National Institute of Business Management

Advanced Database Management Systems

Oracle PL/SQL Practices

Pubudu Withanage

Practice 01

1. Evaluate each of the following declarations. Determine which of them are *not* legal and explain why.

   a) DECLARE v_id NUMBER (4);

   b) DECLARE v_x, v_y, v_z VARCHAR2 (10);

   c) DECLARE v_birthdate DATE NOT NULL;

   d) DECLARE v_in_stock BOOLEAN := 1;

2. Create an anonymous block to output the phrase "**My PL/SQL Block Work**" to the screen.

G_MESSAGE
My PL/SQL Block Work

3. Create a block that declares two variables as *v_char* and *v_num*. Assign the value of these PL/SQL variables to *i*SQL*Plus host variables and print the result of the PL/SQL variables to the screen.

G_CHAR
42 is the answer

G_NUM
42

Practice 02

1. Evaluate the P/SQL block below and determine the data type and variable of each of the following variables according to the rules of scoping.

```
DECLARE
        v_weight     NUMBER (3) := 600;
        v_message    VARCHAR2 (255) :=  'Product 10012';
BEGIN

        DECLARE
                v_weight     NUMBER (3) := 1;
                v_message    VARCHAR2 (255) :=  'Product 11001';
                v_new_locn VARCHAR2 (50) :=  'Europe';

        BEGIN

                v_weight := v_weight + 1;

                v_new_locn := 'Western ' || v_new_locn;

1 →

        END;

        v_weight     := v_weight + 1;
        v_message    :=  v_message  || ' is in stock';
        v_new_locn :=  'Western' || v_new_locn;

2 →

END;
```

a) The value of ***v_weight*** at position 1 is : …………………………………………………….

b) The value of ***v_new_locn*** at position 1 is: …………………………………………………

c) The value of ***v_weight*** at position 2is : …………………………………………………….

d) The value of ***v_message*** at position 2 is :.………………………………………………….

e) The value of ***v_new_locn*** at position 2 is: …………………………………………………

2. Create and execute a PL/SQL block that accepts two numbers through *i*SQL*Plus substitution variables.

   a) Use the DEFINE command to provide the two values.

   b) Pass the two values defined in step a above, to the PL/SQL block through *i*SQL*Plus substitution variables. The first number should be divided by the second number and have the second number added to the result. The result should be stored in a PL/SQL variable and printed on the screen.

3. Build a PL/SQL block that computes the total compensation for one year.

   a) The annual salary and the annual bonus percentage values are defined using the DEFINED command.

   b) Pass the values defined in the above step to the PL/SQL block through *i*SQL*Plus substitution variables. The bonus must be converted from a whole number to a decimal (for example, 15 to .15). If the salary is *null*, set it to zero before computing the total compensation.

      Note : Total Compensation is the sum of the annual salary and the annual bonus. Use the NVL function to handle *null* values.

      Logic :Salary * (1 + Bonus)

      Values : Salary = 50,000 & Bonus = 10%

      Handle Null Values : NVL(v_variable, initialize_value)

Practice 03

SELECT * FROM ***DEPARTMENT***;

DPARTMENT_ID	DEPARTPENT_NAME	MANAGER_ID	LOCATION_ID
230	IT Helpdesk	10255	1700
240	Government Sales	11345	1700
250	Retail Sales	34678	1700
260	Recruiting	90675	1700
270	Payroll	30937	1700

1. Create a PL/SQL block that selects the maximum department number in the DEPARTMENT table and stores in the PL/SQL variable. Print the result to the screen.

G_MAX_DEPT_NO
270

2. Modify the PL/SQL block you created in exercise 1 to insert a new department into the DEPARTMENT table.

   a) Define the department name. Name the new department ***Education***.
   b) Before printing the department number retrieved from exercise 01, add 10 to it and use it as the department number for the new department.
   c) Leave the manager number and location number as null for now.
   d) Display the new department that you created through query.

3. Create a PL/SQL block that update the location ID for the new department.

   a) Name the new location ID as 1700.
   b) Pass the value to the PL/SQL block and update the location Id.
   c) Display the department that you updated by using location Id.

4. Create a PL/SQL block that delete the department that you created in exercise 02.

   a) Provide the department Id.
   b) Print to the screen the number of rows affected.
   c) Confirm the department has been deleted.

