Oracle PL/SQL

Lab Book

Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Revision No. | Author | Summary of Changes |
| 05-Feb-2009 | 0.1D | Rajita Dhumal | Content Creation |
| 09-Feb-2009 |  | CLS team | Review |
| 02-Jun-2011 | 2.0 | Anu Mitra | Integration Refinements |
| 30-Nov-2012 | 3.0 | HareshkumarChandiramani | Revamp of Assignments and Conversion to iGATE format. |
| 22-Apr--2015 | 4.0 | Kavita Arora | Rearranging the lab questions |
| 9-May-2016 | 5.0 | Kavita Arora | Integration Refinements |
| 19th May 2017 | 6.0 | Kavita Arora | Revamped according to 10 week requirement |

Table of Contents

[Getting Started 4](#_Toc454260319)

[Overview 4](#_Toc454260320)

[Setup Checklist for Oracle 9i 4](#_Toc454260321)

[Instructions 4](#_Toc454260322)

[Learning More (Bibliography if applicable) 4](#_Toc454260323)

[Lab 1. Introduction to PL/SQL and Cursors 5](#_Toc454260324)

[1.1 Identify the problems(if any) in the below declarations: 5](#_Toc454260325)

[1.2 The following PL/SQL block is incomplete. 6](#_Toc454260326)

[1.3 Write a PL/SQL program 6](#_Toc454260327)

[1.4 Write a PL/SQL program 6](#_Toc454260328)

[1.5.Write a PL/SQL block to increase the salary of employees 6](#_Toc454260329)

[Lab 2. Lab 2.Exception Handling 7](#_Toc454260330)

[2.1 The following PL/SQL block attempts to calculate bonus of staff. 7](#_Toc454260331)

[2.2 Rewrite the above block. 8](#_Toc454260332)

[2.3: Write a PL/SQL program 8](#_Toc454260333)

[Lab 3. Database Programming 9](#_Toc454260334)

[3.1. Write a function to compute age. 9](#_Toc454260335)

[3.2 Write a procedure to find the manager of a staff. 9](#_Toc454260336)

[3.3. Write a function to compute the following. 9](#_Toc454260337)

[3.4. Write a procedure that accept Staff\_Code 10](#_Toc454260338)

[3.5. Write a procedure to insert details into Book\_Transaction table. 10](#_Toc454260339)

[Appendices 11](#_Toc454260340)

[Appendix A: Oracle Standards 11](#_Toc454260341)

[Appendix B: Coding Best Practices 12](#_Toc454260342)

[Appendix C: Table of Examples 13](#_Toc454260343)

# Getting Started

## Overview

This lab book is a guided tour for learning Oracle 9i. It comprises ‘To Do’ assignments. Follow the steps provided and work out the ‘To Do’ assignments.

## Setup Checklist for Oracle 9i

Here is what is expected on your machine in order for the lab to work.

Minimum System Requirements

Intel Pentium 90 or higher (P166 recommended)

Microsoft Windows 95, 98, or NT 4.0, 2k, XP,7.

Memory: 32MB of RAM (64MB or more recommended)

Please ensure that the following is done:

Oracle Client is installed on every machine

Connectivity to Oracle Server

## Instructions

For all coding standards refer Appendix A. All lab assignments should refer coding standards.

Create a directory by your name in drive <drive>. In this directory, create a subdirectory Oracle 9i\_assgn. For each lab exercise create a directory as lab <lab number>.

## Learning More (Bibliography if applicable)

Oracle10g - SQL - Student Guide - Volume 1 by Oracle Press

Oracle10g - SQL - Student Guide - Volume 2 by Oracle Press

Oracle10g database administration fundamentals volume 1 by Oracle Press

Oracle10g Complete Reference by Oracle Press

Oracle10g SQL with an Introduction to PL/SQL by Lannes L. Morris-Murphy

1. Introduction to PL/SQL

|  |  |
| --- | --- |
| Goals | * The following set of exercises are designed to implement the following * PL/SQL variables and data types * Create, Compile and Run anonymous PL/SQL blocks |
| Time | 1hr |

## 1.1 Identify the problems(if any) in the below declarations:

DECLARE

V\_Sample1 NUMBER(2);

V\_Sample2 CONSTANT NUMBER(2) ;

V\_Sample3 NUMBER(2) NOT NULL ;

V\_Sample4 NUMBER(2) := 50;

V\_Sample5 NUMBER(2) DEFAULT 25;

Example 1: Declaration Block

## 1.2 The following PL/SQL block is incomplete.

Modify the block to achieve requirements as stated in the comments in the block.

DECLARE --outer block

var\_num1 NUMBER := 5;

BEGIN

DECLARE --inner block

var\_num1 NUMBER := 10;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Value for var\_num1:' ||var\_num1);

--Can outer block variable (var\_num1) be printed here.IfYes,Print the same.

