statements to the SQL Engine on the Oracle Server.

If the PL/SQL unit does not have any SQL statements, the entire PL/SQL unit is processed by the PL/SQL engine. The SQL engine may invoke a PL/SQL statement executor to execute any function calls that are present in the SQL statements.

PL/SQL Block Structure

- **DECLARE (optional)**
 - Variables, cursors, user-defined exceptions
- **BEGIN (mandatory)**
 - SQL statements
 - PL/SQL statements
- **EXCEPTION (optional)**
 - Actions to perform when exceptions occur
- **END (mandatory)**



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

The slide shows a basic PL/SQL block. A PL/SQL block consists of four sections:

- **DECLARE (optional):** The declarative section begins with the keyword `DECLARE` and ends when the executable section starts. You may or may not have a declaration section, because you may not always declare variables or constants for the PL/SQL block. You can declare variables, constants, cursors, and user-defined exceptions here.
- **BEGIN (required):** The executable section begins with the keyword `BEGIN`. This is the section where you write the procedural or SQL statements that must be executed. You should have at least one statement. The executable section of a PL/SQL block can include any number of PL/SQL blocks.
Note: Every `BEGIN` statement should have a corresponding `END` statement to demarcate the PL/SQL block.
- **EXCEPTION (optional):** The exception section is nested within the executable section. This section begins with the keyword `EXCEPTION`.
- **END (required):** All PL/SQL blocks must conclude with an `END` statement. Observe that `END` is terminated with a semicolon.

Agenda

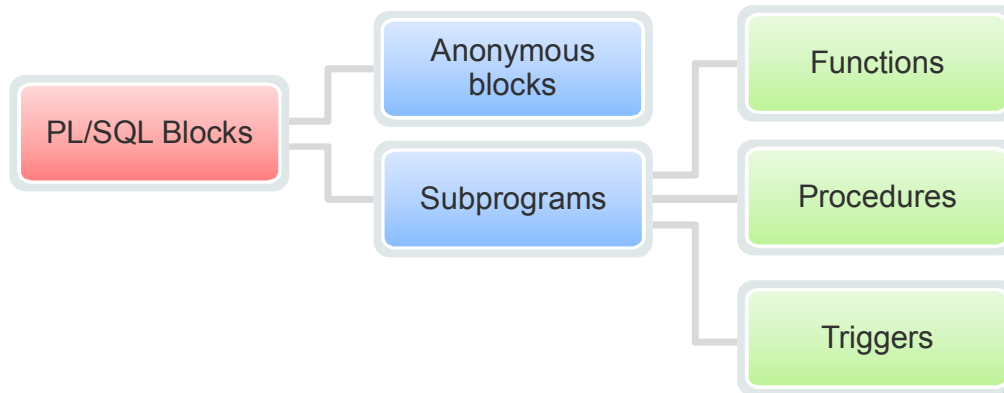
- Understanding the benefits and structure of PL/SQL
- Understanding PL/SQL blocks
- Generating output messages in PL/SQL



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Block Types



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

A PL/SQL program comprises one or more blocks.

There are two types of PL/SQL blocks:

- Anonymous blocks
- Named PL/SQL blocks, which are also termed as subprograms

Anonymous blocks: Anonymous blocks are unnamed blocks. They are declared inline at the point in an application where they are to be executed, and are compiled each time the application is executed. These blocks are not stored in the database. They are passed to the PL/SQL engine for execution at run time. If you want to execute the same block again, you may have to rewrite the block.

Subprograms: Subprograms are named PL/SQL blocks that are stored in the database. They are reusable blocks, and can be invoked by the application as per requirement. Subprograms are stored on the database server, and can also be invoked by other subprograms.

Procedures: Procedures are named PL/SQL blocks that can be explicitly executed. Procedures help in modularizing application logic.

Functions: Functions, like procedures, are named PL/SQL blocks that are explicitly invoked, which are capable of returning a value.

Both procedures and functions can accept some input parameters and process based on these parameters.

