**ASSIGNMENTS**

**1. Programming Basics Assignments**

1) Write a plsql block to accept the empno from the user. Raise the salary of that employee by 20% and display the raised salary. (Note - Do not update the salary, just display the changed salary)

declare

vsal emp.sal%type;

begin

select sal into vsal

from emp

where empno=:EnterEmpno;

dbms\_output.put\_line('Salary $'||vsal||' is incremented to $'||(vsal\*1.20));

end;

2) Accept job type from the user. Display the message depending upon whether

no rows or one row or several rows are selected.

The message should be any one from the below 3 as per the situation.

JOB TYPE FOUND ONCE

JOB TYPE FOUND MORE THAN ONCE

JOB TYPE NOT FOUND

declare

vcount number;

begin

select count(\*) into vcount

from emp

where job=:EnterJobType;

if vcount>1 then

dbms\_output.put\_line('Job type found more than once');

elsif vcount=1 then

dbms\_output.put\_line('Job type found once');

else

dbms\_output.put\_line('Job type not found');

end if;

end;

3) Using basic loop technique display all the multiples of 7 between 31 and 48.

declare

i number:=0;

begin

for i in 31..48

loop

if mod(i,7)=0 then

dbms\_output.put\_line(i);

end if;

end loop;

end;

4) Write a block to accept the Empno from the user and change the salary according to the following condition. If salary is in the range of

RANGE INCREMNENT

1000 – 2000 500

2001 –3000 1000

3001 – 4000 1500

>4000 2000

(Note - Do not update the salary, just display the changed salary)

declare

vsal emp.sal%type;

begin

select sal into vsal

from emp

where empno=:EnterEmpno;

if vsal>999 and vsal<2001 then

dbms\_output.put\_line('Salary $'||vsal||' is increased to $'||(vsal+500));

elsif vsal>2000 and vsal<3001 then

dbms\_output.put\_line('Salary $'||vsal||' is increased to $'||(vsal+1000));

elsif vsal>3000 and vsal<4001 then

dbms\_output.put\_line('Salary $'||vsal||' is increased to $'||(vsal+1500));

elsif vsal>4000 then

dbms\_output.put\_line('Salary $'||vsal||' is increased to $'||(vsal+2000));

end if;

end;

5) Create a table Inspection\_Details\_*EmployeeID* that has one column Readings of numeric type.

Using pl/sql block add numbers, which has the difference of 0.1. The numbers should be between 0.1 and 6.8.

create table inspection\_details\_employeeid (val number)

AND

declare

i inspection\_details\_employeeid.val%type:=0.0;

begin

while(i<6.9)

loop

i:=i+0.1;

dbms\_output.put\_line(i);

insert into inspection\_details\_employeeid

values(i);

end loop;

end;

6) Through while loop display the multiples of 7 till 70 in the descending order.

declare

i number:=71;

begin

while(i>1)

loop

i:=i-1;

if mod(i,7)=0 then

dbms\_output.put\_line(i);

end if;

end loop;

end;

7) Display the difference of salary between the oldest and the latest employee.

declare

vsalmax emp.sal%type;

vsalmin emp.sal%type;

begin

select sal into vsalmax

from emp

where hiredate=(select min(hiredate) from emp);

select sal into vsalmin

from emp

where hiredate=(select max(hiredate) from emp);

dbms\_output.put\_line(vsalmax-vsalmin);

end;

8) Create the table tx1 with the following script.

Create table Oracle\_Batch(student\_name varchar (20);

Create a program that will accept the student\_name form the user and if the user has entered all the characters as alphabets only then enter that name into the Oracle\_Batch table.

9) Write a PL/SQL code to accept an employee number from the user and display whether it is exists or not .

10) Write a PL/SQL code to display the name, salary and grade of the employee by accepting employee code. Grade is ‘A’ if salary >25000, ‘B’ if salary > 15000, ‘C’ for other salaries. Use CASE statement only.

**2. Cursor Assignments**

**Using the cursor mechanism write the following PLSQL blocks.**

1) Check whether the 3rd Clerk’s salary is greater than the 2nd Clerk’s salary. If it is so, then display message as “Third has more salary than the second” otherwise display message as “Second has more salary than the first”

2) Display the sum of first five salaries from the emp table.

