Oracle PL/SQL – L1: Trend.Nxt Hands-on Assignments

Estimated Efforts: 3 PDs

For detailed ToC and other Details: https://wipro365.sharepoint.com/:w:/r/sites/ku-practice-4101/KMSitesContent/GDO/UCF/_layouts/15/WopiFrame.aspx?sourcedoc=%7B8793EEF7-F1AC-4B79-8A5C-13B5CEC832CE%7D&file=PL-SQL_L1_LG.docx&action=default

Author: magesh.babu@wipro.com

Date: 15th May 2018

ToC :

Topic No	Topic Name	Sub Topics	Min No of Assignments to be Done
1	Declaring Variables	PL/SQL Block Syntax, Identifiers, SQL Functions in PL/SQL, Data Type Conversion, Operators in PL/SQL	1
2	Control Structures	Control statements	1
3	Composite Data Types	Composite Data Types	1
4	Cursors (Handling Exceptions)	Explicit Cursor Functions, Explicit Cursor Attributes, Cursor FOR Loops	1
5	Creating Procedures	CREATE PROCEDURE statement Creating Procedures with Parameters, IN OUT Parameters, DEFAULT Option for Parameters, Anonymous PL/SQL Block, Handled Exceptions	1
6	Creating Functions	CREATE Function statement Creating a Function, Invoking Functions in SQL Expressions, Procedure or Function	1
7	Creating Triggers	CREATE TRIGGER statement Types of Triggers, Creating DML Triggers, DROP TRIGGER	1

8	Transaction Management	Database Transactions Database Transactions, COMMIT, ROLLBACK and Save Points, Controlling Transactions, Implicit Transaction Processing, Transaction Properties, Autonomous Transactions Data Concurrency and Consistency Isolation Levels, Read Consistency, Implementation of Read Consistency	1
9	Managing Subprograms	Managing using DBMS_* Managing Stored PL/SQL Objects, USER_OBJECTS, USER_SOURCE, USER_ERRORS, Debugging PL/SQL Program Units, etc	1
10	Creating Packages	Oracle packages management Components of a Package, Developing a Package, Declaring Public Constructs, Public and Private Constructs, Referencing a Public Variable from a Stand-Alone Procedure, Advantages of Packages, Restrictions on Package Functions, Persistent State of Package Variables, PL/SQL Tables and Records in Packages,etc..	1
Total Min No of Assignments to be Done			**10**

Topic 1: Declaring Variables

Assignment 1:

a. Identify which of the following variable declarations are valid with a reason: -

```
i.    DECLARE   iNumber NUMBER(4);
ii.   DECLARE   sFirst,sMiddle,sLast  VARCHAR2(10);
iii.  DECLARE   dDateOfBirth DATE NOT NULL;
iv.   DECLARE   bFlag   BOOLEAN :=1;
```

b. Predict the output of the below mentioned code snippet :-

```
DECLARE
      iNumber NUMBER(7);
BEGIN
      DBMS_OUTPUT.PUT_LINE(iNumber);
END;
```

c. Predict the output of the below mentioned code snippet :-

```
DECLARE
     iWeight NUMBER (3):=300;
     sProdName VARCHAR2(84):= ‘Talent Transformation’;
BEGIN
     DECLARE
          iWeight NUMBER(3) :=1;
          sProdName VARCHAR2 (25):=‘Bangalore’;
          sLocation VARCHAR2(57):=‘Electronic ’;
     BEGIN
           iWeight := iWeight +1;
           sLocation:=sLcation || ‘City‘;
           DBMS_OUTPUT.PUT_LINE(sLocation)
     END;
     iWeight:=iWeight + 1;
     sProdName:=sProdName|| ‘ CTE ’;
     DBMS_OUTPUT.PUT_LINE(sProdName);
END;
```

Assignment 2:

a. Write a PLSQL block which accepts two non zero integer numbers from the user and store it to the PL/SQL variables named iFirst and iSecond , perform the following operation :-
   (iFirst * iSecond / iSecond).
   The result of the above operation should be stored into a PL/SQL variable and print the same.

b. Declare two SQL* plus variables named MAX_SALARY and MIN_SALARY of type NUMBER, Create a PL/SQL block which accepts departmentNumber from the user and identify the Maximum and Minimum salary from the “Employee” table for the specified departmentNumber and assign it to the MAX_SALARY and MIN_SALARY variables. Use appropriate SQL * plus command to print the value of the created variables.

c. Create 2 SQL * plus variables and write a code to assign a value for the created SQL * plus variable inside the PL/SQL block and print the same.

