6.1

Q 1. **Consider table Stud(Roll, Att,Status)**

Write a PL/SQL block for following requirement and handle the exceptions.

Roll no. of student will be entered by user. Attendance of roll no. entered by user will be checked in Stud table. If attendance is less than 75% then display the message “Term not granted” and set the status in stud table as “D”. Otherwise display message “Term granted” and set the status in stud table as “ND”.

🡺

Declare

s\_roll stud.roll%type;

s\_att stud.att%type;

begin

s\_roll :=&Enter\_roll\_no;

select att into s\_att from stud where roll=s\_roll;

if s\_att<75 then

dbms\_output.put\_line('Term not granted');

update stud set status='D' where roll=s\_roll;

else

dbms\_output.put\_line('Term granted');

update stud set status='P' where roll=s\_roll;

end if;

Exception

when no\_data\_found then

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

end;

6.2

. The bank manager has decided to activate all those accounts which were previously marked as inactive for performing no transaction in last 365 days. Write a PL/SQ block (using implicit cursor) to update the status of account, display an approximate message based on the no. of rows affected by the update. (Use of %FOUND, %NOTFOUND, %ROWCOUNT)

create table acc1(acc\_no number(4),name char(10),status char(3));

select \*from acc1;

ACC\_NO NAME STA

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

111 abc IA

222 sdf IA

333 erer IA

Declare

row\_c number(3);

begin

update acc1 set status='A'where status='IA';

//dbms\_output.put\_line('hiii');

if SQL%FOUND then

row\_c:=SQL%ROWCOUNT;

dbms\_output.put\_line(row\_c||'is updated');

elsif SQL%NOTFOUND then

dbms\_output.put\_line(' no row is updated');

end if;

end;

/

=========================================================== ======================

7.1

. Write an SQL code block these raise a user defined exception where business rule is voilated. BR for client\_master table specifies when the value of bal\_due field is less than 0 handle the exception.

create table acc\_exception(acc\_no number(4) UNIQUE,balance number(9));

select \* from acc\_exception;

ACC\_NO BALANCE

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

11 2300

22 0

24 5500

21 500

declare

mwithdraw acc\_exception.balance%type;

mbalance acc\_exception.balance%type;

macc\_no acc\_exception.acc\_no%type;

RULES\_VOILATED Exception ;

Begin

macc\_no:=&Enter\_acc\_no;

mwithdraw:=&Enter\_amount\_to\_wihdraw;

select balance into mbalance from acc\_exception where macc\_no=acc\_no;

if mwithdraw > mbalance then

raise RULES\_VOILATED;

elsif mbalance >=mwithdraw then

update acc\_exception set balance=balance-mwithdraw where macc\_no=acc\_no;

end if;

Exception

when RULES\_VOILATED then

dbms\_output.put\_line(' Exception is occured amount is not sufficient to withdraw');

when OTHERS then

dbms\_output.put\_line(' EROOR OCCURED');

end;

/

=================================================--------------------------------------------------------

7.2

Q 2. Organization has decided to increase the salary of employees by 10% of existing salary, who are having salary less than average salary of organization, Whenever such salary updates takes place, a record for the same is maintained in the increment\_salary table.

**EMP (E\_no , Salary)**

**increment\_salary(E\_no , Salary)**

declare

msalary emp.salary%type;

memp\_no emp.emp\_no %type;

cursor c is select emp\_no ,salary from emp where salary<(select avg(salary) from emp);

begin

open c;

if c%ISOPEN then

loop

fetch c into memp\_no,msalary;

exit when c%NOTFOUND;

if c%FOUND then

update emp set salary=msalary + 0.01\*msalary where salary=msalary;

insert into inc\_salary values(memp\_no,msalary + 0.01\*msalary);

else

dbms\_output.put\_line('no salary is greater than average salary');

end if;

end loop;

close c;

end if;

EXCEPTION

WHEN INVALID\_CURSOR THEN

DBMS\_OUTPUT.PUT\_LINE('attempts are made to make a cursor operation that is not allowed');

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('error occured in arithmatic operaions');

WHEN PROGRAM\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('internal program error occured');

WHEN STORAGE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('PL/SQL got out of memory | memory got corrupted');

END;

/

8

**. Borrower(Roll\_no, Name, DateofIssue, NameofBook, Status)**

**Fine(Roll\_no,Date,Amt)**

1. Accept roll\_no & name of book from user.

* 1. Check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs 5per day.

