```
--------------------------------------------------------------------------------
--------------------------------
-------------------------------------------CREATING & USING
PACKAGES-------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------
----------------- Creating first package specification
CREATE OR REPLACE
PACKAGE EMP AS
  v_salary_increase_rate number := 0.057;
  cursor cur_emps is select * from employees;

  procedure increase_salaries;
  function get_avg_sal(p_dept_id int) return number;
END EMP;
----------------- Creating the package body
CREATE OR REPLACE
PACKAGE BODY EMP AS
  procedure increase_salaries AS
  BEGIN
    for r1 in cur_emps loop
      update employees_copy set salary = salary + salary * v_salary_increase_rate;
    end loop;
  END increase_salaries;
  function get_avg_sal(p_dept_id int) return number AS
  v_avg_sal number := 0;
  BEGIN
    select avg(salary) into v_avg_sal from employees_copy where
          department_id = p_dept_id;
    RETURN v_avg_sal;
  END get_avg_sal;
END EMP;
----------------- using the subprograms in packages
exec EMP_PKG.increase_salaries;
----------------- using the variables in packages
begin
  dbms_output.put_line(emp_pkg.get_avg_sal(50));
  dbms_output.put_line(emp_pkg.v_salary_increase_rate);
end;

--------------------------------------------------------------------------------
----------
-------------------------------VISIBILITY OF VARIABLES IN
PACKAGES-------------------------
--------------------------------------------------------------------------------
----------
create or replace PACKAGE BODY EMP_PKG AS

  v_sal_inc int := 500;
  v_sal_inc2 int := 500;

  procedure print_test is
  begin
    dbms_output.put_line('Test : '|| v_sal_inc);
  end;

  procedure increase_salaries AS
  BEGIN
    for r1 in cur_emps loop
```

```
      update employees_copy set salary = salary + salary * v_salary_increase_rate
      where employee_id = r1.employee_id;
    end loop;
  END increase_salaries;
  function get_avg_sal(p_dept_id int) return number AS
  v_avg_sal number := 0;
  BEGIN
    print_test;
    select avg(salary) into v_avg_sal from employees_copy where
          department_id = p_dept_id;
    RETURN v_avg_sal;
  END get_avg_sal;

END EMP_PKG;
-----------------
create or replace PACKAGE BODY EMP_PKG AS

  v_sal_inc int := 500;
  v_sal_inc2 int := 500;
  function get_sal(e_id employees.employee_id%type) return number;
procedure print_test is
  begin
    dbms_output.put_line('Test : '|| v_sal_inc);
    dbms_output.put_line('Test salary : '|| get_sal(102));
  end;
  procedure increase_salaries AS
  BEGIN
    for r1 in cur_emps loop
      update employees_copy set salary = salary + salary * v_salary_increase_rate
      where employee_id = r1.employee_id;
    end loop;
  END increase_salaries;
  function get_avg_sal(p_dept_id int) return number AS
  v_avg_sal number := 0;
  BEGIN
    print_test;
    select avg(salary) into v_avg_sal from employees_copy where
          department_id = p_dept_id;
    RETURN v_avg_sal;
  END get_avg_sal;

  function get_sal(e_id employees.employee_id%type) return number is
  v_sal number := 0;
  begin
    select salary into v_sal from employees where employee_id = e_id;
  end;

end;

--------------------------------------------------------------------------------
----------
---------------------------------PERSISTENT STATE OF
PACKAGES-------------------------------
--------------------------------------------------------------------------------
----------
-----------------
execute dbms_output.put_line(constants_pkg.v_salary_increase);
grant execute on constants_pkg to my_user;
revoke execute on constants_pkg from my_user;
```

```
-----------------
-----------------
begin
  constants_pkg.v_company_name := 'ACME';
  dbms_output.put_line(constants_pkg.v_company_name);
  dbms_lock.sleep(20);
end;
exec dbms_output.put_line(constants_pkg.v_company_name);
-----------------
create or replace package constants_pkg is
PRAGMA SERIALLY_REUSABLE;
  v_salary_increase constant number:= 0.04;
  cursor cur_emps is select * from employees;
  t_emps_type employees%rowtype;
  v_company_name varchar2(20) := 'ORACLE';
end;
-----------------
begin
  constants_pkg.v_company_name := 'ACME';
