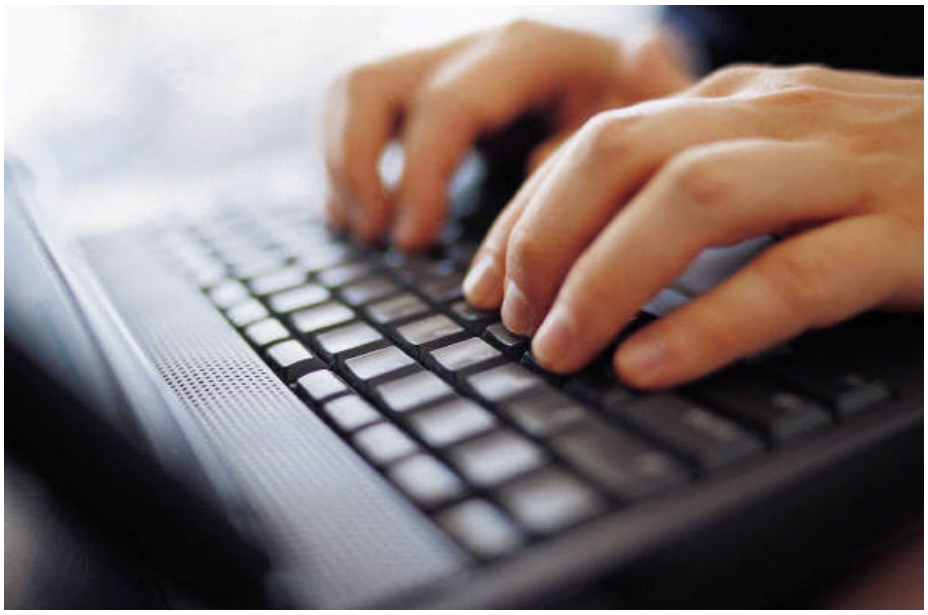
Lab Guide for Oracle



Education and Research

Infosys®
POWERED BY INTELLECT
DRIVEN BY VALUES

Author(s)	Parimala,Vijaykalyani,Mahendra Nath J,Rituza
Authorized by	Dr. M.P. Ravindra
Creation/Revision Date	June 2009
Version	2.0

1 Assignments for Day 1 of Oracle

All the assignments in this section must be completed on Day 1 of your Oracle course.

1.1 *Assignment 1: Solve the DDL and DML exercise*

Objective: To create tables and to insert the required data.

Problem Description:

Create the following tables and insert the sample data as given below(scripts supplied):

Column name	Data type	Description
EMPNO	Number	Employee number, Primary Key
ENAME	Varchar2	Employee name
JOB	Char	Designation
MGR	Number	Manager's Emp. number
HIREDATE	Date	Date of joining
SAL	Number	Basic Salary
COMM	Number	Commission
DEPTNO	Number	Department Number, Foreign Key

DEPT

Column name	Data type	Description
DEPTNO	Number	Department number, Primary Key
DNAME	Varchar2	Department name
LOC	Varchar2	Location of department

Data for EMP

7369	Smith	Clerk	7902	17/12/80	800		20
7499	Allen	Salesman	7698	20/2/81	1600	300	30
7521	Ward	Salesman	7698	22/2/81	1250	500	30
7566	Jones	Manager	7839	2/4/81	2975		20
7654	Martin	Salesman	7698	28/9/81	1250	1400	30
7698	Blake	Manager	7839	1/5/81	2850		30
7782	Clark	Manager	7839	9/6/81	2450		10
7788	Scott	Analyst	7566	9/12/82	3000		20
7839	King	President		17/11/81	5000		10
7844	Turner	Salesman	7698	8/9/81	1500	0	30
7876	Adams	Clerk	7788	12/1/83	1100		20

7900	James	Clerk	7698	3/12/81	950		30
7902	Ford	Analyst	7566	4/12/81	3000		20
7934	Miller	Clerk	7782	23/1/82	1300		10

Data for DEPT table

10	Accounting	New York
20	Research	Dallas
30	Sales	Chicago
40	Operations	Boston

Solve the following queries:-

1. Create the tables with all appropriate constraints.(scripts suppled) **(5 Min)**

2. Commit all the changes that have been made so far. **(1 Min)**

3. Find all employees whose names begin with 'A' and end with 's'. **(5 Min)**

4. Find all Jobs whose descriptions have the characters 'man'. **(5 Min)**

5. Find the total salary paid by each department to employees. **(5 Min)**

6. Display the department names and the names of the employee who belong to that section. **(5 Min)**

7. Display the information for all departments and all employees. **(5 Min)**

8. Display the department name and the name of the employee who heads that department. **(7 Min)**

9. Display employee name, salary and commission. If the commission is null, display 'NO COMMISSION'. Use the NVL function for this. **(7 Min)**

10. Display the employee name and the job code. If the job is 'Salesman' display 'S', if the job is 'Clerk' display 'C', for 'Analyst' display 'A', for 'President' display 'P' , for 'Manager' display 'M'. Use the DECODE function for this. **(10 Min)**

