PL/SQL ASSIGNMENTS For Cybage Freshers Batch 2016

Anonymous Block

1. Write a PL/SQL block that selects the maximum department number in the department table and store it in a SQL\*PLUS variable. And print the results to screen.

set serveroutput on;

CREATE TABLE department

(

deptId number(10) not null,

deptName varchar2(50) not null,

location varchar(50) ,

CONSTRAINT pk\_dept\_key PRIMARY KEY (deptId)

);

insert into department values( 1, 'Security' ,'pune');

insert into department values( 2, 'HR' ,'pune');

insert into department values( 3, 'Training','gandhinagar' );

insert into department values( 4, 'Communication',null );

DECLARE

cnt integer;

BEGIN

select max(deptId) into cnt from department;

dbms\_output.put\_line('Max department no.: ' || cnt);

END;

/

2. Create a PL/SQL block to insert a new department number into the Departments table. Use maximum dept number fetched from above and adds 10 to it. Use SQL\*PLUS substitution variable for department name. Leave the location AS null.

DECLARE

cnt integer;

BEGIN

select max(deptId) into cnt from department;

cnt:=cnt+10;

insert into department values(cnt, '&&deptName' ,null);

--select \* from department;

END;

/

3. Create a PL/SQL block to update the location for an existing department. Use substitution variable for dept no. and dept location.

/\*

BEGIN

update department

set location='&&location' where deptId='&&deptId';

END;

/

select \* from department;

\*/

4. Create a PL/SQL Block to delete the department created in exercise 2. Print to the screen the number of rows affected.

/\*

BEGIN

delete from department where deptName='&&deptName';

dbms\_output.put\_line(TO\_Char(SQL%ROWCOUNT) || ' rows affected.');

END;

/

select \* from department;

\*/

5. 1. Write a PL/SQL block which accepts employee name, basic and should display Employee name, PF and net salary.

HRA=31% of basic salary

DA=15% of basic salary

Net salary=basic+HRA+DA-PF

If the basic is less than 3000 PF is 5% of basic salary.

If the basic is between 3000 and 5000 PF is 7% of basic salary.

If the basic is between 5000 and 8000 PF is 8% of basic salary.

DECLARE

empName varchar(20);

sal integer(7);

HRA integer(7,2);

DA integer(7,2);

PF integer(7,2);

net\_sal integer(7,2);

BEGIN

dbms\_output.put\_line('Eneter employee name:'||'&&empName');

dbms\_output.put\_line('Eneter employee salary:' || '&&sal');

HRA:=((31\*&&sal)/100);

DA:=((15\*&&sal)/100);

net\_sal:=&&sal+HRA+DA;

if(&&sal<3000) then

PF:=((5\*&&sal)/100);

elsif (&&sal between 3000 and 5000) then

PF:=((5\*&&sal)/100);

elsif (&&sal between 5000 and 8000) then

PF:=((5\*&&sal)/100);

else

PF:=null;

end if;

dbms\_output.put\_line( 'net salary:' || net\_sal || ' and PF:' || PF);

END;

/

6. Write a PL/SQL block to find the salary grade of the specified employee.

If grade is 1 display ‘the employee is junior engineer’

If grade is 2 display ‘the employee is engineer’

If grade is 3 display ‘the employee is lead engineer’

If grade is 4 display ‘the employee is Manager’

If grade is 5 display ‘the employee is Project manager’

(Use case expression)

create table employee

(

empId integer not null,

empName char(20) not null,

grade integer,

CONSTRAINT pk\_emp\_key PRIMARY KEY (empId)

);

insert into employee values ('1','rohit','1');

insert into employee values ('2','ganesh','4');

insert into employee values ('3','purva','2');

DECLARE

grade1 integer;

BEGIN

dbms\_output.put\_line('Enter empployee name:' || '&&empName');

select grade into grade1 from employee where empName='&&empName';

case grade1

when '1' then dbms\_output.put\_line('the employee is junior engineer');

when '2' then dbms\_output.put\_line('the employee is engineer');

when '3' then dbms\_output.put\_line('the employee is lead engineer');

when '4' then dbms\_output.put\_line('the employee is Manager');

when '5' then dbms\_output.put\_line('the employee is Project manager');

else dbms\_output.put\_line('No such grade');

end case;

END;

/

7. Wrtie a PL/SQL block to award an employee with the bonus.

Bonus is 15% of commission drawn by the employee. If the employee does not earn any commission then display a message that ‘employee does not earn any commission’. Otherwise add bonus to the salary of the employee. The block should accept an input for the employee number.