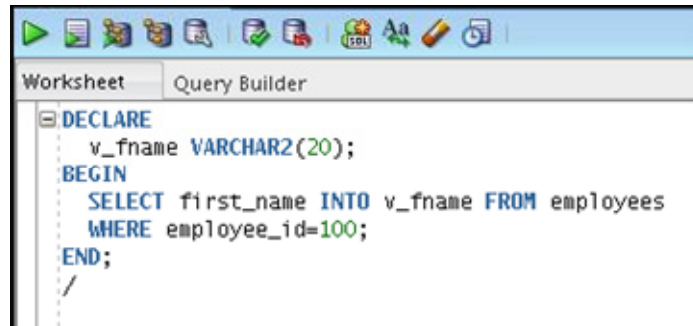
Triggers: Triggers are PL/SQL blocks that are conditionally invoked based on an event or operation that occurs in the database context.

For instance, if you want to write all the operations that occur in the database to a log file, you can define a trigger, which would do just that. You can define the trigger such that each time an insert, a delete, or an update is performed on the database, an entry is made to the log.

We discuss Procedures, Functions, and Triggers in the *Oracle Database 12c: Develop PL/SQL Program Units* course.

Examining an Anonymous Block

An anonymous block in the SQL Developer workspace:

A screenshot of the SQL Developer workspace. The top toolbar contains various icons for execution and editing. Below the toolbar, there are two tabs: 'Worksheet' and 'Query Builder'. The 'Worksheet' tab is active, displaying a PL/SQL anonymous block. The code is as follows:

```
DECLARE
  v_fname VARCHAR2(20);
BEGIN
  SELECT first_name INTO v_fname FROM employees
  WHERE employee_id=100;
END;
```

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

To create an anonymous block by using SQL Developer, enter the block in the workspace (as shown in the slide).

Example

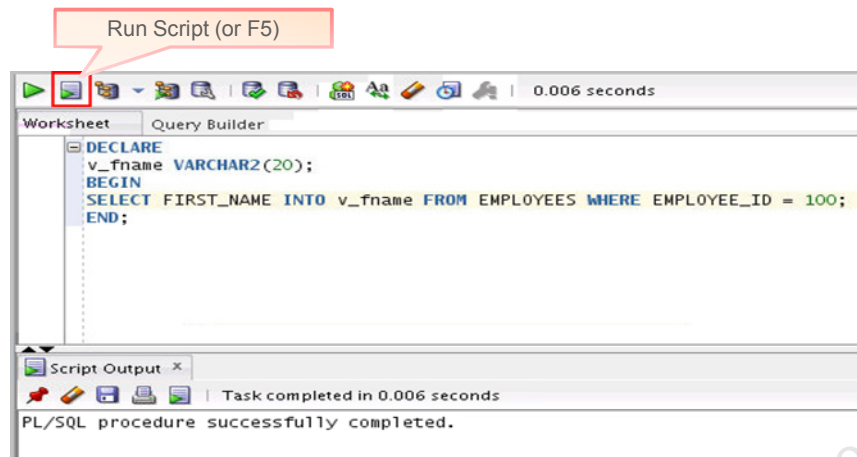
The example block has a declarative section and an executable section. You need not pay attention to the syntax of the statements in the block at this point in time.

The anonymous block gets the `first_name` of the employee whose `employee_id` is 100, and stores it in a variable called `v_fname`.

Note: You can also write PL/SQL code through SQL*Plus, which is a command-line tool.

Executing an Anonymous Block

Click the Run Script icon to execute the anonymous block:



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

To execute an anonymous block, click the Run Script icon (or press F5).

Note: The message “PL/SQL procedures successfully completed” is displayed in the Script Output window after the block is executed.

Agenda

- Understanding the benefits and structure of PL/SQL
- Understanding PL/SQL blocks
- Generating output messages in PL/SQL



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Enabling Output of a PL/SQL Block

1. To enable output in SQL Developer, execute the following command before running the PL/SQL block:

```
SET SERVEROUTPUT ON
```

2. Use a predefined Oracle package and its procedure in the anonymous block:
 - DBMS_OUTPUT.PUT_LINE

```
DBMS_OUTPUT.PUT_LINE(' The First Name of the Employee is ' ||  
v_fname);  
...
```

ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

PL/SQL does not have built-in input or output functionality. Therefore, you need to use predefined Oracle packages for input and output. To generate output, perform the following steps:

