Banking Information System

Scenario:-

ABC Bank Maintaining details of it's customers, such as

-Acount Details

-Day to day transactions(deposits & Withdrawals).

Functional Requirements:-

-Write DB script to add customers and to make transactions

(Deposits and Withdrawals).

Delivarables:-

-3NF

-Table Definitions with relations.

-Applying Integrity Constraints.

-Defining the reusable Components

-Handling Suitable Exceptions.

-Use necessary Packages,Procedures,Functions.

-Use Triggers.

1.CUSTOMER TABLE

CREATE TABLE CUS351(

ACC\_NO NUMBER(30) PRIMARY KEY,

CNAME VARCHAR2(10) NOT NULL,

CR\_BAL NUMBER(7) DEFAULT CHECK (CR\_BAL>=0)

);

2.TRANSACTION TABLE

CREATE TABLE TRANSAC351(

TRAN\_ID NUMBER(5) PRIMARY KEY,

ACC\_NO NUMBER REFERENCES CUS351(ACC\_NO),

TRAN\_TYPE CHAR(1) CHECK(UPPER(TRAN\_TYPE) IN ('W','D')),

AMOUNT NUMBER(7) NOT NULL,

TRAN\_DATE DATE

);

CREATE SEQUENCE TRAN\_SEQ;

CREATE SEQUENCE CUS\_SEQ START WITH 100;

INSERT INTO CUS351 VALUES (CUS\_SEQ.NEXTVAL,'AABB',0);

INSERT INTO CUS351 VALUES (CUS\_SEQ.NEXTVAL,'CCCC',0);

INSERT INTO CUS351 VALUES (CUS\_SEQ.NEXTVAL,'DDDD',0);

select \* from cus351;

select \* from transac351;

CREATE OR REPLACE TRIGGER TRI\_VAL BEFORE INSERT OR DELETE ON TRANSAC351 for each row

declare

v\_cnt number:= 0;

v\_val number;

begin

if inserting then

select cr\_bal into v\_val from cus351 where acc\_no = :new.acc\_no;

select count(acc\_no) into v\_cnt from transac351 where acc\_no = :new.acc\_no;

if lower(:new.tran\_type)='w' then

if v\_cnt=0 OR v\_val<:new.amount then

RAISE\_APPLICATION\_ERROR(-20001,'INSUFFICIENT BALANCE');

end if;

end if;

elsif deleting then

RAISE\_APPLICATION\_ERROR(-20003,'NOT ALLOWED');

end if;

end;

CREATE OR REPLACE TRIGGER TRG\_UPD AFTER INSERT ON TRANSAC351 for each row

Declare

v\_val number;

begin

select cr\_bal into v\_val from cus351 where acc\_no=:NEW.acc\_no;

if inserting then

if lower(:new.tran\_type)='w' and :new.amount<=v\_val then

v\_val:=v\_val-:new.amount;

else

v\_val:=v\_val+:new.amount;

end if;

update cus351 set cr\_bal=v\_val where acc\_no=:new.acc\_no;

end if;

end;

INSERT INTO TRANSAC351 VALUES(TRAN\_SEQ.NEXTVAL,100,'d',600,sysdate);

INSERT INTO TRANSAC351 VALUES(TRAN\_SEQ.NEXTVAL,101,'w',700,sysdate);

INSERT INTO TRANSAC351 VALUES(TRAN\_SEQ.NEXTVAL,102,'w',200,sysdate);

DELETE FROM TRANSAC351 WHERE TRAN\_ID = 2;

2. Creating tables:

Customer Table:

CREATE TABLE CL483\_CUST (ACNO NUMBER(10) PRIMARY KEY,

CNAME VARCHAR2(20) NOT NULL,

BALANCE NUMBER(20));

INSERT INTO CL483\_CUST VALUES(1112223331,'HARIKA',10000);

INSERT INTO CL483\_CUST VALUES(1112223332,'NANI',10000);

INSERT INTO CL483\_CUST VALUES(1112223333,'PRIYA',10000);

INSERT INTO CL483\_CUST VALUES(1112223334,' TEJA',10000);

INSERT INTO CL483\_CUST VALUES(1112223335,'SAHITI',10000);

INSERT INTO CL483\_CUST VALUES(1112223336,' DEEPU',10000);

CREATE SEQUENCE ST

INCREMENT BY 1

START WITH 1000

NOMAXVALUE;

Customer Details Table:

create table cl483\_t2(ACNO NUMBER(10) PRIMARY KEY,

STREET VARCHAR2(10),

CITY VARCHAR2(10),

STATE VARCHAR2(10),

PIN NUMBER(10));

INSERT INTO CL483\_t2 (1112223331,'RAILWAY COLONY','TIRUPATI','A.P',517501);

INSERT INTO CL483\_t2 VALUES(1112223332,'RAILWAY COLONY','TIRUPATI','A.P',517501);

INSERT INTO CL483\_t2 VALUES(1112223333,'K.T.ROAD',' TIRUPATI','A.P',517501);

INSERT INTO CL483\_t2 VALUES(1112223334,' K.T.ROAD','TIRUPATI','A.P',517501);

INSERT INTO CL483\_t2 VALUES(1112223335,' MARUTHI NAGAR','TIRUPATI','A.P',517501);

INSERT INTO CL483\_t2 VALUES(1112223336,' MARUTHI NAGAR','TIRUPATI','A.P',517501);

Transaction Table:

CREATE TABLE CL483\_TRAN(TID NUMBER(10),

TDATE DATE,

TRTP VARCHAR2(5),

ACNO NUMBER(10) REFERENCES CL483\_CUST(ACNO),

AMOUNT NUMBER(20) );

SELECT \* FROM CL483\_CUST;

SELECT \* FROM CL483\_T2;

SELECT \* FROM CL483\_TRAN;

Handling Transactions:

INSERT INTO CL483\_TRAN VALUES(ST.NEXTVAL,'&TDATE','&TRTP',&ACNO,&AMOUNT);

CREATE OR REPLACE TRIGGER MYTRIG483

AFTER INSERT ON CL483\_TRAN FOR EACH ROW

DECLARE

TR CL483\_CUST.BALANCE%TYPE;

BEGIN

SELECT BALANCE INTO TR FROM CL483\_CUST WHERE ACNO = :NEW.ACNO;

IF :NEW.TRTP='D' AND TR<:NEW.AMOUNT THEN

RAISE\_APPLICATION\_ERROR(-20009,'THERE IS NO SUFFICIENT BALANCE');

ELSIF :NEW.TRTP='D' AND TR>:NEW.AMOUNT THEN

UPDATE CL483\_CUST

SET BALANCE=TR-:NEW.AMOUNT WHERE ACNO=:NEW.ACNO;

ELSIF :NEW.TRTP='C' THEN

UPDATE CL483\_CUST SET BALANCE=TR+:NEW.AMOUNT WHERE ACNO=:NEW.ACNO;

END IF;

END;

RESULTS:

CASE 1:

INSERT INTO CL483\_TRAN VALUES(ST.NEXTVAL,'&TDATE','&TRTP',&ACNO,&AMOUNT);

old 1: INSERT INTO CL483\_TRAN VALUES(ST.NEXTVAL,'&TRTP',&ACNO,&AMOUNT)

new 1: INSERT INTO CL483\_TRAN VALUES(ST.NEXTVAL,'C ',1112223334,2000)

1 row created.

CASE 2:

INSERT INTO CL483\_TRAN VALUES(ST.NEXTVAL,'&TDATE','&TRTP',&ACNO,&AMOUNT);

old 1: INSERT INTO CL483\_TRAN VALUES(ST.NEXTVAL,'&TRTP',&ACNO,&AMOUNT)

new 1: INSERT INTO CL483\_TRAN VALUES(ST.NEXTVAL,'D',1112223335,4000)

1 row created.

CASE 3:

INSERT INTO CL483\_TRAN VALUES(ST.NEXTVAL,'&TDATE','&TRTP',&ACNO,&AMOUNT);

1112223333 D 11000

ERROR at line 1:

ORA-20009: THERE IS NO SUFFICIENT BALANCE

ORA-06512: at "ELP1019.MYTRIG483", line 8

ORA-04088: error during execution of trigger 'ELP1019.MYTRIG483'

3. create table customer(id number(5),name varchar2(10),amount number(10),ldate date,periodtype varchar2(1),repay date);

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

create table transaction (tid number(5),id number(5),capital number(10),int number(2),period number(2),intamt number(10),paid number(10),tdate date,due number(10));

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

alter table transaction add constraint fk\_tran foreign key(id) references customer(id);

-----------

create sequence tid start with 1000 increment by 1;

create sequence cid start with 1 increment by 1;

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

pcust FOR NEW CUSTOMER

create or replace procedure pcustomer(a varchar2,b number,c varchar2)

as

m number(2);

x date;

begin

if b<0 then raise\_application\_error(-20110,'------ENTER CORRECT AMOUNT------');

elsif c not in('q','m','y') then raise\_application\_error(-20111,'-----ENTER CORRECT PERIOD TYPE-----');

elsif C='m' OR C='m' then m:=1;

elsif C='q' OR C='q' then m:=3;

elsif C='y' OR C='y' then m:=12;

end if;

x:= add\_months(sysdate,m);

insert into customer values(cid.nextval,a,b,sysdate,c,x);

insert into transaction values(tid.nextval,cid.currval,b,2,0,0,0,sysdate,b);

end;

PTRAN FOR NEW CUSTOMER

SHOW ERRORS;

create or replace procedure ptransaction(a number,b number)

as

p number(10);

q number(2);

r number(2);

x number(10);

y number(10);

m date;

n number(10);

FK EXCEPTION;

PRAGMA EXCEPTION\_INIT(FK,-2291);

cursor c is select tdate,due from transaction where id=a order by tid desc;

k c%rowtype;

begin

if b<0 then raise\_application\_error(-20112,'----------ENTER CURRECT AMOUNT-------');

END IF;

open c;

fetch c into k;

close c;

p:=k.due;

q:=2;

m:=k.tdate;

r:=months\_between(sysdate,m);

x:=p\*q\*r/100;

y:=p+x-b;

if y<0 then raise\_application\_error(-20113,'----YOUR PENDING AMOUNT LESS THAN WHAT YOU ARE PAYING------');

END IF;

insert into transaction values(tid.nextval,a,p,q,r,x,b,sysdate,y);

EXCEPTION

WHEN FK THEN DBMS\_OUTPUT.PUT\_LINE('-----------ENTER VALID CUSTOMER ID------------');

end ;

EXECUTION

NEW CUSTOMER

BEGIN

PCUST('&ENTER\_NAME',&ENTER\_AMOUNT,'&ENTER\_PERIOD\_TYPE');

END;

/

select \* from customer2;

select \* from transaction;

===========

OLD CUSTOMER

==========

SET SERVEROUTPUT ON

BEGIN

PTRAN(&ENTER\_LOAN\_ID,&ENTER\_AMOUNT);

END;

/

select \* from customer2;

select \* from transaction;

4.

