



Database Programming with PL/SQL

1-3 Creating PL/SQL Blocks



Objectives

This lesson covers the following objectives:

- Describe the structure of a PL/SQL block
- Identify the different types of PL/SQL blocks
- Identify PL/SQL programming environments
- Create and execute an anonymous PL/SQL block
- Output messages in PL/SQL

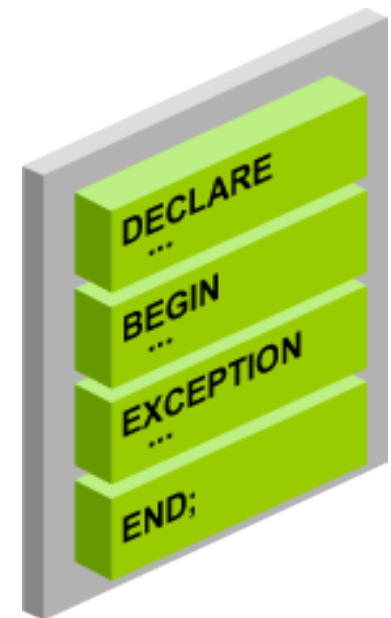
Purpose

- Here you will learn the structure of a PL/SQL block and create one kind of block: an anonymous block.
- Later in the course, you will learn to create Procedures, Functions, and Packages using the basic structure found in anonymous blocks.
- After learning about the different environments into which you can develop your PL/SQL programs, you will also begin coding PL/SQL in the Application Express development environment.

PL/SQL Block Structure

A PL/SQL block consists of three sections.

Section	Description
Declarative (optional)	The declarative section begins with the keyword <code>DECLARE</code> and ends when your executable section starts.
Executable (mandatory)	The executable section begins with the keyword <code>BEGIN</code> and ends with <code>END</code> . Observe that <code>END</code> is terminated with a semicolon. The executable section of a PL/SQL block can include any number of nested PL/SQL blocks.
Exception handling (optional)	The exception section is nested within the executable section. This section begins with the keyword <code>EXCEPTION</code> .



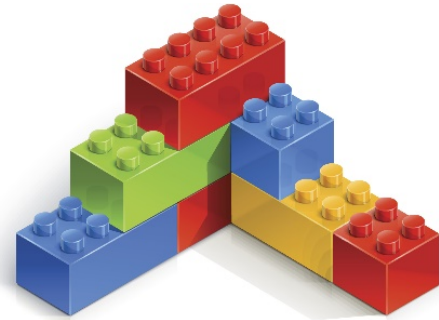
PL/SQL Block Structure Sections

Section	Description	Inclusion
Declarative (DECLARE)	Contains declarations of all variables, constants, cursors, and user-defined exceptions that are referenced in the executable and exception sections.	Optional
Executable (BEGIN ... END;)	Contains SQL statements to retrieve data from the database and PL/SQL statements to manipulate data in the block. Must contain at least one statement.	Mandatory
Exception (EXCEPTION)	Specifies the actions to perform when errors and abnormal conditions arise in the executable section.	Optional

Anonymous Blocks

Characteristics of anonymous blocks:

- Unnamed block
- Not stored in the database
- Declared inline at the point in an application where it is executed
- Compiled each time the application is executed
- Passed to the PL/SQL engine for execution at run time
- Cannot be invoked or called because it does not have a name and does not exist after it is executed



Anonymous Blocks – Basic Structure

- Basic structure of an anonymous block:

```
[ DECLARE ]  
  
BEGIN  
  --statements  
  
[ EXCEPTION ]  
  
END;
```

- The `DECLARE` and `EXCEPTION` keywords/sections are optional.

