

# American International University- Bangladesh

|  |  |  |
| --- | --- | --- |
| **Course Name** | ADVANCE DATABASE MANAGEMENT SYSTEM |  |
| **Report** | Blood Bank Management System. |
| **Submitted To** | REZWAN AHMED |
| **Semester** | Summer 2019-20 |
| **Submission Date** | 20-09-2020 |
| **Section** | A &B |
| **Group Name** | BTR |

**GROUP MEMBERS**

|  |  |  |
| --- | --- | --- |
| **Student Name** | **Student ID** | **Section** |
| 1. RAHMAN MD.SAJJADUR | 17-35034-2 | B |
| 2. ISLAM,MD.RAIHANUL | 17-34988-2 | A |
| 3. NASIF, NABIL AL | 17-35131-2 | B |
| 4. RAHIL,MD.ABU HANIF | 17-35074-2 | B |
| **Teacher Remarks**  (Only for teacher) |  |  |

**TABLE**

**Information**

**Sequence:**

Create sequence Informations\_sequence start with 1

increment by 1

minvalue 1

maxvalue 10000;

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

CREATE TABLE Informations(

Info\_Id NUMBER(5) CONSTRAINT PK\_Informations PRIMARY KEY not null,

Full\_Name varchar(255) NOT NULL,

Username varchar(255) NOT NULL,

Address varchar(255) NOT NULL,

Blood\_Group varchar(255) NOT NULL,

Gender varchar(255) NOT NULL,

DOB date NOT NULL,

Profession varchar(255) NOT NULL,

Gmail varchar(255) NOT NULL

);

**Users**

**Sequence:**

Create sequence Users\_sequence start with 1

increment by 1

minvalue 1

maxvalue 10000;

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

CREATE TABLE Users (

Id NUMBER(5) CONSTRAINT PK\_Users PRIMARY KEY not null,

Username varchar(255) NOT NULL,

Password varchar(255) NOT NULL,

Type varchar(255) NOT NULL,

Permission varchar(255) NOT NULL,

Info\_Id number(5) CONSTRAINT FK\_Info\_Id\_users REFERENCES Informations

);

**Blood\_stocks**

**Sequence:**

Create sequence Blood\_stocks\_sequence start with 1

increment by 1

minvalue 1

maxvalue 10000;

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

CREATE TABLE Blood\_stocks (

Blood\_id NUMBER(5) CONSTRAINT PK\_Blood\_stock PRIMARY KEY not null,

Blood\_Group varchar(255) NOT NULL,

Quantity number(5) NOT NULL

);

**Donates**

**Sequence:**

Create sequence Donates\_sequence start with 1

increment by 1

minvalue 1

maxvalue 10000;

CREATE TABLE Donates (

Serial NUMBER(5) CONSTRAINT PK\_Donates PRIMARY KEY not null,

Info\_Id number(5) CONSTRAINT FK\_Info\_Id\_donate REFERENCES Informations,

Blood\_Group varchar(255) NOT NULL,

Last\_time\_donated date NOT NULL

);

**Orders**

**Sequence:**

Create sequence Orders\_sequence start with 1

increment by 1

minvalue 1

maxvalue 10000;

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

CREATE TABLE Orders (

Order\_serial NUMBER(5) CONSTRAINT PK\_Order PRIMARY KEY not null,

Username varchar(255) NOT NULL,

Order\_status varchar(255) NOT NULL,

Blood\_id number(5) CONSTRAINT FK\_Blood\_Id\_order REFERENCES Blood\_stocks,

Quantity number(5) not null,

Member\_status varchar(255) NOT NULL,

service\_charge number(5) NOT NULL

);

**TRIGGER**

**1. Validate Info to add a user.**

create or replace trigger user\_info

before insert on users

for each row

declare

c int;

begin

c:=0;

select count(\*) into c from Informations where Info\_Id=:new.Info\_Id;

if c=0 then

Raise\_application\_error(-20111,'Invalid Info Id');

end if;

end;

**2. Before insert data on Donates whether it is in Information table or not.**

create or replace trigger Doner\_info

before insert on Donates

for each row

declare

c int;

begin

c:=0;

select count(\*) into c from Informations where Info\_Id=:new.Info\_Id;

if c=0 then

Raise\_application\_error(-20113,'Invalid Info Id');

end if;

end;

3. **Before insert data on Orders whether blood group in Blood\_stocks table or not.**

create or replace trigger BloodGroup\_Checking

before insert on Orders

for each row

declare

c int;

begin

c:=0;

select count(\*) into c from Blood\_stocks where Blood\_id =:new.Blood\_id ;

if c=0 then

Raise\_application\_error(-20114,'Invalid Blood Group');

end if;

end;

**4. Service charge check**

create or replace trigger serviceCharge\_Check

before insert on Orders

for each row

Begin

if :new.service\_charge<0 then

Raise\_application\_error(-20115,'Service charge can not be negetive');

else

dbms\_output.put\_line('insert Successfully');

end if;

end;

**5. Check permission**

create or replace trigger Permission\_Checking

before insert on Donates

for each row

declare

c varchar2(10);

Begin

select permission into c from users where Info\_Id=:new.Info\_Id;

if c='Invalid' then

Raise\_application\_error(-20116,'Permission Invalid');

else

dbms\_output.put\_line('Donated Successfully');

end if;

end;

**6. Check Username**

create or replace trigger CheckUsername

before insert on Informations

for each row

declare

c int;

begin

c:=0;

select count(\*) into c from Informations where Username=:new.Username;

if c>0 then

Raise\_application\_error(-20120,'Username is in used ! Use another one!');

end if;

end;