**CUSTOMER ORDER**

***SCENARIO:***

A customer visits a Super Market to place multiple orders and an order can include multiple

products.

***INITIAL INFORMATION:***

CUSTOMER ORDER (CustName, OrderNo,

ProdNo, ProdDesc, Qty, UnitPrice, CustAddress, DateOrdered)

***TABLE STRUCTURES:***

**CUSTOMER**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **CID(pk)** | **CNAME** | **A\_STREET** | **A\_CITY** | **A\_STATE** | **PHONE** | **E-MAIL** |
|  |  |  |  |  |  |  |

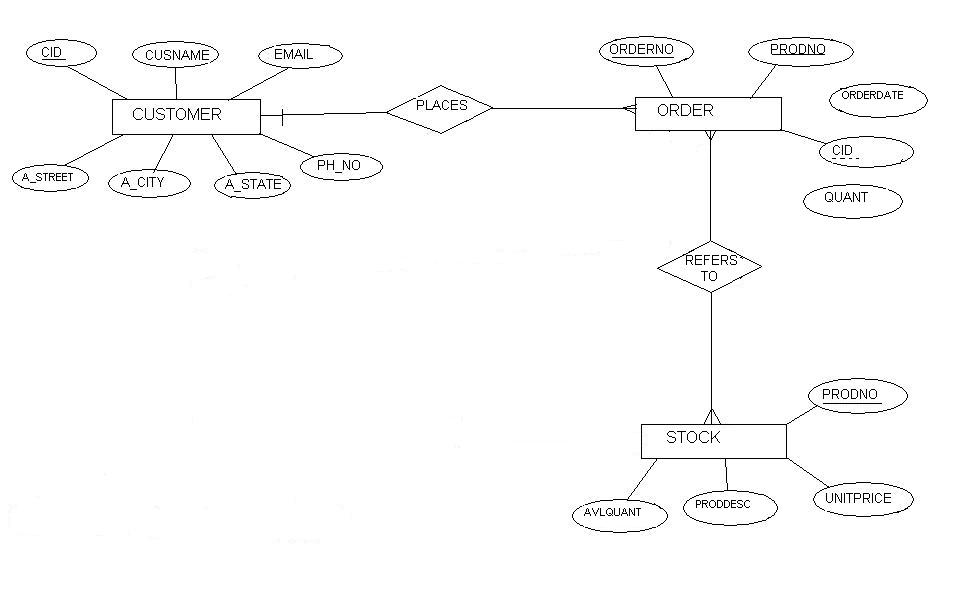
**ORDER**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ORDERNO (comp-pk)** | **PRODNO (comp-pk)** | **ORDERDATE** | **QUANT** | **CID(fk)** |
|  |  |  |  |  |

**STOCK**

|  |  |  |  |
| --- | --- | --- | --- |
| **PRODNO(pk)** | **PRODDESC** | **UNITPRICE** | **AVLQUANT** |
|  |  |  |  |

***ER DIAGRAM:***

******

***DDL SCRIPTS:***

create table customer\_364( CID number(5) PRIMARY KEY,CUSNAME varchar2(10) NOT NULL,A\_STREET varchar2(15),A\_CITY varchar2(15),A\_STATE varchar2(15),PH\_NO number(10) UNIQUE,EMAIL varchar2(25) UNIQUE);

create table order\_364( ORDERNO number(5),PRODNO number(5),ORDERDATE date NOT NULL,QUANT number(3) NOT NULL,CID number(5) REFERENCES customer\_364(CID),PRIMARY KEY (ORDERNO,PRODNO));

create table stock\_364 (PRODNO number(5) PRIMARY KEY ,PRODDESC varchar2(20) NOT NULL,UNITPRICE number(7,2) NOT NULL,AVLQUANT number(5));

***DML SCRIPTS:***

insert into customer\_364 values (1023,'PRASHANT','AUTONAGAR','VIZAG','AP',9885700351,'prashant\_kamsu@msat.com');

insert into customer\_364 values (1364,'PINTU','SHAWSTREET','BHUBANESHWAR','ORISSA',7207504527,'pintu\_shaw@msat.com');

insert into customer\_364 values (2361,'VISHESH','CHADHALANE','LUCKNOW','UP',7702053008,'vishesh\_chadha@msat.com');

insert into customer\_364 values (2849,'CHIRASMITA','SAHOOMARG','BHUBANESHWAR','ORISSA',8596421703,'chirasmita\_sahoo@msat.com');

insert into customer\_364 values (3265,'HARIKA','MADILAPALEM','VIZAG','AP',9640092139,'harika\_pusuluri@msat.com');

insert into customer\_364 values (5432,'PRACHI','CHOUSTREET','CHANDRAPUR','MAHARASHTRA',9491785365,'prachi\_chaudhari@msat.com');

insert into customer\_364 values (3456,'HINA','CHOULTRY','BHOPAL','MP',8886455330,'hina\_gokhe@msat.com');

insert into customer\_364 values (2341,'DEEPIKA','SRINAGAR','VIZAG','AP',8019481894,'deepika\_rapeti@msat.com');

insert into order\_364 values (8354, 100,'17-dec-99',10,1023);

insert into order\_364 values (8354, 101,'17-dec-99',5,1023);

insert into order\_364 values (8216,101,'21-aug-92',5,5432);

insert into order\_364 values (8934,234,'25-feb-96',5,2849);

insert into order\_364 values (8614,289,'09-jun-04',3,2361);

insert into order\_364 values (8922,365,'28-nov-92',4,3265);

insert into order\_364 values (8364,345,'31-jan-06',5,1023);

insert into stock\_364 values (100,'DAIRYMILK',20,40);

insert into stock\_364 values (101,'5STAR',20,40);

insert into stock\_364 values (234,'PERK',5,40);

insert into stock\_364 values (289,'KITKAT',5,40);

insert into stock\_364 values (365,'MILKYBAR',15,40);

insert into stock\_364 values (345,'BARONE',10,40);

***DISPLAY:***

select\* from customer\_364;

select\* from order\_364;

select\* from stock\_364;

***QUERIES:***

**>List the no of orders state wise**

select count(distinct orderno),a\_state from customer\_364 c,order\_364 o where c.cid=o.cid group by a\_state;

**>List the total price for each order**

Create view tot\_order\_price as select o.orderno,sum(s.unitprice\*o.quant) as total from order\_364 o,stock\_364 s where s.prodno=o.prodno group by orderno;

**>List the no of orders of a particular state**

select count(orderno) as no\_of\_orders from customer\_364 c,order\_364 o where c.cid=o.cid and a\_state='AP';

**>List the total amount spent on each product**

select s.prodno,sum(s.unitprice\*o.quant) as total from order\_364 o,stock\_364 s where s.prodno=o.prodno group by s.prodno;

**>Get the availability of a particular product**

Set serveroutput on;

Declare

v\_avl number(5);

prodno number(5);

Begin

Select avlquant into v\_avl from stock\_364 where prodno=&getprod;

Dbms\_output.put\_line(v\_avl);

Exception

When no\_data\_found then

Dbms\_output.put\_line('PRODUCT NOT PRESENT');

End;

**>Get the list of products ordered by a particular customer**

set serveroutput on;

Create or replace procedure cust\_order(p\_orderno in order\_364.orderno%type) is

begin

declare

Cursor prolist\_cur is

Select o.cid,o.prodno,s.proddesc from order\_364 o,stock\_364 s where orderno=p\_orderno and o.prodno=s.prodno;

Pro\_list prolist\_cur%rowtype;

begin

Open prolist\_cur;

loop

Fetch prolist\_cur into pro\_list;

exit when prolist\_cur%notfound;

Dbms\_output.put\_line(pro\_list.cid ||' ordered ' ||pro\_list.proddesc ||' having product no '||pro\_list.prodno);

end loop;

Close prolist\_cur;

End;

End cust\_order;

/

execute cust\_order(8354);

**>Create trigger when customer places an order**

create or replace trigger order\_t\_364

before insert or update on order\_364

for each row

declare

avl number;

begin

select avlquant into avl from stock\_364 where prodno=:new.prodno;

if (:new.quant> avl) then

raise\_application\_error(-20001,'product out of stock');

else

update stock\_364 set avlquant = avl-:new.quant where prodno=:new.prodno;

end if;

end;

/

show errors;

--insert into order\_364 values(8531,100,'02-feb-06',50,2341);

--select \* from order\_364;

--delete from order\_364 where orderno=8531;

5.

TO CREATE STOCK TABLE:

create table stock456(itemid number(3),itemname varchar2(20),stock number(3),minquantity number(3));

Insert into stock456 values(1,'Olay',40,5);

Insert into stock456 values(2,'Healthy white',40,5);

Insert into stock456 values(3,'Spinz',30,5);

Insert into stock456 values(4,'lakme facewash',35,5);

Insert into stock456 values(5,'Parachute',45,5);

Insert into stock456 values(6,'Clinic plus',50,5);

TO CREATE SALES TABLE:

create table sales456(itemid number(3) refernces stock456(itemid),itemname varchar2(20),transdate date,quantity number(3));

1.CREATE OR REPLACE TRIGGER new456

BEFORE INSERT OR DELETE ON SALES456 FOR EACH ROW

DECLARE

s\_quantity NUMBER(5);

BEGIN

SELECT stock INTO s\_quantity FROM STOCK456 WHERE itemid=:NEW.itemid;

IF INSERTING THEN

IF :NEW.Quantity>s\_quantity

THEN

RAISE\_APPLICATION\_ERROR(-20222,'NO STOCK');

ELSIF DELETING

THEN

RAISE\_APPLICATION\_ERROR(-22222,'NOT ALLOWED')

ENDIF;

END;

2.CREATE OR REPLACE TRIGGER newstock

AFTER INSERT ON SALES456 FOR EACH ROW

BEGIN

UPDATE STOCK456 SET stock=stock-:NEW.quantity WHERE

itemid=:NEW.itemid;

END;

3.CREATE OR REPLACE TRIGGER STOCK\_456

AFTER UPDATE ON STOCK456 FOR EACH ROW

BEGIN

IF (:stock-:new.quantity <=5) THEN

RAISE\_APPLICATION\_ERROR(-55555,'ORDER THE PRODUCT');