3) The management has decided to increase salary of employees by 10%. Starting with lowest paid earner and working up. If at any time salary bill exceeds 45000/- then no further employees are to be given an increase.

4) Display the names of the employees who are earning salary less than the average salary of their own jobs.

5) Use a parameterized cursor which will take deptno as the parameter and will display the highest two salaries of that deptno.

7) Create the table Emp\_Coupons with the two fields Name and Coupon\_No

Enter the following records in it.

|  |  |
| --- | --- |
| John | 80 |
| Martin | 83 |
| Allen | 87 |
| Roger | 78 |
| Adams | 88 |
| Kim | 89 |

Make a PLSQL block that will check whether the coupon number of the current record is greater than the previous. If any record’s coupon number is less than the previous one then display the name of the person whose coupon number is less.

8) Display the name, sal and the cumulative total of salaries.

Cumulative salary would be first salary +0 for the first record, second + first for the second and so on.

9) Comparing the total of salaries of the even number records and the odd number records from the emp table and display the highest total.

10) Check the total of salaries of the first 3 employees. If that total is greater than or equal to 3500 then increment those first 3 employees salaries by 15% else do not increment the salary.

**3. PLSQL Table Assignments**

1. Add multiples of 5 from 5 to 130 in a plsql table and display them from the plsql table in the descending order.
2. Using a cursor add the rows of all SALESMAN in the plsql table and then display only the last row of that plsql table
3. Take all the salaries from the emp table in a plsql block using a cursor. Then display only those salaries from the plsql block which have value >= 2800.

**4. Exception Handling Assignments**

1. Accept the name of the employee. Display the salary of that employee.

Handle all possible run-time errors.

1. Create the ex\_tab table with the following specifications:

|  |  |  |
| --- | --- | --- |
| **Name** | **Data type** | **Constraint** |
| A | Integer | Primary key |
| B | Integer | Unique |
| C | Char(3) | Not Null |
| D | Integer | Must be >= 10 |
| E | Integer | Foreign key from A column |

For trapping all the constraint violation errors create another table error\_log that has 3 columns Record\_Number, Error\_Number and Error\_Message.

Through a plsql block accept all the 5 column values from the user run-time, if any error violation takes place then the error\_log table should get the record displaying the details of that error.

If there is no error then the record should get added in the ex\_tab table.

1. Create a table Emp\_Details which has the following specifications.

|  |  |  |
| --- | --- | --- |
| **Name** | **Data type** | **Constraint** |
| Empno | Integer | Primary Key |
| Membership\_no | Char | Unique |
| Name | Varchar | Not Null |
| Salary | Integer | Above 10000 |
| Mgr | Integer | Refers to the Empno column |

Take all the column values from the user and enter the record in the table Emp\_Details. If any constraint violation happens then provide user friendly error message according to the error.

1. Run the following script.

create table mm1

(empid number, ename varchar2(10), job varchar2(10));

insert into mm1 values(1,'John','Manager');

insert into mm1 values(2,'Martin','Clerk');

insert into mm1 values(3,'Smith','Clerk');

Create a plsql block which will accept empid, ename and job from the user. If the complete record is duplicated then raise an exception FOUND\_DUPLICATE\_RECORD. If the record in not duplicate then add it into the mm1 table.

1. Create a table Customer\_Data which has 5 fields like Custid, Qty, Required\_In\_Days, Qty\_Per\_Day and Rate\_Per\_Day.

Take the first 3 columns values from the user.

You want to calculate Quantity per day which would be qty given divided by Required\_In\_Days.

Once Quantity per day is calculated do the calculation of Rate\_Per\_Day which will be Quantity per day \* 100.

Finally when all the values are ready add a new record in the table Customer\_Data.

Handle all types of error that may occur at different point of times.

