**PL/SQL**

1. Write a PL/SQL block to display the factorial of a number.

set serveroutput on;

declare

n number;

ans number;

I number;

BEGIN

n:=&n;

ans:=1;

For I in 1..n

LOOP

Ans:=ans\*I;

END LOOP;

dbms\_output.put\_line('Factorial is '||ans);

END;

1. Write a PL/SQL block to find the area of a circle and insert these data into a table called Areas.

PL/SQL

CREATE TABLE Areas

(radius NUMBER, area NUMBER);

set serveroutput on;

DECLARE

r Areas.radius%Type;

area1 Areas.area%Type;

pi CONSTANT NUMBER:=3.14;

BEGIN

r:=&r;

area1:=pi\*r\*r;

INSERT INTO Areas VALUES(r,area1);

END;

1. Write a PL/SQL block to enter a new row into university table.

|  |  |
| --- | --- |
| **UN** | **NAME** |
| 1 | Kerala University |
| 2 | M.G. University |
| 3 | Calicut University |
| 4 | Anna University |
| 5 | Mysore University |
| 6 | Osmania University |
| 7 | kameron university |

PL/SQL

set serveroutput on;

DECLARE

ucode university.univcode%Type;

uname university.name%Type;

BEGIN

ucode:='&ucode';

uname:='&uname';

insert into university values(ucode,uname);

end;

1. Write a pl/sql program to increase the commsion\_rate by 10%.

PL/SQL

set serveroutput on;

DECLARE

comrate NUMBER;

BEGIN

update salesperson set comrate=comrate+comrate\*0.1;

dbms\_output.put\_line(comrate); commit;

END;

1. Write a pl/sql program to display the name of the customer with the highest commission rate.

PL/SQL

set serveroutput on;

DECLARE

comrate salesperson.commision\_rate%type;

hname salesperson.f\_name%type;

BEGIN

select max(commission\_rate) into comrate from salesperson;

select f\_name into hname from salesperson where commission\_rate= comrate;

dbms\_output.put\_line('The student with the highest percentage is'|| hname);

1. Write a PL/SQL program to accept the customer number and print the bill for the same. The charges are calculated as follows:

**Units consumed Charge**

<20 Nil

20-100 50% per unit

101-300 75% per unit

301-500 150% per unit

>500 225% per unit

create table cust(id int primary key,name varchar(12),pre\_read float,past\_read float);

insert into cust values(1,'aparna',45,20)

insert into cust values(2,'mohan',100,50);

insert into cust values(3,'nikhil',900,605);

insert into cust values(4,'zoya',1500,600);

insert into cust values(5,'hamna',100,90);

insert into cust values(6,'george',600,100);

**DECLARE**

CUSTOMER\_NO NUMBER;

P1 NUMBER;

P2 NUMBER;

U NUMBER;

CHARGE VARCHAR(5);

**BEGIN**

CUSTOMER\_NO:= &customer\_no;

Select pre\_read,past\_read into p1,p2 from cust where id=CUSTOMER\_NO;

u:=p1-p2;

If u<20 then charge:='nil';

Else if u> 20 and u<100 then charge:=u\*50/100;

Else if u>101 and u<300 then charge:= u\*75/100;

Else if u>301 and u<500 then charge:= u\*150/100;

Else charge:=u\* 225/100;

End if;

End if;

End if;

End if;

dbms\_output.put\_line('electricity bill');

dbms\_output.put\_line('customer no:'|| CUSTOMER\_NO);

Dbms\_output.put\_line('present reading:'||P1);

dbms\_output.put\_line('past reading:'||P2);

Dbms\_output.put\_line('units consumed:'||U);

Dbms\_output.put\_line('charge:'||CHARGE);

**END;**

1. Write a PL/SQL block for inserting rows into EMP((ENO, ENAME, DEPTNO, BASIC, HRA, DA, PF, NETPAY) table with the following Calculations:

HRA=50% OF BASIC

DA=20% OF BASIC

PF=7% OF BASIC

NETPAY=BASIC+DA+HRA-PF

DECLARE

ENO1 empy.id%type;

