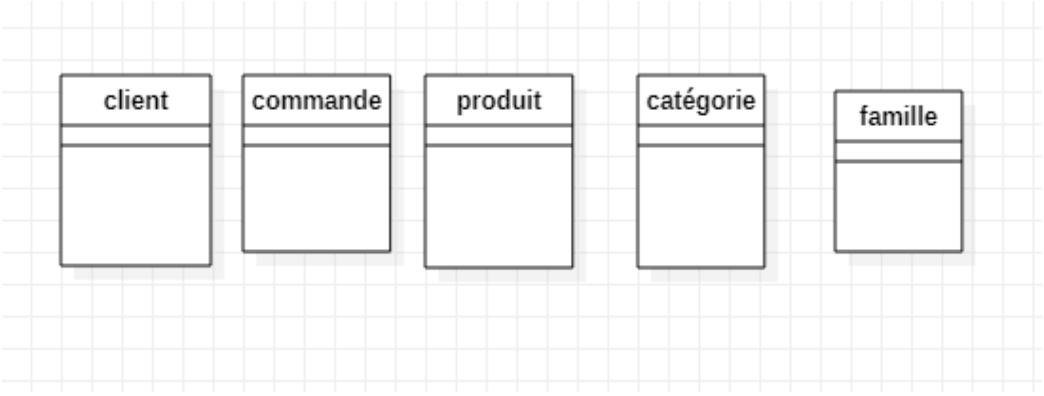
Rapport :

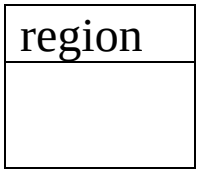
TP 1

Réalisé par : Laila EL HAJJAMY

Modélisation :

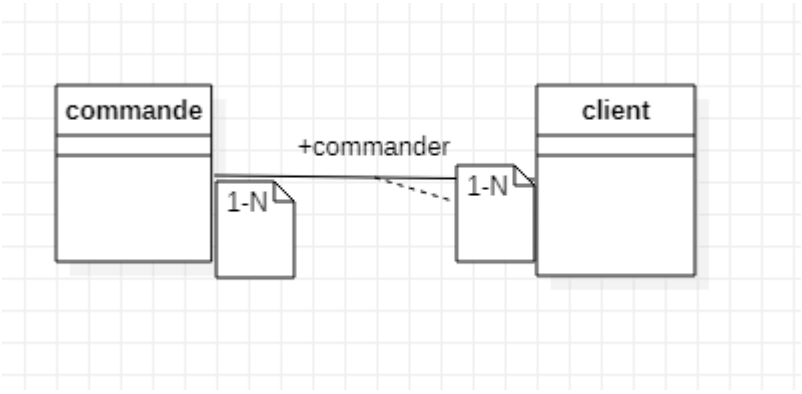
1) A- On distingue **les entités** suivantes :



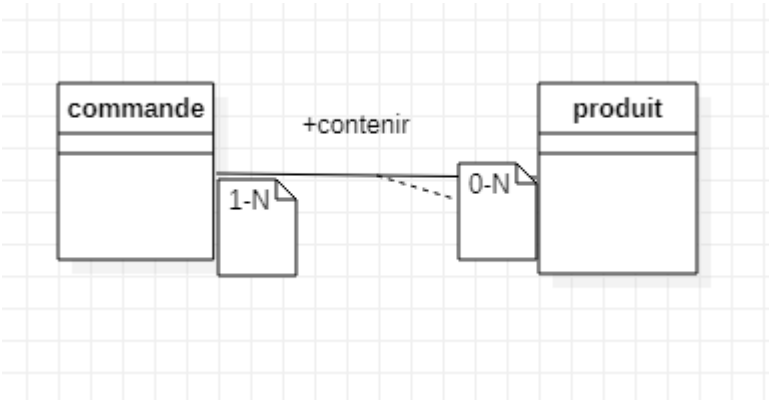


B-Les associations :