11. List all departments who have no employees. **(7 Min)**

12. Display the average salary of employees. **(5 Min)**

13. Display the employee name and department name to which they belong. The Employee name must have the first letter in capital case and the rest in lower case, department name must be in upper case. **(7 Min)**

14. Display the employees who have joined more than a month ago.*(Hint: Use months_between() function)* **(5 Min)**

15. Display employee names and the date joined for employees. Date should be in the format 'DD-MMM-YYYY'. **(10 Min)**

16. Display employee names and the number of months between today's date and the joining date. **(10 Min)**

17. Display the gross salary rounded to the nearest integer. **(5 Min)**

18. Find the highest paid employee in every department. **(5 Min)**

19. Write a PL/SQL block of code which will let you to insert a record into the Emp table. Use substitution variables to represent the values of input. Use anchored declarations for these variables. **(15 Min)**

20. Write a PL/SQL anonymous block which will insert 50 records into test_table. Insert the current date value into the table. (Use SYSDATE to get the system date). [To see the value of SYSDATE type the following at the SQL Prompt: SELECT SYSDATE FROM DUAL;]. (create a table called test_table, with 2 columns RecordNumber (type: Number(3)) and CurrentDate(type: Date)). **(15 Min)**

21. Solve assignment 20 using the following:
    a) Declare a constant MAX_RECORDS which has a value of 50. Use this value as the maximum number of records to be inserted **(8 Min)**
    b) Do this using a FOR Loop ,
    c) DO this using a While Loop **(8 Min)**

22. At the SQL prompt, issue a ROLLBACK. Try displaying the records in test_table. Can you see all the records? If not, how can you make sure that the changes made by the block are permanent? **(3 Min)**

23. Given an employee number, display the following information to the user:
    Emp name
    Employee Dept no
    Employee Job
    Employee Band
    Obtain the band as follows:

President	1
Manager	2
Analyst	3
Salesman	4
Clerk	5

    Use the %ROWTYPE declaration for obtaining the employee information from the tables. **(20 Min)**

24. Write a PL/SQL anonymous block, which will display the deptno , dname,location and number of employees working in it. **(20 Min)**
25. Write a PL/SQL anonymous block which will increase the salary by 10%, of all employees working for Research department. Use a FOR ..UPDATE clause with the cursor declaration for this. **(15 Min)**

Oracle Day 2 Assignments

2 Assignments for Day 2 of Oracle

All the assignments in this section must be completed on Day 2 of your Oracle course.

2.1 *Assignment 1: Partitioning the tables*

Objective: To create partitioned tables

Problem Description:

1) Create a Range partitioning on EMP based on Hiredate (3 partitions).
2) Create a Hash partitioning on EMP based on Job (5 partitions).
3) Create a List partitioning on EMP based on deptno (4 partitions).
4) Create a Range List composite partitioning on EMP table based on deptno and Hiredate.
5) Create a Range Hash composite partitioning on EMP table based on Job and Hiredate.

2.2 *Assignment 2: Materialized Views*

Objective: To create Materialized Views

Problem Description:

1) Create a read-only materialized view on primary key of emp table.
   The view must get refreshed every 3 minutes.
2) Create a read-only materialized view having the emp details along with the dept details. The view must be refreshed every 1 minute.

2.3 *Assignment 3: Indexes and Clusters*

Objective: To create Indexes and clusters

Problem Description:

1) Create a non unique index on job column of emp table
2) Create a unique index on dname column of dept table
3) Create a non unique index on location column of dept table
4) Drop the index created on location column
5) Create a function based index on the rounded gross salary(sal + comm)
6) Disable the function based index.
7) Create a cluster emp_dept.
8) Drop EMP and DEPT tables.
9) Create emp and dept tables and add them to emp_dept cluster.
10) Create an index on the cluster

2.4 *Assignment 4: Synonyms and Sequences*

Objective: To create Synonym and Sequence

Problem Description:

1) Create a sequence named seq_empid starting with your E#.
2) Check the current value of the sequence(it results in error, why?)
3) Check the next value of the sequence(it does not result in error, why?)
4) Check the current value of the sequence(it does not result in error, why?)
5) create a private synonym named employee for the emp table.
6) Add a row in the emp table.(E# must be generated through the sequence)
7) try to access the contents of the table by the following queries
    a. Select * from Emp;
    b. Select * from employee;
8) Grant select permission on this synonym to your neighbor
9) Ask your neighbor to access this synonym
   (Does the output shows your Emp table contents?)
10) Drop the synonym
11) Create it as a public synonym and try steps 4 and 5
12) Set a maximum value for the sequence as 29999
13) Set the cycle on for the sequence
14) What value will be generated after 25999

2.5 *Assignment 5: Multi-table Insert and Merge command*

Objective: To insert into Multiple tables and Merge command

Problem Description:

/

1) Create a table emp_copy1 which is a replica of Emp table.
2) Create a table emp_copy2 which is a replica of Emp table.
3) From the emp_copy2 table, delete all the records for deptno 10 and commit.