ENAME1 empy.name%type;

DEPTNO1 empy.deptno%type;

BASIC1 empy.salary%type;

HRA1 empy.HRA%type;

DA1 empy.DA%type;

PF1 empy.pf%type;

NETPAY1 empy.netpay%type;

BEGIN

ENO1:=&ENO1;

ENAME1:='&ENAME1';

DEPTNO1:=&DEPTNO1;

BASIC1:=&BASIC1;

HRA1:=(BASIC1\*50)/100;

DA1:=(BASIC1\*20)/100;

PF1:=(BASIC1\*7)/100;

NETPAY1:=BASIC1+HRA1+DA1-PF1;

INSERT INTO EMPY VALUES (ENO1, ENAME1, BASIC1, DEPTNO1, HRA1, DA1, PF1, NETPAY1);

END;

1. Create table with fields of accno, name and balance.Insert values into the table. Write a PL/SQL block to implement deposit and withdraw. Withdraw Should not be allowed if the balance goes below Rs.500.

DECLARE

BAL number(10);

ACNO varchar(10);

MIN\_BAL CONSTANT number(7,2):=500;

AMOUNT number(10);

CH number(1);

BEGIN

DBMS\_OUTPUT.PUT\_LINE('ENTER YOUR CHOICE');

DBMS\_OUTPUT.PUT\_LINE(TO\_CHAR(1)||'.DEPOSIT');

DBMS\_OUTPUT.PUT\_LINE(TO\_CHAR(2)||'.WITHDRAW');

CH:=&CH;

ACNO:=&ACNO;

Select balance into bal from bank where acc\_no=acno;

amount:=&amount;

IF(CH=1) THEN

UPDATE BANK SET BALANCE=BALANCE+AMOUNT WHERE ACC\_NO=ACNO;

DBMS\_OUTPUT.PUT\_LINE('DEPOSITED');

ELSE IF (BAL-AMOUNT)<MIN\_BAL THEN

DBMS\_OUTPUT.PUT\_LINE('Insufficient Funds..CANT WITHDRAW’);

ELSE

UPDATE BANK SET BALANCE=BALANCE-AMOUNT WHERE ACC\_NO=ACNO AND (BAL-AMOUNT)>MIN\_BAL;

DBMS\_OUTPUT.PUT\_LINE('WITHDRAWN'|| BAL- AMOUNT);

END IF;

END IF;

END;

/

1. Create table bank with fields of accno,name and balance. Insert values into bank table ,Write a PL/SQL block transfer the given amount of money from one account to another account.

**DECLARE**

BAL INT;

BAL1 INT;

BAL2 INT;

WITHDRAWAL\_AMT INT;

ACCNO1 INT;

ACCNO2 INT;

**BEGIN**

accno1:=&accno1;

accno2:=&accno2;

withdrawal\_amt:=& withdrawal\_amt;

select bank.balance into bal from bank where acc\_no=accno1;

**if**(bal- withdrawal\_amt >= 500)then

update bank set balance=balance- withdrawal\_amt where acc\_no=accno1;

update bank set balance=balance+ withdrawal\_amt where acc\_no=accno2;

select balance into bal1 from bank where acc\_no=accno1;

select balance into bal2 from bank where acc\_no=accno2;

dbms\_output.put\_line(‘NEW BALANCE OF ‘||accno1 ||’ IS ’|| bal1 ||‘ and ’ ||accno2 ||‘ IS ’||bal2);

**else**

dbms\_output.put\_line('balance not sufficient');

**end if;**

**END;**

/

1. Write a PL/SQL program to grade the student according to the following rules Student(name,rollno,mark1,mark2,mark3)

**TOTAL MARKS GRADE**

>=250 Distinction

180-250 First Class

120-179 Second Class

80-119 Third Class

<80 Fail

The result should be in the following Format

STUDENT NAME :

ROLL NO :

TOTAL MARKS :

GRADE :

Create table Stud(rollno int primary key,name char(10),mark1 float,mark2 float,mark3 float);