END IF;

ND;

PROCEDURE TO GET ITEM ID & NAME :

CREATE OR REPLACE PROCEDURE newstock456

(n\_itemid IN STOCK456.itemid%TYPE)

IS

V1 STOCK456.itemname%TYPE;

V2 STOCK456.stock%TYPE;

BEGIN

SELECT itemname,stock INTO V1,V2 FROM STOCK456 WHERE n\_itemid=itemid;

DBMS\_OUTPUT.PUT\_LINE(V1 || V2);

END newstock456;

SET SERVEROUTPUT ON;

EXECUTE newstock456(&n\_itemid);

6. STOCK-INFORMATION SYSTEM

* NORMALIZED TABLES:

STOCK TABLE:STOCKAVAIL

|  |  |  |  |
| --- | --- | --- | --- |
| ITEMCODE | ITEMNAME | ITEMPRICE | QUANTITY\_ AVAIL |
|  |  |  |  |

CREATE TABLE STOCKAVAIL(ITEMCODE NUMBER (10) PRIMARY KEY, ITEMNAME VARCHAR2(20),

ITEMPRICE NUMBER (10) NOT NULL, QUANTITY\_AVAIL NUMBER(10));

ORDER TABLE:ORDER1001

|  |  |  |  |
| --- | --- | --- | --- |
| ITEMCODE | ORDERID | QUANTITY\_REQ | PRICE |
|  |  |  |  |

CREATE TABLE ORDER1001(

ITEMCODE NUMBER(10) REFERENCES STOCK\_AVAIL1(ITEMCODE),

ORDERID NUMBER(10) PRIMARY KEY,

QUANTITY\_REQ NUMBER(10) NOT NULL

);

E-R DIAGRAM

STOCK

DEPENDS

**ORDER**

INSERTING VALUES INTO STOCKAVAIL:

INSERT INTO STOCKAVAIL VALUES (1101,'USHA\_FAN',1750,100);

INSERT INTO STOCKAVAIL VALUES (1102,'GOODDAY\_BIKIS',22,2000);

INSERT INTO STOCKAVAIL VALUES (1103,'SNICKERS',15,500);

INSERT INTO STOCKAVAIL VALUES (1104,'EVEREADY',8,1000);

INSERT INTO STOCKAVAIL VALUES (1105,'LAYS',20,200);

INSERT INTO STOCKAVAIL VALUES (1106,'AK\_47',20000,10);

INSERT INTO STOCKAVAIL VALUES (1107,'GRENADE',12000,5);

INSERT INTO STOCKAVAIL VALUES (1108,'UPMA\_RAVVA',60,100);

INSERT INTO STOCKAVAIL VALUES (1109,'TIMEX\_WATCH',2500,50);

INSERT INTO STOCKAVAIL VALUES (1110,'BOOST',110,200);

INSERT INTO STOCKAVAIL VALUES (1111,'VERTU',30000,2);

INSERT INTO STOCKAVAIL VALUES (1112,'ROLEX',70000,2);

INSERT INTO STOCKAVAIL VALUES (1113,'AASHA\_CHOC',1,2000);

INSERT INTO STOCKAVAIL VALUES (1114,'REEBOK',2500,50);

TO DISPLAY ABOVE TABLES:

SELECT \* FROM ORDER1001;

SELECT \* FROM STOCKAVAIL;

TRIGGER

CREATE OR REPLACE TRIGGER TRG\_SAL

AFTER INSERT OR DELETE ON ORDER1001

FOR EACH ROW

DECLARE

EXP EXCEPTION;

QUAN\_PRES STOCKAVAIL.QUANTITY\_AVAIL%TYPE;

BEGIN

SELECT QUANTITY\_AVAIL INTO QUAN\_PRES FROM STOCKAVAIL WHERE ITEMCODE = :NEW.ITEMCODE;

IF (:NEW.QUANTITY\_REQ < QUAN\_PRES) THEN

UPDATE STOCKAVAIL SET QUANTITY\_AVAIL = STOCKAVAIL.QUANTITY\_AVAIL- :NEW.QUANTITY\_REQ WHERE ITEMCODE =:NEW.ITEMCODE;

ELSIF UPDATING THEN

UPDATE STOCKAVAIL SET QUANTITY\_AVAIL = STOCKAVAIL.QUANTITY\_AVAIL + :NEW.QUANTITY\_REQ WHERE ITEMCODE =:NEW.ITEMCODE;

ELSE

RAISE EXP;

END IF;

EXCEPTION

WHEN EXP THEN

DBMS\_OUTPUT.PUT\_LINE('OUT OF STOCK');

END;

PL-SQL BLOCK

DECLARE

ODR ORDER1001%ROWTYPE;

BEGIN

ODR .ITEMCODE:=&ITEMCODE;

ODR.QUANTITY\_REQ :=&QO;

ODR.ORDERID:=&OID;

SELECT ITEMPRICE INTO ODR.PRICE FROM STOCKAVAIL WHERE ITEMCODE=ODR.ITEMCODE;

ODR.PRICE:=ODR.QUANTITY\_REQ\*ODR.PRICE;

INSERT INTO ORDER1001 VALUES ODR;

COMMIT;

END;

7.EMPLOYEE INFORMATION SYSTEM

CREATE SEQUENCE myseq458 START WITH 1 INCREMENT BY 1 NOMAXVALUE;

CREATE TYPE ph\_a AS VARRAY(2) OF NUMBER(10);

MY TABLES:

CREATE TABLE dept458(

deptno number(2) primary key,

dname char(20),

dlocation char(20) );

insert into dept458 values(10,'TRAINING','hyd');

insert into dept458 values(20,'IT','hyd');

insert into dept458 values(30,'IT','banglore');

insert into dept458 values(40,'MANAGEMENT','USA');

CREATE TABLE emp458(

eid number(10),

ename char(20),

gender char(1) default 'm',

des char(20),

hiredate date NOT NULL,

sal number,

deptno number(2) REFERENCES dept458(deptno),

phoneno ph\_a,

address varchar2(30) );

insert into emp458

values(myseq458.nextval,'anil','m','trainee','10-dec-90',10000,10,ph\_a(8008128926,8008954955),'vizag');

insert into emp458

values(myseq458.nextval,'chintan','m','learning officer','10-aug-89',20000,20,ph\_a(7008127926,8007954855),'hyd');

insert into emp458

values(myseq458.nextval,'pradeep','m','team leader','10-dec-89',30000,30,ph\_a(7008127926,8007954855),'hyd');

insert into emp458

values(myseq458.nextval,'harika','f','trainee','10-jun-90',10000,10,ph\_a(7008127926,8007954855),'mp');

insert into emp458

values(myseq458.nextval,'sony','f','manager','10-nov-87',40000,40,ph\_a(7008127926,8007954855),'mp');

drop table maintain;

CREATE TABLE maintain(

Ename char20),

id number,

oldDeptNo number(2),

newDeptNo number(2),

olddes char(20),

newdes char(20),

oldSal number,

newSal number,

date\_of\_modify localtimestamp );

TRIGGER TO MY EMP458 TABLE:

CREATE OR REPLACE TRIGGER mytrig458 after update on emp458 for each row

begin

insert into maintain (ename,eid,olddeptno,newdeptno,olddes,newdes,oldsal,newsal,date\_of\_modify)values (:new.ename,:new.eid,:old.DeptNo,:new.DeptNo,:old.des,:new.des,:old.Sal,:new.Sal,localtimestamp);

end;

/

show errors

select \* from emp458;

select \* from dept458;

update emp458 set deptno=20 where eid=1;

select \* from emp458;

select \* from maintain;

update emp458 set deptno=30 where eid=2;

select \* from emp458;

select \* from maintain;

CURSOR TO MY TABLE FOR UPDATING VALUES BY LOCKING THE ROWS:

SET SERVEROUTPUT ON;

DECLARE

CURSOR cur(e IN number) IS

SELECT \* FROM emp458 WHERE eid=e for update of deptno;

eno number;

dno number(2);

s number;

ds char(20);

BEGIN

eno :=&EnterEmployeeId;

dno :=&EnterTheNewDepartment;

open cur(eno);

s:=

case dno

when 10 then 10000

when 20 then 20000

when 30 then 30000

when 40 then 50000

end;

ds:=

case dno

when 10 then 'trainee'

when 20 then 'learing officer'

when 30 then 'team leader'

when 40 then 'manager'

end;

update emp458 set deptno=dno,sal=s,des=ds where eid=eno;

close cur;

commit;

end;

/

show errors;

commit;

select \* from emp458;

select \* from maintain;

PROCEDURE TO DISPLAY EMPLOYEE SALARY FOR THE GIVEN EMPLOYEE ID

CREATE OR REPLACE PROCEDURE MYPROC(eno IN NUMBER,V\_EID OUT EMP458.EID%TYPE,V\_ENAME OUT EMP458.Ename%TYPE,V\_SAL OUT EMP458.SAL)

IS

BEGIN

SELECT EID,ENAME,SAL INTO V\_EID,V\_ENAME,V\_SAL FROM EMP458 WHERE EID=eno;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

dbms\_output.put\_line('the employee details are not found PLEASE CHECK THE EMPLOYEE ID');

END NoDisplay;

/

show errors

VARIABLE V\_EID NUMBER

VARIABLE V\_ENAME CHAR(20)

VARIABLE V\_SAL NUMBER

EXECUTE NoDisplay(40,:V\_EID,:V\_ENAME,:V\_SAL)

PRINT V\_EID

PRINT V\_ENAME

PRINT V\_SAL

SOME QUERIES RELATED TO MY CASELET:

Q1..LIST THE EMPLOYEE WHO CHANGED DEPARTMENT MORE THAN TWICE?

MY SOLUTION: SELECT ENAME,EID FROM MAINTAIN WHERE GROUPBY deptno HAVING COUNT(\*)>2;

Q2..LIST THE EMPLOYEES WHO HAD CHANGED THEIR DEPATRMENT IN THE PARTICULAR MONTH?