Topic 2: Control Statements

Assignment 1:

Write a PL/SQL block for the following scenarios using control structures :-

a. Accept a number from the user during the runtime and check whether the given number is even or odd.
b. Generate the prime numbers between 1 to 100
c. Accept an integer value between 1 to 12 from the user and display the name of the corresponding month [ie. 1 – January, 2 – February, …]. If the specified input is either < 1 or > 12 then display " Invalid Month ".
d. Accept three different integer values from the user ,identify and display the largest of 3 numbers.
e. Generate perfect numbers between 1 to 100

   Note.: If sum of the divisors is equal to given number then it is said to be perfect

   i.e. Given Number : 6

   Divisors : 1, 2, 3 Sum of Divisors : 6

Assignment 2:

Write a PL/SQL block to implement the below scenarios :-

a. Accept employee number from the user and retrieve the salary from the "Employee" table for the specified employee number and display the employee details (EmployeeNumber, EmployeeeName, Salary and Class) determine the Class as per the below mentioned criteria:-
   - If the salary < 2500, then class = 'Deluxe'
   - If the salary >= 2500 and salary < 5000, then class = 'Super Deluxe'.
   - If the salary >= 5000, then class='Ultra Deluxe'.

b. Accept the department number from the user and check for its existence in "Department" table , If it exists then retrieve the department name and display the remarks as per the department name mentioned below :-

DepartmentName	Remarks
TT	Talent Transformation
BAS	Business Administration Services
PES	Product Engineering Service
FS	Financial Service
CTE	Center For Technology Excellence
Else	New Department

c. Display the numbers from 1 to 50 in words.

d. Accept the CourseId from the user, check for the existence in Course table, if exists remove the records if the user wishes to delete else display appropriate error message.
e. Insert the records into the "Department" Table recursively :-

i. Value of DeptNumber column must be an odd integer between 1 to 20.
ii. Accept DeptName from the user.

Note.: Create appropriate tables with respect to the problem statements.

Topic 3: Composite Datatypes

Assignment 1:

Create and implement the PL/SQL block for the below mentioned problem statements :

a. Accept employee number from the user, retrieve the employee details [EmployeeNumber, EmployeeName, Salary, DepartmentNumber] from the "Employee" table and store it to a PLSQL record type variable and print the same.

b. Create a composite type named DeptRecord (iDeptno, sDeptName, sDeptLoc) as attributes with appropriate datatypes and accept the value for these attributes from the user as mentioned below :-
   i. iDeptNo : Value should be Max(deptno) From the "Department" Table and increment 1 to it.
   ii. sDeptName : must be a not null string with <= 21 characters
   iii. sDeptLoc : Can be either "BDC" or "CDC" or "HDC"

   If the above conditions are met insert the above record to the "Department" table.

c. To find the 1st Maximum element on the given array

Assignment 2:

Create and implement the PL/SQL block for the below mentioned problem statements :

a. To sort the contents of an integer array both in ascending and descending order.

b. To find the 'n'th minimum element from the given array [Note.: 'n' should be taken as an input from the user]

c. Accept a number from the user and find whether the given element is available on the existing collection, if yes display the index position along with the element else display " Element not Found ".

d. Accept a number between 1 to 10 from the user and delete the corresponding element from the given array and after deleting display the contents of the array.

Topic 4: Cursors and Handling Exceptions

Assignment 1:

a. Write a PL/SQL block that retrieves information from the “Employee” and “Department” table and displays it in the below mentioned format :-

```
Department Number :10                           Department Name : CTE
______________________________________________________________________

EMPLOYEE NUMBER          EMPLOYEE NAME     SALARY            JOB
      XXXXXX               XXX             XXX               XX
      XXXXXX               XXX             XXX               XX
      .........

Department Number : 11                          Department Name : BAS
______________________________________________________________________

EMPLOYEE NUMBER           EMPLOYEE NAME    SALARY            JOB
      XXXXXX                XXX             XXX              XX
      XXXXXX                XXX             XXX              XX
      .............
```

for all the department numbers available in “Department” table.

Note.: In a loop, use a cursor to retrieve the deptNumber and the deptName from the “Department” table, pass the deptNumber to a ref cursor to retrieve the employees who belong to that deptNumber from the table named “Employee”.

b. Write a PL/SQL block to retrieve and display the empName and empSalary of those employees whose job =’CLERK’ from the table named “Employee”.

c. Create a user defined exception named “DUPLICATE_EMPLOYEE_JOBS’ . The above exception has to be raised provided if there are any duplicates available for empJob column in the table named “Employee”, handle the same with an error message “More Than one Employee for the same Job”.