8. Write a PL/SQL block which displays the department name, total no of employees in the department, avg salary of the employees in the department for all the departments from department 10 to department 40 in the Dept table.If no employees are working in the department ,then display a message that no emplyees are working in that department.

9 .Write a PL/SQL block which accepts employee number and finds the average salary of the employees working in the department where that employee works.

If his salary is more than the average salary of his department, then display message that ‘employee’s salary is more than average salary’ else display ‘employee’s salary is less than average salary’

**Procedures and functions**

1. Create a procedure that deletes rows from the emp table. It should accept 1 parameter, job; only delete the employee’s with that job. Display how many employees were deleted. Write a PL/SQL block to invoke the procedure.

create table employee

(

empId integer not null,

empName char(20) not null,

job char(20),

CONSTRAINT pk\_emp\_key PRIMARY KEY (empId)

);

insert into employee values (1,'ramesh','BI');

insert into employee values (2,'reshma','QA');

insert into employee values (3,'hari','BI');

insert into employee values (4,'puja','BI');

create or replace procedure del\_job

as

begin

delete from employee where job='&&job';

dbms\_output.put\_line(TO\_Char(SQL%ROWCOUNT) || ' rows affected.');

end;

/

execute del\_job;

2. Change the above procedure so that it returns the number of employees removed via an OUT parameter. Write a PL/SQL block to invoke the procedure and display how many employees

Were deleted.

insert into employee values (1,'ramesh','BI');

--insert into employee values (2,'reshma','QA');

insert into employee values (3,'hari','BI');

insert into employee values (4,'puja','BI');

declare

a number;

procedure del1\_job(x out number) is

begin

delete from employee where job='&&job';

x := SQL%ROWCOUNT;

dbms\_output.put\_line(TO\_Char(SQL%ROWCOUNT) || ' rows affected...');

end;

begin

del1\_job(a);

dbms\_output.put\_line(' rows affected...'|| a);

end;

/

3. Convert the above program to a function. Instead of using an OUT parameter for the number of employees deleted, use the functions return value. Write a program to invoke the function and display how many employees were deleted.

declare

a number;

function del\_job

return number

is

z number;

begin

delete from employee where job='&&job';

z := SQL%ROWCOUNT;

dbms\_output.put\_line(TO\_Char(SQL%ROWCOUNT) || ' rows affected...');

return z;

end;

begin

a:=del\_job();

dbms\_output.put\_line(' rows affected...'|| a);

end;

/

4. Create a table having the following structure

Accounts(Account\_id, branch\_name, amount\_balance)

a. Write a PL/SQL procedure to perform withdraw operation that only permits a withdrawal if there sufficient funds in the account. The procedure should take Account\_id and withdrawal amount as input.

create table Accounts

(

Account\_id integer not null,

branch\_name char(20),

amount\_balance integer

);

insert into Accounts values(1,'pune',2000);

insert into Accounts values(2,'pune',4000);

declare

a number;

b number;

c number;

procedure withdraw(x in number,y in number) is

begin

select amount\_balance into b from Accounts where Account\_id=x;

if(y < b) then

dbms\_output.put\_line('Amount is withdrew...');

update Accounts

set amount\_balance=amount\_balance-y where Account\_id=x;

else

dbms\_output.put\_line('Low balance...');

end if;

end;

begin

a:=1;

c:=1000;

withdraw(a,c);

end;

/

b. Write a procedure to deposit money into someone's account. The procedure should accept account\_id and deposit amount.