1. Execute the following command:

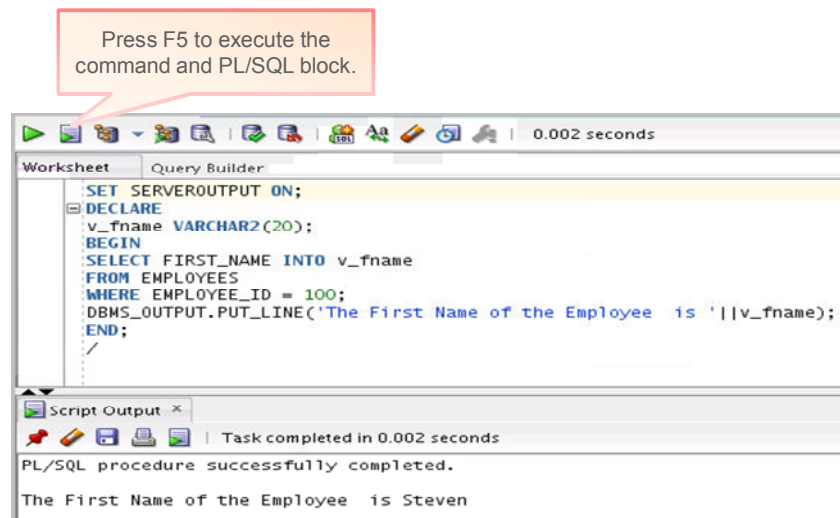
```
SET SERVEROUTPUT ON
```

Note: To enable output in SQL*Plus, you must explicitly issue the SET SERVEROUTPUT ON command.

2. In the PL/SQL block, use the PUT_LINE procedure of the DBMS_OUTPUT package to display the output. Pass the value that must be printed as an argument to this procedure (as shown in the slide). The procedure then outputs the argument.

Note: We discuss Packages in the *Oracle Database 12c: Program Units* course. A package is a logical grouping of different PL/SQL program blocks and other database objects. It may have procedures, functions, triggers, sequences, and other database objects grouped according to context.

Viewing the Output of a PL/SQL Block



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Press F5 (or click the Run Script icon) to view the output for the PL/SQL block. This action:

1. Executes the `SET SERVEROUTPUT ON` command
2. Runs the anonymous PL/SQL block

The output appears on the Script Output tab.

Quiz

Q

A PL/SQL block *must* consist of the following three sections:

- A Declarative section, which begins with the keyword `DECLARE` and ends when the executable section starts
 - An Executable section, which begins with the keyword `BEGIN` and ends with `END`
 - An Exception handling section, which begins with the keyword `EXCEPTION` and is nested within the executable section
- a. True
b. False



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

Answer: b

A PL/SQL block consists of three sections:

- **Declarative (optional):** The optional declarative section begins with the keyword `DECLARE`, and ends when the executable section starts.
- **Executable (required):** The required executable section begins with the keyword `BEGIN` and ends with `END`. This section essentially needs to have at least one statement. Observe that `END` is terminated with a semicolon. The executable section of a PL/SQL block can, in turn, include any number of PL/SQL blocks.
- **Exception handling (optional):** The optional exception section is nested within the executable section. This section begins with the keyword `EXCEPTION`.

Summary

In this lesson, you should have learned how to:

- Explain the need for PL/SQL
- Explain the benefits of PL/SQL
- Identify the different types of PL/SQL blocks
- Output messages in PL/SQL



ORACLE

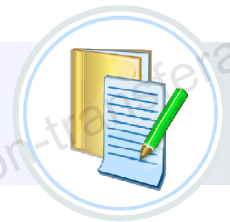
Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

PL/SQL is a language that has programming features that serve as extensions to SQL. SQL, which is a non-procedural language, is made procedural with the PL/SQL programming constructs. PL/SQL applications can run on any platform or operating system on which an Oracle Server runs. In this lesson, you learned how to build basic PL/SQL blocks.

Practice 2: Overview

This practice covers the following topics:

- Identifying PL/SQL blocks that execute successfully
- Creating and executing a simple PL/SQL block



ORACLE

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.

This practice reinforces the basics of PL/SQL covered in this lesson.

- Exercise 1 is a paper-based exercise in which you identify PL/SQL blocks that execute successfully.
- Exercise 2 involves creating and executing a simple PL/SQL block.

Claudia Dinuca (claudia.dinuca@oracle.com) has a non-transferable license to use this Student Guide.