**Answers to exercises related to HR Schema in Oracle Database 11g**

Here are the answers to exercises related to queries and PL/SQL programs

The following is the structure of the tables provided by Oracle in Human Resource Schema (HR).

  
**Note:** Columns in RED color indicate primary key(s).

**PL/SQL Programs**

1. Write a program to interchange the salaries of employee 120 and 122.

Declare

V\_salary\_120 employees.salary%type;

Begin

Select salary into v\_salary\_120

From employees where employee\_id = 120;

Update employees set salary = (select salary from employees where employee\_id = 122)

Where employee\_id = 120;

Update employees set salary = v\_salary\_120 Where employee\_id = 122;

Commit;

End;

1. Increase the salary of employee 115 based on the following conditions: If experience is more than 10 years, increase salary by 20% If experience is greater than 5 years, increase salary by 10% Otherwise 5% Case by Expression:

declare

v\_exp number(2);

v\_hike number(5,2);

begin

select floor((sysdate-hire\_date) / 365 ) into v\_exp

from employees

where employee\_id = 115;

v\_hike := 1.05;

case

when v\_exp > 10 then

v\_hike := 1.20;

when v\_exp > 5 then

v\_hike := 1.10;

end case;

update employees set salary = salary \* v\_hike

where employee\_id = 115;

end;

1. Change commission percentage as follows for employee with ID = 150. If salary is more than 10000 then commission is 0.4%, if Salary is less than 10000 but experience is more than 10 years then 0.35%, if salary is less than 3000 then commission is 0.25%. In the remaining cases commission is 0.15%.

declare

v\_salary employees.salary%type;

v\_exp number(2);

v\_cp number(5,2);

begin

select salary, ( (sysdate-hire\_date)/365) into v\_salary, v\_exp

from employees

where employee\_id = 150;

if v\_salary > 10000 then

v\_cp := 0.4;

elsif v\_exp > 10 then

v\_cp := 0.35;

elsif v\_salary < 3000 then

v\_cp := 0.25;

else

v\_cp := 0.15;

end if;

update employees set commission\_pct = v\_cp

where employee\_id = 150;

end;

1. Find out the name of the employee and name of the department for the employee who is managing for employee 103.

declare

v\_name employees.first\_name%type;

v\_deptname departments.department\_name%type;

begin

select first\_name , department\_name into v\_name, v\_deptname

from employees join departments using (department\_id)

where employee\_id = ( select manager\_id from employees where employee\_id = 103);

dbms\_output.put\_line(v\_name);

dbms\_output.put\_line(v\_deptname);

end;

1. Display missing employee IDs.

declare

v\_min number(3);

v\_max number(3);

v\_c number(1);

begin

select min(employee\_id), max(employee\_id) into v\_min, v\_max

from employees;

for i in v\_min + 1 .. v\_max - 1

loop

select count(\*) into v\_c

from employees

where employee\_id = i;

if v\_c = 0 then

dbms\_output.put\_line(i);

end if;

end loop;

end;

1. Display the year in which maximum number of employees joined along with how many joined in each month in that year.

declare

v\_year number(4);

v\_c number(2);

begin

select to\_char(hire\_date,'yyyy') into v\_year

from employees

group by to\_char(hire\_date,'yyyy')

having count(\*) =

( select max( count(\*))

from employees

group by to\_char(hire\_date,'yyyy'));

dbms\_output.put\_line('Year : ' || v\_year);

for month in 1 .. 12

loop

select count(\*) into v\_c

from employees

where to\_char(hire\_date,'mm') = month and to\_char(hire\_date,'yyyy') = v\_year;

dbms\_output.put\_line('Month : ' || to\_char(month) || ' Employees : ' || to\_char(v\_c));

end loop;

end;

1. Change salary of employee 130 to the salary of the employee with first name ‘Joe’. If Joe is not found then take average salary of all employees. If more than one employee with first name ‘Joe’ is found then take the least salary of the employees with first name Joe.

declare

v\_salary employees.salary%type;

begin

select salary into v\_salary

from employees where first\_name = 'Joe';

update employees set salary = v\_salary

where employee\_id = 130;

exception

when no\_data\_found then

update employees set salary = (select avg(salary) from employees)

where employee\_id = 130;

end;

1. Display Job Title and Name of the Employee who joined the job first day.

declare

cursor jobscur is select job\_id, job\_title from jobs;

v\_name employees.first\_name%type;

begin

for jobrec in jobscur

loop

select first\_name into v\_name

from employees

where hire\_date = ( select min(hire\_date) from employees where job\_id = jobrec.job\_id)

and job\_id = jobrec.job\_id;

dbms\_output.put\_line( jobrec.job\_title || '-' || v\_name);

end loop;

end;