declare

a number;

b number;

c number;

m number;

procedure deposit(x in number,y in number) is

begin

update Accounts

set amount\_balance=amount\_balance+y where Account\_id=x;

dbms\_output.put\_line('Amount is deposited...');

end;

begin

a:=1;

c:=1000;

deposit(a,c);

select amount\_balance into m from Accounts where Account\_id=a;

dbms\_output.put\_line('Amount deposited is:'||m);

end;

/

c. Write a procedure to transfer money from one person's account to another. The procedure should table two account\_id’s one for giver and one for receiver and the amount to be transferred.

declare

a1 number;

a2 number;

b number;

c number;

d number;

m number;

procedure transfer(x in number,y in number,z number) is

begin

select amount\_balance into d from Accounts where Account\_id=x;

if(d>z) then

update Accounts

set amount\_balance=amount\_balance-z where Account\_id=x;

update Accounts

set amount\_balance=amount\_balance+z where Account\_id=y;

dbms\_output.put\_line('Amount is deposited...');

else

dbms\_output.put\_line('Not enough balance...');

end if;

end;

begin

a1:=2;

a2:=1;

c:=1000;

transfer(a1,a2,c);

end;

/

**Cursors and Data Types as in 3GL**

1. Write a PL/SQL block to accept an employee number. and use a record variable to store the record of that employee. and insert it into retired\_employee table.

Retired\_employee table has the following structure

Retired\_employee (empno, ename, hiredate, leaveDate, salary, mgr\_id, deptno).

Set the leavedate to the current date.

2. Write a PL/SQL Block to create a PL/SQL table which can store grade and on of employees with that grade. Get the information about the grade and number of employees with that grade and store it in the PL/SQL table. Then retrieve the information from the PL/SQL table and display it on the screen in the following way.

No of employees with the grade 1 are 3

No of employees with the grade 2 are 2

No of employees with the grade 3 are 1

No of employees with the grade 4 are 2

No of employees with the grade 5 are 5

**Cursors**

1. Write a program that gives all employees in department 10 a

15% pay increase. Display a message displaying how many

Employees were awarded the increase.

create table Emp1

(

eId int not null,

sal int,

dept char(20)

);

insert into Emp1 values(1,2000,'10a');

insert into Emp1 values(2,2000,'20a');

insert into Emp1 values(3,4000,'10a');

declare

sal1 Emp1.sal%type;

dep Emp1.dept%type;

c integer :=0;

cursor emp is

select sal,dept from Emp1 where dept='10a';

begin

open emp;

loop

fetch emp into sal1,dep;

if dep='10a' then

sal1:=(sal1\*15)/100;

update Emp1

set sal=sal1+sal

where dept='10a';

dbms\_output.put\_line(sal1 || 'and' || dep);

c:=c+1;

end if;

exit when emp%notfound;

end loop;

dbms\_output.put\_line('No. of employees awarded:'|| c);

close emp;

end;

/

2. Write a PL/SQL block and use cursor to retrieve the details of the employees with grade 5.and then display employee no,job\_id ,max\_sal and min\_sal and grade for all these employees.

create table Emp1

(

eId int not null,

jobId int,

min\_sal int,

max\_sal int,

grade int

);

insert into Emp1 values(1,1,2000,4000,5);

insert into Emp1 values(2,2,2000,4000,1);

insert into Emp1 values(3,3,4000,6000,5);

declare

eId1 Emp1.eId%type;

jobId1 Emp1.jobId%type;

min\_sal1 Emp1.min\_sal%type;

max\_sal1 Emp1.max\_sal%type;

grade1 Emp1.grade%type;

cursor emp is

select eId,jobId,min\_sal,max\_sal,grade from Emp1 where grade='5';

begin

open emp;

loop

fetch emp into eId1,jobId1,min\_sal1,max\_sal1,grade1;

dbms\_output.put\_line(eId1||' '||jobId1||' '||min\_sal1||' '||max\_sal1||' '||grade1);

exit when emp%notfound;

end loop;

close emp;

end;

