DIGITAL ASSIGNMENT-2 PL/SQL

REG NO. :- 18MCA0207 **NAME :- SAHIL GANDHE**

```
create table customer
  (
   cus_code number(5) constraints c_pk primary key,
  cus_fname char(20),
  cus_lname char(20),
  cus_balance number(6,2));
```

```
SQL> create table customer
  2     (
  3       cus_code number(5) constraints c_pk primary key,
  4      cus_fname char(20),
  5      cus_lname char(20),
  6      cus_balance number(6,2));

Table created.

SQL>
SQL> desc customer
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUS_CODE                                  NOT NULL NUMBER(5)
 CUS_FNAME                                          CHAR(20)
 CUS_LNAME                                          CHAR(20)
 CUS_BALANCE                                        NUMBER(6,2)
```

```
create table invoice
  (inv_no number(5) constraints i_pk primary key,
  cus_code number(5),
  inv_date date,
  inv_amount number(6,2));
```

```
SQL> create table invoice
  2      (inv_no number(5) constraints i_pk primary key,
  3      cus_code number(5),
  4      inv_date date,
  5      inv_amount number(6,2));

Table created.

SQL> desc invoice;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 INV_NO                                    NOT NULL NUMBER(5)
 CUS_CODE                                           NUMBER(5)
 INV_DATE                                           DATE
 INV_AMOUNT                                         NUMBER(6,2)
```

```
create table line
  ( lnv_no number(5),
```

line_no number(5),
p_code number(5),
line_units number(5),
line_price number(6,2),
constraints l_pk primary key(lnv_no,line_no));

```
SQL> create table line
  2      ( lnv_no number(5),
  3      line_no number(5),
  4      p_code number(5),
  5      line_units number(5),
  6      line_price number(6,2),
  7      constraints l_pk primary key(lnv_no,line_no));

Table created.

SQL> desc line;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 LNV_NO                                    NOT NULL NUMBER(5)
 LINE_NO                                   NOT NULL NUMBER(5)
 P_CODE                                             NUMBER(5)
 LINE_UNITS                                         NUMBER(5)
 LINE_PRICE                                         NUMBER(6,2)
```

> create table product
(p_code number(5) constraints p_pk primary key,
p_desc char(10),
p_qoh integer,
p_min integer,
p_price number(6,2),
v_code number(5));

```
SQL> create table product
  2      ( p_code number(5) constraints p_pk primary key,
  3      p_desc char(10),
  4      p_qoh integer,
  5      p_min integer,
  6      p_price number(6,2),
  7      v_code number(5));

Table created.

SQL> desc product;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 P_CODE                                    NOT NULL NUMBER(5)
 P_DESC                                             CHAR(10)
 P_QOH                                              NUMBER(38)
 P_MIN                                              NUMBER(38)
 P_PRICE                                            NUMBER(6,2)
 V_CODE                                             NUMBER(5)
```

create table vendor
(v_code number(5) constraint v_pk primary key,

v_name char(10),
v_contact number(10));

```
SQL> create table vendor
  2      ( v_code number(5) constraint v_pk primary key,
  3      v_name char(10),
  4      v_contact number(10));

Table created.

SQL> desc vendor;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 V_CODE                                    NOT NULL NUMBER(5)
 V_NAME                                             CHAR(10)
 V_CONTACT                                          NUMBER(10)
```

FOREIGN KEY:--

alter table invoice add constraints i_fk foreign key(cus_code) references customer on delete cascade initially deferred deferrable;

```
SQL> alter table invoice add constraints i_fk foreign key(cus_code) references customer on delete cascade initially defe
rred deferrable;

Table altered.
```

alter table line add constraints l_fk1 foreign key(lnv_no) references invoice on delete cascade initially deferred deferrable;

alter table line add constraints l_fk2 foreign key(p_code) references product on delete cascade initially deferred deferrable;

alter table product add constraints p_fk foreign key(v_code) references vendor on delete cascade initially deferred deferrable;

```
SQL> alter table line add constraints l_fk2 foreign key(p_code) references product on delete cascade initially deferred deferrable;

Table altered.

SQL>
SQL> alter table product add constraints p_fk foreign key(v_code) references vendor on delete cascade initially deferred deferrable;

Table altered.

SQL> alter table line add constraints l_fk1 foreign key(lnv_no) references invoice on delete cascade initially deferred deferrable;

Table altered.
```