MY SOLUTION: SELECT ENAME,EID,DATE\_OF\_MODIFY FROM MAINTAIN WHERE =

SUBSTR(TO\_CHARDATE\_OF\_MODIFY,'DD-MON-YY'),4,3)='&GIVENDATE';

note:

select \* from user\_constraints where table\_name='EMP458';

example:

sys\_coo5457

sys\_coo5458

alter table emp458 drop constraint SYS\_C005457

alter table emp458 drop constraint SYS\_C005458

8. CREATING DEPARTMENT TABLE:

create table departments385( deptno number(2) primary key,

deptname varchar2(10),

deptloc varchar2(10);

desc departments385;

CREATING EMPLOYEE TABLE

create table employees385(empnum number(5) primary key,

empname varchar2(10),

deptno number(5) references departments385(deptno),

empsal number(5));

desc employees385;

CREATING SALARY TABLE

create table salaries385(deptno number(5) references departments385 (deptno),

gross\_sal number(5),

basic\_sal number(5),

hra number(3),

da number(3),

pf number(3),

net\_sal number(5));

desc salaries385;

INSERTING VALUES INTO DEPARTMENT USING SQL:

insert into departments385 (deptno, deptname,deptloc)

values(1, 'JAVA', 'TEXAS');

insert into departments385 (deptno, deptname,deptloc)

values(2, 'ORACLE', 'LONDON');

insert into departments385 (deptno, deptname,deptloc)

values(3, 'MAINFRAMES', 'CALIFORNIA');

insert into departments385 (deptno, deptname,deptloc)

values(4, 'MICROSOFT', 'CHICAGO');

insert into departments385 (deptno, deptname,deptloc)

values(5, 'TESTING', 'TOKYO');

select \* from departments385;

DEPTNO DEPTNAME DEPTLOC

1 JAVA TEXAS

2 ORACLE LONDON

3 MAINFRAMES CALIFORNIA

4 MICROSOFT CHICAGO

5 TESTING TOKYO

INSERTING VALUES INTO SALARIES USING PL/SQL BLOCK:

set serveroutput on;

declare

v\_sal385 salaries385%rowtype;

greaterbasic exception;

begin

v\_sal385.deptno:=&getdeptno;

v\_sal385.basic\_sal:=&getbasic\_sal;

if v\_sal385.basic\_sal>6500 then

raise greaterbasic;

end if;

v\_sal385.da:=v\_sal385.basic\_sal\*(0.45);

v\_sal385.hra:=v\_sal385.basic\_sal\*(0.3);

v\_sal385.gross\_sal:=v\_sal385.da+v\_sal385.hra+v\_sal385.basic\_sal;

v\_sal385.pf:=v\_sal385.basic\_sal\*0.1;

v\_sal385.net\_sal:=v\_sal385.gross\_sal-v\_sal385.pf;

insert into salaries385 values v\_sal385;

exception

when greaterbasic then

dbms\_output.put\_line('not a valid basic');

end;

select \* from salaries385;

DEPTNO GROSS\_SAL BASIC\_SAL HRA DA PF NET\_SAL

1 11200 6400 1920 2880 640 10560

2 9975 5700 1710 2565 570 9405

3 8575 4900 1470 2205 490 8085

4 7263 4150 1245 1868 415 6848

5 5530 3160 948 1422 316 5214

CREATING SEQUENCE:

create sequence empsequence

start with 10730

increment by 1

maxvalue 10780

nocycle;

CREATING A PROCEDURE FOR INSERTING VALUES INTO EMPLOYEES TABLE:

CREATE OR REPLACE PROCEDURE procemp385

(p\_empname IN employees385.empname%TYPE,p\_deptno IN employees385.deptno%TYPE,p\_empsalary in employees385.empsal%type)

IS

BEGIN

INSERT INTO employees385(empnum,

empname,deptno,empsal)

VALUES (empsequence.NEXTVAL, p\_empname, p\_deptno,p\_empsalary);

END;

EXECUTING A PROCEDURE:

set serveroutput on;

begin

procemp385('varun',1,10560);

procemp385('ravali',2,9405);

procemp385('avinash',3,8085);

procemp385('navya',4,6848);

procemp385('deepika',5,5214);

end;

select \* from employees385;

EMPNUM EMPNAME DEPTNO EMPSAL

10731 varun 1 10560

10732 ravali 2 9405

10733 avinash 3 8085

10734 navya 4 6848

10735 deepika 5 5214

CREATING A PROCEDURE TO CALCULATE THE SALARY OF EMPLOYEES WHO HAVE WORKED FOR LESS NUMBER OF DAYS THAN THE ACTUAL WORKING DAYS

create or replace procedure sal\_gen(days in number,empid in number)

is

wrkdays number(2);

sal number(6);

dsal number(4);

begin

wrkdays:=to\_number(to\_char(last\_day(sysdate),'dd'));

select empsal into sal from employees385 e,salaries385 s where e.deptno=s.deptno and empnum=empid;

dsal:=sal/to\_number(to\_char(last\_day(sysdate),'dd'));

dbms\_output.put\_line('No.of working days='||wrkdays);

if days<wrkdays then

dbms\_output.put\_line('Monthly salary of'|| empid ||'='||days\*dsal);

end if;

end;

EXECUTING A PROCEDURE:

set serveroutput on;

execute sal\_gen(20,10731);

OUTPUT:

No.of working days=30

Monthly salary of10731=7040

PL/SQL procedure successfully completed.

9. caselet 09-student information system Abhinav Anishetty(107467)

CREATE TABLE STU\_467

(STU\_ID NUMBER(10),

S\_NAME VARCHAR2(20) NOT NULL,

COR\_ID NUMBER(10)NOT NULL,

DOJ DATE NOT NULL,

DOB DATE,CREDITS NUMBER(2),

CONSTRAINT PK1\_STU\_STUNUM PRIMARY KEY (STU\_ID),

CONSTRAINTS FK1\_STU\_CORID FOREIGN KEY (COR\_ID) REFERENCES COR\_467 (COR\_ID),

CONSTRAINT CK\_STU\_CRE CHECK (CREDITS BETWEEN 1 AND 10));

DECLARE

STU STU\_467%ROWTYPE;

TODATE DATE;

WRONGDOB EXCEPTION;

BEGIN

SELECT NVL(MAX(STU\_ID),0)+1 INTO STU.STU\_ID FROM STU\_467;

STU.S\_NAME :='&SNAME';

STU.COR\_ID :=&CID;

STU.DOJ :='&DOJ';

STU.DOB :='&DOB';

TODATE :=TO\_DATE(SYSDATE);

IF STU.DOB=TODATE THEN

RAISE WRONGDOB;

END IF;

STU.CREDITS:=&CREDITS;

INSERT INTO STU\_467 VALUES STU;

COMMIT;

EXCEPTION

WHEN WRONGDOB THEN

DBMS\_OUTPUT.PUT\_LINE('WRONG DOB GIVEN GIVE THE CORRECT ONE');

END;

CREATE TABLE COR\_467(COR\_ID NUMBER(10),C\_NAME VARCHAR2(20)NOT NULL,LOC VARCHAR2(20),DOC NUMBER(4),

CONSTRAINT PK1\_COR\_CORID PRIMARY KEY (COR\_ID));

DECLARE

COR COR\_467%ROWTYPE;

BEGIN

COR.COR\_ID :=&CID;

COR.C\_NAME :='&CNAME';

COR.LOC :='&LOCATION';

COR.DOC :=&DURATION;

INSERT INTO COR\_467 VALUES COR;

COMMIT;

END;

CREATING VIEWS:

CREATE OR REPLACE VIEW CHECKVIEW467 AS SELECT STU\_ID,S\_NAME,CREDITS FROM STU\_467;

CREATE OR REPLACE VIEW SIMPLEVIEW467 AS SELECT SUBSTR(DOB,-2) YEAR,COUNT(\*) NUMB FROM STU\_467 GROUP BY COR\_ID;

CREATE OR REPLACE VIEW CHKVIEW467 AS SELECT \* FROM STU\_467 WHERE COR\_ID=103 WITH CHECK OPTION;

EXCEPTIONS:

DECLARE

STU STU\_467%ROWTYPE;

TODATE DATE;

WRONGDOB EXCEPTION;

BEGIN

SELECT NVL(MAX(STU\_ID),0)+1 INTO STU.STU\_ID FROM STU\_467;

STU.S\_NAME :='&SNAME';

STU.COR\_ID :=&CID;

STU.DOJ :='&DOJ';

STU.DOB :='&DOB';

TODATE :=TO\_DATE(SYSDATE);

IF STU.DOB>=TODATE THEN

RAISE WRONGDOB;

END IF;

STU.CREDITS:=&CREDITS;

INSERT INTO STU\_467 VALUES STU;

COMMIT;

EXCEPTION

WHEN WRONGDOB THEN

DBMS\_OUTPUT.PUT\_LINE('WRONG DOB GIVEN GIVE THE CORRECT ONE');

END;

declare

s\_invalid\_class exception;

begin

update COR\_467

set c\_name='&c\_class\_desc'

where COR\_id=&c\_class\_number;

If sql%notfound then

raise s\_invalid\_class;

end if;

commit;

exception

when s\_invalid\_class then

dbms\_output.put\_line('no such class id');

end;

declare

s\_invalid\_class exception;

begin

delete from COR\_467

where C\_NAME='&CNAME';

If sql%notfound then

raise s\_invalid\_class;

end if;

commit;

exception

when s\_invalid\_class then

dbms\_output.put\_line('no such class');

end

CURSORS:

DECLARE

CURSOR CORCUR IS

SELECT \* FROM COR\_467;

CURSOR STUCUR(C\_CID IN NUMBER) IS

SELECT STU\_ID,S\_NAME,CREDITS,DOJ FROM STU\_467 WHERE COR\_ID=C\_CID;

CORREC CORCUR%ROWTYPE;

STUREC STUCUR%ROWTYPE;

BEGIN

OPEN CORCUR;

LOOP

FETCH CORCUR INTO CORREC;

EXIT WHEN CORCUR%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('STUDENT DETAILS BELONGS TO '||CORREC.COR\_ID);

OPEN STUCUR(CORREC.COR\_ID);

LOOP

FETCH STUCUR INTO STUREC;

EXIT WHEN STUCUR%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(STUREC.STU\_ID||STUREC.S\_NAME||STUREC.CREDITS);

END LOOP;

CLOSE STUCUR;

END LOOP;

CLOSE CORCUR;

END;

DECLARE

CURSOR cre\_cursor IS

SELECT \* from stu\_467

FOR UPDATE OF credits NOWAIT;

BEGIN

FOR st\_record IN cre\_cursor

LOOP

IF st\_record.credits<10 then

UPDATE stu\_467

SET credits = st\_record.credits +1

WHERE CURRENT OF cre\_cursor;

END IF;

END LOOP;

END;

PROCEDURE:

CREATE OR REPLACE PROCEDURE DEFAULT\_STU467

(CID IN COR\_467.COR\_ID%TYPE DEFAULT 200,NAME IN COR\_467.C\_NAME%TYPE DEFAULT 'C-LANG',LOC IN COR\_467.LOC%TYPE DEFAULT 'MSLW',duration in cor\_467.doc%type default 12)