  dbms_output.put_line(constants_pkg.v_company_name);
  dbms_lock.sleep(20);
end;
-----------------
declare
v_emp employees%rowtype;
begin
 open constants_pkg.cur_emps;
 fetch constants_pkg.cur_emps into v_emp;
 dbms_output.put_line(v_emp.first_name);
 close constants_pkg.cur_emps;
end;
-----------------
declare
v_emp employees%rowtype;
begin
 fetch constants_pkg.cur_emps into v_emp;
 dbms_output.put_line(v_emp.first_name);
end;

--------------------------------------------------------------------------------
----------
---------------------------------USING COLLECTIONS IN
PACKAGES------------------------------
--------------------------------------------------------------------------------
----------
create or replace PACKAGE EMP_PKG AS
  type emp_table_type is table of employees%rowtype index by pls_integer;
  v_salary_increase_rate number := 1000;
  v_min_employee_salary number := 5000;
  cursor cur_emps is select * from employees;

  procedure increase_salaries;
  function get_avg_sal(p_dept_id int) return number;
  v_test int := 4;
  function get_employees return emp_table_type;
  function get_employees_tobe_incremented return emp_table_type;
  procedure increase_low_salaries;
  function arrange_for_min_salary(v_emp employees%rowtype) return employees
%rowtype;
```

```
END EMP_PKG;
----------------- package body
create or replace PACKAGE BODY EMP_PKG AS

  v_sal_inc int := 500;
  v_sal_inc2 int := 500;
  function get_sal(e_id employees.employee_id%type) return number;
  procedure print_test is
  begin
    dbms_output.put_line('Test : '|| v_sal_inc);
    dbms_output.put_line('Tests salary :'|| get_sal(102));
  end;

  procedure increase_salaries AS
  BEGIN
    for r1 in cur_emps loop
      update employees_copy set salary = salary + salary * v_salary_increase_rate
      where employee_id = r1.employee_id;
    end loop;
  END increase_salaries;
  function get_avg_sal(p_dept_id int) return number AS
  v_avg_sal number := 0;
  BEGIN
  print_test;
    select avg(salary) into v_avg_sal from employees_copy where
          department_id = p_dept_id;
    RETURN v_avg_sal;
  END get_avg_sal;

  function get_sal(e_id employees.employee_id%type) return number is
  v_sal number := 0;
  begin
    select salary into v_sal from employees where employee_id = e_id;
    return v_sal;
  end;
  /*
    This function returns all the employees in employees table
  */
  function get_employees return emp_table_type is
  v_emps emp_table_type;
  begin
    for cur_emps in (select * from employees_copy) loop
      v_emps(cur_emps.employee_id) := cur_emps;
    end loop;
    return v_emps;
  end;
  /*
    This function returns the employees which are under the minimum salary
    of the company standards and to be incremented by the new minimum salary
  */
  function get_employees_tobe_incremented return emp_table_type is
  v_emps emp_table_type;
  i employees.employee_id%type;
  begin
  v_emps := get_employees;
  i := v_emps.first;
    while i is not null loop
      if v_emps(i).salary > v_min_employee_salary then
        v_emps.delete(i);
```

```
      end if;
      i := v_emps.next(i);
    end loop;
    return v_emps;
  end;
  /*
    This procedure increases the salary of the employees who has a less salary
    then the company standard
  */
  procedure increase_low_salaries as
  v_emps emp_table_type;
  v_emp employees%rowtype;
  i employees.employee_id%type;
  begin
  v_emps := get_employees_tobe_incremented;
    i := v_emps.first;
    while i is not null loop
    v_emp := arrange_for_min_salary(v_emps(i));
      update employees_copy set row = v_emp
        where employee_id = i;
    i := v_emps.next(i);
    end loop;
  end increase_low_salaries;
  /*
    This function returns the employee by arranging the salary based on the
    company standard.
  */
  function arrange_for_min_salary(v_emp in out employees%rowtype) return employees
%rowtype is
  begin
    v_emp.salary := v_emp.salary + v_salary_increase_rate;
    if v_emp.salary < v_min_employee_salary then
      v_emp.salary := v_min_employee_salary;
    end if;
    return v_emp;
  end;
  /*********************************************/
BEGIN
  v_salary_increase_rate := 500;
  insert into logs values ('EMP_PKG','Package initialized!',sysdate);
END EMP_PKG;
```