**PACKAGE:**

create or replace package proj2 as

procedure select\_emp(emp\_details OUT SYS\_REFCURSOR);

procedure select\_cust(cust\_details OUT SYS\_REFCURSOR);

procedure select\_dis(dis\_details OUT SYS\_REFCURSOR);

procedure select\_products(products\_details OUT SYS\_REFCURSOR);

procedure select\_suppliers(suppliers\_details OUT SYS\_REFCURSOR);

procedure select\_supplies(supplies\_details OUT SYS\_REFCURSOR);

procedure select\_purchases(purchases\_details OUT SYS\_REFCURSOR);

procedure select\_logs(logs\_details OUT SYS\_REFCURSOR);

function purchase\_saving(pno in NUMBER)

return NUMBER;

procedure monthly\_sale\_activities(empno IN char,p\_cur OUT sys\_refcursor);

procedure add\_customer(cid IN char,name IN varchar, telephone# IN char);

procedure add\_purchase(eid IN char,p\_id IN char,cid IN char, pur\_qty

IN char);

procedure delete\_purchase(p\_pur# IN char);

end proj2;

**PACKAGE BODY:**

create or replace package body proj2 as

procedure select\_emp(

emp\_details OUT SYS\_REFCURSOR)

IS

BEGIN

open emp\_details for select \* from employees;

END select\_emp;

procedure select\_cust(

cust\_details OUT SYS\_REFCURSOR)

IS

BEGIN

open cust\_details for select \* from customers;

END select\_cust;

procedure select\_dis(

dis\_details OUT SYS\_REFCURSOR)

IS

BEGIN

open dis\_details for select \* from discounts;

END select\_dis;

procedure select\_products(

products\_details OUT SYS\_REFCURSOR)

IS

BEGIN

open products\_details for select \* from products;

END select\_products;

procedure select\_suppliers(

suppliers\_details OUT SYS\_REFCURSOR)

IS

BEGIN

open suppliers\_details for select \* from suppliers ;

END select\_suppliers;

procedure select\_supplies(

supplies\_details OUT SYS\_REFCURSOR)

IS

BEGIN

open supplies\_details for select \* from supplies;

END select\_supplies;

procedure select\_purchases(

purchases\_details OUT SYS\_REFCURSOR)

IS

BEGIN

open purchases\_details for select \* from purchases;

END select\_purchases;

procedure select\_logs(

logs\_details OUT SYS\_REFCURSOR)

IS

BEGIN

open logs\_details for select \* from logs;

END select\_logs;

**Function to get total saving of a particular purchase.**

function purchase\_saving(pno in NUMBER)

return NUMBER

IS savings number(7,2);

BEGIN

select s.original\_price\*p.qty-total\_price

into savings

from purchases p join products s on p.pid=s.pid

where pur#=pno;

return (savings);

END;

**Procedure to get total monthly sales of an employee**

procedure monthly\_sale\_activities(

empno IN char,

p\_cur OUT sys\_refcursor)

AS

BEGIN

open p\_cur for

select

p.eid,e.name,to\_char(p.ptime,&#39;Mon&#39;),to\_char(p.ptime,&#39;YYYY&#39;),count(p.eid)

as total\_sales,sum(p.qty) as total\_qty,sum(p.total\_price) as total\_amt

from purchases p

inner join employees e on e.eid=p.eid

where p.eid=empno

group by p.eid,e.name,to\_char(p.ptime,&#39;Mon&#39;),to\_char(p.ptime,&#39;YYYY&#39;);

END monthly\_sale\_activities;

procedure add\_customer(cid IN char,name IN varchar, telephone# IN

char)

AS

BEGIN

insert into customers values (cid,name,telephone#,1,sysdate);

END add\_customer;

procedure add\_purchase(eid IN char,p\_id IN char,cid IN char, pur\_qty

IN char)

IS

varTotal number(7,2);

pqoh number(5);

pthreshold number(4);

ssid char(2);

cursor c1 is

select qoh from products where pid=p\_id;

cursor c2 is

select qoh\_threshold from products where pid=p\_id;

cursor c3 is

select sid from supplies where pid=p\_id;

BEGIN

open c1;

fetch c1 into pqoh;

close c1;

open c2;

fetch c2 into pthreshold;

close c2;

open c3;

fetch c3 into ssid;

close c3;

if pqoh&lt;pur\_qty then

dbms\_output.put\_line(&#39;Insufficient Quantity in stock&#39;);

else

select p.original\_price\*pur\_qty\*(1-d.discnt\_rate) into varTotal

from products p join discounts d on p.discnt\_category=d.discnt\_category

where p.pid=p\_id;

insert into purchases values (seq\_pur.nextval,eid,p\_id,cid,pur\_qty,sysdate,varTotal);

if pqoh-pur\_qty &lt; pthreshold then

dbms\_output.put\_line(&#39;the current qoh of the product is below the required threshold

and new supply is required&#39;);

insert into supplies

values(seq\_sup.nextval,p\_id,ssid,sysdate,pthreshold-(pqoh- pur\_qty)+10+1);

end if;

end if;