IS

BEGIN

INSERT INTO COR\_467 VALUES(cid,name,loc,duration);

end;

BEGIN

DEFAULT(101,'JAVA','VALLABHI',12);

END;

CREATE OPR REPLACE PROCEDURE DELETE\_stu

(SID IN STU\_467.STU\_ID%TYPE)

IS

BEGIN

DELETE FROM STU\_467 WHERE STU\_ID=SID;

END;

BEGIN

DELETE\_stu(5);

END;

FUNCTION:

SHOW ERRORS;

CREATE OR REPLACE FUNCTION GET\_DOCOM(P\_ID STU\_467.STU\_ID%TYPE)

RETURN date

IS

P\_DATE date;

BEGIN

SELECT ADD\_MONTHS(DOJ,12) INTO P\_DATE FROM STU\_467 WHERE STU\_ID=P\_ID;

RETURN P\_DATE;

END;

VARIABLE V\_DATE VARCHAR2;

EXECUTE :V\_DATE :=GET\_DOCOM(1);

PRINT V\_DATE;

TRIGGERS:

CREATE OR REPLACE TRIGGER STOP\_TRIGGER

BEFORE INSERT OR DELETE OR UPDATE

ON STU\_467

BEGIN

IF TO\_CHAR(SYSDATE,'D')=1

THEN

RAISE\_APPLICATION\_ERROR(-20001,'TODAY IS SUNDAY U CANNOT DO ANY THING...');

END IF;

END;

CREATE OR REPLACE TRIGGER STOP\_TRIGGER

BEFORE INSERT OR DELETE OR UPDATE

ON COR\_467

BEGIN

IF TO\_CHAR(SYSDATE,'D')=1

THEN

RAISE\_APPLICATION\_ERROR(-20001,'TODAY IS SUNDAY U CANNOT DO ANY THING...');

END IF;

END;

CREATE OR REPLACE TRIGGER DEFAULT\_VALUES BEFORE INSERT ON COR\_467

FOR EACH ROW

WHEN (NEW.COR\_ID IS NOT NULL AND NEW.C\_NAME IS NULL)

BEGIN

IF :NEW.COR\_ID=101

THEN

:NEW.C\_NAME:='JAVA';

ELSIF :NEW.COR\_ID=102

THEN

:NEW.C\_NAME:='ORACLE';

ELSIF :NEW.COR\_ID=103

THEN

:NEW.C\_NAME:='TESTING';

END IF;

END;

10. COLLEGE INFORMATION SYSTEMS

create table student\_2470 (

sid number(10) primary key,

sname varchar2(20) not null,

doj date not null,

year number(2) not null,

cid number(10) references course\_2470(cid) on delete cascade

);

create table course\_2470(

cid number(10) primary key,

cname varchar2(20),

max\_strength number(3),

fee1 number(10),

fee2 number(10),

fee3 number(10)

);

create table fees\_2470 (

sid number(10) references student\_2470(sid) on delete cascade,

cid number(10),

t\_date date not null,

t\_id varchar2(10) primary key,

paid\_fees number(10)

);

insert into student\_2470 values(101,'ANAND','21-JAN-12',1,100);

insert into student\_2470 values(102,'RAMU','05-FEB-12',2,100);

insert into student\_2470 values(103,'AKHIL','20-JAN-12',3,100);

insert into student\_2470 values(201,'VICKY','27-JAN-12',1,200);

insert into student\_2470 values(202,'AJJU','21-JAN-12',2,200);

insert into student\_2470 values(203,'SURI','22-JAN-12',3,200);

insert into student\_2470 values(301,'VINNU','05-FEB-12',1,300);

insert into student\_2470 values(302,'ROHITH','02-FEB-12',2,300);

insert into student\_2470 values(303,'KITTU','28-JAN-12',3,300);

insert into course\_2470 values(100,'ece',5,8000,12000,20000);

insert into course\_2470 values(200,'csc',5,8000,12000,20000);

insert into course\_2470 values(300,'eee',5,8000,12000,20000);

insert into fees\_2470 values(101,100,'22-apr-12','ax0001',15000);

insert into fees\_2470 values(103,100,'25-apr-12','ax0002',5000);

insert into fees\_2470 values(301,300,'18-apr-12','ax0003',8000);

insert into fees\_2470 values(302,300,'12-apr-12','ax0004',12000);

insert into fees\_2470 values(203,200,'22-feb-12','ax0005',13000);

insert into fees\_2470 values(102,100,'22-mar-12','ax0006',20000);

declare

stu student\_2470%rowtype;

begin

stu.sid := &studentid;

stu.sname := '&name';

stu.doj := '&dateofjoining';

stu.year := &year;

stu.cid := &courseid;

insert into student\_2470 values stu;

commit;

end;

declare

cor course\_2470%rowtype;

begin

cor.cid := &courseid;

cor.cname := '&cname';

cor.maxstrength := &maximumstrength;

cor.fees1 := &fees1;

cor.fees2 := &fees2;

cor.fees3 := &fees3;

insert into course\_2470 values cor:

commit;

end;

view

< total fee to be paid by each student >

create view due\_student2470 as

select s.sid, s.sname, s.year, c.cid,c.cname,

(decode(s.year,1,c.fee1,2,fee2,3,fee3)) total\_fee

from student\_2470 s, course\_2470 c where

s.cid = c.cid;

< VIEW FOR TOTAL FEE, PAID FEE AND DUE FEE>

CREATE VIEW v\_duefees2470 AS

SELECT S.SID, S.SNAME, S.YEAR, C.CID,

(DECODE(S.YEAR,1,C.FEE1, 2, C.FEE2, 3, C.FEE3)) FEETOBEPAID,

(SELECT SUM(paid\_fees) FROM Fees\_2470 WHERE SID = S.SID

GROUP BY SID, YEAR ) FEEPAID,

(DECODE(S.YEAR,1,C.FEE1, 2, C.FEE2, 3, C.FEE3)) -

(SELECT SUM(paid\_fees) FROM Fees\_2470 WHERE SID = S.SID

GROUP BY SID, YEAR ) BALANCEFEE

FROM STUDENT\_2470 S , Course\_2470 C WHERE S.CID = C.CID;

exception

< details of particular transaction >

DECLARE

s\_tid fees\_2470.t\_id%type;

s\_amtpd fees\_2470.paid\_fees%type;

BEGIN

s\_tid:='&transaction\_ID';

select paid\_fees into s\_amtpd from fees\_2470 where t\_id=s\_tid;

DBMS\_OUTPUT.PUT\_LINE(' FEES PAID =' || s\_amtpd);

EXCEPTION

WHEN no\_data\_found THEN

DBMS\_OUTPUT.PUT\_LINE('INVALID TRANSACTION......');

END;

trigger

< tigger on fees table >

set serveroutput on;

create or replace trigger tri\_fees470

before insert or update or delete on fees\_2470

begin

if to\_char(sysdate,'dy')in('sun') then

if inserting then

raise\_application\_error(-20001,'not allowed to pay fees on sunday');

elsif updating then

raise\_application\_error(-20001,'not allowed to update fees on sunday');

elsif deleting then

raise\_application\_error(-20001,'not allowed to delete on sunday');

end if;

end if;

end;

procedure

< INSERTING INTO STUDENT TABLE >

create or replace procedure enter\_student470

( s\_id in student\_2470.sid%type,

s\_name in student\_2470.sname%type,

s\_doj in student\_2470.doj%type,

s\_year in student\_2470.year%type,

s\_cid in student\_2470.cid%type)

is

begin

insert into student\_2470(sid,sname,doj,year,cid) values

(s\_id,s\_name,s\_doj,s\_year,s\_cid);

end enter\_student470;

passing parameters :

begin

enter\_student470(&studentid,'&studentname','&dateofjoining',&year,&courseid);

end;

function

< amount paid by each student >

set serveroutput on;

create or replace function get\_details470

(id fees\_2470.sid%type)

return number

is

amount\_paid fees\_2470.pfees%type;

begin

select pfees into amount\_paid from fees\_2470 where sid=id;

return amount\_paid;

end get\_details470;

variable amt\_paid number

execute :amt\_paid := get\_details470(&idno)

print amt\_paid

cursor

< retrieve details of particular student >

declare

cursor stu470\_cur(stu\_id number) is

select sid,sname,cid,cname,year, total\_fee from due\_student2470 where sid=stu\_id;

stu\_rec student\_2470%rowtype;

s\_idno student\_2470%type;

begin

s\_idno := &sid;

open stu470\_cur(s\_idno);

loop

fetch stu470\_cur into stu\_rec;

exit when stu470\_cur%notfound;

dbms\_output.put\_line(stu\_rec.sid||'-'||stu\_rec.sname||'-'||stu\_rec.cid||'-'||stu\_rec.year||'-'||stu\_rec.total\_fee);

end loop;

close stu470\_cur;

end;

< due list of students >

set serveroutput on;

declare

cursor curr\_2470 is

select v\_duefees2470.sid,v\_duefees2470.sname from v\_duefees2470 join v\_due470

on v\_duefees2470.sid=v\_due470.sid where sysdate < deadline and balancefee >0;

due\_rec curr\_2470%rowtype;

begin

open curr\_2470;

fetch curr\_2470 into due\_rec;

dbms\_output.put\_line(due\_rec.sid||','||due\_rec.sname);

close curr\_2470;

end;

11. --TRAINING INSTITUTE

create table inst471(

c\_id number(3) primary key,

cname varchar2(10),

shift varchar2(1),

timings varchar2(10),

class varchar2(2),

strength number(2),

fee number(5),

doj varchar2(15)

);

insert into inst471 values(101,'java','m','6am-8am','c1','0',1000,'16-may-12');

insert into inst471 values(102,'c','m','6am-8am','c2','0',2000,'25-may-12');

insert into inst471 values(103,'.net','m','8am-10am','c1','0',2000,'10-jun-12');

insert into inst471 values(104,'c++','m','9am-11am','c2','0',3000,'01-jun-12');

insert into inst471 values(105,'c#','e','4pm-6pm','c1','0',3000,'30-may-12');

insert into inst471 values(106,'oracle','e','6pm-8pm','c2','0',4000,'16-jun-12');

insert into inst471 values(107,'sql','e','8pm-10pm','c1','0',4000,'19-jun-12');

create table std\_reg(

s\_id number(3) primary key,

sname varchar2(10),

c\_id number(3) references inst471(c\_id),

amt\_paid number(5)