**5. Procedures Assignments**

1. Create a procedure named Job\_Details, which will show the total, highest, lowest and average of the salary for the job. (Job type is the parameter to be passed.)

create or replace procedure job\_details(vjob emp.job%type)

is

a number;

b number;

c number;

d number;

begin

select sum(sal),max(sal) ,min(sal) ,avg(sal) into a,b,c,d

from emp

where job=vjob;

dbms\_output.put\_line('Total:'||a||' Max:'||b||' Min:'||c||' Avg:'||d);

end job\_details;

1. Create a procedure named Change\_Salary that will accept salary and will return the 40% raised salary. The value returned should be given to a separate parameter.

create or replace procedure change\_salary(sal in number,salout out number)

is

begin

salout:=(sal+sal\*0.4);

end change\_salary;

1. Create a table Emp\_Proc having structure and records same as that of emp table.

Create a procedure Delete\_Employee that will take department name as a parameter. The procedure should delete the record of the employees from the Emp\_EmployeeID table only for the department name passed.

4. Create a procedure, NEW\_EMP to insert a new employee into the Emp\_Proc table . The procedure should ensure whether the department ID specified for the new employee exists in the DEPARTMENTS table.

1. Create a table Company\_Data with columns empid, ename and job. There is no primary key or unique key constraint to the empid column.

Create a procedure Add\_Company\_Data which will take empid as the parameter and adds a new record in that table if the empid with the same value does not exist.

1. Create a table Emp\_Sal that will have 2 columns – Empno and Gross\_Salary

Create a procedure Salary\_Details which will take Empno,Basic salary and Allowance as the three parameters from the user.

There has to be a sub procedure Tax\_Calculations that will calculate the tax value as per the salary values given.

There is a small chart for tax calculations.

|  |  |
| --- | --- |
| **Basic** | **Tax** |
| Above or equal to 5000 | 20% of the basic |
| Between 4000 and 4999 | 15% of the basic |
| Between 3000 and 3999 | 10% of basic and 2% of Allowance |
| Between 2000 and 2999 | 5% of basic and 1% of Allowance |
| Less than 2000 | 100 |

Once the tax value is calculated the Gross calculation will be done as ( Basic +Allowance ) – Tax.

After calculating the tax main procedure will add the record in the Emp\_sal table.

7. Considering the emp table create a procedure IsHighest that will take the ename as the parameter. The procedure should reply whether that employee is the highest earner or not.

**6. FUNCTIONS Assignments**

1. Create a table student. It has four fields rollno, name, marks and grade.

Note – Grade column has be generated by a function get\_Grade(). This function takes marks as argument and generates the grade as per the following table

|  |  |
| --- | --- |
| **Marks Criteria** | **Grade** |
| Greater than or equal to 70 | Distinction |
| Between 60 and 69 | First Class |
| Between 50 and 59 | Second Class |
| Less than 50 | Failed. |

When the record is inserted then the grade value should be generated.

1. Create a function Raise\_Sal that will accept salary as a parameter and returns 15% raised salary. Use this function in the select statement of the emp table to see the result.
2. Create a function Yearly\_Raise that will take the salary, deptno and job as the parameters and raise the salary according to different criteria.

|  |  |
| --- | --- |
| **Criteria** | **Raise** |
| Clerk employees of deptno 20 earning salary above 1000 | 20% |
| Clerk employees of deptno 20 earning salary less 1000 | 15% |
| Clerk employees of deptno 20 earning salary above 1000 | 25% |
| Clerk employees of deptno 20 earning salary less than 1000 | 18% |
| Clerk employees of deptno 30 having any salary | 10% |

Use this function to update salaries of the employees of job Clerk in the table emp.

4. Run the following script.

Create Table Company\_Product (ProductId Varchar(20));

Create a function check\_productid that will take a productid (string) as a parameter. The function should check the last 4 characters of that productid. If they are KPIT (only upper case) then a new record must get created in the table Company\_Product else throw a relevant error message.

5. Considering the emp table create a function Last\_Employee which will take the job type as the parameter and display the last employee (s) joined in that job.

6. Considering the emp table create a function TopN\_Salary that will take the top Nth number as the parameter and returns the highest salary at that position.

**7. Packages Assignments**

1. Create a package Clerk\_Salary. This package should have the following members.
   1. Basic has a fixed value of 10000. Only package members should be able to use the basic figure.
   2. Function get\_HRA(). It should be 40% of basic.
   3. Function get\_DA(). It should be 35% of basic.
   4. Function get\_TA(). It should be 10% of HRA.
   5. Function get\_PF(). It should be 15% of (basic+hra+da+ta)
   6. Function get\_TAX(). It should be a fixed value 7500.
   7. Function get\_Gross(). It should be basic+hra+da+ta – (pf + tax).
   8. Procedure display\_Gross. It should show the gross figure.