1) . **Write a procedure to add a new customer to the CUSTOMER table. Use the following values in the new record: <1002, 'Rauthor', 'Peter', 0.00>.**
Run a query to see if the record has been added

```
create or replace procedure customer_insert(c_code in customer.cus_code%type,c_fname in
customer.cus_fname%type,c_lname in customer.cus_lname%type,c_bal in
customer.cus_balance%type) is
      begin
        insert into customer values(c_code,c_fname,c_lname,c_bal);
        commit;
      end;
```

```
SQL> create or replace procedure customer_insert(c_code in customer.cus_code%type,c_fname in customer.cus_fname%type,c_lname in customer.
cus_lname%type,c_bal in customer.cus_balance%type) is
  2     begin
  3       insert into customer values(c_code,c_fname,c_lname,c_bal);
  4       commit;
  5     end;
  6  /

Procedure created.
```

```
SQL> execute customer_insert(1002,'Rauthor','Peter',0.00);

PL/SQL procedure successfully completed.

SQL> select * from customer;

  CUS_CODE CUS_FNAME            CUS_LNAME            CUS_BALANCE
---------- -------------------- -------------------- -----------
      1002 Rauthor              Peter                          0
```

2) **. Write a procedure to add a new invoice record to the INVOICE table. Use the following values in the new record: <8006, 1000, '30-APR-16', 301.72>.**
Run a query to see if the record has been added.

```
create or replace procedure invoice_insert(i_no number,c_code number,i_date date,i_amount
number) is
      begin
        insert into invoice values(i_no,c_code,i_date,i_amount);
        commit;
      end;
```

```
SQL> create or replace procedure invoice_insert(i_no number,c_code number,i_date date,i_amount number) is
  2         begin
  3           insert into invoice values(i_no,c_code,i_date,i_amount);
  4           commit;
  5         end;
  6  /

Procedure created.
```

```
SQL> execute invoice_insert(8006,1000,'30-APR-16',301.72);

PL/SQL procedure successfully completed.

SQL> select * from invoice;

    INV_NO   CUS_CODE INV_DATE  INV_AMOUNT
---------- ---------- --------- ----------
      8006       1000 30-APR-16     301.72
```

3. Write a PL/SQL function to compute purchase made by a given customer for a particular invoice. Test the function in another function to compute the total purchase made by a customer.

```
create or replace function com_purchase(to_purchase in number)
        return number is
         total_purchase number;
        begin
       select inv_amount into total_purchase from invoice
       where inv_no=to_purchase;
       return total_purchase;
       end;

      /
```

```
SQL> create or replace function com_purchase(to_purchase in number)
  2            return number is
  3             total_purchase number;
  4            begin
  5           select inv_amount into total_purchase from invoice
  6           where inv_no=to_purchase;
  7           return total_purchase;
  8           end;
  9  /

Function created.
```