/

3. Write a PL/SQL block that copies all departments to a table called old\_dept. Do not use a cursor FOR loop. Display how many rows were copied.

create table dept

(

dId int not null,

deptName char(20)

);

create table old\_dept

(

dId int not null,

deptName char(20)

);

insert into dept values(1,'BI');

insert into dept values(2,'java');

insert into dept values(3,'c');

declare

dId1 dept.dId%type;

deptName1 dept.deptName%type;

c integer:=0;

cursor dep is

select dId,deptName1 from dept;

begin

open dep;

loop

fetch dep into dId1,deptName1;

insert into old\_dept values(dId1,deptName1);

dbms\_output.put\_line(dId1||' '||deptName1);

c:=c+1;

exit when dep%notfound;

end loop;

dbms\_output.put\_line('No.of rows copied:'||c);

close dep;

end;

/

4. Display the names of employees who are working for Department 30.

create table Emp1

(

eId int not null,

name char(20),

dept int

);

insert into Emp1 values(1,'raj','30');

insert into Emp1 values(2,'rama','10');

insert into Emp1 values(3,'raju','30');

declare

name1 Emp1.name%type;

cursor emp is

select name from Emp1 where dept='30';

begin

open emp;

loop

fetch emp into name1;

dbms\_output.put\_line(name1);

exit when emp%notfound;

end loop;

close emp;

end;

/

5. Write a PL/SQL Block that mimics selecting all columns and rows from the dept table. There is no need to format the output, just select all columns and all rows. Use a cursor FOR loop.

6. Write a PL/SQL block to display the top 6 employees with respect to salaries using cursors.

create table Emp1

(

eId int not null,

sal int,

dept char(20)

);

insert into Emp1 values(1,3000,'10a');

insert into Emp1 values(2,2000,'20a');

insert into Emp1 values(3,4000,'10a');

declare

sal1 Emp1.sal%type;

eId1 Emp1.eId%type;

c integer :=0;

cursor emp is

select eId,sal from Emp1 order by sal desc;

begin

open emp;

for i in 1..6 loop

fetch emp into eId1,sal1;

dbms\_output.put\_line(eId1 || 'and' || sal1);

exit when emp%notfound;

end loop;

close emp;

end;

/

7. Use a cursor to retrieve the department number and the department name from the dept table. Pass the department number to another cursor to retrieve from the emp table the details of employee name, job, hiredate and salary of all the employees who work in that department.

create table Emp1

(

eId int not null,

eName char(20),

job char(20),

hireDate varchar(20),

sal int,

deptNo char(20)

);

insert into Emp1 values(1,'ram','coder','2/2/2000',3000,10);

insert into Emp1 values(2,'rama','coder','3/2/2000',3000,10);

insert into Emp1 values(3,'ramesh','tester','4/2/2000',3000,20);

create table dept

(

dId int ,

dName char(20)

);

insert into dept values(10,'HR');

insert into dept values(20,'training');

declare

dId1 dept.dId%type;

dName1 dept.dName%type;

eId1 Emp1.eId%type;

eName1 Emp1.eName%type;

job1 Emp1.job%type;

hireDate1 Emp1.hireDate%type;

sal1 Emp1.sal%type;

cursor dep is

select dId,dName from dept;

cursor emp is

select eName,job,hireDate,sal from Emp1 where deptNo=dId1;

begin

open dep;

loop

fetch dep into dId1,dName1;

dbms\_output.put\_line(dId1||' '||dName1);

open emp;

loop

fetch emp into eName1,job1,hireDate1,sal1;

--dbms\_output.put\_line(eId1||' '||dName1);

dbms\_output.put\_line(eName1||' '||job1|| ' '||hireDate1||' '||sal1||' '||dId1);

exit when emp%notfound;

end loop;

close emp;

exit when dep%notfound;

end loop;

close dep;

end;

