1)1.Create a plsql procedure, that copy the all record of department\_id, department\_name and

-- location\_id of all departments that located in country\_id of "US" to a table called "Department\_US" using the

-- cursor with basic loop.

-- Display how many rows were copied , if not then display the message "NO RECORDS FOUND".

-- Note :create a table Department\_US with

-- structure ((HR SCHEME)

-- Field Name Datatype

-- ----------------- -------------

-- Department\_Id number

-- Department\_name varchar2(40)

-- Location\_id number

create or replace procedure hr\_procedure is

v\_department\_id departments.department\_id%type;

v\_department\_name departments.department\_name%type;

v\_location\_id locations.location\_id%type;

v\_rows\_count number := 0;

cursor c\_hrs is

select d.department\_id,d.department\_name , l.location\_id

from hr.departments d

join hr.locations l on d.location\_id = l.location\_id

where l.country\_id = 'US';

Begin

open c\_hrs;

loop

fetch c\_hrs into v\_department\_id , v\_department\_name , v\_location\_id;

exit when c\_hrs%notfound;

insert into departments\_us(department\_id,department\_name , location\_id ) values(v\_department\_id,v\_department\_name,v\_location\_id);

v\_rows\_count := v\_rows\_count + 1;

end loop;

close c\_hrs;

if v\_rows\_count > 0 then

dbms\_output.put\_line('Data Found');

else

dbms\_output.put\_line('No Data Found');

end if;

exception

when others then

dbms\_output.put\_line('NO RECORDS FOUND');

end hr\_procedure;

/

-- Create a procedure named as "deleteemp( depit in number) " that delete rows from the employees table.

-- It should accept 1 parameter, departmentId; only delete the employee records belonging to that departmentid.

-- Display how many employees were deleted else raise"DeptIDNotFound" and print the message 'No Records Found' (HR SCHEME)

create or replace procedure deleteemp(depit in number) is

no\_rows\_del number := 0;

DeptIDNotFound Exception;

begin

delete from hr.employees where department\_id = depit;

no\_rows\_del := sql%rowcount;

if no\_rows\_del = 0 then

raise DeptIDNotFound;

else

dbms\_output.put\_line(no\_rows\_del || 'no records deleted');

end if ;

exception

when DeptIDNotFound then

dbms\_output.put\_line('No Records Found');

when others then

dbms\_output.put\_line('An error Found'|| SQLERRM);

end deleteemp;

/

-- create a proedurce that gives all employees in IT job\_id, with 22 % pay increase in there salary

-- Display a message how many Employees were give salary hike the increase. If no employee found then print the message 'No Records Found' ((HR SCHEME))

create or replace procedure sal\_hike\_emp as

v\_row\_count number ;

norecordsfound Exception;

begin

update hr.employees set salary = salary \* 0.22 where job\_id like '%IT%';

v\_row\_count := sql%rowcount;

if v\_row\_count = 0 then

raise norecordsfound;

else

dbms\_output.put\_line(v\_row\_count);

end if;

exception

when norecordsfound then

dbms\_output.put\_line('No records Found.');

end sal\_hike\_emp;

/

-- Write a procedure to perform salary hike operation that only permits a hike ,

-- if there is salary is less than 8000, then update the salary with employee table and print the message 'Salary Hike is successful'

-- else print 'Salary is greater than 8000'.((HR SCHEME)) procedure named as salhike(hike as number)

create or replace procedure salhike(hike in number) as

begin

update hr.employees set salary = salary\*commission\_pct + salary where salary < 8000;

if salary < 8000 then

dbms\_output.put\_line('Salary hike is Successful');

else

dbms\_output.put\_line('Salary is greater than 8000');

end if;

end salhike;

/

-- 5.Write procedure to display the number of department to the location\_id as 'Number of Department in location id[50] is [25]'

-- by pass the argument to procedure as location\_id with named as deptcount(loc number).

create or replace procedure deptcount(loc in number) is

v\_dept\_count number ;

begin

select count(department\_id) into v\_dept\_count from hr.departments where location\_id = loc;

dbms\_output.put\_line('Number of Departments in location id [' || loc || '] is [' || v\_dept\_count ||']');

end deptcount;

-- Write the procedure to copy the manager record from employees table and move records to managers table.