Examples of Anonymous Blocks

- Executable section only (minimum required)

```
BEGIN
  DBMS_OUTPUT.PUT_LINE('PL/SQL is easy!');
END;
```

- Declarative and executable sections

```
DECLARE
  v_date      DATE := SYSDATE;
BEGIN
  DBMS_OUTPUT.PUT_LINE(v_date);
END;
```

Examples of Anonymous Blocks

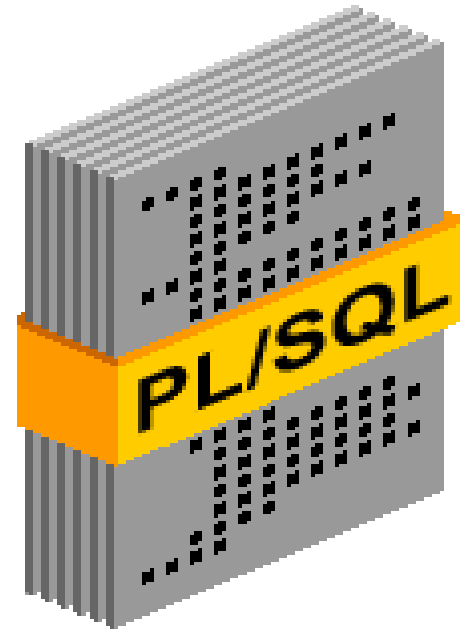
Declarative, executable, and exception sections

```
DECLARE
  v_first_name  VARCHAR2(25);
  v_last_name   VARCHAR2(25);
BEGIN
  SELECT first_name, last_name
     INTO v_first_name, v_last_name
    FROM employees
   WHERE last_name = 'Oswald';
  DBMS_OUTPUT.PUT_LINE ('The employee of the month is: '
                        || v_first_name || ' ' || v_last_name || '.');
EXCEPTION
  WHEN TOO_MANY_ROWS THEN
    DBMS_OUTPUT.PUT_LINE ('Your select statement retrieved
                          multiple rows. Consider using a cursor or changing
                          the search criteria.');
```

END;

The PL/SQL Compiler

- The anonymous block found on the previous slide is automatically compiled when it is executed.
- If the code has errors that prevent it from compiling, it will not execute, but will return the first compile error it detects.



The PL/SQL Compiler

- Every program written in a high-level programming language (C, Java, PL/SQL, and so on) must be checked and translated into binary code (ones and zeros) before it can execute.
- The software that does this checking and translation is called a compiler.
- The PL/SQL compiler executes automatically when needed.
- It checks not only that every command is spelled correctly, but also that any referenced database objects (such as tables) exist, and that the user has the necessary privileges to access them.

Subprograms

Subprograms:

- Are named PL/SQL blocks
- Are stored in the database
- Can be invoked whenever you want depending on your application
- Can be declared as procedures or as functions
 - Procedure: Performs an action
 - Function: Computes and returns a value

```
PROCEDURE name
IS
  -- variable declarations
BEGIN
  -- statements
[EXCEPTION]
END;
```

```
FUNCTION name
RETURN datatype
  -- variable declaration(s)
IS
BEGIN
  -- statements
  RETURN value;
[EXCEPTION]
END;
```

Examples of Subprogram Code Blocks

- Code block to create a procedure called PRINT_DATE.

```
CREATE OR REPLACE PROCEDURE print_date IS
  v_date VARCHAR2(30);
BEGIN
  SELECT TO_CHAR(SYSDATE, 'Mon DD, YYYY')
    INTO v_date
    FROM DUAL;
  DBMS_OUTPUT.PUT_LINE(v_date);
END;
```

- Code block to create a function called TOMORROW.

```
CREATE OR REPLACE FUNCTION tomorrow (p_today IN DATE)
  RETURN DATE IS
  v_tomorrow DATE;
BEGIN
  SELECT p_today + 1 INTO v_tomorrow
    FROM DUAL;
  RETURN v_tomorrow;
END;
```

PL/SQL Programming Environments

There are many tools available from Oracle that provide an environment for developing database-driven applications using PL/SQL.

ORACLE

Application Express	Browser-based, database-driven, application development environment.
SQL Workshop	A component of Application Express.
Application Builder	A component of Application Express.
SQL Developer	An IDE for database development and management.
JDeveloper	An IDE for Java-based development.
NetBeans	An IDE for Java, HTML5, PHP, and C++.