Practice 04

1. Create the MESSAGE table. The column name is the Results. Write a PL/SQL block to insert numbers into the MESSAGE table.

   a) Insert the numbers 1 to 10, excluding 6 and 8.
   b) Commit before the end of the block.
   c) Select from the MESSAGE table to verify that your PL/SQL block worked.

2. Create PL/SQL block that computes the commission amount for a given employee based on the employee's salary.

   a) Use the DEFINE command to provide employee ID. Pass the value to the PL/SQL block through an*i*SQL*Plus substitution variable.

      DEFINE p_empno = 100

   b) If the employee's salary less than Rs. 5,000, display the bonus amount for the employee as 10% of the salary.

   c) If the employee's salary between Rs. 5,000 and Rs.10, 000, display the bonus amount for the employee as 15% of the salary.

   d) If the employee's salary exceeds Rs. 10,000, display the bonus amount for the employee as 20% of the salary.

   e) If the employee's salary is NULL, display the bonus amount for the employee as 0.

      Note : Table name is **employee** and column namesare**emp_id** and **salary.**

Practice 05

1. Create a new table for storing the salaries of the employee

   Table Name : F_SALARY

   Column : Salary

2. Crate a PL/SQL block that determines the top employees with respect to salaries.

   a) Accept a number ***n*** from the user where ***n*** represents the number of top***n*** earners from the EMPLOYEE table. For example, to view the top five earners, enter 5.

      Note : Use the DEFINE command to provide the value for ***n***. Pass the value to the PL/SQL block through a *i*SQL*Plus substitution variable.

   b) In a loop use the *i*SQL*Plus substitution parameter created in step 01 and gather the salaries of the top *n* people from the EMPLOYEES table. There should be no duplication in the salaries. If two employees earn the same salary, the salary should be picked up only once.

   c) Store the salaries in the F_SALARY table.

   d) Test a variety of special cases, such as *n* = 0 or where *n* is greater than the number of employees in the Employee table. Empty the F_SALARY table after each test.

Practice 06

1. In a loop, use a cursor (**dept_cursor**) to retrieve the department number (**Dept_No**) and the department name (**Dept_Name**) from the DEPARTMENTS table for those whose Dept_No is less than 100. Pass the department number to another cursor (**emp_cursor**) to retrieve from the EMPLOYEES table the details of employees last name (**L_name**), designation (**Desig_Name**), hire date (**Hire_Date**) and salary (**Salary**) of those employees whose employee id (**Emp_ID**) is less than 120 and who work in that department.

   Results will be

```
Dept.No : 10            Dept. Name : Administration

Dept.No : 20            Dept. Name : Marketing

Dept.No : 30            Dept. Name : IT
Perera     Programmer       11-NOV-14      90900

Silva      DBA              19-NOV-14      120200

…………

Dept.No : 40            Dept. Name : Human Resource
```

2. Create a PL/SQL block that rewards an employee by appending an asterisk (*) in the STARS column for every Rs.1000.00 of the employee's salary. Update the STARS column for the employee with the string of asterisk. Use the FOR UPDATE and WHERE CURRENT OF functionality in cursor processing.

   Note : If the employee has a salary amount of Rs.12500.00, the string of asterisk should contain 13.

EMPLOYEE_ID	SALARY	STARS
104	6000	
176	11000	
200	8600	

Output will be

EMP_ID	SALARY	STARS
104	6000	******
176	11000	***********
200	8600	*********

Practice 07

1. Write a PL/SQL block to select the name of the employee with a given salary value.
   a) Use the DEFINED command to provide the salary.

   b) Pass the value to the PL/SQL block structure through PL/SQL substitution variable. If the salary entered returns more than one row, handle the exception with an appropriate exception handler and insert into the MESSAGES table in the message "More than one employee with a salary of <*salary*>."

   c) If the salary entered does not return any rows, handle the exception with an appropriate exception handler and insert into the MESSAGES table the message "No employee with a salary of <*salary*>."

   d) If the salary entered returns only one row, insert into the MESSAGES table the employee's last name and the salary amount.

   e) Handle any other exception with an appropriate exception handler and insert into the MESSAGES table the message "Some other error occurred."

Practice 08

1. Create a procedure called ADD_JOB procedure and consider the result.

   a) Create a procedure called ADD_JOB insert a new job into the JOBS table. Provide the ID and the title of the job, using two parameters.

   b) Compile the code, and invoke the procedure with **IT_DBA** and '**Database Administrator**' as job title. Query the JOBS table to view the results.