**7. Manage User in office time**

create or replace trigger ManageUser

before

insert or update or delete on Users

begin

if (to\_char(sysdate,'HH24:MM') not between '9:00' and '17:00') then

Raise\_application\_error(-20005,'Yoy take normal hours!');

elsif to\_char(sysdate,'Day') in ('Saturday','Sunday') then

Raise\_application\_error(-20006,'Yoy take weekends!');

else

dbms\_output.put\_line('Operation Successful');

end if;

end;

**Function**

**1.Total service charge calculate**

create or replace function TotalServiceCharge

return float

is

summation float:=0;

begin

select sum(service\_charge) into summation from Orders;

return summation ;

end;

**2.Unit wise service charge calculate**

create or replace function UnitTotalServiceCharge(uname Orders.Username%type)

return float

is

summation float:=0;

begin

select sum(service\_charge) into summation from Orders where Username=uname;

return summation ;

end;

**3. Calculate charge**

create or replace function CalCharge(q Orders.Quantity%type,c number)

return number

is

total number(5);

begin

total:=q\*c;

return total;

end;

**4. Check Quantity**

create or replace function CheckValidQuantity(bg Blood\_stocks.Blood\_Group%type,quan Blood\_stocks.Quantity%type)

return boolean

is

q number(5);

begin

select Quantity into q from Blood\_stocks where Blood\_Group=bg;

if quan>q then

return false;

else

return true;

end if;

end;

**5. Check Username**

create or replace function CheckUsername(uname Users.Username%type)

return boolean

is

c int;

begin

c:=0;

select count(\*) into c from users where Username=uname;

if c!=0 then

return true;

else

return false;

end if;

end;

**6. Check Password**

create or replace function CheckPassword(uname Users.Username%type,pass Users.Password%type)

return boolean

is

c int;

begin

c:=0;

select count(\*) into c from users where Username=uname and Password=pass;

if c!=0 then

return true;

else

return false;

end if;

end;

**PROCEDURE**

**1. Insert\_Order**

create or replace procedure InsertOrders(uname Orders.Username%type,ot Orders.Order\_Status%type,

bg Blood\_Stocks.Blood\_Group%type,qt Orders.Quantity%type,mt Orders.Member\_Status%type,c number)

is

bld Blood\_Stocks.Blood\_Id%type;

sc Orders.Service\_Charge%type;

begin

sc:=CalCharge(qt,c);

select Blood\_Id into bld from Blood\_Stocks where Blood\_Group=bg;

Insert into Orders values (Orders\_sequence.nextval,uname,ot,bld,qt,mt,sc);

exception

when no\_data\_found then

dbms\_output.put\_line('Invalid Blood group');

when others then

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

end;

**2.Total service charge**

create or replace procedure Total\_ServiceCharge(sc out NUMBER)

is

begin

sc:=TotalServiceCharge;

Dbms\_Output.Put\_line(sc);

end;

**3.Usernam to Info\_id& blood group**

create or replace procedure Username\_Info(uname Informations.Username%type,iid OUT Informations.Info\_Id%type,bld out Informations.Blood\_Group%type)

is

begin

select Info\_Id into iid from Informations where Username=uname;

select Blood\_Group into bld from Informations where Username=uname;

exception

when no\_data\_found then

dbms\_output.put\_line('No information');

when others then

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

end;

**4.Unit service charge**

create or replace procedure UnitTotal\_ServiceCharge(sc out number,uname Users.Username%type)

is

begin

sc:=UnitTotalServiceCharge(uname);

Dbms\_Output.Put\_line(sc);

end;

**5. Update Informations**

create or replace procedure update\_info(u Users.Username%type,m Informations.Gmail%type,a informations.Address%type,c out number)

is

begin

if CheckUsername(u) then

update Informations set Gmail=m,Address=a where Username=u;

c:=1;

else

c:=0;

dbms\_output.put\_line('Invalid username');

end if;

end;

**6. Update Password**

create or replace procedure updatepassword(u Users.Username%type,old Users.Password%type,new Users.Password%type,c out number)

is

begin

if CheckPassword(u,old) then

update Users set Password=new where Username=u;

c:=1;

else

c:=0;

dbms\_output.put\_line('Invalid username');

end if;

end;

**PACKAGE**

**1.Donate dml**

create or replace package donates\_dml as

procedure donateInfo\_insert(s Donates.Serial%type,id Donates.Info\_Id%type,bg Donates.Blood\_Group%type,t Donates.Last\_time\_donated%type);

procedure donateInfo\_delete(s Donates.Serial%type);

end donates\_dml;

create or replace package body donates\_dml as

function info\_check(id Donates.Info\_Id%type)

return boolean

is

c int;

begin

c:=0;

select count(\*) into c from Informations where Info\_Id=id;

if c>0 then

return true;

else

return false;

end if;

end;

procedure donateInfo\_insert(s Donates.Serial%type,id Donates.Info\_Id%type,bg Donates.Blood\_Group%type,t Donates.Last\_time\_donated%type)

is

begin

if info\_check(id) then

insert into Donates values(s,id,bg,t);

else

dbms\_output.put\_line('Invalid Identity');

end if;

end;

procedure donateInfo\_delete(s Donates.Serial%type)

is

begin

delete from Donates where Serial=s;

exception

when no\_data\_found then

dbms\_output.put\_line('Invalid Serial);

when others then

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

end;

end donates\_dml;

***USER INTERFACE***