Insert into stud values(&rollno,’&name’,&mark1,&mark2,&mark3);

**DECLARE**

Name Char(10);

No int;

TOTMARK NUMBER(5,2);

**BEGIN**

Select rollno,name,(mark1+mark2+mark3) into No,name, TOTMARK from stud where rollno=&no;

IF TOTMARK >=250 THEN

DBMS\_OUTPUT.PUT\_LINE ('----------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE(‘ ROLL NO :'||no);

DBMS\_OUTPUT.PUT\_LINE(' STUDENT NAME :'|| name);

DBMS\_OUTPUT.PUT\_LINE(' TOTAL MARK :'|| TOTMARK);

DBMS\_OUTPUT.PUT\_LINE(' GRADE :DISTINCTION’);

DBMS\_OUTPUT.PUT\_LINE ('----------------------------------------------');

ELSE IF TOTMARK <250 AND TOTMARK >=180 THEN

DBMS\_OUTPUT.PUT\_LINE ('----------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE(‘ROLL NO :'||no);

DBMS\_OUTPUT.PUT\_LINE(' STUDENT NAME :'|| name);

DBMS\_OUTPUT.PUT\_LINE(' TOTAL MARK :'|| TOTMARK);

DBMS\_OUTPUT.PUT\_LINE(' GRADE :First Class’);

DBMS\_OUTPUT.PUT\_LINE ('----------------------------------------------');

ELSE IF TOTMARK <=179 AND TOTMARK >=120 THEN

DBMS\_OUTPUT.PUT\_LINE ('----------------------------------------------');

DBMS\_OUTPUT.PUT\_LINE(‘ROLL NO :'||no);

DBMS\_OUTPUT.PUT\_LINE(' NAME :'|| name);

DBMS\_OUTPUT.PUT\_LINE(' TOTAL MARK :'|| TOTMARK);

DBMS\_OUTPUT.PUT\_LINE(' GRADE :SECOND Class’);

DBMS\_OUTPUT.PUT\_LINE ('----------------------------------------------');

ELSE

DBMS\_OUTPUT.PUT\_LINE ('----------------------------------------------'); DBMS\_OUTPUT.PUT\_LINE(‘ROLL NO :'||no);

DBMS\_OUTPUT.PUT\_LINE(' NAME :'|| name);

DBMS\_OUTPUT.PUT\_LINE(' TOTAL MARK :'|| TOTMARK);

DBMS\_OUTPUT.PUT\_LINE('FAILED ');

DBMS\_OUTPUT.PUT\_LINE ('----------------------------------------------');

END IF;

END IF;

END IF;

**END;**

/

1. Write a PL/SQL block of code that first inserts a record in an Emp table. Update salaries 1001 and emp 1002 by Rs 2000 and Rs 1500. Then check to see that the total salary does not exceed Rs 20000. If total salary is greater than Rs 20000 then undo the updates made to emp 1001 and emp 1002.

create table emp(emp\_id integer, name varchar(15), sal integer);

select \* from emp;

EMP\_ID NAME SAL

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

1001 raju 2000

1002 rani 4000

DECLARE

total\_sal number(9);

BEGIN

insert into emp values (1005, ‘Ram’, 20000);

SAVEPOINT *no\_update;*

update emp set sal = sal + 2000 where emp\_id = 1001;

update emp set sal = sal + 1500 where emp\_id = 1002;

select sum(sal) into total\_sal from emp;

if total\_sal > 20000 then

ROLLBACK to SAVEPOINT *no\_update;*

end if;

COMMIT;

END;

**Exception Handling**

1. Write a program to accept empno and print the details as follows:

EmpId, name, sal. Also handle errors when user entered empno not exist in the table(Use Exception handling).