-- Procedure is named with mgremp and display the "Number of record is copied is" if not then "No record insert into manager" using the basic loop with cursor.

-- Note: create a table mgremp with stucture

-- empid number,

-- first\_name varchar2(30);

-- job\_id varchar2(30);

-- department\_id number..

create table mgremp(empid number,first\_name varchar2(30),job\_id varchar2(30),department\_id number);

create or replace procedure v\_mgremp is

v\_manager employees.manager\_id%type;

v\_rows\_count number := 0;

cursor c\_emp is

insert into mgremp (manager\_id ) select manager\_id from employees ;

begin

loop

open c\_emp ;

fetch c\_emp into v\_manager;

exit c\_emp%notfound;

if v\_rows\_count > 0 then

dbms\_output.put\_line('number of records copied is ' || v\_rows\_count);

else

dbms\_output.put\_line('No records copied');

end if;

end loop;

close c\_emp;

end v\_mgremp;

/

FUNCTIONS :

-- 1. Write a PLSQL user defined function as empname to concatenate firstname and lastname of an employee. Pass employee id as an input to the functions

-- empname Output of function returns :- Mr/Mrs. Sam Peter

create or replace function empname(v\_employee\_id in number,v\_empname out varchar2) return varchar2 as

message varchar2(50);

Begin

select (first\_name || ' ' || last\_name) as full\_name into v\_empname from hr.employees where employee\_id = v\_employee\_id;

message := 'Mr/Mrs' || v\_empname;

return message;

end empname;

/

-- Write user define function as deptloc to return departmentname-city-Country\_name example "IT-NewYork-USA".

-- Function is passed with department\_id as an input

create or replace function deptloc(v\_deptid in number) return varchar2 is

message varchar2(50);

v\_department\_name hr.departments.department\_name%type;

v\_city hr.locations.city%type;

v\_country\_name hr.countries.country\_name%type;

begin

select d.department\_name , l.city , c.country\_name into v\_department\_name , v\_city , v\_country\_name from hr.departments d

join hr.locations l on d.location\_id = l.location\_id

join hr.countries c on l.country\_id = c.country\_id where d.department\_id = v\_deptid;

message := v\_department\_name || ' - ' || v\_city || ' - ' || v\_country\_name;

return message;

end ;

/

-- Write function as empexp( empid number) which returns the experence of the employee,

-- When function is passed as input of employee\_id return as experence of that employee.

create or replace function empexp(empid in number) return number is

message number;

v\_months number;

v\_years number;

begin

select floor(months\_between(sysdate,hire\_date)/12) as years\_exp , mod(floor(months\_between(sysdate,hire\_date)),12) as year\_exp into v\_years , v\_months

from hr.employees where employee\_id = empid;

message := v\_years || 'years' || v\_months ||'months';

return message ;

end ;

-- 4.Write a function to calculate the total salary by salary commission

-- and deduction of professional tax of 6% from total salary if total salary is greater than or equal to $15000 and less than 15000 is 2.5%.

-- function named as salcal with two parameter of salary and commission.

create or replace function salcal (p\_salary in NUMBER, p\_commission in NUMBER) return number is

v\_total\_salary number;

v\_final\_salary number;

begin

-- Calculate the total salary by adding the salary and commission

v\_total\_salary := p\_salary + p\_commission;

-- Deduct the professional tax based on the total salary

if v\_total\_salary >= 15000 then

v\_final\_salary := v\_total\_salary \* (1 - 0.06); -- Deduct 6% tax

else

v\_final\_salary := v\_total\_salary \* (1 - 0.025); -- Deduct 2.5% tax

end if;

return v\_final\_salary;

end salcal;

/

TRIGGERS :

-- 1. Write the trigger to the employee table when the deleted event happened.