1. If no. of days>30, per day fine will be Rs 50 per day & for days less than 30, Rs. 5 per day.
2. If condition of fine is true, then details will be stored into fine table.
3. Also handles the exception by named exception handler or user define exception handler.

create or replace procedure compute(mdays in number,mroll\_no in number)is

famount number(5);

BEGIN

if(mdays>15 and mdays<30) then

famount:=mdays\*5;

insert into fine values(mroll\_no,SYSDATE,famount);

elsif(mdays >30) then

famount:=(mdays-30)\*50+75;

insert into fine values(mroll\_no,SYSDATE,famount);

else

famount:=0;

end if;

update borrow set status='R'where mroll\_no=roll\_n;

Exception

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('error occured in arithmatic operaions');

WHEN PROGRAM\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('internal program error occured');

WHEN STORAGE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('PL/SQL got out of memory | memory got corrupted');

END;

/

declare

mroll\_no number(3);

mdoi date;

mbook\_name varchar(20);

mdays number(3);

Begin

mroll\_no:= &Enter\_roll\_no;

select doi into mdoi from borrow where roll\_n=mroll\_no;

mdays:=SYSDATE-mdoi;

compute(mdays,mroll\_no);

Exception

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('error occured in arithmatic operaions');

WHEN PROGRAM\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('internal program error occured');

WHEN STORAGE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('PL/SQL got out of memory | memory got corrupted');

END;

/

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

9:::::::::::::::::::::::::::::

Write PL/SQL block using explicit cursor for following requirements:

College has decided to mark all those students detained (D) who are having attendance less than 75%.

Whenever such update takes place, a record for the same is maintained in the D\_Stud table.

**create table stud21(roll number(4), att number(4), status varchar(1)); create table d\_stud(roll number(4), att number(4));**

Declare

mroll\_no stud9.roll\_no%type;

matt stud9.att%type;

mstatus stud9.status%type;

cursor c is

select roll\_no,att,status from stud9 where att<75;

begin

open c;

if c%ISOPEN then

loop

Fetch c into mroll\_no,matt,mstatus;

exit when c%NOTFOUND;

if c%found then

update stud9 set status='D'where roll\_no=mroll\_no;

insert into stud\_d values(mroll\_no,matt);

end if;

end loop;

end if;

close c;

Exception

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('error occured in arithmatic operaions');

WHEN PROGRAM\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('internal program error occured');

WHEN STORAGE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('PL/SQL got out of memory | memory got corrupted');

END;

/

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

10.1 already covered 6.2

10.2

. Write a update, delete trigger on clientmstr table. The System should keep track of the records that ARE BEING updated or deleted. The old value of updated or deleted records should be added in audit\_trade table. (separate implementation using both row and statement triggers)

|  |
| --- |
| > CREATE TABLE clientmstr10\_2 (emp\_name VARCHAR(30) NOT NULL UNIQUE, salary NUMBER(10,2) NOT NULL); |
|  | > CREATE TABLE audittrade10\_2 (emp\_name VARCHAR(30) NOT NULL, salary NUMBER(10,2) NOT NULL); |
|  | > ALTER TABLE audittrade10\_2 ADD opt VARCHAR(30); |

create or replace trigger t

after DELETE OR UPDATE on client\_mstr

for each row

enable

Declare

opt audit\_trade.opt%type;

begin

if UPDATING then

opt:='update';

elsif DELETING then

opt:='Delete';

end if;

insert into audit\_trade values(:old.emp\_name,:old.salary,opt);

end;