4) Through the MERGE command, insert the records of deptno 10 from emp_copy1 into emp_copy2. For the existing records of emp_copy1 in emp_copy2 increment the salary by 10%.

2.6 *Assignment 6: Data Dictionary Objects*

Objective: To View information from data dictionary objects

Problem Description:

1) Retrieve the names and types of constraints on emp table.
2) Add a unique constraint to dname column as given below
   Alter table dept
   Add unique(dname);
3) Retrieve the names and types of constraints on dept table.
4) What is the name of the constraint applied on dname column?
5) Get the INDEX NAME,TYPE and STATUS of all the constraints on the EMP table.
6) Get the details of the indexed column in Emp table.
7) Get all the information about the sequence seq_EMpid.
8) Get a list of all the privileges in your schema.

3. Assignments for Day 3 of Oracle

All the assignments in this section must be completed on Day 3 of your Oracle course.

3.1 Assignment 1: Working with Collections

Objective: To understand how to use Collections in PL/SQL program.

Background: The creation and usage of PL/SQL Collections (Varray,Nested table,Records) has been explained to you in the classroom. This step by step guide will let you understand the usage of PL/SQL collections

Estimated time: 5 Minutes

Step 1: Write the following code to declare, initialize and display from a VARRAY Object

```
SQL> SET SERVEROUTPUT ON
DECLARE
 TYPE t_collect is VARRAY(4) OF NUMBER(10);
```

```
     V_A t_collect;
     V_idx NUMBER;
 BEGIN
--Initialize the collectin with two values.
  V_A :=t_collect(1,2);

--Extend the collection with extra values
<<load loop>>
     FOR I IN 3...5 LOOP
          V_A.extend;
          V_A (V_A.LAST):=1;
     END LOOP load_loop;

     --Traverse collection
V_idx:=V_A.FIRST;
     <<Display Loop>>
     WHILE V_idx IS NOT NULL LOOP
          DBMS_OUTPUT.PUT_LINE('The number'|| V_A(V_idx);
          V_idx :=V_A.NEXT(V_idx);
          END LOOP display_loop;
 END;
```

Step 3: Execute the above code at SQL prompt .

After execution the output will be as follows

```
     The number 1
     The number 2
     The number 3
      The number 4
```

Summary of this exercise:
You have learnt

- Step by step procedure to use a PL/SQL collection.

3.2 Assignment 2: Working with PL/SQL Record

Objective :To declare and reference a PL/SQL Record

Estimated time: 5 Minutes

Step 1: Write the following code to declare,

```
DECLARE
TYPE EMPRECTYPE IS RECORD
(
EMPNO NUMBER(4),
ENAME VARCHAR2(20),
SAL NUMBER(9,2)
);--DEFINING A RECORD
EMPREC EMPRECTYPE;--DECLARING A RECORD
CURSOR C1 IS
SELECT EMPNO,ENAME,SAL FROM EMP WHERE EMPNO=7839;
BEGIN
      OPEN C1;
      FETCH C1 INTO EMPREC;
      DBMS_OUTPUT.PUT_LINE(EMPREC.EMPNO || '   ' ||      EMPREC.ENAME ||
'   ' || EMPREC.SAL);--REFERENCING RECORD VARIABLES
      CLOSE C1;
END;
```

Step 3: Execute the above code at SQL prompt .

Summary of this exercise:
You have learnt

- Step by step procedure to use a PL/SQL Record

3.3 Assignment 3:Solve the following

Please write code for the following requirements:

1. Code a PL/SQL Program that uses a PL/SQL table Collection .
   a. Create the collection that must be dense , having consecutive subscripts for the elements
   b. Once created delete an element using the DELETE method to make the collection sparse
   c. Use the NEXT method and traverse the sparse collection. **(10 minutes)**

Summary of this assignment:

You have just learnt the use of Collections and Records

3.4 Assignment 4: Working with Bulk Collect

Objective : Populating Collections Using Bulk Operations.

Estimated Time : 15 Minutes

Step 1:

```
      CREATE TABLE BULK_TEST AS
            SELECT OWNER,OBJECT_NAME,
            OBJECT_ID FROM ALL_OBJECTS;
```

Step 2 :

```
      SET SERVEROUTPUT ON
      DELCARE
            TYPE T_BULK IS TABLE OF BULK_TEST%ROWTYPE;
            V_TAB  T_BULK:=T_BULK();
            V_START NUMBER;
          BEGIN
             --TIME A REGULAR POPULATION
                  V_START :=DBMS_UTILITY.GET_TIME;

            FOR REC IN (SELECT * FROM BULK_TEST)
            LOOP
                  V_TAB.EXTEND;
                  V_TAB(V_TAB.LAST):=REC;
            END LOOP;
      DBMS_OUTPUT.PUT_LINE('REGULAR'||V_TAB.COUNT || ':'||
DBMS_UTILITY.GET_TIME-V_START);

--TIME BULK OPERATIONS
      V_START :=DBMS_UTILITY.GET_TIME;
      SELECT *
            BULK COLLECT INTO V_TAB FROM BULK_TEST;
      DBMS_OUTPUT.PUT_LINE('REGULAR'||V_TAB.COUNT || ': '||
DBMS_UTILITY.GET_TIME-V_START);

END;
```