);

create or replace procedure trainproc(

v\_sid number,

v\_sname varchar2,

v\_cid number,

v\_amt number

)

is

v\_s number(2);

v\_fee number(5);

begin

select strength,fee into v\_s,v\_fee from inst471 where c\_id=v\_cid;

if(v\_s<20) AND (v\_amt=v\_fee) then

insert into std\_reg values(v\_sid,v\_sname,v\_cid,v\_amt);

v\_s:=v\_s+1;

update inst471 set strength=v\_s where c\_id=v\_cid;

else

dbms\_output.put\_line('be in queue');

end if;

end;

alter table inst471 add(

tdate varchar2(15)

);

update inst471 set tdate='1-jun-12' where cname='&cname';

create or replace trigger triginst

after update of tdate on inst471

for each row

declare

v\_doj varchar2(15);

v\_cid varchar2(15);

v\_strength number(2);

v\_sid number(3);

begin

select doj,c\_id,strength into v\_doj,v\_cid,v\_strength from inst471 where tdate=:=new.tdate;

if((:new.tdate)-(v\_doj)=30) then

delete from std\_reg where c\_id=v\_cid;

v\_strength=v\_strength-1;

update inst471 set strength=v\_strength where c\_id=v\_cid;

else

dbms\_output.put\_line('allow student to attend session');

end if;

end;

12. ~~~~~~~~~~HOSPITAL REGISTRATION~~~~~~~~

TABLE DIVISION322 ( DOCTOR DIVISIONS/DEPARTMENTS ) :-

create table division322

(

divid number(5) ,

divname varchar2(20),

docid number(5),

fee number(5),

primary key(divid,docid)

);

INSERTION IN DIVISION322 :-

insert into division322 values(1,'Cardiologist',101,500);

insert into division322 values(1,'Cardiologist',102,500);

insert into division322 values(2,'Physician',103,700);

insert into division322 values(3,'Neurologist',104,600);

insert into division322 values(3,'Neurologist',105,600);

TABLE DOCTOR322 ( DOCTOR DETAILS ):-

create table doctor322

(

docid number(5) primary key,

docname varchar2(20),

divid number(5) ,

patient number(5) check(patient <= 10)

);

INSERTION IN DOCTOR322 :-

insert into doctor322 values(101,'Kapil',1,0);

insert into doctor322 values(102,'Dhawal',1,0);

insert into doctor322 values(103,'Devendra',2,0);

insert into doctor322 values(104,'Ankit',3,0);

insert into doctor322 values(105,'Rahul',3,0);

TABLE PATIENTDTL322 ( FOR PATIENT DETAILS ):-

create table patientdtl322

(

pid number(5) primary key,

pname varchar2(20),

docid number(5) references doctor322(docid) on delete set null,

feepaid number(5),

appdate date

);

SEQUENCE ( GENERATE PATIENT ID ):-

create sequence seq\_pid322

start with 1000

increment by 1;

TRIGGER :-

create or replace trigger patient\_doctor322

before insert on patientdtl322

for each row

declare

vdt number(5);

begin

update doctor322 set patient = patient + 1 where docid = :new.docid;

end;

PROCEDURE (FOR PATIENT REGISTRATION) :-

create or replace procedure appointment322( pname in varchar2, docid in number)

is

v\_docid patientdtl322.docid%type;

v\_appdate patientdtl322.appdate%type;

v\_pid patientdtl322.pid%type;

v\_feepaid patientdtl322.feepaid%type;

vdate date;

app date;

d date;

vdt number;

begin

v\_docid := docid;

select sysdate into v\_appdate from dual;

select seq\_pid322.nextval into v\_pid from dual;

select fee into V\_feepaid from division322 where docid = v\_docid;

select appdate into app from patientdtl322

where pid = (select max(pid) from patientdtl322 ) ;

if ( to\_char(app) <> to\_char(v\_appdate) ) then

update doctor322 set patient = 0 where docid = v\_docid;

end if;

select patient into vdt from doctor322 where docid = v\_docid;

if ( vdt <= 10) then

insert into patientdtl322 values (v\_pid, pname, v\_docid, V\_feepaid,v\_appdate);

else

dbms\_output.put\_line('Doctor has too many appointments');

end if;

end appointment322;

13. STUDENT TABLE::

CREATE TABLE STUD107421(STID NUMBER(4) PRIMARY KEY,

SNAME VARCHAR2(15)

);

subject TABLE::

CREATE TABLE SUBJ107421(YEAR NUMBER(2) PRIMARY KEY,

SUB1 VARCHAR2(10),

SUB2 VARCHAR2(10),

SUB3 VARCHAR2(10));

MARKS TABLE::

CREATE TABLE MARKS107421(STID NUMBER(5),

YEAR NUMBER(2) CHECK(YEAR BETWEEN 1 AND 4),

M1 NUMBER(3) CHECK(M1 BETWEEN 1 AND 100),

M2 NUMBER(3) CHECK(M2 BETWEEN 1 AND 100),

M3 NUMBER(3) CHECK(M3 BETWEEN 1 AND 100),

PRIMARY KEY(STID,YEAR));

INSERTING VALUES INTO TABLES

SUBJECTS

YR SUB1 SUB2 SUB3

INSERT INTO SUBJ107421 VALUES(1,'XX','YY','ZZ');

INSERT INTO SUBJ107421 VALUES(2,'AA','BB','CC');

INSERT INTO SUBJ107421 VALUES(3,'DD','MM','NN');

INSERT INTO SUBJ107421 VALUES(4,'JJ','KK','LL');

STUDENTS

STID SNAME

INSERT INTO STUD107421 VALUES(2001,'INDU');

INSERT INTO STUD107421 VALUES(2002,'SANTHU');

INSERT INTO STUD107421 VALUES(2003,'NANDU');

MARKS

STID YR M1 M2 M3

INSERT INTO MARKS107421 VALUES(2001,1,50,60,80);

INSERT INTO MARKS107421 VALUES(2001,2,60,60,80);

INSERT INTO MARKS107421 VALUES(2001,3,53,60,65);

INSERT INTO MARKS107421 VALUES(2001,4,60,60,80);

INSERT INTO MARKS107421 VALUES(2002,1,75,98,89);

INSERT INTO MARKS107421 VALUES(2002,2,63,67,80);

INSERT INTO MARKS107421 VALUES(2002,3,78,60,95);

INSERT INTO MARKS107421 VALUES(2002,4,85,90,80);

INSERT INTO MARKS107421 VALUES(2003,1,66,58,65);

INSERT INTO MARKS107421 VALUES(2003,2,78,77,80);

INSERT INTO MARKS107421 VALUES(2003,3,38,85,99);

INSERT INTO MARKS107421 VALUES(2003,4,85,90,100);

QUERIES

1. CREATE A PROCEDURE TO FIND MARKS MEMO OF EACH STUDENT OF EACH YEAR WITH GRADES..

SELECT \* FROM SUBJ107421;

SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE PROCE421

IS

VSUB1 VARCHAR2(10);

VSUB2 VARCHAR2(10);

VSUB3 VARCHAR2(10);

VM1 NUMBER(3);

VM2 NUMBER(3);

VM3 NUMBER(3);

VSTID NUMBER(4);

VYEAR NUMBER(1);

MSTID NUMBER(4);

MSNAME VARCHAR2(10);

TOTALMARKS NUMBER(5);

BEGIN

VSTID:=&STID;

VYEAR:=&YEAR;

SELECT STID ,SNAME INTO MSTID,MSNAME FROM STUD107421 WHERE STID=VSTID;

SELECT M1,M2,M3 INTO VM1,VM2,VM3 FROM MARKS107421 WHERE STID=VSTID AND YEAR=VYEAR;

SELECT SUB1,SUB2,SUB3 INTO VSUB1,VSUB2,VSUB3 FROM SUBJ107421 WHERE YEAR=VYEAR;

TOTALMARKS:=VM1+VM2+VM3;

DBMS\_OUTPUT.PUT\_LINE(MSTID||' '||MSNAME);

DBMS\_OUTPUT.PUT\_LINE('YEAR'||' '||VSUB1||' '||VSUB2||' '||VSUB3);

DBMS\_OUTPUT.PUT\_LINE(VYEAR||' '||VM1||' '||VM2||' '||VM3||' '||TOTALMARKS);

IF(TOTALMARKS)/3 > 75 THEN

DBMS\_OUTPUT.PUT\_LINE('GRADE'||' '||'A' );

ELSIF(TOTALMARKS)/3 BETWEEN 60 AND 75 THEN

DBMS\_OUTPUT.PUT\_LINE('GRADE'||' '||'B' );

ELSE

DBMS\_OUTPUT.PUT\_LINE('GRADE'||' '||'C' );

END IF;

END;

EXECUTION

EXECUTE PROCE421;

2.CREATE PL/SQL BLOCK FOR CMM....

SET SERVEROUTPUT ON;

DECLARE

Cursor STUD\_CUR(PSTID IN NUMBER) is

Select STID,SNAME from STUD107421 where stid=pstid;

Cursor MARKS\_CUR(vSTID IN number) is

Select YEAR,M1,M2,M3 from MARKS107421 where STID = vSTID;

CURSOR SUBJ\_CUR(PYEAR IN NUMBER) IS

SELECT SUB1,SUB2,SUB3 FROM SUBJ107421 WHERE YEAR=PYEAR;

STUD\_REC STUD\_CUR%rowtype;

MARKS\_REC MARKS\_CUR%rowtype;

SUBJ\_REC SUBJ\_CUR%ROWTYPE;

PSTID STUD107421.STID%TYPE;

TOTAL NUMBER(3);

BEGIN

PSTID:=&STID;

OPEN STUD\_CUR(PSTID);

LOOP

FETCH STUD\_CUR into STUD\_REC;

EXIT when STUD\_CUR%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE( STUD\_REC.SNAME ||' '|| STUD\_REC.STID);

OPEN MARKS\_CUR(STUD\_REC.STID);

LOOP

FETCH MARKS\_CUR into MARKS\_REC;

EXIT when MARKS\_CUR%NOTFOUND;

OPEN SUBJ\_CUR(MARKS\_REC.YEAR);

LOOP

FETCH SUBJ\_CUR INTO SUBJ\_REC;

EXIT WHEN SUBJ\_CUR%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('YEAR'||' '||SUBJ\_REC.SUB1||' '||SUBJ\_REC.SUB2||' '||SUBJ\_REC.SUB3||' '||'TOTAL');

END LOOP;

CLOSE SUBJ\_CUR;

TOTAL:=MARKS\_REC.M1+MARKS\_REC.M2+MARKS\_REC.M3;

DBMS\_OUTPUT.PUT\_LINE(MARKS\_REC.YEAR||' '|| MARKS\_REC.M1||' '||MARKS\_REC.M2||' '||MARKS\_REC.M3||' '||TOTAL);