/

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

13

. Write a stored function in PL/SQL for given requirement and use the same in PL/SQL block. Account no. and branch name will be accepted from user. The same will be searched in table acct\_details. If status of account is active then display appropriate message and also store the account details in active\_acc\_details table, otherwise display message on screen “account is inactive”.

declare

macc\_no account\_13.acc\_no%type;

mbranch\_name account\_13.branch\_name%type;

mstatus account\_13.status%type;

begin

macc\_no:=&Enter\_acc\_number;

select status into mstatus from account\_13 where acc\_no=macc\_no;

select branch\_name into mbranch\_name from account\_13 where acc\_no=macc\_no;

checkstatus(macc\_no,mbranch\_name,mstatus);

Exception

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('error occured in arithmatic operaions');

WHEN PROGRAM\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('internal program error occured');

WHEN STORAGE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('PL/SQL got out of memory | memory got corrupted');

END;

/

create or replace procedure checkstatus(macc\_no number, mbranch\_name varchar,mstatus varchar)is

begin

if mstatus='A' then

dbms\_output.put\_line('Account:'||macc\_no|| 'is active');

insert into active\_account\_13 values(macc\_no,mbranch\_name,mstatus);

else

dbms\_output.put\_line('Account:'||macc\_no|| 'is NOT active');

end if;

Exception

WHEN VALUE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('error occured in arithmatic operaions');

WHEN PROGRAM\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('internal program error occured');

WHEN STORAGE\_ERROR THEN

DBMS\_OUTPUT.PUT\_LINE('PL/SQL got out of memory | memory got corrupted');

END;

14

**. Write a Stored Procedure namely proc\_Grade for the categorization of student. If marks scored** by students in examination is <=1500 and marks>=990 then student will be placed indistinction category if marks scored are between 989 and900 category is first class, if marks 899 and 825 category is Higher Second Class.

Write a PL/SQL block for using procedure created with above requirement.

**Stud\_Marks(name, total\_marks)**

**Result(Roll,Name, Class)**

|  |
| --- |
| > CREATE TABLE studmarks14\_1 (roll\_no NUMBER(3,0)UNIQUE NOT NULL , sname VARCHAR(30) NOT NULL, total\_marks NUMBER(5,0) NOT NULL); |
|  | > CREATE TABLE result14\_1 (roll\_no NUMBER(3,0) UNIQUE NOT NULL, sname VARCHAR(30) NOT NULL, rclass VARCHAR(30) NOT NULL); |
|  |  |
|  | ######procedure |
|  | CREATE OR REPLACE PROCEDURE procgrade14\_1(mroll\_no NUMBER, msname VARCHAR, mtotal\_marks NUMBER) IS |
|  | mclass result14\_1.rclass%type; |
|  | BEGIN |
|  | IF mtotal\_marks <= 1500 AND mtotal\_marks >= 990 THEN |
|  | mclass := 'Distinction'; |
|  | DBMS\_OUTPUT.PUT\_LINE('Marks : ' || mtotal\_marks || ' falls in category : DISTINCTION'); |
|  | ELSIF mtotal\_marks <= 900 AND mtotal\_marks >= 989 THEN |
|  | mclass := 'First Class'; |
|  | DBMS\_OUTPUT.PUT\_LINE('Marks : ' || mtotal\_marks || ' falls in category : FIRST CLASS'); |
|  | ELSIF mtotal\_marks <= 899 AND mtotal\_marks >=825 THEN |
|  | mclass := 'Higher Second Class'; |
|  | DBMS\_OUTPUT.PUT\_LINE('Marks : ' || mtotal\_marks || ' falls in category : HIGHER SECOND CLASS'); |
|  | ELSE |
|  | mclass := 'Fail'; |
|  | DBMS\_OUTPUT.PUT\_LINE('Marks : ' || mtotal\_marks || ' falls in category : FAIL'); |
|  | END IF; |
|  | INSERT INTO result14\_1 VALUES (mroll\_no, msname, mclass); |
|  |  |
|  | EXCEPTION |
|  | WHEN VALUE\_ERROR THEN |
|  | DBMS\_OUTPUT.PUT\_LINE('error occured in arithmatic operations'); |
|  | WHEN PROGRAM\_ERROR THEN |
|  | DBMS\_OUTPUT.PUT\_LINE('internal program error occured'); |
|  | WHEN STORAGE\_ERROR THEN |
|  | DBMS\_OUTPUT.PUT\_LINE('PL/SQL got out of memory | memory got corrupted'); |
|  | END; |
|  | / |
|  |  |
|  | ###calling |
|  | DECLARE |
|  | mroll\_no studmarks14\_1.roll\_no%type; |
|  | msname studmarks14\_1.sname%type; |
|  | mtotal\_marks studmarks14\_1.total\_marks%type; |
|  |  |
|  | BEGIN |
|  | mroll\_no := &roll\_no; |
|  | SELECT roll\_no, sname, total\_marks INTO mroll\_no, msname, mtotal\_marks |
|  | FROM studmarks14\_1 |
|  | WHERE |
|  | roll\_no = mroll\_no; |
|  | procgrade14\_1(mroll\_no, msname, mtotal\_marks); |
|  |  |
|  | EXCEPTION |
|  | WHEN VALUE\_ERROR THEN |
|  | DBMS\_OUTPUT.PUT\_LINE('error occured in arithmatic operations'); |
|  | WHEN PROGRAM\_ERROR THEN |
|  | DBMS\_OUTPUT.PUT\_LINE('internal program error occured'); |
|  | WHEN STORAGE\_ERROR THEN |
|  | DBMS\_OUTPUT.PUT\_LINE('PL/SQL got out of memory | memory got corrupted'); |
|  | END; |
|  | / |