Step 3:

Execute the above PL/SQL Code

You will get the following output

```
REGULAR 61204 : 29
BULK    61204 : 11
```

Summary of this assignment:

You have just learnt the use of BULK COLLECT

3.5 Assignment 5: Solve the following

Please write code for the following requirements:

1. Write a PL/SQL block that uses a LIMIT clause to split the above collection into chunks of 1000. (**5 Minutes)**
2. Write a PL/SQL code that delete all the rows from the emp table (use FOR ALL statement. **(5 Minutes)**

Summary of this assignment:

You have just learnt the use of FOR ALL and BULK COLLECT

3.6 Assignment 6: Creating PL/SQL stored procedures

Objective: To understand how to write, compile and execute a PL/SQL stored procedure.

Background: The creation of PL/SQL stored procedure has been explained to you in the classroom. This step by step guide will let you understand the procedure of creation of a PL/SQL stored procedure.

Estimated time: 20 Minutes

Note: Before running this block type 'SET SERVEROUTPUT ON' at the SQL Prompt in SQL* Plus

Step 1: Consider the emp relation in your Oracle schema

Step 2: Write the following code in notepad to create the stored procedure Get_Grade and save the file with name “getgrade.sql”:

```
CREATE OR REPLACE PROCEDURE Get_Grade( p_eNo IN EMP.EmpNo%TYPE :=0 ,
p_eGrade OUT EMP.Grade%TYPE) IS
BEGIN
  SELECT grade into p_eGrade FROM emp WHERE EmpNo = p_eNo;
EXCEPTION
WHEN NO_DATA_FOUND THEN
  p_eGrade := 'Z';
WHEN OTHERS THEN
  p_eGrade :='Z';
  dbms_output.put_line('*** Error occurred ***');
  dbms_output.put_line('SQLCODE: '||to_char(SQLCODE));
  dbms_output.put_line('SQLERRM: '||SQLERRM);
END;
/
```

Step 3: Execute the following statement at SQL prompt to compile the stored procedure (assuming you have saved your procedure in E:\workarea directory):

```
@ e:\workarea\getgrade.sql
```

You will get the following output:
Procedure created.

Note: If there are some errors while compiling the same you can see the errors using **SHOW ERRORS** command at SQL PROMPT. Rectify error, if any and than proceed with the next step.

Step 4: Stored procedure is compiled and ready to use. Create an anonymous PL/SQL block as given below to call the stored procedure Get_Grade.

```
DECLARE
 v_employeeNo EMPLOYEE.EmpNo%TYPE;
 v_employeeGrade EMPLOYEE.Grade%TYPE;
BEGIN
v_ employeeNo := 1;
 Get_Grade(v_employeeNo, v_employeeGrade);
 IF v_employeeGrade = 'Z' THEN
  dbms_output.put_line('Employee No Not Found');
 ELSE
  dbms_output.put_line('Employee Grade is '||v_employeeGrade);
 END IF;
EXCEPTION
 WHEN OTHERS THEN
```

```
  dbms_output.put_line('*** Error occurred ***');
  dbms_output.put_line('SQLCODE: '||to_char(SQLCODE));
  dbms_output.put_line('SQLERRM: '||SQLERRM);
END;
/
```

You will get the following output (assuming the grade of employee no 1 is 'A'):

```
Employee Grade is A

PL/SQL procedure successfully completed.
```

Summary of this exercise:
You have learnt

- Step by step procedure to create a PL/SQL stored procedure.
- Remember the stored procedures are saved into your database in a compiled format.

3.7 Assignment 7: Solve the following

Please write code for the following requirements:

1. Code a PL/SQL block which calls a procedure (**not** a stored procedure) to get the total quantity supplied for a particular product. The procedure should total the quantity, and the calling block should print the product no, product description and the total quantity supplied. The procedure should raise an exception if the total quantity supplied is 0, or the product does not exist. **(15 Min)**

2. Convert problem above to invoke a function, instead of a procedure. The function should return the total qty supplied. **(10 Min)**

3. Code a PL/SQL block that will call a procedure to insert a new record into the Suppliers_Products table, and also update the corresponding Qty_On_Hand for the product. The procedure should return the status of completion to the calling program. **(15 Min)**

4. Code a stored procedure to insert a new record into the products table (initial Qty_On_Hand should be 0). The procedure should handle appropriate exceptions. The productno should be obtained by using the sequence Seq_Product. Call the stored procedure from the SQL prompt or from an anonymous block of code. **(15 Min)**

5. *Add a column called Location varchar2(20) to the Employees table. Run a stored procedure, which will update the location of each employee to the same location as that of the Department.* **(15 Min)**

6. For each of the above blocks of code, add the appropriate exception handlers. If you do not know of the exception which may occur, print the sql error code and error message when an unknown exception occurs. **(10 Min)**

Summary of this assignment:

You have just learnt

- How to create various database schema objects like Stored Procedure & Stored Functions.