/

8.Write a procedure Raise\_salary which gives a specified hike to all the employees working in a specified department.The procedure should take department number and percemtage of hike as input.(Use for update and where current of)

**Exception Handling**

1. Write a PL/SQL block to select the name of the employee with a given salary value.

If the salary entered returns more than one row,Handle the exception with an appropriate exception handler and insert into SALARY\_MESSAGES table the message “more than one employee with a salary of <salary>”

If the salary entered does not return any rows ,handle the exception

With an appropriate exception handler and and insert into the SALARY\_MESSAGES table the message “no employee with a salary of <salary>”

If the salary entered returns only one row,insert into the SALARY\_MESSAGES table ,the emloyee’s name and the salary amount.

Handle any other exception with an appropriate handler and insert into the SALARY\_MESSAGES table the message “some other error occurred”.Test the block for a variety of test cases.

2. Write a PL/SQL block to remove a specified department from the department table. If there are employees in that department, print a message to the user that the department cannot be removed.

(Use pragma exception init)

3. Write a PL/SQL program to update the salary and Department number in the employees

table using SQL\*Plus bind variables (or initialized local variables if you prefer). Include the following

exception handling capabilities:

- Define, raise and handle an exception for salary values not in the range of 800 to 5000.

- Define and associate (EXCEPTION\_INIT) and exception with the referential integrity constraint

violated system exception (-2291) and use it to handle attempts to set the deptno to a

value not found in the Department table.

- Provide simple error handling for any other exceptions

**Packages**

Read the following specification:-

Develop a package that will act as an API (Application Programming Interface) for the items table. We need to protect our data and want to ensure no one writes any code that will directly Access this table.

Here is the structure of the items table: -

Column Data Type Description

item\_id NUMBER Internal ID for item

item\_number VARCHAR2(10) User item number

description VARCHAR2(30) Item description

status VARCHAR2(1) [T]est or [L]ive

Cost NUMBER Standard cost of item

We need the item\_id column to be a sequential number (use

items\_item\_id\_s sequence)

The following business rules must be applied:-

• An item is created as a test item and with a zero cost.

• A procedure or function must be called to promote the item from test to live. An item cannot be made live with a zero cost.

• Only test items can be removed

We need an API to provide the developer the following facilities:-

• Create new items

• Promote items from test to live

• Remove items

• Change item cost

All API functions and procedures should work with the item\_id.

Create a package to implement the above. Remember, try and work out the requirements for the package first. Determine your public and private procedures/functions and any data that might

Be needed.

**Triggers**

1. To compliment the package developed in the last section, the user has come up with the following addition to the specification. Implement the above specification using triggers. When items are removed using the new API you provided, we need to ensure the item is archived in a table called items\_archive.

We also want any changes in item cost to be audited, record the details of each change in the auditing\_cost table.

2. An HR system has an emp table that holds a row for each employee within the company. Each record in the table has a manager field, (mgr) that holds the id for the employee's manager. Write a trigger so that when a manager record is deleted, the mgr field of that manager's employees is set to NULL. In other words, implement the following SQL statement:

WHEN AN EMPLOYEE IS DELETED,

UPDATE employee SET

mgr = null

WHERE

mgr = employee id of the deleted employee

3. Create one additional table job\_salary having columns Job\_salary(job,min\_sal,max\_sal)

And insert following records into the table.

('CLERK', 800, 1300);

('ANALYST', 3000, 3500);

('SALESMAN', 1250, 1600);

('MANAGER', 2450, 2975);

('PRESIDENT', 5000, 5500);

Write a trigger on emp table so that when a new employee record is inserted or updated and If the new salary has been decreased or does not lie within the salary range for that job or if more than 10% hike is given, then it should raise exceptions for these three cases mentioned above. And trigger should be executed for employees other than the president.

4. Write a trigger on the table job\_salary so that when a record is deleted or updated from this table, the trigger should fire and has to check whether employees having the deleted job exist. If yes, it should raise an exception. and if it is updation, and if there are employees in the emp table whose salary does not lie within the modified range, then restore old salary ranges.