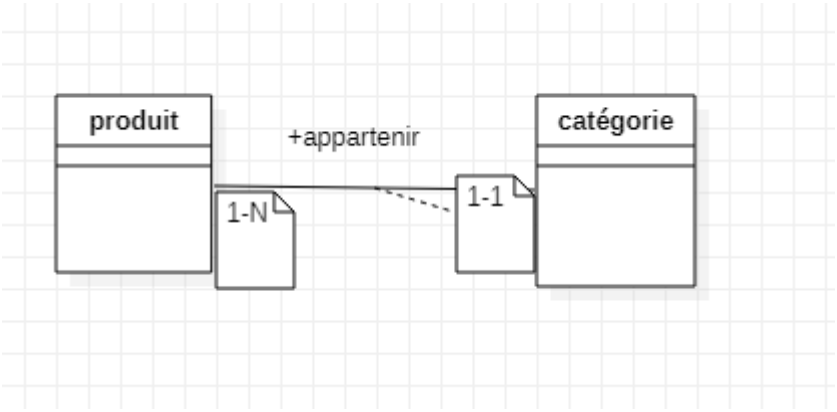
Entre commande et client



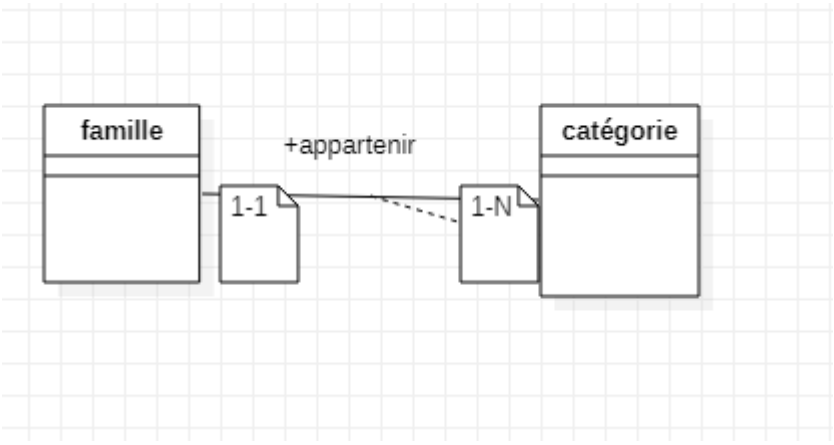
Entre commande et produit



Entre produit et catégorie



Entre catégorie et famille



Entre client et région :

client	habiter	région
	0-N 1-N	

C- Les dépendances fonctionnelles :