3.8 Assignment 8: Recursive Functions

Objective: Showing the Factorial series in context to recursion

Step 1:

```
SQL> CREATE FUNCTION fac(x NUMBER) RETURN NUMBER
      IS
      BEGIN
      IF x=1 THEN
      RETURN 1;
      ELSE
       RETURN(x * fac(x-1));
      END IF;
      END;
```

Step 2: Calling the fac function

```
SQL> SET SEREVEROUTPUT ON
      DECLARE
      Y NUMBER;
      BEGIN
      Y:=fac(5);
      DBMS_OUTPUT.PUT_LINE('Factorial of five is' || y);
      END;
```

Step 3: Execute the above code
You will get the following output.

```
Factorial of five is 120
```

Summary of this assignment:

You have just learnt the use of recursive functions

4. Assignments for Day 4 of Oracle

All the assignments in this section must be completed on Day 4 of your Oracle course.

4.1 Assignment 1 : Packages

Objective: To Learn to create and Implement Packages .

Estimated Time : 5 Minutes.

Step 1 : Create Package Specification comm_package to compute the commission .

```
CREATE OR REPLACE PACKAGE comm_package IS
  g_comm NUMBER := 100;  --initialized to 100
  PROCEDURE reset_comm  (p_comm   IN  NUMBER);
END comm_package;
```

Step 2 : Create Package body comm_package to compute the commission .

```
CREATE OR REPLACE PACKAGE BODY comm_package
  IS
  FUNCTION  validate_comm
    (p_comm   IN   NUMBER)
    RETURN BOOLEAN
  IS
    v_max_comm    NUMBER;
  BEGIN
    SELECT   max(comm)
    INTO v_max_comm
```

```
    FROM      emp;
    IF   p_comm > v_max_comm
      THEN   RETURN(FALSE);
    ELSE   RETURN(TRUE);
    END IF;
  END validate_comm;
  PROCEDURE  reset_comm
    (p_comm   IN  NUMBER)
  IS
  BEGIN
    IF  validate_comm(p_comm)
      THEN   g_comm:=p_comm;  --reset global variable
    ELSE
      RAISE_APPLICATION_ERROR (-20210,'Invalid commission');
    END IF;
  END reset_comm;
END comm_package;
/
```

Step 3 : Executing the Packaged Procedure.

```
EXECUTE comm_package.reset_comm(1000)
```

Summary of this assignment:

You have just learnt

- How to create and executing Packages.

4.2 Assignment 2 Solve the following

Create and implement the following packaged subprograms:

1. Create and invoke a package that contains private and public constructs.

    1. Create a package specification and body called PACK_EMPLOYEE. (You can save the package body and specification in two separate files.) This package contains HIRE_EMPLOYEE, UPD_EMPLOYEE, and FIRE_EMPLOYEE procedures, as well as Q_EMPLOYEE function..

        a. HIRE_EMPLOYEE Procedure should take employee details as the input parameters. It should validate the DEPTNO by using a private function VALID_DEPTNO which checks the existence of DEPTNO in DEPT table and

returns a BOOLEAN value accordingly. After successful validation of the DEPTNO inserts the employee details into the EMP table **(5 Minutes)**

b. UPD_EMPLOYEE Procedure should take existing EMPNO as the input Parameter and Raise the Salary of the Employee based on the following criteria
   i. 10% - if the employee is a Manager
   ii. 15% -if the employee is a Analyst
   iii. 5% -any other JobType. **(5 Minutes)**

c. FIRE_EMPLOYEE Procedure should take the EMPNO as the input Parameter and delete the employee data **(5 Minutes)**

d. Q_EMPLOYEE Function should pass the EMPNO as INPUT Paramenter and return the SAL + COMM of the employee. The function should address NULL values. **(5 Minutes)**

Test and invoke the Packaged Procedures and Functions.

Summary of this assignment:

You have just learnt

- How to create Packages.

4.3 Assignment 3: Package Overloading

Objective: To Learn about Package Overloading

Estimated time: 10 minutes

Step 1: Create Package Specification PACK_NAME with two overloaded functions NAME.

```
CREATE OR REPLACE PACKAGE PACK_NAME
IS
FUNCTION NAME(eno NUMBER)
RETURN VARCHAR2;

FUNCTION NAME(Emp_name VARCHAR2)
RETURN NUMBER;

END PACK_NAME;
```

Step 2: Create a Package Body implementing the Overloaded Functions.(Assume that there are unique names in the TABLE EMP).

```
CREATE OR REPLACE PACKAGE BODY PACK_NAME
IS

FUNCTION NAME(eno NUMBER)
RETURN VARCHAR2
IS
Emp_name VARCHAR2(15);
BEGIN
SELECT ENAME INTO emp_name FROM EMP WHERE EMPNO=eno;
RETURN(emp_name);
END;

FUNCTION NAME(Emp_name VARCHAR2)
RETURN NUMBER
IS
   Eno NUMBER;
BEGIN
SELECT EMPNO INTO eno FROM EMP WHERE ENAME=emp_name;
 RETURN(eno);
END;

END PACK_NAME;
```