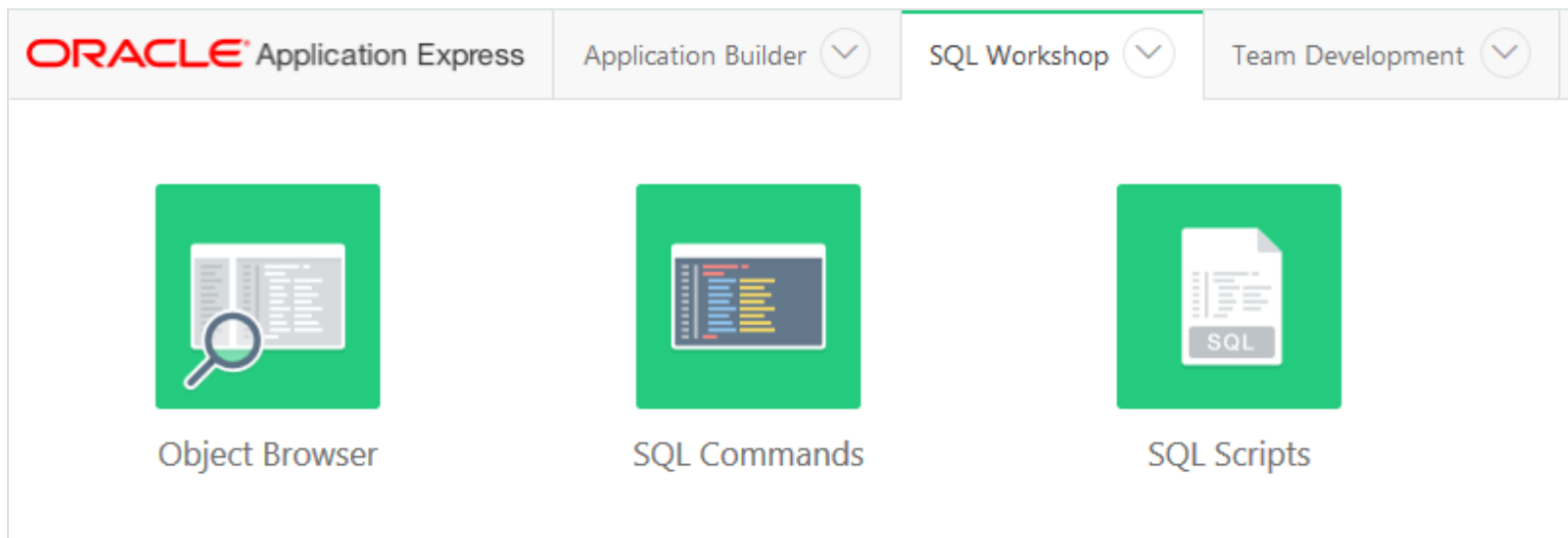
Oracle Application Express

Oracle Application Express is a browser-based web application environment that offers a SQL Workshop component.

The screenshot shows the Oracle Application Express dashboard. At the top, there is a navigation bar with the Oracle logo and the text "Application Express". Below the logo are five tabs: "Application Builder", "SQL Workshop", "Team Development", and "Packaged Apps", each with a dropdown arrow. Below the navigation bar are four large tiles representing different components: "Application Builder" (blue tile with a pencil icon), "SQL Workshop" (green tile with a document and upload icon), "Team Development" (yellow tile with a circular flow icon), and "Packaged Apps" (red tile with a grid icon). Below these tiles are three sections: "Top Applications", "Top Users", and "News and Messages". The "Top Users" section shows a user named "us_z201_plsql_s01" with a blue bar representing their activity and the number "41".