Num_com→ nom_prop ,date_com ,detail_pro,somme,reglee

detail_pro→somme

---

cd_client→nom_client,region , adresse, date_con,email,observation

adresse→region

---

cd_categorie→libelle

---

cd_falmille→libelle

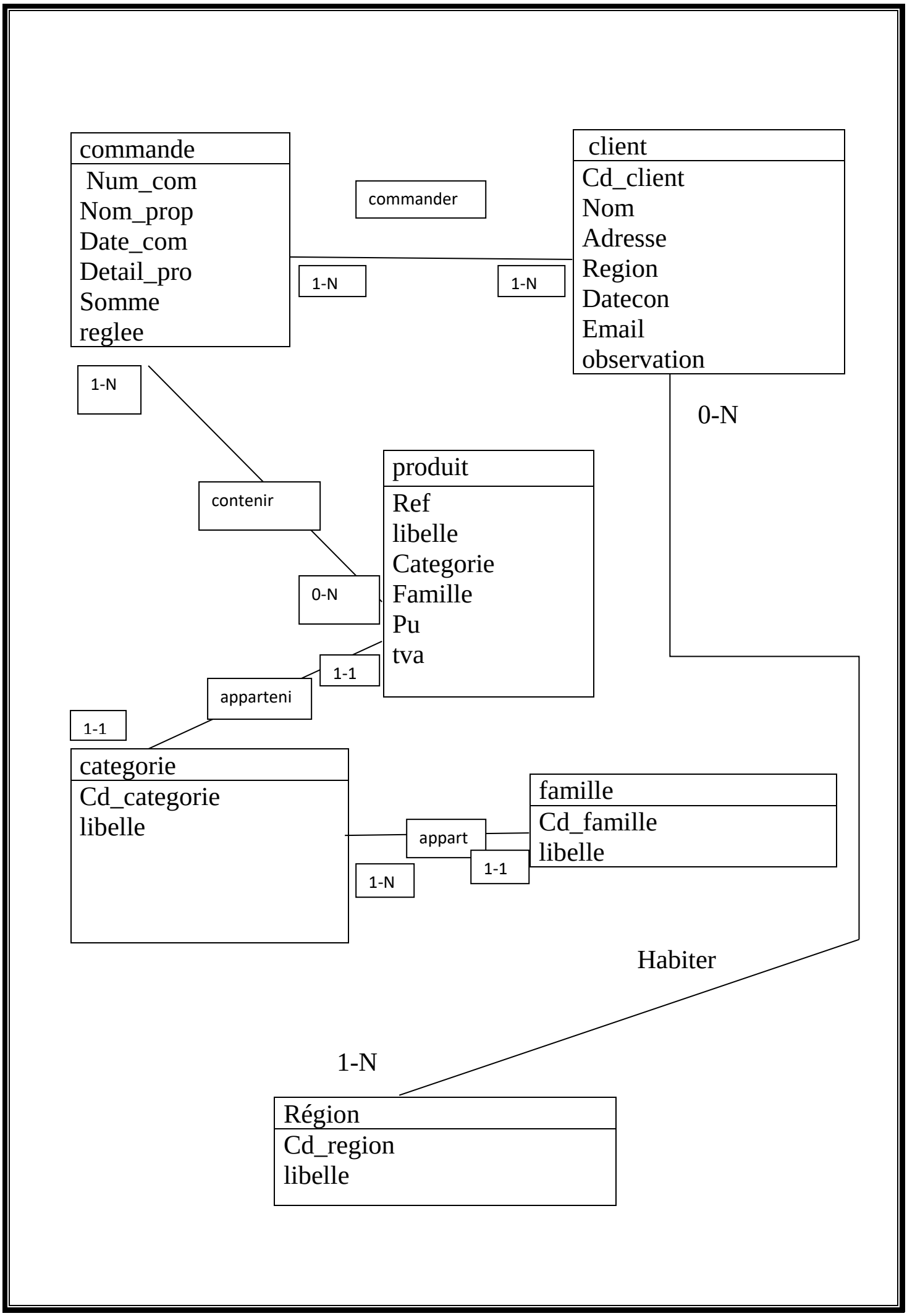
---

ref→libelle, categorie, famille,pu,tva

libelle→categorie,famille

libelle,categorie→famille

D-schéma conceptuel


commande
Num_com
Nom_prop
Date_com
Detail_pro
Somme
reglee
commander
1-N
1-N
client
Cd_client
Nom
Adresse
Region
Datecon
Email
observation
1-N
0-N
contenir
produit
Ref
libelle
Categorie
Famille
Pu
tva
0-N
1-1
apparteni
1-1
categorie
Cd_categorie
libelle
appart
1-N
1-1
famille
Cd_famille
libelle
Habiter
1-N
Région
Cd_region
libelle

MLD :

Commande(nom_prop ,date_com,detail_pro,somme,reglee)

Client(cd_client,nom,adresse,region,datecom,email,observation)

Produit(ref,libelle,categorie,famille,pu,tva)

Categorie(cd_categorie,libelle)

Famille(cd_famille, libelle)

E-Scripts pour la création des tables :

La table client :

```
CREATE TABLE client(
Cd_client  INT,
Nom VARCHAR(50),
Adresse VARCHAR(255),
Region INT,
Date_con DATE,
Email VARCHAR(255),
Observation VARCHAR(500)
```

)

la table commande :

CREATE TABLE commande(

Num_com INT ,

Nom_prop VARCHAR(50),

Date_com DATE,

Detail_pro VARCHAR(255),

Qte INT,

Ref VARCAR(255),

Somme INT,

Reglee BOOLEAN

)

la table produit :

CREATE TABLE produit(

Ref VARCHAR(50),

Libelle VARCHAR(255),

Categorie INT,

Famille INT,

Pu INT,

Tva INT

)

la table categorie :

```
CREATE TABLE categorie(
Cd_categorie INT,
Libelle VARCHAR(255)
)
```

la table famille :

```
CREATE TABLE famille(
Cd_famille INT,
Libelle VARCHAR(255)
)
```

La table region :

```
CREATE TABLE region(
Cd_region INT,
Libelle VARCHAR(255)
)
```