Step 3: Calling both functions.

```
DECLARE
x NUMBER;
y VARCHAR2(15);
BEGIN
Y:=PACK_NAME.NAME(7839);
DBMS_OUTPUT.PUT_LINE('Employee Name retrieved corresponding to the
employee number is : || y);
x:=PACK_NAME.NAME('King');
DBMS_OUTPUT.PUT_LINE('Employee Number retrieved corresponding to the
employee name is :'|| x);
END;
```

Step 4 :

Executing the PL/SQL block will call the appropriate functions.

```
Employee Name retrieved corresponding to the employee number is : King
Employee Number retrieved corresponding to the employee Name is : 7839
```

Summary of this assignment:

Subprograms inside a package can be overloaded.

4.4 Assignment 4: Package Overloading

Objective: To solve exercise on Package Overloading

Estimated time: 15 minutes

1. Create a package called OVER_LOAD. Create two functions in this package; name each function PRINT_IT. The function accepts a date or character string and prints a date or a number, depending on how the function is invoked.

Note:

- To print the date value, use DD-MON-YY as the input format, and FmMonth,dd yyyy as the output format. Make sure you handle invalid input.
- To print out the number, use 999,999.00 as the input format.

a.Test the first version of PRINT_IT with the following set of commands:

```
VARIABLE display_date VARCHAR2(20)
EXECUTE :display_date := over_load.print_it (TO_DATE('08-MAR-01'))
PRINT display_date
```

b.Test the second version of PRINT_IT with the following set of commands:

```
VARIABLE g_emp_sal NUMBER
EXECUTE :g_emp_sal := over_load.print_it('33,600')
PRINT g_emp_sal
```

4.5 Assignment 5 : DML Triggers

Objective: To Learn DML Triggers

Let us try to understand how the DML Trigger works

Background: The types of triggers and the syntax to create triggers has been explained to you in the classroom. This step by step guide will let you understand the procedure of creation of database trigger.

Estimated time: 15 Minutes

Note: Before running this block type 'SET SERVEROUTPUT ON' at the SQL Prompt in SQL* Plus

Step 1: Consider the emp relation in your Oracle schema

Step 2: Write the following code in notepad and save the file with name "trig_insert.sql":

```
create or replace trigger trig_test
after insert on emp
begin
 dbms_output.put_line('Data Inserted into the Emp Table
                                                  successfully');
end;

/
```

Step 3: Execute the following statement at SQL prompt to compile the trigger code (assuming you have saved your trigger in E:\workarea directory):

```
@ e:\workarea\trig_insert.sql
```

You will get the following output:
Trigger created.

Note: If there are some errors while compiling the same you can see the errors using **SHOW ERRORS** command at SQL PROMPT. Rectify error, if any and then proceed with the next step.

Step 4: Trigger is created now successfully. Whenever a new record is inserted into Emp table then the trigger will be fired.

```
SQL> insert into emp(empno,ename,sal) values (100, 'kite',10000);
```

```
1 row created.
```

You will get the following output .

```
Data Inserted into the Emp Table successfully
```

Summary of this exercise:

You have learnt

- Step by step procedure to create statement level trigger.

4.6 Assignment 6 : Use of New qualifier Trigger

(i) Execute the following Code snippet to access new qualifier in a before update row trigger

Step 1 : Create a before row trigger that modifies the commission of the employee for the job type as 'EDU'

```
create or replace trigger Trig_upd
 before update of sal on emp
 for each row
 when (NEW.JOB = 'EDU')
 begin
 :NEW.comm := 100;
 end;
```

Step 2 :

```
update emp set job='EDU', sal= 50000 where empno = 7934;
```

Step 3 : Execute the below query to see the updated salary for the job entered as 'EDU'

```
select * from emp;
```

(ii) Execute the following Code snippet to access new qualifier in a after update row trigger

Step 1 : Create a after row trigger that modifies the commission of the employee for the job type as 'EDU'

```
 create or replace trigger g
```

```
after update of sal on emp
for each row
when (NEW.JOB = 'EDU')
begin
:NEW.comm := 100;
end;
```

Executing the statement gives the following error

```
ERROR at line 1:
ORA-04084: cannot change NEW values for this trigger type
```

Note: cannot change NEW values in the after Triggers.