1. Display 5th and 10th employees in Employees table.

declare

cursor empcur is

select employee\_id, first\_name

from employees;

begin

for emprec in empcur

loop

if empcur%rowcount > 4 then

dbms\_output.put\_line( emprec.first\_name);

exit when empcur%rowcount > 10;

end if;

end loop;

end;

1. Update salary of an employee based on department and commission percentage. If department is 40 increase salary by 10%. If department is 70 then 15%, if commission is more than .3% then 5% otherwise 10%.

declare

cursor empcur is

select employee\_id, department\_id, commission\_pct

from employees;

v\_hike number(2);

begin

for emprec in empcur

loop

if emprec.department\_id = 40 then

v\_hike := 10;

elsif emprec.department\_id = 70 then

v\_hike := 15;

elsif emprec.commission\_pct > 0.30 then

v\_hike := 5;

else

v\_hike := 10;

end if;

update employees set salary = salary + salary \* v\_hike/100

where employee\_id = emprec.employee\_id;

end loop;

end;

1. Create a function that takes department ID and returns the name of the manager of the department.

create or replace function get\_dept\_manager\_name(deptid number)

return varchar is

v\_name employees.first\_name%type;

begin

select first\_name into v\_name

from employees

where employee\_id = ( select manager\_id from departments where department\_id = deptid);

return v\_name;

end;

1. Create a function that takes employee ID and return the number of jobs done by the employee in the past.

create or replace function get\_no\_of\_jobs\_done(empid number)

return number is

v\_count number(2);

begin

select count(\*) into v\_count

from job\_history

where employee\_id = empid;

return v\_count;

end;

1. Create a procedure that takes department ID and changes the manager ID for the department to the employee in the department with highest salary. (Use Exceptions).

create or replace procedure change\_dept\_manager(deptid number)

is

v\_empid employees.employee\_id%type;

begin

select employee\_id into v\_empid

from employees

where salary = ( select max(salary) from employees where department\_id = deptid)

and department\_id = deptid;

update departments set manager\_id = v\_empid

where department\_id = deptid;

end;

1. Create a function that takes a manager ID and return the names of employees who report to this manager. The names must be returned as a string with comma separating names.

create or replace function get\_employees\_for\_manager(manager number)

return varchar2

is

v\_employees varchar2(1000) := '';

cursor empcur is

select first\_name from employees

where manager\_id = manager;

begin

for emprec in empcur

loop

v\_employees := v\_employees || ',' || emprec.first\_name;

end loop;

-- remove extra , at the beginning

return ltrim(v\_employees,',');

end;

1. Ensure no changes can be made to EMPLOYEES table before 6am and after 10pm in a day.

create or replace trigger trg\_employees\_time\_check

before update or insert or delete

on employees

for each row

begin

if to\_char(sysdate,'hh24') < 6 or to\_char(sysdate,'hh24') > 10 then

raise\_application\_error(-20111,'Sorry! No change can be made before 6 AM and after 10 PM');

end if;

end;

1. Create a Trigger to ensure the salary of the employee is not decreased.

create or replace trigger trg\_employees\_salary\_check

before update

on employees

for each row

begin

if :old.salary > :new.salary then

raise\_application\_error(-20111,'Sorry! Salary can not be decreased!');

end if;

end;

1. Create a trigger to ensure the employee and manager belongs to the same department.

**Note:**  This trigger need to read the row that is being modified, which causes mutating problem. The solution to mutating problem is

explained at : [Work around for mutating problem in Oracle Triggers](http://www.srikanthtechnologies.com/blog/oracle/mutatingsolution.aspx). Please check it out.

1. Whenever the job is changed for an employee write the following details into job history. Employee ID, old job ID, old department ID, hire date of the employee for start date, system date for end date. But if a row is already present for employee job history then the start date should be the end date of that row +1.

create or replace trigger trg\_log\_job\_change

after update of job\_id

on employees

for each row

declare

v\_enddate date;

v\_startdate date;

begin

-- find out whether the employee has any row in job\_history table

select max(end\_date) into v\_enddate

from job\_history

where employee\_id = :old.employee\_id;

if v\_enddate is null then

v\_startdate := :old.hire\_date;

else

v\_startdate := v\_enddate + 1;

end if;

insert into job\_history values (:old.employee\_id, v\_startdate, sysdate, :old.job\_id, :old.department\_id);

end;

**Note:** Before testing the above trigger, you need to disable UPDATE\_JOB\_HISTORY trigger, which is already present in HR account, as it does the same.