2) Définir les contraintes clef primaire , et clef étrangère :

```
ALTER TABLE client
ADD PRIMARY KEY(Cd_client);
ADD FOREIGN KEY (Region) REFERENCES region(Cd_region);
```

```
ALTER TABLE commande
ADD PRIMARY KEY(Num_com)
ADD FOREIGN KEY (Nom_prop) REFERENCES client(Nom);

ALTER TABLE  produit
ADD PRIMARY KEY(Ref)
ADD FOREIGN KEY (categorie) REFERENCES categorie((Cd_categorie);

ALTER TABLE  categorie
ADD PRIMARY KEY(Cd_categorie);

ALTER TABLE  famille
ADD PRIMARY KEY(Cd_famille);
```

3) Remplissage des tables :

INSERT INTO client VALUES(1 ,'bentager', 'Mhannech 2, Teouan', 1 , 2019-11-2 , 'ahmed@mail.com', 'a passe une grosse commande en janvier 2018 ;depuis , plus rien .penser à reprendre contact'), (2, 'esseghiri','res naoufel app8 ,maarif,casablanca', 2, 2018-10-27 ,'youssef@mail.com' , 'client contacte grace a m .houty de casablanca .remise de 10 pourcent sur la derniere commande suite a un retard d'envoi'),(3, 'houty' ,'settat', 2 , 2018-11-20 ,'karim@mail.com' , 'client fidele qui passe des commandes regulierment (environ une par moi) ') ;

INSERT INTO produit VALUES('A01 ' ,'barette memoire 2Go', 1 ,1 ,150 ,12),('A02' ,'barette mémoire 4Go' ,1, 1 , 240 , 12),('B15','carte graphique AMD Radeon RX 570 pulse 4Go', 5 , 2 , 2349 , 20),('A11' ,'carte son Asus Xonar DG ' , 4 ,2 ,750 , 12),('C80' ,'Disque Externe Toshiba usb 3 .0 1000Go ', 2 , 3 ,546 ,33) ;

INSERT INTO region VALUES(1 , 'tanger –tetouan-al hoceima),(2 , 'region de l'oriental'),(3 ,'region de fes-meknes),(4 ,'region de rabat-sale-kenitra'),(5 ,'region de beni mellah-khenifra'),(6, 'region de casablanca-settat'),(7, 'region de marrakech-safi ') ;

INSERT INTO famille VALUES(1, 'MEMOIRE'),(2, 'CARTE'),(3 ,'DISQUE');

INSERT INTO categorie VALUES(1, ‘RAM’),(2, ‘DISQUE DUR EXTERNE’),(3 ,’DISQUE DUR INTERNE’) ,(4 ,’CARTE SON’),(5, ‘CARTE GRAPHIQUE’);

PL /SQL:

Curseurs :

1) Curseur pour afficher les infos d’un produit :

```
DECLARE
   CURSOR c_infpro IS
   SELECT * FROM produit;
   v_infpro  c_infpro%ROWTYPE;
   BEGIN
        OPEN c_infpro;
        BEGIN c_infpro;
        LOOP
        FETCH c_infpro INTO  v_infpro;
```

```
            EXIT WHEN c_infpro%NOTFOUND;

            dbms_output.put_line('Ref' || v_infpro.Ref);
        dbms_output.put_line('Libelle' ||
        v_infpro.Libelle);
        dbms_output.put_line('Categorie ||
        v_infpro.Categorie);
        dbms_output.put_line('Famille' ||
        v_infpro.Famille);
        dbms_output.put_line('Pu || v_infpro.Pu);
        dbms_output.put_line('Tva' || v_infpro.Tva);

        END LOOP;
        CLOSE c_infpro;
        END;
END;
/
```

2) Curseur pour afficher la phrase prédéfinie ;

```
DECLARE

   CURSOR c_infprop(marque  IN OUT
commande.detail_pro%TYPE  ,ref  IN OUT
commande.Ref%TYPE , quantite  IN OUT
commande.Qte%TYPE , datedebut  IN OUT  DATE ,
datefin  IN OUT DATE ) IS
```

```
    SELECT COUNT (*)FROM  commande WHERE
  detail_pro = marquee  AND Qte = quantite AND
  Date_com BETWEEN datedebut AND datefin;

    v_infprop  c_infprop%ROWTYPE;

    BEGIN

        OPEN c_infprop;

        BEGIN c_infprop;

        LOOP

        FETCH c_infprop  INTO  v_infprop;

        EXIT WHEN c_infprop%NOTFOUND;

       dbms_output.put_line(marque || ' ' || ' Ref:'||'a
ete vendu' || v_infprop||'fois' ||'entre le '||
datedebut||'et' || datefin || 'dune quantite de '||
quantite||'pieces vendues');

      END LOOP;
      CLOSE c_infprop;
      END;
END;
/
```