Testing of com_purchase in another function:-

```
declare
inv_no number:=&inv_no;
answer number;
function compute_purchase(cus_purchase in number)
   return number is
     total number;
     begin
inv_no:=com_purchase(inv_no);
dbms_output.put_line('inv_amount is'||inv_no);
select sum(inv_amount) into total from invoice
 where cus_code=cus_purchase;
dbms_output.put_line('total purchase made by customer is'||total);
return total;
end ;
begin
answer:=compute_purchase(1001);
dbms_output.put_line(answer);
end;
/
```

```
SQL> declare
  2  inv_no number:=&inv_no;
  3  answer number;
  4  function compute_purchase(cus_purchase in number)
  5      return number is
  6         total number;
  7         begin
  8  inv_no:=com_purchase(inv_no);
  9  dbms_output.put_line('inv_amount is'||inv_no);
 10  select sum(inv_amount) into total from invoice
 11   where cus_code=cus_purchase;
 12  dbms_output.put_line('total purchase made by customer is'||total);
 13  return total;
 14  end ;
 15  begin
 16  answer:=compute_purchase(1001);
 17  dbms_output.put_line(answer);
 18  end;
 19  /
Enter value for inv_no: 8006
old   2: inv_no number:=&inv_no;
new   2: inv_no number:=8006;

PL/SQL procedure successfully completed.
```

```
SQL>   declare
  2  inv_no number:=&inv_no;
  3  answer number;
  4  function compute_purchase(cus_purchase in number)
  5      return number is
  6         total number;
  7         begin
  8  dbms_output.put_line('Enter inv_no');
  9  inv_no:=com_purchase(inv_no);
 10  dbms_output.put_line('inv_amount is'||inv_no);
 11  select sum(inv_amount) into total from invoice
 12   where cus_code=cus_purchase;
 13  dbms_output.put_line('total purchase made by customer is'||total);
 14  return total;
 15  end ;
 16  begin
 17  answer:=compute_purchase(1001);
 18  dbms_output.put_line(answer);
 19  end;
 20  /
Enter value for inv_no: 8006
old   2: inv_no number:=&inv_no;
new   2: inv_no number:=8006;
Enter inv_no
inv_amount is340.45
total purchase made by customer is840.95
840.95

PL/SQL procedure successfully completed.

SQL>
```

4. Write a procedure to delete an invoice, giving the invoice number as a parameter. Test the procedure by deleting invoices 8005 and 8006.

```
create or replace procedure prc_del(d number) AS
  BEGIN
   IF d is not NULL THEN
  DELETE FROM INVOICE WHERE inv_no=d;
   END IF;
   END;
   /

execute prc_del(8085);
```

```
SQL> create or replace procedure prc_del(d number) AS
  2     BEGIN
  3      IF d is not NULL THEN
  4     DELETE FROM INVOICE WHERE inv_no=d;
  5      END IF;
  6      END;
  7      /

Procedure created.

SQL> execute prc_del(8085);

PL/SQL procedure successfully completed.
```

execute prc_del(8086);

```
SQL> execute prc_del(8086);

PL/SQL procedure successfully completed.
```

5. Write a procedure to display the INV_SUBTOTAL, INV_TAX, and INV_TOTAL. The procedure takes the invoice number as a parameter. The INV_SUBTOTAL is the sum of the LINE_TOTAL amounts for the invoice, the INV_TAX is the product of the INV_SUBTOTAL and the tax rate (8 percent), and the INV_TOTAL is the sum of the INV_SUBTOTAL and the INV_TAX.

```
CREATE OR REPLACE PROCEDURE DISP(I_NO IN NUMBER) IS
    INV_SUBTOTAL NUMBER;
   INV_TOTAL NUMBER;
   BEGIN
   SELECT SUM(LINE_UNITS*LINE_PRICE)  INTO INV_SUBTOTAL
   FROM LINE WHERE LNV_NO=I_NO;
   INV_TOTAL:=INV_SUBTOTAL+.08*INV_SUBTOTAL;
   DBMS_OUTPUT.PUT_LINE('INV_TOTAL: ' ||INV_TOTAL);
   END;
/
```

```
SQL>  CREATE OR REPLACE PROCEDURE DISP(I_NO IN NUMBER) IS
  2        INV_SUBTOTAL NUMBER;
  3       INV_TOTAL NUMBER;
  4       BEGIN
  5       SELECT SUM(LINE_UNITS*LINE_PRICE)  INTO INV_SUBTOTAL
  6       FROM LINE WHERE LNV_NO=I_NO;
  7       INV_TOTAL:=INV_SUBTOTAL+.08*INV_SUBTOTAL;
  8       DBMS_OUTPUT.PUT_LINE('INV_TOTAL: ' ||INV_TOTAL);
  9       END;
 10  /

Procedure created.
```