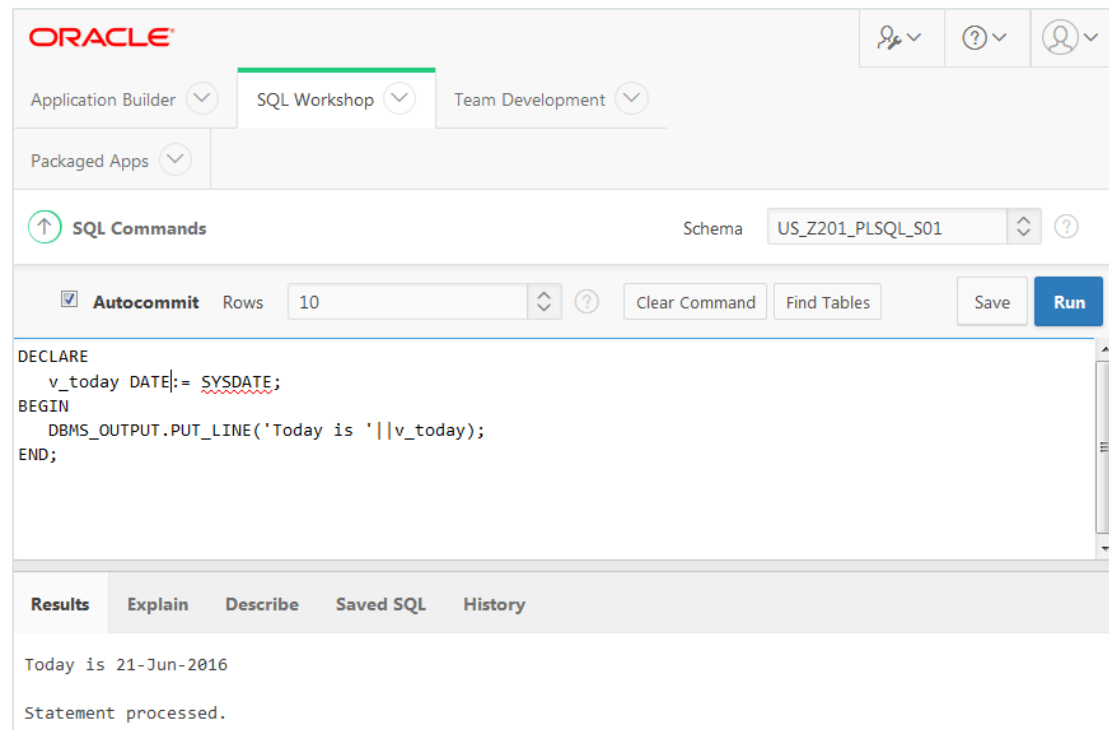
Developing with SQL Workshop

When you log in to Oracle Application Express and choose SQL Workshop, you can choose to use the SQL Commands option to use the SQL command-line editor, or you can choose the SQL Scripts option to work within the Script Editor.



SQL Commands

- As you did in the SQL course, you can use SQL Commands to enter and run a SINGLE SQL statement.
- You also use SQL Commands to enter and run a SINGLE PL/SQL block.