Assignment 2:

a. Write a PL/SQL block that accepts the employee numbers of two different employees as user inputs namely iFirstEmpNo and iSecondEmpNo and check for its existence in the table named “Employee” and perform the following :
   i. If “iFirstEmpNo” exist then update his/her salary by 10%, else if “iSecondEmpNo” exist then update his/her salary by 20%
   ii. If both “iFirstEmpNo and iSecondEmpNo” exist/not exist raise the exception.

b. Write a PL/SQL block to display the empName, empSalary, empSalGrade for the first 10 employees in the table named “Employee” using the following :-

i. Simple for loop with “EXIT WHEN” condition
ii. Using While loop
iii. Using cursor for loop

c. Write a PL/SQL block which accepts the job from the user and display (empName, empSalary and empDeptNo) for the specified job from the table named “Employee”
   Note.: User input has to be passed as an argument to the cursor.

Topic 5: Creating Procedures

Assignment 1:

a. Create a procedure named DISP_EMP_DETAILS with 3 parameters out of which one [ie. iEmpNo] is an “IN” mode and other two are [ie. sGrade and sSalary] “OUT” mode parameters. The procedure should retrieve the empGrade and empSalary for the specified employee number [ie. iEmpNo] in the table named “Employee” and assign the retrieved values to the “OUT” mode parameters.
   Note.:
   i. It should display an appropriate error message if the specified employee number does not exist in “Employee” table.
   ii. Call the created procedure using Bind variable and print the details.

b. Create a procedure named DISPLAY_RECORDS which accepts the JoinDate as a parameter and display all the employees (empNo,empSalary,empDeptNo) from the “Employee” table who joined after the specified parameter in the following format :-

```
EmployeeNumber    Salary        DepartmentNumber
________________________________________________
  XXXXXXXX        99,999              99
  XXXXXXXX         9,999              12
```

   Note.:
   i. It should display an appropriate error message if there are no employees who have joined after the specified date.
   ii. Invoke the above mentioned procedure inside the anonymous PL/SQL block.

Assignment 2:

a. Create a procedure named FIND_COURSE_CREDITS which accepts employeeNumber as a parameter. Find and display the various e-learning modules completed by the specified employee along with the total duration spent.
   Note.:
   i. Handle appropriate exceptions where ever required
   ii. Display empNumber, CourseName, Duration in Minutes
   iii. Invoke the above mentioned procedure inside the anonymous PL/SQL block.

b. Create a procedure named ADD_EMPLOYEE to hire an employee with the following specifications :-
   i. Accepts job,managerId,hireDate,empSalary,commission and deptNo as parameters.
      a. **job** should be either 'CLERK' or 'ANALYST' or 'SALESMAN'. The input value can be entered in any case (upper or lower or initcap).
      b. **managerId** must be an existing employee in "Employee" table
      c. **deptNo** must exist in "Department" table.
      d. **empSalary** must be > 1200
   ii. In addition to this employeeNumber must be auto generated by using a sequence.

Insert the record if the above validations are met and display a message '1 row inserted'. Otherwise handle appropriate exceptions.

Topic 6: Creating Functions

Assignment 1:

a. Create a function named VALIDATE_EMP which accepts employeeNumber as a parameter, Returns TRUE if the specified employee exist in the table named "Employee" else FALSE.
b. Create a function named CALCULATE_ROYALTY that takes employeeNumber as a parameter and identify whether the specified employeeNumber exist and he is a developer from the table named "Employee",
   i. If yes return the royalty that the developer as got, for the software he/she has developed.
   ii. It should return 1 if the specified employeeNumber is not a developer,
   iii. It should return 2 if the specified employeeNumber is a developer and has a value null to royalty column.

Assignment 2:

a. Create a function named USER_ANNUAL_COMP accepts three parameters [employeeNumber, salary, commission]. It should calculate and returns the annual compensation of the employee as mentioned blow :-
   annual_compensation = (salary + commission)*12.

   Note.:
   i. If the salary or commission value is NULL then zero should be substituted for it.

b. Create a function named SHOW_STRENGTH that accepts departmentNumber as a parameter and invokes another function named "USER_VALID_DEPTNO" with the specified parameter to check whether the specified depratmentNumber exist in "Department" table. If exist returns "TRUE" otherwise "FALSE".

Note.:
i. If USER_VALID_DEPTNO returns "TRUE" then count the total number of employees for the specified departmentNumber from the table named "Employee" and return the same.
ii. Otherwise handle appropriate exceptions and return a value "0" .

Topic 7: Triggers

Assignment 1:

Implement the following business rule with the help of a Procedure and a Trigger :-

i. Changes to the data in the Department table, will be allowed only in the month of March.
ii. Create a procedure named SECURE_DML that prevents the DML statement from executing in any other month other than March. In case, if a user tries to modify the table in any other month apart from March, the procedure should display a message "You can modify or add a department only at the end of a financial year"
iii. Create a statement level trigger named TR_CHECK_DEPT on the Department table that calls the above procedure.
iv. Test it by inserting a new record in the Department table.