ID NAME DESIGNATION SAL

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

101 sam MANAGER 10000

102 syam CLERK 1500

103 ram SUPDT 10000

106 ramu TECNICIAN 2000

107 ramu TECNICIAN 2000

declare

emp\_miss exception;

nam varchar(20);

desig varchar(20);

eid int;

csal int;

emp\_id int;

begin

emp\_id:=&emp\_id;

select id,name,designation,sal into eid,nam, desig, csal from em where id=emp\_id;

if eid is null then

raise emp\_miss;

else

dbms\_output.put\_line('EMP ID :'||eid);

dbms\_output.put\_line('NAME :' ||nam);

dbms\_output.put\_line('DESIGNATION :'||desig);

dbms\_output.put\_line('SALARY : ' ||csal);

end if;

exception

when no\_data\_found then

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

end raise\_emp\_miss;

**Cursor**

1. **To write a Cursor to display the details of Employees from emp table.**

create table emply(id number primary key, name varchar(15), job varchar(15), deptno number);

insert into emply values(&id,'&name', ‘&job’,&deptno);

select \* from emply;

ID NAME JOB DEPTNO

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

1 kiran manager 10

2 riya office asst. 10

3 raju salesman 20

4 ram office asst. 30

DECLARE

cursor c1 is select \* from emp;

….

…….

……..

BEGIN

for i in c1 loop

dbms\_output.put\_line('Department is :' || i.emp\_id);

**........**

**……..**

END

1. To write a Cursor to display the list of Employees and Total Salary Department wise.

*Dept (deptno, deptname);*

create table dept(deptno number primary key, deptname varchar(20));

insert into dept values(&deptno,'&deptname');

DECLARE

cursor c1 is select \* from dept;

cursor c2 is select \* from empy;

s empy.salary%type;

BEGIN

for i in c1 loop

s:=0;

dbms\_output.put\_line('Department is :' || i.deptno);

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

for j in c2 loop

if ( i.deptno=j.deptno) then

s:=s+j.salary;

dbms\_output.put\_line (j.id|| ' '|| j.name || ' '|| j.salary );

end if;

end loop;

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

dbms\_output.put\_line ('Total salary is: '|| s);

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

end loop;

END;

1. Write a PLSQL block to declare a cursor and display the information about all customers.

.

**Function**

1. Write a function to find the square of a given number.

Code for creating the function

CREATE OR REPLACE FUNCTION

SQR(num number)

Return number as sqnum number;

BEGIN

Sqnum := num \* num;

RETURN sqnum;

END

PL/SQL code to use the function

DECLARE

inno number;

BEGIN

inno : = ‘&inno’

Dbms\_output.put\_line(SQR(inno));

END;

1. Create a function called FindCourse. It has one parameter called cname and it returns a number. The function will return the course number if it finds a match based on the course name ( in table course); otherwise it returns 999.

Course( course\_no number, course\_name varchar(10))

SQL> select \* from course;

COURSE\_NO COURSE\_NAM

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

101 BSc.

102 BA

104 BTech.

105 MTech.

106 MBA

103 MA

106 MSc.

7 rows selected.

CREATE OR REPLACE FUNCTION FindCourse(cname IN varchar2) RETURN number IS

cnumber number:=null ;

begin

Select course\_no into cnumber from course where course\_name=cname;

Return cnumber;

exception

when no\_data\_found then

cnumber:=999;

End;

Declare

Cs varchar2(10);

Cno number;

begin

Cs:=’&Cs’;

Cno:=FindCourse(Cs);

        dbms\_output.put\_line(Cno);

End;

/

Output

Enter value for cs: BA

old 5: Cs:='&Cs';

new 5: Cs:='BA';

102

PL/SQL procedure successfully completed.