IF(TOTAL/3)>75 THEN

DBMS\_OUTPUT.PUT\_LINE('GRADE'||' '||'A');

ELSIF(TOTAL/3) BETWEEN 60 AND 75 THEN

DBMS\_OUTPUT.PUT\_LINE('GRADE'||' '||'B');

ELSE

DBMS\_OUTPUT.PUT\_LINE('GRADE'||' '||'C');

END IF;

END LOOP;

CLOSE MARKS\_CUR;

END LOOP;

CLOSE STUD\_CUR;

END;

3) FINDING ABOUT STUDENT RESULT(PASS OR FAIL)

CREATE OR REPLACE PROCEDURE PF107421 (VSTID IN NUMBER, VYEAR IN NUMBER, VRES OUT VARCHAR2)

IS

VM1 NUMBER(3);

VM2 NUMBER(3);

VM3 NUMBER(3);

BEGIN

SELECT M1,M2,M3 INTO VM1,VM2,VM3 FROM MARKS107421 WHERE STID=VSTID AND YEAR=VYEAR;

IF VM1>35 AND VM2>35 AND VM3>35 THEN

DBMS\_OUTPUT.PUT\_LINE('STUDENT PASSED');

ELSE

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

END IF;

END;

BIND VARIABLES:

VARIABLE ARES VARCHAR2(25)

EXECUTE PF107421(2001,2,:ARES)

SELECT \* FROM STUD107421;

SELECT \* FROM SUBJ107421;

SELECT \* FROM MARKS107421;

SELECT SUB1,SUB2,SUB3,M1,M2,M3 FROM SUBJ107421 S,MARKS107421 M WHERE S.YEAR=M.YEAR;

SELECT M1,M2,M3 FROM MARKS107421 WHERE YEAR=1 AND STID=2001;

SELECT SNAME,M1,M2,M3 FROM STUD107421 S,MARKS107421 M WHERE S.STID=M.STID;

15. Pharmacy

customer

cid-pk

mid-pk,fk(inventory)

qty

date

inventory

mid-pk,fk(customer)

mname

qty

supplier

sid

mid(cpk)

ph.no

customer table

CREATE TABLE customer107347(

cid number(10) ,

mid number(10) ,

qpurchased number(10) not null,

dop date,

CONSTRAINT fk\_midcust107347 FOREIGN KEY (mid) REFERENCES inventory107347(mid),

CONSTRAINT pk\_cidmidcust107347 PRIMARY KEY (cid,mid)

);

inventory table

CREATE TABLE inventory107347(

mid number(10),

stock number(10),

CONSTRAINT fk\_midinv107347 FOREIGN KEY (mid) REFERENCES supplier107347(mid),

CONSTRAINT pk\_midinv107347 PRIMARY KEY (mid)

);

supplier table

CREATE TABLE supplier107347(

sid number(10) unique not null ,

mid number(10),

phone number(10) not null,

CONSTRAINT pk\_sidmidinv107347 PRIMARY KEY (mid)

);

INSERT INTO SUPPLIER107347 VALUES(4,3,987654331);

INSERT INTO SUPPLIER107347 VALUES(2,4,987654351);

INSERT INTO SUPPLIER107347 VALUES(3,5,987654371);

INSERT INTO INVENTORY107347 VALUES(3,100);

INSERT INTO INVENTORY107347 VALUES(4,800);

INSERT INTO INVENTORY107347 VALUES(5,50);

create or replace trigger H107347 after insert on CUSTOMER107347 for each row

begin

update INVENTORY107347 set STOCK=STOCK-:new.QPURCHASED where Mid=:new.Mid;

end;

INSERT INTO CUSTOMER107347 VALUES(2,5,40,SYSDATE);

INSERT INTO CUSTOMER107347 VALUES(3,3,90,SYSDATE);

create or replace view caseview as

select i.mid,i.stock,s.sid,s.phone from inventory107347 i,supplier107347 s where i.mid=s.mid and i.stock<=10 and TO\_CHAR(SYSDATE,'HH24:MI') BETWEEN '12:00' AND '21:00';

create or replace procedure caseprocedure (qty in number)

is

cursor casecursor is select \* from caseview;

casevar casecursor%rowtype;

qtyvar number(3);

begin

qtyvar:=qty;

open casecursor;

loop

fetch casecursor into casevar;

exit when casecursor %notfound;

DBMS\_OUTPUT.PUT\_LINE('mid:'||casevar.mid||' stock:'||casevar.stock||' sid:'||casevar.sid||' phone: '||casevar.phone||' quantity: '||qtyvar);

end loop;

close casecursor;

end caseprocedure;

16. stock107434 TABLE (HOLDS THE VALUES OF STOCK PRESENT)

create table stock107434 (product\_id varchar2(3) primary key, product\_name varchar2(20), qty number(5),unit\_price number(6));

insert into stock107434 values ('E01','2.0L TFSI I4', 1000,60000);

insert into stock107434 values ('E02','2.0L TDI DIESEL I4',1000,80000);

insert into stock107434 values ('E03','5.4L SUPERCHARGED V8',100,100000);

insert into stock107434 values ('B01','4-DOOR SEDAN',1000,60000);

insert into stock107434 values ('B02','4-DOOR SEDAN 2',1000,60000);

insert into stock107434 values ('B03','2-DOOR ROADSTER', 100,70000);

insert into stock107434 values ('S01','FRONT DRIVING SEAT',2100,10000);

insert into stock107434 values ('S02','FRONT PASSENGER SEAT',2100,10000);

insert into stock107434 values ('S03','REAR SEAT',2000,30000);

insert into stock107434 values ('T01','RADIAL',10500,5000);

OBJECT DATA TYPE

CREATE TYPE SEATS\_TYPE AS OBJECT (FRONT\_SEAT\_DRIVE VARCHAR2(3), FRONT\_SEAT

VARCHAR2(3), REAR\_SEAT VARCHAR2(3));

CARS107434 (HOLDS THE VALUES OF THE MODELS OF CARS)

CREATE TABLE CARS107434(CAR\_ID NUMBER(1) PRIMARY KEY, CAR\_NAME VARCHAR2(20),ENGINE\_TYPE VARCHAR2(3),BODY\_TYPE VARCHAR2(3),SEATS SEATS\_TYPE, DATE\_OF\_ORDER DATE, QTY NUMBER(5));

INSERT INTO CARS107434 VALUES(1,'VIPER PETROL','E01','B01',SEATS\_TYPE('S01','S02','S03'),SYSDATE,0);

INSERT INTO CARS107434 VALUES (2,'VIPER DIESEL','E02','B02',SEATS\_TYPE('S01','S02','S03'),SYSDATE,0);

INSERT INTO CARS107434 VALUES (3,'VENOM MCLAREN','E03','B03',SEATS\_TYPE('S01','S02',''),SYSDATE,0);

RECORD107434 TABLE (HOLDS THE VALUES OF MANUFACTURED CARS)

CREATE TABLE RECORD107434 (R\_NO NUMBER(4) PRIMARY KEY, CAR\_ID NUMBER(1) REFERENCES CARS107434(CAR\_ID),QTY NUMBER(5),DATE\_OF\_ORDER DATE);

RE\_ORDER107434 TABLE (HOLDS THE VALUES FOR RE ORDER IF STOCK IS LOW)

CREATE TABLE RE\_ORDER107434 (ORDER\_ID NUMBER(4) PRIMARY KEY,PRODUCT\_ID VARCHAR2(3) REFERENCES STOCK107434(PRODUCT\_ID), QTY NUMBER (5), AMOUNT NUMBER(9));

insert into RE\_ORDER107434 values (1,'E01', 500,3000000);

insert into RE\_ORDER107434 values (2,'E02', 500,4000000);

insert into RE\_ORDER107434 values (3,'E03', 100,10000000);

insert into RE\_ORDER107434 values (4,'B01', 500,30000000);

insert into RE\_ORDER107434 values (5,'B02', 500,30000000);

insert into RE\_ORDER107434 values (6,'B03', 100,7000000);

insert into RE\_ORDER107434 values (7,'S01', 600,6000000);

insert into RE\_ORDER107434 values (8,'S02', 600,6000000);

insert into RE\_ORDER107434 values (9,'S03', 500,15000000);

insert into RE\_ORDER107434 values (10,'T01',1100,5500000);

ALTER TABLE RE\_ORDER107434 ADD (NUMBER\_OF\_TIMES NUMBER(6), DATE\_OF\_REORDER DATE);

UPDATE RE\_ORDER107434 SET NUMBER\_OF\_TIMES=0;

PROCEDURE CARS434 (IT UPDATES THE ORDER IN CARS TABLE THE ID AND THE QUANTITY OF CARS TO BE MANUFACTURED AND INSERT THE VALUE IN RECORD IF THE VALUES HAVING CAR ID AND DATE\_OF\_ORDER AS SYSDATE ARE PRESENT IN THE RECORD TABLE IT'LL UPDATE THE TABLE)

CREATE OR REPLACE PROCEDURE CARS434 (CID IN OUT CARS107434.CAR\_ID%TYPE,Q IN OUT CARS107434.QTY%TYPE)

IS

VCAR CARS107434%ROWTYPE;

VREC RECORD107434%ROWTYPE;

RNO NUMBER;

C NUMBER;

BEGIN

IF CID in (1,2,3) THEN

UPDATE CARS107434 SET QTY=Q, DATE\_OF\_ORDER=SYSDATE WHERE CAR\_ID = CID;

SELECT \* INTO VCAR FROM CARS107434 WHERE CAR\_ID=CID AND TO\_CHAR(DATE\_OF\_ORDER)=TO\_CHAR(SYSDATE) ;

DBMS\_OUTPUT.PUT\_LINE('AN ORDER FOR QUANTITY OF '||VCAR.QTY||' OF '||VCAR.CAR\_NAME||' HAVING ID '||VCAR.CAR\_ID||' HAS BEEN REGISTERED');

SELECT NVL(MAX(R\_NO),0)+1 INTO RNO FROM RECORD107434;

VREC.R\_NO:=RNO;

VREC.CAR\_ID:=VCAR.CAR\_ID;

VREC.QTY:=VCAR.QTY;

VREC.DATE\_OF\_ORDER:=VCAR.DATE\_OF\_ORDER;

SELECT NVL(MAX(CAR\_ID),0)+10 INTO C FROM RECORD107434 WHERE CAR\_ID=CID AND TO\_CHAR(DATE\_OF\_ORDER)=TO\_CHAR(SYSDATE);

IF C=10 THEN

INSERT INTO RECORD107434 VALUES VREC;