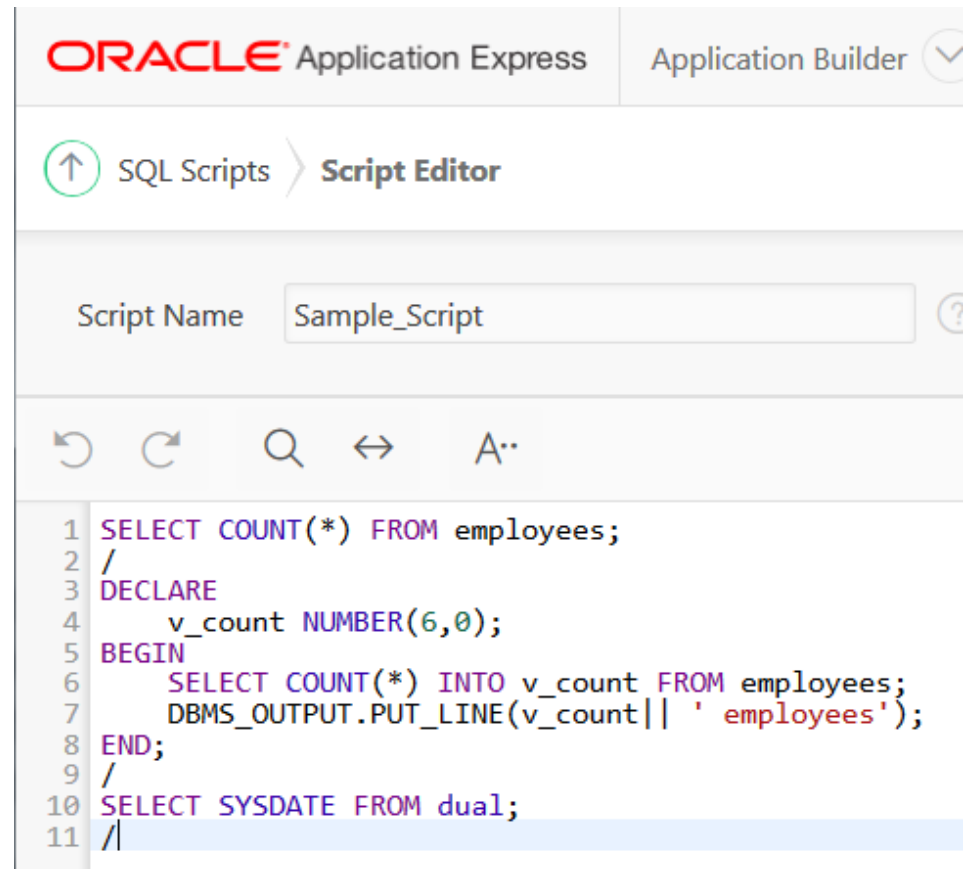
The screenshot shows the Oracle SQL Workshop interface. At the top, the Oracle logo is visible. Below it, there are navigation menus for 'Application Builder', 'SQL Workshop', and 'Team Development'. The 'SQL Commands' section is active, showing a schema of 'US_Z201_PLSQL_S01'. The 'Autocommit' checkbox is checked, and the 'Rows' limit is set to 10. The 'Run' button is highlighted in blue. The SQL code entered is:

```
DECLARE
  v_today DATE := SYSDATE;
BEGIN
  DBMS_OUTPUT.PUT_LINE('Today is '||v_today);
END;
```

The results section shows the output: 'Today is 21-Jun-2016' and 'Statement processed.'.

SQL Scripts

- SQL Scripts can contain one or more SQL statements and/or PL/SQL blocks.
- Use SQL Scripts to enter and run multi-statement scripts.
- In SQL Scripts, anonymous PL/SQL blocks must be followed by a forward slash (/).