END add\_purchase;

procedure delete\_purchase(p\_pur# IN char)

IS

BEGIN

delete from purchases where pur#=p\_pur#;

END delete\_purchase;

END proj2;

**TRIGGERS:**

**Trigger to add new entry to logs after inserting new customer**

create or replace trigger one

after insert

on customers

for each row

declare

uname varchar2(30);

BEGIN

select user into uname from dual;

insert into logs values(seq\_log.nextval, uname ,'INSERT',sysdate,'Customers',:new.cid);

END one;

**Trigger to add entry to logs after updating customers last visit date and number of visits.**

create or replace trigger two

after update

on customers

for each row

declare

uname varchar2(30);

BEGIN

select user into uname from dual;

insert into logs values(seq\_log.nextval, uname ,'update',sysdate,'Customers',:new.cid);

end two;

**trigger to add new entry to logs after inserting a tuple to purchases.**

create or replace trigger three

after insert

on purchases

for each row

declare

uname varchar2(30);

BEGIN

select user into uname from dual;

insert into logs values(seq\_log.nextval, uname ,'INSERT',sysdate,'PURCHASES',:new.pur#);

end three;

**trigger to add new entry to logs after updating qoh of products**

create or replace trigger four

after update

on products

for each row

declare

uname varchar2(30);

BEGIN

select user into uname from dual;

insert into logs values(seq\_log.nextval, uname ,'update',sysdate,'Products',:new.pid);

end four;

**trigger to add new entry to logs after ordering new supply**

create or replace trigger five

after insert

on supplies

for each row

declare

uname varchar2(30);

BEGIN

select user into uname from dual;

insert into logs values(seq\_log.nextval, uname ,'INSERT',sysdate,'SUPPLIES',:new.sup#);

end five;

**trigger after updating quantity**

create or replace trigger seven

after insert

on purchases

for each row

BEGIN

update products set qoh=qoh-:new.qty where pid=:new.pid;

end seven;

**trigger for updating qoh from supplies**

create or replace trigger sevenc

after insert

on supplies

for each row

BEGIN

update products set qoh=qoh+:new.quantity where pid=:new.pid;

end sevenc;

**trigger to print updated qoh**

create or replace trigger sevend

after update

on products

for each row

BEGIN

dbms\_output.put\_line('new qoh= ' || :new.qoh);

end sevend;

**trigger to update last visit date and total visits of customers.**

create or replace trigger sevenfinal

after insert

on purchases

for each row

declare

last date;

cursor c1 is

select last\_visit\_date from customers where cid=:new.cid;

begin

open c1;

fetch c1 into last;

close c1;

update customers set last\_visit\_date=sysdate where cid=:new.cid;

if to\_char(last,'MM/DD/YYYY') <> to\_char(sysdate,'MM/DD/YYYY') then

update customers set visits\_made=visits\_made+1 where cid=:new.cid;

end if;

end sevenfinal;

**trigger to update visits made and last visit date after deleting purchase entry**

create or replace trigger del

AFTER DELETE

ON purchases

for each row

declare

last date;

cursor c1 is

select last\_visit\_date from customers where cid=:old.cid;

begin

open c1;

fetch c1 into last;

close c1;

update products

set qoh=qoh+:old.qty

where pid=:old.pid;

update customers set last\_visit\_date=sysdate where cid=:old.cid;

if to\_char(last,'MM/DD/YYYY') <> to\_char(sysdate,'MM/DD/YYYY') then

update customers set visits\_made=visits\_made+1 where cid=:old.cid;

end if;

end;

**trigger to add new entry to logs after deleting a purchase.**

create or replace trigger delp

after delete

on purchases

for each row

declare

uname varchar2(30);

BEGIN

select user into uname from dual;

insert into logs values(seq\_log.nextval, uname ,'delete',sysdate,'Purchases',:old.pur#);

end delp;

**SEQUENCES-**

**Sequence for purchases**

create SEQUENCE seq\_pur

MINVALUE 111111

START WITH 111111

INCREMENT BY 1

CACHE 10

**Sequence for supplies**

create SEQUENCE seq\_sup

MINVALUE 1111

START WITH 1111

INCREMENT BY 1

CACHE 10

**Sequence for logs.**

create SEQUENCE seq\_log

MINVALUE 11111

START WITH 11111

INCREMENT BY 1

CACHE 10