Summary of this exercise:
You have learnt

- Use of New qualifiers in TRIGGERS

4.7 Assignment 7 : Solve the following

1. Add a column location in the Emp table.Whenever the location of the department changes, update the employee location also. (Use an AFTER UPDATE trigger for this). **(20 Min)**

2. Add a new column called Performance_Measure Number(3,1) to the EMP table. The column can take values between 1 and 10(Use a check constraint to implement this). Whenever the performance measure of an employee is 8 or more, insert a record into a table called EXCEPTIONAL_EMPLOYEES, which contains the columns EMPNO and PERFORMANCE_LINKED_BONUS. Write a [AFTER INSERT OR AFTER UPDATE] trigger for this. **(20 Min)**

3. Write a Statement Level Trigger that gets fired whenever employee salaries are updated. The trigger inserts a record into the table SALARY_UPDATE_LOG that contains the USERNAME, DATE OF UPDATION of the user who updated the salary.(Hint: To get the username of the user currently logged in use a data dictionary view. Consult the Oracle Server Reference Manual on the Oracle Online documentation for the data dictionary views.) **(20 Min)**

4.8 Assignment 8 : Instead of Triggers

Solve the following :

1. Create a view Emp_Dept_Upd that queries empno,ename,deptno,dname from EMP and DEPT table. Create an INSTEAD of Trigger on the view Emp_Dept_Upd that updates the data on the Join View.

4.9 Assignment 9 : DDL Triggers

Objective: To Learn DDL Triggers

Let us try to understand how the DDL Trigger works

Note: DDL Triggers cannot be applied on individual Tables , you must create DDL Triggers either on the database or on the current schema.

Estimated time: 5 minutes

Step 1: First Let us create a DDL Trigger named cre_ate on the current schema.This trigger will be fired when a current user executes the create statement.

```
SET SERVEROUTPUT ON
CREATE OR REPLACE TRIGGER cre_ate
AFTER CREATE ON SCHEMA
BEGIN
DBMS_OUTPUT.PUT_LINE('Execution successful');
END cre_ate;
```

The output of the above code is

```
Trigger Created
```

Step 2: Create a table to check whether the trigger is fired when we executed the CREATE statement.

```
CREATE TABLE STUDENT (sname VARCHAR2(50),age NUMBER(2));
```

Step 3: Executing the statement fires the trigger created.

```
Execution successful
Table Created
```

Summary of this exercise:
You have learnt

- Step by step procedure to create a DDL Trigger.

4 .10 Assignment 10 : Solve the following

1. Create a DDL Trigger that fires whenever CREATE ,ALTER , DROP is executed on the current schema and displays the message " **Execution successful".(5 Minutes)**

2. Create a Table Audit_DDL (User_name,DDl_Event_Type,Date_of_Exceution) .Write a DDL Trigger that fires whenever CREATE ,ALTER , DROP is executed on the current schema and inserts into into the Audit_DDL table the current user , DDL Type and Date of execution.**(10 Minutes)**

Hint : Use **`ora_sysevent`** `built in function to get the type of DDL event.`

3. Create a Trigger Restrict_DROP that restricts DROP operation of any database object in the current schema on a Saturday or a Sunday..**(5 Minutes)**

Summary of this assignment:

You have just learnt

- How to create a DDL Trigger.

4 .11 Assignment 11 : Cascading Triggers

Objective: To understand Cascading Triggers .

Estimated time: 10 minutes

Step 1:

```
CREATE TABLE cascade (
testcol VARCHAR2(10));
```

Step 2: .

```
CREATE OR REPLACE TRIGGER t_cascade
AFTER INSERT
ON cascade
BEGIN
  INSERT INTO cascade
  (testcol)  VALUES  ('change');
END t_cascade;
/
```

Step 3:

```
INSERT INTO cascade (testcol) VALUES ('ABC');
```

Step 4 :

Executing the statement fires the trigger created.

```
ERROR at line 1:
ORA-00036: maximum number of recursive SQL levels (50) exceeded
ORA-06512: at "TEST1.T_CASCADE1", line 2
ORA-04088: error during execution of trigger 'TEST1.T_CASCADE'
ORA-06512: at "TEST1.T_CASCADE", line 2
ORA-04088: error during execution of trigger 'TEST1.T_CASCADE'
ORA-06512: at "TEST1.T_CASCADE", line 2
ORA-04088: error during execution of trigger 'TEST1.T_CASCADE'
ORA-06512: at "TEST1.T_CASCADE", line 2
ORA-04088: error during execution of trigger 'TEST1.T_CASCADE'
ORA-06512: at "TEST1.T_CASCADE", line 2
ORA-04088: error during execution of trigger 'TEST1.T_CASCADE'
ORA-06512: at "TEST1.T_CASCADE", line 2
ORA-04088: error during execution of trigger 'TEST1.T_CASCADE'
ORA-06512: at "TEST1.T_CASCADE", line 2
```

Summary of this exercise:

When a statement in a trigger body causes another trigger to be fired, the triggers are said to be cascading. This may lead to recursive calls to the Trigger .

4.12 Assignment 12 : Mutating Triggers

Objective: To understand Mutating Triggers .