**----------------------------------------------------------------------------------**

**reverse number**

**-------------------------------------------------------------------------------**

**Declare**

**num number(10);**

**snum varchar(10);**

**lnum number(10);**

**output varchar(10);**

**begin**

**num:=&Enter\_number;**

**lnum:=length(num);**

**snum:=to\_char(num);**

**for i in reverse 1..lnum**

**loop**

**output:=output || substr(snum,i,1);**

**end loop;**

**dbms\_output.put\_line('Reverse of the number:'||output);**

**Exception**

**WHEN VALUE\_ERROR THEN**

**DBMS\_OUTPUT.PUT\_LINE('error occured in arithmatic operaions');**

**WHEN PROGRAM\_ERROR THEN**

**DBMS\_OUTPUT.PUT\_LINE('internal program error occured');**

**WHEN STORAGE\_ERROR THEN**

**DBMS\_OUTPUT.PUT\_LINE('PL/SQL got out of memory | memory got corrupted');**

**END;**

**/**

create a row level trigger for the CUSTOMERS table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old values and new values

|  |
| --- |
| > CREATETABLE customer27\_1 (cust\_id NUMBER(3,0) UNIQUE NOT NULL, cust\_name VARCHAR(30) NOT NULL); |
|  | > ALTER TABLE customer27\_1 ADD csalary NUMBER(10,2); |
|  |  |
|  | >> |
|  | CREATE OR REPLACE TRIGGER tcust27\_1 |
|  | AFTER UPDATE OR DELETE OR INSERT ON customer27\_1 |
|  | FOR EACH ROW |
|  | DECLARE |
|  | mcsalary customer27\_1.csalary%type; |
|  | mdiff NUMBER(10,2) := 0.00; |
|  | BEGIN |
|  |  |
|  | IF UPDATING THEN |
|  | mdiff := :NEW.csalary - :OLD.csalary; |
|  | DBMS\_OUTPUT.PUT\_LINE('updating, difference : ' || mdiff); |
|  | ELSIF INSERTING THEN |
|  | DBMS\_OUTPUT.PUT\_LINE('inserting, difference : ' || mdiff); |
|  | ELSIF DELETING THEN |
|  | DBMS\_OUTPUT.PUT\_LINE('deleting, difference : ' || mdiff); |
|  | END IF; |
|  | END; |
|  | / |