END;

--Can inner block variable(var\_num1) be printed here.IfYes,Print the same.

END;

Example 2: PL/SQL block

## 1.3 Write a PL/SQL program

Write a PL/SQL program to display the details of the employee number 7369.

## 1.4 Write a PL/SQL program

Write a PL/SQL program to accept the Employee Name and display the details of that Employee including the Department Name.

## 1.5.Write a PL/SQL block to increase the salary of employees

Write a PL/SQL block to increase the salary of employees either by 30 % or 5000 whichever is minimum for a given Department\_Code.

Find out 30% of salary, if it is more than 5000, increase by 5000. If it is less than 5000, increase by 30% of salary

1. Lab 2.Exception Handling

|  |  |
| --- | --- |
| **Goals** | Implementing Exception Handling ,Analyzing and Debugging |
| **Time** | 30 mins |

## 2.1 The following PL/SQL block attempts to calculate bonus of staff.

The following PL/SQL block attempts to calculate bonus of staff for a given MGR\_CODE. Bonus is to be considered as twice of salary. Though Exception Handling has been implemented but block is unable to handle the same.

Debug and verify the current behavior to trace the problem.

DECLARE

V\_BONUS V\_SAL%TYPE;

V\_SAL STAFF\_MASTER.STAFF\_SAL%TYPE;

BEGIN

SELECT STAFF\_SAL INTO V\_SAL

FROM STAFF\_MASTER

WHERE MGR\_CODE=100006;

V\_BONUS:=2\*V\_SAL;

DBMS\_OUTPUT.PUT\_LINE('STAFF SALARY IS ' || V\_SAL);

DBMS\_OUTPUT.PUT\_LINE('STAFF BONUS IS ' || V\_BONUS);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('GIVEN CODE IS NOT VALID.ENTER VALID CODE');

END;

Example 3: PL/SQL block

## 2.2 Rewrite the above block.

Rewrite the above block to achieve the requirement.

## 2.3: Write a PL/SQL program

Write a PL/SQL program to check for the commission for an employee no 7369. If no commission exists, then display the error message. Use Exceptions.

1. Database Programming

|  |  |
| --- | --- |
| **Goals** | * The following set of exercises are designed to implement the following * Implement business logic using Database Programming like Procedures and Functions * Implement validations in Procedures and Functions |
| **Time** | 2 Hrs |

**Note:** Procedures and functions should handle validations, pre-defined oracle server and user defined exceptions wherever applicable. Also use cursors wherever applicable.

## 3.1. Write a function to compute age.

The function should accept a date and return age in years.

## 3.2 Write a procedure to find the manager of a staff.

Procedure should return the following – Staff\_Code, Staff\_Name, Dept\_Code and Manager Name.

## 3.3. Write a function to compute the following.

Function should take Staff\_Code and return the cost to company.

DA = 15% Salary, HRA= 20% of Salary, TA= 8% of Salary.

Special Allowance will be decided based on the service in the company.

< 1 Year Nil

>=1 Year< 2 Year 10% of Salary

>=2 Year< 4 Year 20% of Salary

>4 Year 30% of Salary

## 3.4. Write a procedure that accept Staff\_Code

Write a procedure that accept Staff\_Code and update the salary and store the old salary details in Staff\_Master\_Back (Staff\_Master\_Back has the same structure without any constraint) table.

Exp< 2 then no Update

Exp> 2 and < 5 then 20% of salary

Exp> 5 then 25% of salary

## 3.5. Write a procedure to insert details into Book\_Transaction table.

Procedure should accept the book code and staff/student code. Date of issue is current date and the expected return date should be 10 days from the current date. If the expected return date falls on Saturday or Sunday, then it should be the next working day

# Appendices

## Appendix A: Oracle Standards

Key points to keep in mind:

1. Write comments in your stored Procedures, Functions and SQL batches generously, whenever something is not very obvious. This helps other programmers to clearly understand your code. Do not worry about the length of the comments, as it will not impact the performance.
2. Prefix the table names with owner names, as this improves readability, and avoids any unnecessary confusion.

Some more Oracle standards:

To be shared by Faculty in class

## Appendix B: Coding Best Practices

1. Perform all your referential integrity checks and data validations by using constraints (foreign key and check constraints). These constraints are faster than triggers. So use triggers only for auditing, custom tasks, and validations that cannot be performed by using these constraints.
2. Do not call functions repeatedly within your stored procedures, triggers, functions, and batches. For example: You might need the length of a string variable in many places of your procedure. However do not call the LENGTH function whenever it is needed. Instead call the LENGTH function once, and store the result in a variable, for later use.

## Appendix C: Table of Examples

[Example 1: Declaration Block 5](#_Toc454206203)

[Example 2: PL/SQL block 6](#_Toc454206204)

[Example 3: PL/SQL block 7](#_Toc454206205)