Estimated time: 10 minutes

Step 1: Create tables t1 and t2. Insert the data into the table t1 .Query the data from t1 and t2 tables.

```
CREATE TABLE t1 (x int);
CREATE TABLE t2 (x int);

INSERT INTO t1 VALUES (1);

SELECT * FROM t1;
SELECT * FROM t2;
```

Step 2: Create a Trigger after insert on the t1 table , count the number of rows in the table t1 and insert the data into the table t2.

```
CREATE OR REPLACE TRIGGER t_trigger
AFTER INSERT
ON t1
FOR EACH ROW

DECLARE
 i PLS_INTEGER;
BEGIN
  SELECT COUNT(*)
  INTO i
  FROM t1;

  INSERT INTO t2
  VALUES
  (i);
END;
/
```

Step 3:

```
INSERT INTO t1 VALUES (1);
```

Step 4 :

Executing the statement fires the trigger created and results in a mutating condition.

```
ERROR at line 1:
ORA-04091: table TEST1.T1 is mutating, trigger/function may not see it
ORA-06512: at "TEST1.T_TRIGGER", line 4
ORA-04088: error during execution of trigger 'TEST1.T_TRIGGER'
```

Summary of this exercise:

The insert into t1 fires the trigger on t1 which attempts to count the number of records in t1 ... which is ambiguous and leads to mutating Condition.

Assignments for Day 5 of Oracle

All the assignments in this section must be completed on Day 5 of your Oracle course.

5.1 Assignment 1: Programming using Pro * C

Objective: To write a Pro C program which will query the employee table and display the information.

Background: The creation of Pro C program has been explained to you in the classroom. This step by step guide will let you understand the procedure of creation of Pro C program.

Note: It is assumed that user name is **<tEmpid>** , password is **<tEmpid>** and host string is **GEORLI01.** To use this program. you can use your username password and host string. Please do the necessary changes with respect to this in the forthcoming steps.

Step 1: Create the employee relation in your Oracle schema:

```
Create table employee (
empNo number,
empName varchar2(20),
job varchar2(9),
mgr number(4),
hiredate date,
sal number(7,2),
comm number(7,2),
deptno number(2)
);
```

Step 2 : Use notepad to create the following program which will query the employee table and display the information.Save as the file employeeinfo.pc

/* Program to query Employee table and display information */

#include<stdio.h>
#include<string.h>

```
exec sql begin declare section;
char username[20];

/*input host variable*/
int empno1;

/*outut host variable*/
char empname[20];
float salary;
float commission;

/*indicator variables*/
short empname_ind;
short salary_ind;
short comm_ind;

exec sql end declare section;
exec sql include sqlca;

main()
{
  exec sql whenever sqlerror goto sqlerror;
   /*connect to ORACLe*/
   strcpy((char*)username,"tEmpid/tEmpid@GEORLI01");

    exec sql connect :username;

     /*get input data*/
     printf("\nEnter the employee number:");
     scanf("%d",&empno1);

     exec sql
      select ename,sal,comm into
:empname:empname_ind,:salary:salary_ind,:commission:comm_ind from emp where
empno=:empno1;

      if(salary_ind==-1)
         salary=0;
      if(comm_ind==-1)
         commission=0;
/* print output data*/
          printf("\nemployee name: %s",empname);
          printf("\nsalary: %f",salary);
           printf("\ncommission: %f",commission);


 /*exit from oracle*/
        exec sql commit work release;
        exit(0);
```

```
        sqlerror:

         printf("\n***sql error has occured****");
         printf("\nsql code: %d",sqlca.sqlcode);
         printf("\nsql error message: %s",sqlca.sqlerrm.sqlerrmc);
}
```

Step 3 : Execute the following command in the command prompt to prepare the employeeinfo.c file.

C:\Documents and Settings\user>proc iname=employeeinfo.pc SQLCHECK=SEMANTICS userid=scott\infy@georli01

Step 4 : Open an empty project in VC++ editor and add this file to that. Also add the following file from the mentioned location.

orasql10.lib

\ORACLE_HOME\product\10.2.0\db_1\precomp\LIB

Step 5 : Now compile and execute the file.

1. Write a Pro C program using cursor to count the total number of employees. Use the same employee table as used above. The procedure should handle appropriate exceptions.

2. Write a Pro C program which will display the details of those employees whose dept is MGR. Also display the total number of employees of MGR category.

3. Write a Pro C program which will add the comm. For all the employees as 10% of their salary, for those employees whose salary is less than 10000.

4. Write a Pro C program to update the COMM column of EMP table with the following calculation. Handle the exceptions where ever possible
    a. If SAL is in the range SAL>1000 and SAL<=2000, then commission is 10% of SAL
    b. If SAL is in the range SAL>2000 and SAL<=3000, then commission is 20% of SAL
    c. If SAL is in the range SAL>3000 and SAL<=4000, then commission is 30% of SAL
    d. If SAL is in the range SAL>4000 and SAL<=5000, then commission is 40% of SAL
5. Write a Pro C program to do the following. Handle the exceptions where ever possible
    a. Prompt the user for Add, Update and Display Employee records.
        i. Insertion of employee records; Get the details from the user and do the insert operation in the database
        ii. Update of employee records; Get the relevant details from the user and do the update operation in the database
        iii. Report generation of Employee records; Get the employee number from the user and display the details of that employee

6. Repeat the above programs using C++ by implementing appropriate classes and methods