Assignment 2:

Implement the following business rule

i. Create a table named salaryLog with the appropriate columns and insert the empno, new grade, old salary and new salary values in salaryLog table if the grade of an employee changes.
ii. Create a trigger named TR_CHECK_GRADE that will be fired when a user modifies the EMP table. It will check whether the grade has changed by making use of the SALGRADE table. (Grade is dependent on Salary.) If grade is changed, the trigger will log the corresponding employee number, old salary, new salary and new grade into salaryLog table.
iii. Test the working of the trigger by firing an appropriate DML query.

Topic 8: Transaction Management

Assignment 1:

Start required number of session and write code to illustrate concept of

a. reader does not lock another reader
b. writer does not lock a reader
c. writer locks another writer

Topic 9: Managing Subprograms

Assignment 1:

Write Query to:

a. List Object names of all Stored Procedures and Functions in your schema

b. List Source Code of a given Stored Procedure

c. List Compilation error - Text , Line, Position of a given stored procedure that got created with compilation error.

Topic 10: Creating Packages

Assignment 1:

- Create a package named MANAGE_EMP_PACK that has two public procedures, two package level variables and a private function. The public procedure HIRE_EMP adds an employee record in EMP table and the public procedure FIRE_EMP deletes an employee record from the EMP table. The two variables v_insert_cnt and v_delete_cnt are used in the package, for keeping record of the numbers of times insert / delete has been executed.
  Create a private function VALIDATE_EMP in the package to validate employee number. This function can be called from HIRE_EMP and FIRE_EMP.
- The function VALIDATE_EMP accepts an employee number with a parameter p_eno and returns TRUE if the specified employee number exists in the EMP table else it returns FALSE.
- The procedure HIRE_EMP takes all the column values of the EMP table as parameters. It gives a call to VALIDATE_EMP by passing employee number as a parameter and if the function returns TRUE then it displays message 'Employee number already in use'. If the function returns FALSE then it inserts a new record in to the table named 'EMP', displays a message 'One employee added' and increments the value of v_insert_cnt by 1.
- The procedure FIRE_EMP accepts an employee number as a parameter and gives a call to VALIDATE_EMP by passing employee number as a parameter. If the function returns TRUE then it deletes the corresponding record from the EMP table, displays message 'One employee deleted' and increments the value of v_delete_cnt by 1. If the function returns FALSE then it displays message 'Invalid Employee Number'.
- Invoke the public procedures inside the package named MANAGE_EMP_PACK and display the values of package variable, inside an anonymous PLSQL block

Assignment 2 :

Create the below mentioned tables and insert appropriate records to automate the conference room rental process.

HALLS [Used to have the information about various halls available to conduct a conference]

Column Name	**Data Type**	**Description**
HName	Varchar2(4)	Name of the hall; Primary key
HType	Varchar2(10)	Capacity of the hall
Rent	Number(5)	Rent for the hall per day in Rs; Should be a positive value
NoOfBookings	Number(2)	Total number of bookings for the hall

COMPANY [Used to store their client information who used to utilize the conference rooms]

Column Name	**Data Type**	**Description**
CCode	Number(3)	Company code; Primary key
CompanyName	Varchar2(10)	Name of the company;
ContactNo	Number(10)	Contact number of the company

BOOKING [Maintains the transactions made by various clients along with the schedule]

Column Name	**Data Type**	**Description**
TransId	Number(2)	Transaction ID; Primary key
CCode	Number(3)	Company code; Should be an existing company
Hname	Varchar2(4)	Name of the hall; Should be an existing hall
StartDate	Date	Start date of booking
EndDate	Date	End date of booking

1. Create a sequence to generate transaction id.
2. Create a package named Conference_Automation with the below mentioned details :-
   a. Write a procedure to insert one record in 'BOOKING' table :-
      i. Accept CCode,HName,StartDate and EndDate as inputs
      ii. Validate CCode andHName
      iii. Insert a record into the "BOOKING' table by using above mentioned values and use sequence for TransId.
   b. Write a procedure to retrieve the total number of bookings :-
      i. Accept HType (Capacity of the hall) as input
   c. Write a procedure to generate the report :-
      i. Company Code must be passed as an argument
      ii. Validate for the existence of the company code , if present then display all the booking details [Transaction Id, Hall Name, Start Date and End Date]
   d. Write a procedure to get the rent from "HALLS" table and calculate the cost
      [Note. Cost = No. Of Days * Rent]

e. Create 2 private functions to validate
    i. Company Code : Must be a three digit positive integer between 100 and 999
    ii. Hall Capacity : Must be either Small or Medium or Large