ELSIF C IN (11,12,13) THEN

UPDATE RECORD107434 SET QTY = Q WHERE CAR\_ID=VCAR.CAR\_ID AND TO\_CHAR(DATE\_OF\_ORDER)=TO\_CHAR(SYSDATE);

END IF;

ELSE

DBMS\_OUTPUT.PUT\_LINE('WE ARE RIGHT NOW MANUFACTURING ONLY 3 MODELS HAVING ID 1 2 AND 3');

END IF;

END CARS434;

VAR CARID NUMBER;

VAR QUANT NUMBER;

BEGIN

:CARID:=&CAR\_ID;

:QUANT:=&QUANTITY;

END;

EXECUTE CARS434(:CARID,:QUANT);

PROCEDURE REORDER434 (IF THE STOCK OF CAR 1 AND 2 IS LESS THAN 500 OR THE STOCK OF CAR 3 IS LESS THAN 100 IT WILL UPDATE THE STOCK BY ADDING 500 AND 100 RESPECTIVELY AND WILL UPDATE THE RE ORDER TABLE)

CREATE OR REPLACE PROCEDURE REORDER434 (CID IN RECORD107434.CAR\_ID%TYPE)

IS

Q NUMBER;

T NUMBER;

BEGIN

IF CID = 1 THEN

SELECT QTY INTO Q FROM STOCK107434 WHERE PRODUCT\_ID = 'E01';

ELSIF CID = 2 THEN

SELECT QTY INTO Q FROM STOCK107434 WHERE PRODUCT\_ID = 'E02';

ELSIF CID = 3 THEN

SELECT QTY INTO Q FROM STOCK107434 WHERE PRODUCT\_ID = 'E03';

END IF;

SELECT UNIT\_PRICE INTO T FROM STOCK107434 WHERE PRODUCT\_ID='T01';

IF CID=1 AND Q < 500 THEN

UPDATE STOCK107434 SET QTY=QTY+500 WHERE PRODUCT\_ID IN('E01','B01','S01','S02','S03');

UPDATE STOCK107434 SET QTY=QTY+2500 WHERE PRODUCT\_ID IN ('T01');

UPDATE RE\_ORDER107434 SET QTY=2500, AMOUNT=T\*2500, number\_of\_times=number\_of\_times+1,

DATE\_OF\_REORDER=SYSDATE WHERE PRODUCT\_ID IN ('T01');

FOR UPDRE434 IN (SELECT \* FROM STOCK107434 WHERE PRODUCT\_ID IN('E01','B01','S01','S02','S03','T01'))

LOOP

UPDATE RE\_ORDER107434 SET QTY=500,AMOUNT=UPDRE434.UNIT\_PRICE\*500 ,number\_of\_times=number\_of\_times+1,

DATE\_OF\_REORDER=SYSDATE WHERE PRODUCT\_ID IN ('E01','B01','S01','S02','S03');

END LOOP;

UPDATE RE\_ORDER107434 SET number\_of\_times=number\_of\_times+1

WHERE PRODUCT\_ID IN ('E01','B01','S01','S02','S03');

END IF;

IF CID=2 AND Q < 500 THEN

UPDATE STOCK107434 SET QTY=QTY+500 WHERE PRODUCT\_ID IN('E02','B02','S01','S02','S03');

UPDATE STOCK107434 SET QTY=QTY+2500 WHERE PRODUCT\_ID IN ('T01');

UPDATE RE\_ORDER107434 SET QTY=2500, AMOUNT=T\*2500, number\_of\_times=number\_of\_times+1,

DATE\_OF\_REORDER=SYSDATE WHERE PRODUCT\_ID IN ('T01');

UPDATE RE\_ORDER107434 SET number\_of\_times=number\_of\_times+1

WHERE PRODUCT\_ID IN ('E02','B02','S01','S02','S03');

FOR UPDRE434 IN (SELECT \* FROM STOCK107434 WHERE PRODUCT\_ID IN('E02','B02','S01','S02','S03'))

LOOP

UPDATE RE\_ORDER107434 SET QTY=500,AMOUNT=UPDRE434.UNIT\_PRICE\*500,

DATE\_OF\_REORDER=SYSDATE WHERE PRODUCT\_ID IN ('E02','B02','S01','S02','S03');

END LOOP;

END IF;

IF CID=3 AND Q < 100 THEN

UPDATE STOCK107434 SET QTY=QTY+100 WHERE PRODUCT\_ID IN ('E03','B03', 'S01', 'S02');

UPDATE STOCK107434 SET QTY=QTY+500 WHERE PRODUCT\_ID IN ('T01');

UPDATE RE\_ORDER107434 SET QTY=500 , AMOUNT=T\*500,number\_of\_times=number\_of\_times+1,

DATE\_OF\_REORDER=SYSDATE WHERE PRODUCT\_ID IN ('T01');

UPDATE RE\_ORDER107434 SET number\_of\_times=number\_of\_times+1

WHERE PRODUCT\_ID IN ('E03','B03','S01','S02');

FOR UPDRE434 IN (SELECT \* FROM STOCK107434 WHERE PRODUCT\_ID IN ('E03','B03', 'S01', 'S02'))

LOOP

UPDATE RE\_ORDER107434 SET QTY=100 , AMOUNT=UPDRE434.UNIT\_PRICE\*100, DATE\_OF\_REORDER=SYSDATE WHERE PRODUCT\_ID IN ('E03','B03','S01','S02');

END LOOP;

END IF;

DBMS\_OUTPUT.PUT\_LINE('STOCK IS UPDATED NOW YOU CAN GIVE ORDER');

END REORDER434;

/

show error;

TRIGGER (ON INSERTING OR UPDATING ON THE RECORD TABLE THE QTY WILL BE SUBTRACTED FROM STOCK TABLE DEPENDING UPON THE MODEL OF THE CAR IF THE STOCK IS LESS IT WILL PROMPT TO EXECUTE RE ORDER PROCEDURE)

CREATE OR REPLACE TRIGGER UPD107434

AFTER INSERT OR UPDATE ON RECORD107434

FOR EACH ROW

DECLARE

Q NUMBER;

T NUMBER;

BEGIN

IF INSERTING THEN

IF :NEW.CAR\_ID = 1 THEN

SELECT QTY INTO Q FROM STOCK107434 WHERE PRODUCT\_ID = 'E01';

ELSIF :NEW.CAR\_ID = 2 THEN

SELECT QTY INTO Q FROM STOCK107434 WHERE PRODUCT\_ID = 'E02';

ELSIF :NEW.CAR\_ID = 3 THEN

SELECT QTY INTO Q FROM STOCK107434 WHERE PRODUCT\_ID = 'E03';

END IF;

SELECT UNIT\_PRICE INTO T FROM STOCK107434 WHERE PRODUCT\_ID='T01';

IF Q < 500 AND :NEW.CAR\_ID IN(1,2) THEN

REORDER434(:NEW.CAR\_ID);

ELSIF Q < 100 AND :NEW.CAR\_ID=3 THEN

REORDER434(:NEW.CAR\_ID);

ELSE

UPDATE STOCK107434 SET QTY=QTY-(5\*:NEW.QTY) WHERE PRODUCT\_ID = 'T01';

IF :NEW.CAR\_ID = 1 THEN

UPDATE STOCK107434 SET QTY=QTY-:NEW.QTY WHERE PRODUCT\_ID IN ('E01','B01','S01','S02','S03');

ELSIF :NEW.CAR\_ID = 2 THEN

UPDATE STOCK107434 SET QTY=QTY-:NEW.QTY WHERE PRODUCT\_ID IN ('E02','B02','S01','S02','S03');

ELSE

UPDATE STOCK107434 SET QTY=QTY-:NEW.QTY WHERE PRODUCT\_ID IN ('E03','B03','S01','S02');

END IF;

END IF;

ELSIF UPDATING THEN

IF :NEW.CAR\_ID = 1 THEN

SELECT QTY INTO Q FROM STOCK107434 WHERE PRODUCT\_ID = 'E01';

ELSIF :NEW.CAR\_ID = 2 THEN

SELECT QTY INTO Q FROM STOCK107434 WHERE PRODUCT\_ID = 'E02';

ELSIF :NEW.CAR\_ID = 3 THEN

SELECT QTY INTO Q FROM STOCK107434 WHERE PRODUCT\_ID = 'E03';

END IF;

IF (:OLD.QTY>:NEW.QTY) THEN

UPDATE STOCK107434 SET QTY=QTY+(5\*(:OLD.QTY-:NEW.QTY)) WHERE PRODUCT\_ID = 'T01' ;

IF :OLD.CAR\_ID =1 THEN

UPDATE STOCK107434 SET QTY=QTY+(:OLD.QTY-:NEW.QTY) WHERE PRODUCT\_ID IN ('E01','B01','S01','S02','S03');

ELSIF :OLD.CAR\_ID=2 THEN

UPDATE STOCK107434 SET QTY=QTY+(:OLD.QTY-:NEW.QTY) WHERE PRODUCT\_ID IN ('E02','B02','S01','S02','S03');

ELSE

UPDATE STOCK107434 SET QTY=QTY+(:OLD.QTY-:NEW.QTY) WHERE PRODUCT\_ID IN ('E03','B03','S01','S02') ;

END IF;

ELSIF (:OLD.QTY<:NEW.QTY) THEN

IF (:NEW.QTY-:OLD.QTY) < 500 AND :NEW.CAR\_ID IN(1,2) THEN

REORDER434(:NEW.CAR\_ID);

ELSIF (:NEW.QTY-:OLD.QTY) < 100 AND :NEW.CAR\_ID=3 THEN

REORDER434(:NEW.CAR\_ID);

ELSE

UPDATE STOCK107434 SET QTY=QTY-(5\*(:NEW.QTY-:OLD.QTY)) WHERE PRODUCT\_ID = 'T01' ;

IF :OLD.CAR\_ID =1 THEN

UPDATE STOCK107434 SET QTY=QTY-(:NEW.QTY-:OLD.QTY) WHERE PRODUCT\_ID IN('E01','B01','S01','S02','S03');

ELSIF :OLD.CAR\_ID=2 THEN

UPDATE STOCK107434 SET QTY=QTY-(:NEW.QTY-:OLD.QTY) WHERE PRODUCT\_ID IN ('E02','B02','S01','S02','S03');

ELSE

UPDATE STOCK107434 SET QTY=QTY-(:NEW.QTY-:OLD.QTY) WHERE PRODUCT\_ID IN ('E03','B03','S01','S02') ;

END IF;

END IF;

END IF;

END IF;

END;

/

SHOW ERRORS;