# PL-SQL – L2: Sample Hands-on Assignments

**Tables in HR Schema used.**

**Topic 1 and 2: Explicit Cursors and Managing Dependencies (work on any 2)**

## Assignment 1:

Demonstrate the Usage of CURSOR FOR UPDATE (declare cursor cursor\_name is …. **for update** and DML ending with **WHERE CURRENT** of cursor\_name ) with an appropriate example.

## Assignment 2:

Create a ladder of dependent and referenced objects Create Table

Create View on the Table

Create a Function that reference the View Create a Procedure that calls the Function

Now, Write the necessary code to display both direct and indirect dependencies on the Table in the Tree Format

## Assignment 3:

Create a View – CREATE VIEW EMP\_VIEW AS SELECT FIST\_NAME, SALARY FROM EMPLOYEES;

Writer query to check the status of the view created Add a new column to the EMPLOYEES table.

Writer query to check the status of the view created Drop SALARY column from EMPLOYEES table

Writer query to check the status of the view created.

SQL> create or replace package pack\_parent 2 as

3 function f1 return number; 4 end;

5 /

Package created.

SQL> create or replace package pack\_dependent 2 as

3 global\_var1 number := pack\_parent.f1; 4 end;

5 /

Package created.

Writer query to check the status of the Package Spec and Body created SQL> create or replace package pack\_parent

1. as
2. function f1 return number;
3. procedure p1;
4. end;

6 /

Writer query to check the status of the Package Spec and Body SQL> create or replace package pack\_parent

1. as
2. procedure p2;
3. function f1 return number;
4. procedure p1;
5. end;

7 /

Writer query to check the status of the Package Spec and Body

Based on above Hands-on – explain that is Fine Grained Dependency Tracking and best practices that can be followed to reduce dependent invalidation.

# Topic No 3: Adv Triggers (work on any 4)

## Assignment 1:

Create following 2 (T1, T2) triggers of the same type on the same table (before insert on employees for each row)

create or replace trigger T1 before insert on employees for each row

begin

DBMS\_OUTPUT.PUT\_LINE(‘Trigger T1 executed’);

end;

/

create or replace trigger T2 before insert on employees for each row

begin

DBMS\_OUTPUT.PUT\_LINE(‘Trigger T2 executed’);

end;

/

Insert many rows into employees table – but insert one by one – note down the order in which the 2 triggers are getting executed and share your understanding.

Make necessary changes to the trigger code to ensure that trigger T2 always gets executed only after trigger T1 execution.

Insert many rows into employees table – but insert one by one – note down the order in which the 2 triggers are getting executed and share your understanding.

## Assignment 2:

Demonstrate the usage of COMPUND TRIGGER with an appropriate example.

## Assignment 3:

Create DDL Trigger that Prevents DROPPING of DB Objects on Saturday and Sunday

## Assignment 4:

Login as SYS and Write a database trigger called as AFTER\_LOGON in SYS schema to track the login of every DB User into a table called LOGON\_TRACK with 2 columns – USERNAME and LOGON\_DATE

Test the trigger created by logging as HR or SCOTT multiple times.

## Assignment 5:

Write a trigger that prevents decreasing/reducing the Salary of employees.

## Assignment 6:

Write a Trigger to track/audit all UPDATE operations done on SALARY column of EMPLOYEES table by DB users – capture the data into a table called AUD\_EMP with columns – USERNAME, UPDATE\_DATE, EMP\_ID, OLD\_SALRY and NEW\_SALARY.

## Assignment 7:

Create a PRODUCT table with columns – PROD\_ID, DESC, QUANTUTY\_ON\_HAND, REORDER\_LEVEL, and REORDER\_QUANTITY.

Populate PRODUCT table with good number of rows.

Create a REORDER table with columns – PROD\_ID , REORDER\_QUANTITY

Write trigger that INSERTs a row into REORDER table with PROD\_ID and REORDER\_QUANTUTY values -- whenever QUANTITY\_ON\_HAND of a product goes below the REORDER\_LEVEL.

Test by updating QUANTUTY\_ON\_HAND column of PRODUCT table for specific products

UPDATE TABLE PRODUCT SET QUANTUTY\_ON\_HAND = (value < REORDER\_LEVEL ) WHERE PROD\_ID=10

# Topic No 4: Adv. PL-SQL Programming (work on any 4)

## Assignment 1:

Demonstrate the usage and benefits of BULK BINDING and BULK COLLECT with appropriate examples

## Assignment 2:

Create a stored procedure (called PROC\_ARITH) that performs lot of arithmetic operations inside a long running loop.

Write query to note down the PLSQL\_CODE\_TYPE of the procedure created Execute the procedure (with set timing on) – note down the elapsed time Write code to NATIVEly compile the procedure using ALTER PROCEDURE … Write query to note down the PLSQL\_CODE\_TYPE of the procedure Execute the procedure (with set timing on) – note down the elapsed time Do you see any difference in the elapsed time?

Share code and your understanding on INTERPRETED v/s NATIVE compilation

## Assignment 3:

Write another stored procedure (called CALL\_PROC\_ARITH) that calls the stored procedure (PROC\_ARITH) created in above Assignment inside a long running loop.

Note down what is the default value of PLSQL\_OPTIMIZE\_LEVEL Execute the CALL\_PROC\_ARITH – note down the elapsed time Change the PLSQL\_OPTIMIZE\_LEVEL to 3 at session level Recompile both procedures with ALTER PROCEDURE … COMPILE Execute the CALL\_PROC\_ARITH – note down the elapsed time Do you see any difference in the elapsed time?

Share code and your understanding of PLSQL\_OPTIMIZE\_LEVEL parameter

## Assignment 4:

Demonstrate the usage on PRAGMA INLINE and NOCOPY

## Assignment 5:

Write a PL/SQL Function called as DEL\_ROWS that takes TAB\_NAME and WHERE\_CONDITION as parameters and deletes the rows from the given table (TAB\_NAME) that satisfies the given condition (WHERE\_CONDITION) and returns number of rows deleted.

Test the function by passing table name and where condition as inputs.

Ex: SELECT DEL\_ROWS (‘EMPLOYEES’, ‘DEPARTMENT\_ID=20’) FROM DUAL;

## Assignment 6:

Write a PL/SQL procedure called DROP\_OBJ that takes OBJ\_NAME and OBJ\_TYPE as parameters and drops the given object if it exists otherwise display message – ‘OBJECT DOES NOT EXISTS’

Test the Procedure

# Topic No 5: Adv. Packages (work on any 1)

## Assignment 1:

Create a procedure called TEST\_PROC that just insert a row into a table.

Create a scheduler job (using DBMS\_SCHEDULER) for execution of TEST\_PROC procedure every 5 mins from current time

Check the details of scheduler job using USER/DBA\_SCHEDULER\_JOBS DD view Check – is it getting executed every 5 mins

Disable the Scheduler Job

Check the details of scheduler job using USER/DBA\_SCHEDULER\_JOBS DD view Drop the Scheduler Job

Check the details of scheduler job using USER/DBA\_SCHEDULER\_JOBS DD view

## Assignment 2:

Demonstrate encrypting / obfuscating PL/SQL source code by using WRAP.EXE utility and DBMS\_DDL package