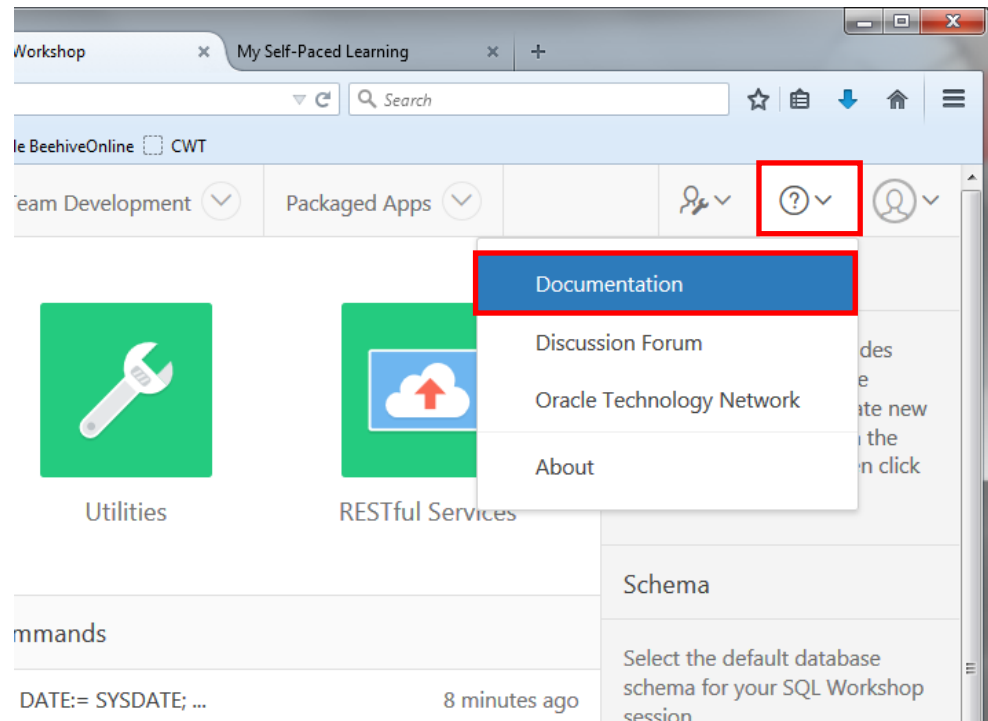


The screenshot shows the Oracle Application Express interface. At the top, there is a navigation bar with "ORACLE Application Express" and "Application Builder". Below this, the breadcrumb "SQL Scripts" is followed by "Script Editor". A text input field labeled "Script Name" contains the value "Sample_Script". Below the input field is a toolbar with icons for undo, redo, search, and other actions. The main area displays a SQL script with line numbers 1 through 11. The script is as follows:

```
1 SELECT COUNT(*) FROM employees;
2 /
3 DECLARE
4     v_count NUMBER(6,0);
5 BEGIN
6     SELECT COUNT(*) INTO v_count FROM employees;
7     DBMS_OUTPUT.PUT_LINE(v_count|| ' employees');
8 END;
9 /
10 SELECT SYSDATE FROM dual;
11 /
```

APEX SQL Workshop Guide

- Oracle provides an extensive HELP function to assist you in using the Application Express environment.
- Object Browser, SQL Commands, and SQL Scripts, as well as many other topics, are covered.



Using DBMS_OUTPUT.PUT_LINE Example

- Look at this simple PL/SQL block and its output.
- How can you display the result?

The screenshot shows the Oracle Application Express interface. At the top, it says "ORACLE Application Express". Below that are navigation tabs: "Application Builder", "SQL Workshop" (which is selected and highlighted with a green border), and "Team Development". Under "SQL Workshop", there is a "SQL Commands" section with an upward arrow icon. Below this, there is a control bar with a checked "Autocommit" checkbox, a "Rows" label, and a dropdown menu set to "10". The main area contains the following PL/SQL code:

```
DECLARE
  v_result NUMBER;
BEGIN
  v_result := 10*25;
END;
```

At the bottom of the interface, there is a navigation bar with five tabs: "Results", "Explain", "Describe", "Saved SQL", and "History". The "Results" tab is currently selected.

Statement processed.

0.00 seconds

Using DBMS_OUTPUT.PUT_LINE Example

- Let's add a call to the PUT_LINE function in the DBMS_OUTPUT package.
- Now you can see the result!

```
 Autocommit Rows 10
```

```
DECLARE
  v_result NUMBER;
BEGIN
  v_result := 10*25;
  DBMS_OUTPUT.PUT_LINE(v_result);
END;
```

Results	Explain	Describe	Saved S
---------	---------	----------	---------

250

Statement processed.

0.00 seconds

Using DBMS_OUTPUT.PUT_LINE

- The DBMS_OUTPUT.PUT_LINE allows you to display results so that you can check that your block is working correctly.
- It allows you to display one character string at a time, although this can be concatenated.

```
DECLARE
  v_emp_count    NUMBER;
BEGIN
  DBMS_OUTPUT.PUT_LINE('PL/SQL is easy so far!');
  SELECT COUNT(*) INTO v_emp_count FROM employees;
  DBMS_OUTPUT.PUT_LINE('There are ' || v_emp_count || '
                        rows in the employees table');
END;
```

Terminology

Key terms used in this lesson included:

- Anonymous PL/SQL block
- Compiler
- Subprograms
- Procedures
- Functions

Summary

In this lesson, you should have learned how to:

- Describe the structure of a PL/SQL block
- Identify the different types of PL/SQL blocks
- Identify PL/SQL programming environments
- Create and execute an anonymous PL/SQL block
- Output messages in PL/SQL