-- so that when an employee record is deleted the record details need to be inserted into an table called X\_Emp along with deleted date

-- Note : Assume X\_emp table is existing with

-- Field Name datatype

-- --------------------------------

-- Employee\_ID number

-- First\_name Varchar2(30)

-- department\_id number

-- deleted\_date date

create table X\_emp(Employee\_ID number,First\_name Varchar2(30), department\_id number,deleted\_date date);

create or replace trigger emp\_trigger before delete on hr.employees for each row

declare

v\_emp\_id hr.employees.employee\_id%type;

v\_first\_name hr.employees.first\_name%type;

v\_deptid hr.employees.department\_id%type;

begin

v\_emp\_id := :old.employee\_id;

v\_first\_name := :old.first\_name;

v\_deptid := :old.department\_id;

insert into X\_emp values(v\_emp\_id,v\_first\_name,v\_deptid,sysdate);

end emp\_trigger;

/

-------------------------or--------------------------------------

-- create or replace trigger emp\_trigger before delete on hr.employees for each row

-- declare

-- begin

-- insert into X\_emp values(:old.employee\_id,:old.first\_name,:old.department\_id,sysdate);

-- end emp\_trigger;

-- /

-- 2. Create a tigger to display the message "Place a order for the Product <product\_name>",

-- when ever a item quantity reached 10 and below in product table when updating or inserting an item in order table.

-- Products table

-- -----------------------

-- Pid ProductName Qty

-- 100 Mouse 50

-- 101 Keyboard 32

-- 102 Pendrive 5

-- 103 RAM 12

-- Order Table

-- ------------------

-- Oid Pid qty

-- 1000 100 45

-- 1001 101 2

-- 1003 102 25

-- Output:

-- "Place a Order for the product Mouse

create or replace trigger trigger\_name before update or insert on orders for each row

declare

begin

if inserting then

insert into products values(:new.Pid,:new.ProductName,:new.Qty);

elsif updating then

update products set qty = qty - :new.qty where pid = :old.pid;

if (select Qty from products where pid = :old.pid <= 10) then

dbms\_output.put\_line('Place a order for the Product ' || :old.productname);

end if;

end if;

end trigger\_name;

----------------------------or----------------------------------

-- CREATE OR REPLACE TRIGGER trigger\_name

-- AFTER INSERT OR UPDATE ON orders

-- FOR EACH ROW

-- DECLARE

-- v\_qty products.qty%TYPE;

-- v\_product\_name products.productname%TYPE;

-- BEGIN

-- -- Fetch the current quantity and product name from the products table

-- SELECT qty, productname

-- INTO v\_qty, v\_product\_name

-- FROM products

-- WHERE pid = :NEW.pid;

-- -- Check if the quantity is 10 or below

-- IF v\_qty <= 10 THEN

-- DBMS\_OUTPUT.PUT\_LINE('Place a order for the Product ' || v\_product\_name);

-- END IF;

-- END trigger\_name;

-- /

-- 3. ) Create trigger on the employee table when the update is happened to employee's salary field.

-- wirte the action to trigger to Insert the employee's old salary in empoldsal table

-- empoldsal table

-- ----------------------------

-- Field Name datatype

-- empid number

-- firstname varchar2(30)

-- Oldsalary number

create or replace trigger emp\_trg before update of salary on hr.employees for each row

declare

begin

insert into empoldsal values(:new.fieldName,:new.empid,:new.firstname,:old.oldsalary);

end emp\_trg;

/

-- 4.Write trigger on emp table for the event of update to column of salary.

-- when the salary is update to emp table stored old salary into empoldsal table. Display 'Record is inserted into empoldsal'.

create or replace trigger emp\_trg before update of salary on hr.employees for each row

declare

begin

insert into empoldsal values(:new.fieldName,:new.empid,:new.firstname,:old.oldsalary);

dbms\_output.put\_line('Record is inserted into empoldsal');

end emp\_trg;