Enter value for cs: mca

old 5: Cs:='&Cs';

new 5: Cs:='mca';

999

Procedure

1. Write a procedure to increase the salary for the specified employee based on the following criteria

Clerk - 5%

Salesman - 7%

Analyst - 10%

Manager - 20%

GM - 25%

emptable (empno, ename, job, salary,dept)

create table emptable(empno number primary key, ename varchar2(15), job varchar2(15), salary number, dept varchar(20));

insert into emptable values(&empno, '&ename', '&job', &salary, '&dept');

EMPNO ENAME JOB SALARY DEPT

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

1 anu clerk 7000 admin

2 binu clerk 7000 purchase

3 manu manager 15000 purchase

4 dany gm 50000 sales

5 dennis analyst 35000 sales

6 raju salesman 8000 sales

create or replace procedure incsal(eno number) is

begin

update emptable set salary=salary+ (salary\*0.05) where job=’clerk’ and empno=eno;

update emptable set salary=salary+ (salary \*0.07) where job=’salesman’ and empno=eno; update emptable set salary=salary+ (salary \*0.1) where job=’analyst’ and empno=eno; update emptable set salary=salary+ (salary \*0.2) where job=’manager’ and empno=eno; update emptable set salary=salary+ (salary \*0.25) where job=’gm’ and empno=eno;

end incsal;

/

Procedure created.

SQL> declare

no emptable.empno%type;

begin

no:=&no;

incsal(no);

end; /

1. Write a procedure to transfer the specified employee to the specified department

emptable (empno, ename, job, salary,dept)

EMPNO ENAME JOB SALARY DEPT

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

1 anu clerk 7350 admin

2 binu clerk 7000 purchase

3 manu manager 15000 purchase

4 dany gm 50000 sales

5 dennis analyst 35000 sales

6 raju salesman 8000 sales

create or replace procedure trans(eno number, dno varchar) is

begin

update emptable set dept=dno where empno=eno;

end trans;

/

Procedure created.

SQL> declare

employee\_no emptable.empno%type;

dept\_to \_trans emptable.dept%type;

begin

employee\_no:=&employee\_no;

dept\_to \_trans:=’&dept\_to \_trans’;

trans(employee\_no, dept\_to \_trans);

end;

/

1. Write a PL/SQL procedure to find the net salary of an employee in EMP table (ENO, ENAME, BASIC) with the following Calculations:

HRA=50% OF BASIC

DA=20% OF BASIC

PF=7% OF BASIC

NETPAY=BASIC+DA+HRA-PF

create or replace procedure netsalary(BASIC1 number) is

HRA1 float;

DA1 float;

PF1 float;

NETPAY float;

BEGIN

HRA1:=(BASIC1\*50)/100;

DA1:=(BASIC1\*20)/100;

PF1:=(BASIC1\*7)/100;

NETPAY:=BASIC1+HRA1+DA1-PF1;

DBMS\_OUTPUT.PUT\_LINE(‘Net Salary = ‘||NETPAY);

END netsalary;

/

SQL> declare

sal emptable.salary%type:=null;

ENO1 emptable.empno%type;

Begin

ENO1:=&ENO1;

Select salary into sal from emptable where empno=ENO1;

If sal is not null then

Netsalary(sal);

Else

dbms\_output.put\_line(‘Not a valid Employee’);

End if;

End;

/

**Trigger**

1. Create a database containing tables

Depositor(acc\_no primary key, name)

Loan(loan\_no, amount, branch)

Borrower(loan\_no, cust\_name)

Accnt(acc\_no, balance, branch)

User can update account details on table ***Accnt***. If the user decrement the balance amount in the Accnt table then the trigger must insert the account number and borrower’s name to the table borrower and the trigger must insert the account number, loan amount and branch name to table loan also.

Create table depositor(acc\_no primary key, name varchar(15));

Create table accnt(acc\_no, balance number, branch varchar(10), foreign key(acc\_no) references depositor(acc\_no);

Create table borrower (loan\_no number, name varchar(15));

Create table loan (loan\_no number, branch varchar(10), amount number);

Insert into depositor values(&acc\_no, &name);

Insert into accnt values (&acc\_no, &balance, ‘&branch’);

Create or replace trigger account after update on balance on accnt for each row

Declare

c\_name varchar(15);

begin

if(:new.balance<:old.balance) then

insert into loan values(:new.acc\_no, :new.branch, :old.balance-:new.balance);

select name into c\_name from depositor where acc\_no = :new.acc\_

no;

insert into borrower values(:new.accno,c\_name);

end if;

end;

/