============================================================= = = = = = = = = = == = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Write PL/SQL block to update the Customer table and increase the salary of each customer by 500 and use the **SQL%ROWCOUNT**attribute to determine the number of rows affected.

|  |
| --- |
| > CREATE TABLE customer28\_1 (cust\_id NUMBER(3,0) UNIQUE NOT NULL, cust\_name VARCHAR(30) NOT NULL, csalary NUMBER(10,2) NOT NULL); |
|  |  |
|  | >>> |
|  | DECLARE |
|  | increment CONSTANT NUMBER(10,2) := 500; |
|  | total\_rows NUMBER(3,0); |
|  | BEGIN |
|  | UPDATE customer28\_1 SET csalary = csalary + increment; |
|  | IF SQL%FOUND THEN |
|  | total\_rows := SQL%ROWCOUNT; |
|  | DBMS\_OUTPUT.PUT\_LINE('number of rows affected : ' || total\_rows); |
|  | END IF; |
|  | END; |
|  | / |

= = = === = == = = = === = == = = = == = = = = == == = = = = = = = ==== === === === == == = == = = = = =

A stored function is created to perform the ACCOUNT\_NO check operation .F\_checkAccNO() is the name of function which accept a variable ACCOUNT\_NO and returns the value to host environment The value changes from 0(if ACCOUNT\_NO does not exist) to 1(if ACCOUNT\_NO exist) depending on the records retrieved.

|  |
| --- |
| > CREATE TABLE account\_master\_25\_1 (acc\_no NUMBER(3,0) UNIQUE NOT NULL, ahname VARCHAR(30) NOT NULL); |
|  |  |
|  | CREATE OR REPLACE FUNCTION fcheckacc\_no\_26\_1 (macc\_no NUMBER) |
|  | RETURN NUMBER |
|  | IS |
|  | CURSOR ccheckacc\_no(macc\_no NUMBER) IS |
|  | SELECT \*FROM account\_master\_25\_1 |
|  | WHERE acc\_no = macc\_no; |
|  | mlacc\_no account\_master\_25\_1.acc\_no%type; |
|  | mlahname account\_master\_25\_1.ahname%type; |
|  | mstatus NUMBER(1,0); |
|  | BEGIN |
|  | OPEN ccheckacc\_no(macc\_no); |
|  | FETCH ccheckacc\_no INTO mlacc\_no, mlahname; |
|  | IF ccheckacc\_no%FOUND THEN |
|  | mstatus := 1; |
|  | ELSE |
|  | mstatus := 0; |
|  | END IF; |
|  | CLOSE ccheckacc\_no; |
|  | RETURN mstatus; |
|  | EXCEPTION |
|  | WHEN INVALID\_CURSOR THEN |
|  | DBMS\_OUTPUT.PUT\_LINE('attempts are made to make a cursor operation that is not allowed'); |
|  | WHEN VALUE\_ERROR THEN |
|  | DBMS\_OUTPUT.PUT\_LINE('error occured in arithmatic operaions'); |
|  | WHEN PROGRAM\_ERROR THEN |
|  | DBMS\_OUTPUT.PUT\_LINE('internal program error occured'); |
|  | WHEN STORAGE\_ERROR THEN |
|  | DBMS\_OUTPUT.PUT\_LINE('PL/SQL got out of memory | memory got corrupted'); |
|  | END; |
|  | / |
|  |  |
|  | >>####calling |
|  | DECLARE |
|  | macc\_no account\_master\_25\_1.acc\_no%type; |
|  | BEGIN |
|  | macc\_no := &acc\_no; |
|  | IF fcheckacc\_no\_26\_1(macc\_no) = 1 THEN |
|  | DBMS\_OUTPUT.PUT\_LINE('Account found in database.'); |
|  | ELSE |
|  | DBMS\_OUTPUT.PUT\_LINE('Account not found in database.'); |
|  | END IF; |
|  | END; |
|  | / |

=============================================================================