Execute the display\_Gross procedure and display the gross amount.

create or replace package clerk\_salary

is

basic number:=10000;

function get\_HRA return number;

function get\_TA return number;

function get\_DA return number;

function get\_PF return number;

end;

AND

create or replace package body clerk\_salary

is

function get\_HRA return number

is

begin

return basic\*1.40;

end get\_HRA;

function get\_DA return number

is

begin

return basic\*1.35;

end get\_DA;

function get\_TA return number

is

begin

return (basic\*0.10);

end get\_TA;

function get\_pf return number

is

a number;

begin

a:=(basic+clerk\_salary.get\_HRA+clerk\_salary.get\_TA+clerk\_salary.get\_DA)\*0.15;

return (a);

end get\_pf;

end clerk\_salary;

AND

declare

vhra number;

begin

vhra:=clerk\_salary.get\_HRA();

dbms\_output.put\_line(vhra);

dbms\_output.put\_line(clerk\_salary.get\_pf);

end;

1. Create a package OverLoaded\_Total that has function Show\_total. This function takes salary and/or commission as parameters and returns the total as:
   * 1. If only salary was passed then the total should be 30% raised salary.
     2. If salary and commission were passed then the total should be salary+commission.

create or replace package overload\_pack

is

procedure p;

procedure p(x number);

procedure p(x varchar2,y varchar2);

end;

AND

create or replace package body overload\_pack

is

procedure p

is

begin

dbms\_output.put\_line('in procedure p');

end;

procedure p(x number)

is

begin

dbms\_output.put\_line(x);

end;

procedure p(x varchar2,y varchar2)

is

begin

dbms\_output.put\_line(x||y);

end;

end overload\_pack;

AND

begin

overload\_pack.p;

overload\_pack.p(5);

overload\_pack.p('Commission','Salary');

end;

1. Create a package Emp\_Package that will have a plsql table called Emp\_Info containing Enames and Sal from the emp table.

The package has two procedure Show\_Clerks and Show\_Managers which will display the enames and salaries of the Clerks and Managers respectively. These procedures should take the names and salaries from the Emp\_Info plsql table only.

The Emp\_Info always should get loaded within the session first when any of the package member is getting called for the first time within that session.

**8. Triggers Assignments**

The following triggers have to be created on the emp and dept tables –

1. Create a trigger No\_DML\_in\_Lunch, which will not allow records to be entered between 1 and 2 pm on weekdays.
2. Create a trigger DML\_on\_View which will allow insert, update and delete for a view which is based on an inner join between tables Emp and Dept
3. Create a trigger ClerkAndAnalystOnly, which will allow records getting inserted in the table emp if the job of the employee is CLERK and ANALYST only.

create or replace trigger ClerkAndAnalystOnly

after insert on emp

begin

if :new.job='CLERK' or :new.job='ANALYST' then

dbms\_output.put\_line('Ok');

else

delete from emp where empno=(:new.empno);

raise\_application\_error(-20001,'Sorry');

end if;

end;

4. Create a table Emp\_Summary that has 4 columns Record number, Lowest\_Sal Highest\_Sal and Average\_Sal. (Record\_no is the auto generated number). Create a trigger Emp\_Stats that will add a new record into the table Emp\_Summary, which will get the auto record number, lowest, highest and the average salaries of the emp table, whenever a new record is added in the emp table or whenever any record is updated.

1. Create a table New\_Top\_employees that will have empno,ename,sal as the columns. Create a trigger Check\_New\_Top that will get fired whenever a new employee record is added in the emp table. If the new record’s salary is greater than or equal to 2500 then only that record’s empno, name and salary should be transferred into the New\_Top\_employees table.

create or replace trigger new\_top\_emp\_tri

after insert or update on emp

for each row

begin

if (:new.sal)>=2500 then

insert into new\_top\_emp

values(:new.empno,:new.ename,:new.sal);

dbms\_output.put\_line('Record added in new\_top\_emp');

end if;

end;