   Table Name : JOBS

JOB_ID	**JOB_TITLE**
IT_DBA	Database Administrator

2. Create a procedure called UPD_JOB to modify a job in the JOBS table.

   a) Create a procedure called UPD_JOB to update the job title. Provide the job ID and a new title using two parameters. Include the necessary exception handling if no update occurs.

   b) Compile the code, and invoke the procedure with **IT_DBA** and '**Data Administrator**' as job title. Query the JOBS table to view the results.

Table Name : JOBS

JOB_ID	**JOB_TITLE**
IT_DBA	Data Administrator

Practice 09

1. Create and invoke the Q_JOB function to return a job title.
   a) Create a function called Q_JOB to return a job title to a host variable.
   b) Compile the code, create a hot variable G_TITLE and invoke the function with job ID SA_REP. Query the host variable to view the result.

G_TITLE
Sales Representative

2. Create a function called ANNUAL_COMP to return the annual salary by accepting two parameters (Employee monthly salary and commission). The function addresses NULL values.
   a) Create and invoke the functionANNUAL_COMP, passing in values for monthly salary and commissions. Either or both values passed can be NULL, but the function should still return an annual salary, which is not NULL. The annual salary is defined by the basic formula :
   (salary * 12) + (commission * salary * 12)
   b) Use the function in a SELECT statement against the EMPLOYEE table for department number 80.
   Columns:Emp_id, Last_name, salary, commission,dept_id

PL/SQL INSTEAD OF Trigger Example

The following example creates two new tables, *NEW_EMPS* and *NEW_DEPTS*, based on the *EMPLOYEES* and *DEPARTMENTS* table respectively. It also creates an *EMP_DETAILS* view from the *EMPLOYEES* and *DEPARTMENTS* tables. The example also creates an INSTEAD OF trigger, *NEW_EMP_DEPT*. When a row is inserted into the *EMP_DETAILS* view, instead of inserting the row directly into the view, rows are added into the *NEW_EMPS* and *NEW_DEPTS* tables, based on the data in the INSERT statement. Similarly, when a row is modified or deleted through the *EMP_DETAILS* view, corresponding rows in the *NEW_EMPS* and *NEW_DEPTS* tables are affected.

```
CREATE TABLE new_emps AS
    SELECT employee_id, last_name, salary, department_id, email, job_id, hire_date
    FROM employees;

CREATE TABLE new_depts AS
    SELECT d.department_id, d.department_name, d.location_id, SUM(e.salay) tot_dept_sal
    FROM employee e, departments d
    WHERE e.department_id = d.department_id
    GROUP BY d.department_id, d.department_name, d.location_id;

CREATE VIEW emp_details AS
    SELECT e.employee_id, e._last_name, e.salary, e.department_id, e.email, e.job_id,
        d.department_name, d.location_id
        FROM employee e, departments d
        WHERE e.department_id = d.department_id;


CREATE OR REPLACE TRIGGER new_emp_dept
INSTEAD OF INSERT OR UPDATE OR DELETE ON emp_details
FOR EACH ROW
BEGIN
    IF INSERTING THEN
            INSERT INTO new_emps
            VALUES (:NEW.employee_id, :NEW.last_name, :NEW.salary, :NEW.department_id,
                        :NEW.email, :NEW.job_id, SYSDATE);

            UPDATE new_depts
            SET tot_dept_sal = tot_dept_sal + :NEW.salary
            WHERE department_id = :NEW.department_id;

        ELSEIF DELETING THEN
            DELETE FROM new_emps
            WHERE employee_id = :OLD.employee_id;

            UPDATE new_depts
            SET tot_dept_sal = tot_dept_sal  -  :OLD.salary
          WHERE department_id = :OLD.department_id;

        ELSEIF UPADTING (‘salay’) THEN
            UPDATE new_emps
            SET salary = :NEW.salary
            WHERE employee_id = :OLD.employee_id;

          UPDATE new_depts
            SET tot_dept_sal = tot_dept_sal + ( :NEW.salary -  :OLD.salary)
          WHERE department_id = :OLD.department_id;

          ELSEIF UPADTING (‘department_id’) THEN
          UPDATE new_emps
            SET department_id = :NEW. department_id
            WHERE employee_id = :OLD.employee_id;

          UPDATE new_depts
            SET tot_dept_sal = tot_dept_sal -  :OLD.salary
          WHERE department_id = :OLD.department_id;

          UPDATE new_depts
            SET tot_dept_sal = tot_dept_sal -  :NEW.salary
          WHERE department_id = :NEW.department_id;


        END IF;
      END;
      /
```