6. Write suitable PL/SQL code to display the list of vendors who must be contacted whenever a product reaches reorder level.

```
create or replace trigger ven_list
   after insert or delete or update on PRODUCT
   for each row
   when (new.P_qoh<old.P_min)
   declare
   vcode number :=:old.V_code;
   cursor c1 is select V_name,V_contact from VENDOR where V_code=vcode;
   vname VENDOR.V_name%type;
```

```
 vno vendor.V_contact%type;
BEGIN
open c1;
loop
fetch c1 into vname,vno;
exit when c1%notfound;
dbms_output.put_line('VENDOR NAME:-'||vname||'contact number:-'||vno);
end loop;
close c1;
end;
/
```

```
SQL> create or replace trigger ven_list
  2      after insert or delete or update on PRODUCT
  3      for each row
  4      when (new.P_qoh<old.P_min)
  5      declare
  6      vcode number :=:old.V_code;
  7      cursor c1 is select V_name,V_contact from VENDOR where V_code=vcode;
  8      vname VENDOR.V_name%type;
  9      vno vendor.V_contact%type;
 10     BEGIN
 11     open c1;
 12     loop
 13     fetch c1 into vname,vno;
 14     exit when c1%notfound;
 15     dbms_output.put_line('VENDOR NAME:-'||vname||'contact number:-'||vno);
 16     end loop;
 17     close c1;
 18     end;
 19  /

Trigger created.
```

7. Write the trigger to update the CUST_BALANCE in the CUSTOMER table when a new invoice record is entered. (Assume that the sale is a credit sale.) Test the trigger using the following new INVOICE record:<8005, 1001, '27-APR-16', 225.40>.

```
create or replace trigger cust_UPDATE_Bal
    After insert on INVOICE
    for each row
    BEGIN
   UPDATE CUSTOMER
   SET Cus_balance=Cus_balance + :NEW.inv_amount
    where Cus_code= :NEW.Cus_code;
   END;
   /
```

```
SQL> create or replace trigger cust_UPDATE_Bal
  2       After insert on INVOICE
  3       for each row
  4       BEGIN
  5      UPDATE CUSTOMER
  6      SET Cus_balance=Cus_balance + :NEW.inv_amount
  7       where Cus_code= :NEW.Cus_code;
  8      END;
  9      /

Trigger created.
```

8. Write a trigger to update the customer balance when an invoice is deleted.

```
create or replace trigger trg1 after delete on INVOICE
    for each row
     begin
     update CUSTOMER set Cus_balance=Cus_balance + :old.inv_amount
     where Cus_code= :old.Cus_code;
    END;
     /
```

```
SQL>  create or replace trigger trg1 after delete on INVOICE
  2        for each row
  3         begin
  4         update CUSTOMER set Cus_balance=Cus_balance + :old.inv_amount
  5         where Cus_code= :old.Cus_code;
  6        END;
  7         /

Trigger created.
```

9. Write a trigger that automatically updates the quantity on hand for each product sold after a new LINE row is added.

```
create or replace trigger trg2 after insert on Line
    for each row
    Begin
    update product set P_qoh=P_qoh - :new.Line_units
    where product.P_code = :new.P_code;
    end;
    /
```

```
SQL>  create or replace trigger trg2 after insert on Line
  2         for each row
  3         Begin
  4         update product set P_qoh=P_qoh - :new.Line_units
  5         where product.P_code = :new.P_code;
  6         end;
  7         /

Trigger created.
```

10. Write a trigger to throw exception whenever the invoice amount exceeds customer balance.

```
 create or replace trigger inv_amt_exceed
after insert or update on invoice
for each row
declare
my_exc exception;
cus_bal customer.cus_balance%type;
inv_amt invoice.inv_amount%type;
begin
select cus_balance
into
cus_bal
from customer
where cus_code = :new.cus_code;
if(cus_bal < :new.inv_amount) then
raise my_exc;
end if;
exception
when my_exc then
raise_application_error(-20000,'Message');
end;
/
```

```
SQL> create or replace trigger inv_amt_exceed
  2  after insert or update on invoice
  3  for each row
  4  declare
  5  my_exc exception;
  6  cus_bal customer.cus_balance%type;
  7  inv_amt invoice.inv_amount%type;
  8  begin
  9  select cus_balance
 10  into
 11  cus_bal
 12  from customer
 13  where cus_code = :new.cus_code;
 14  if(cus_bal < :new.inv_amount) then
 15  raise my_exc;
 16  end if;
 17  exception
 18  when my_exc then
 19  raise_application_error(-20000,'Message');
 20  end;
 21  /

Trigger created.
```