Procedures :

1) Procédure donnant la liste des produits triés par PU

```
CREATE OR REPLACE PROCEDURE affichage_trie IS
```

```
BEGIN
  DECLARE
        CURSOR c_prodtrie IS
        SELECT * FROM produit GROUP BY Pu ;
        BEGIN
              FOR  v_prop IN c_prodtrie LOOP
              dbms_output.put_putline('Ref' || v_prop.Ref ||
'Libelle' || v_prop.Libelle || 'Categorie'|| v_prop.Categorie ||
'Famille' || v_prop.Famille || 'Pu' || v_prop.Pu || 'tva'||
v_prop.Tva);
END LOOP;
END;
END;
/
CALL affichage_trie ;
```

2) <u>Procédure permettant de lancer une comande:</u>

```
CREATE OR REPLACE PROCEDURE lancer_commande
(nom_pro IN OUT VARCHAR(255) ,Num_com IN OUT INT ,
Date_com  IN OUT DATE , detail_pro  IN OUT
VARCHAR(255) ,qte IN OUT INT , Ref IN OUT
VARCHAR(255), somme IN OUT INT , reglee  IN OUT
BOOLEAN DEFAULT FALSE )IS
BEGIN

        BEGIN
        DECLARE
        CURSOR cnom IS
        SELECT nom FROM client WHERE nom = nom_pro ;
        OPEN cnom;
        IF cnom %NOTFOUND THEN

        INSERT INTO commande VALUES (nom_pro,
Num_com, date_com , detail_pro , qte, Ref, somme ,reglee );
        CLOSE cnom;
        END;
```

```
END;
/
CALL lancer_commande ;
```

3) Procedure pour lancer une commande avec remise :

```
CREATE OR REPLACE PROCEDURE commande_remise
(nom_pro IN OUT VARCHAR(255) ,Num_com IN OUT INT ,
Date_com  IN OUT DATE , detail_pro  IN OUT
VARCHAR(255) ,qte IN OUT INT , Ref IN OUT
VARCHAR(255), somme IN OUT INT , reglee  IN OUT
BOOLEAN DEFAULT FALSE , remise IN INT)IS
BEGIN

        BEGIN
        DECLARE
        CURSOR cnom IS
        SELECT nom FROM client WHERE nom = nom_pro ;
        OPEN cnom;
        Prix_total : =somme;
        IF cnom %NOTFOUND THEN
        Prix_total := Prix_total-Prix_total*remise ;
        INSERT INTO commande VALUES (nom_pro,
Num_com, date_com , detail_pro , qte, Ref, Prix_total ,reglee );
        CLOSE cnom;
        END;

END;
/
CALL commande_remise ;
```

Fonctions:

1)fonction permettant d'afficher la somme des commandes passées par un client

```
DECLARE OR REPLACE FUNCTION somme(nom_client
IN VARCHAR(255))
 RETURN DOUBLE
IS
BEGIN
DECLARE
CURSOR c_some IS
SELECT somme FROM commande WHERE nom_prop =
nom_client);
v_somme c_somme%TYPE :=0;
v_total  c_somme%TYPE :=0 ;
BEGIN
OPEN c_somme ;
LOOP
FETCH c_somme INTO v_somme ;
EXIT WHEN v_somm%NOTFOUND;
IF v_somme IS NOT NULL THEN
v_total:=v_total+ v_somme;
dbms_output.put_line('somme totale est '|| v_total) ;
END IF ;
END LOOP;
CLOSE c_somme;
END;
END;
/
```

3)
```
DECLARE OR REPLACE FUNCTION
somme_conversion(numero IN INT , choix IN CHAR)
 RETURN DOUBLE
IS
BEGIN
DECLARE
CURSOR c_some IS
```

```
    SELECT somme FROM commande WHERE
    num_commande= numero);
v_euro c_somme%TYPE :=0;
v_dollar c_somme%TYPE :=0 ;
v_somme c_somme%TYPE :=0;

BEGIN
OPEN c_somme ;
LOOP
FETCH c_somme INTO v_somme ;
   EXIT WHEN v_somme%NOTFOUND;
IF v_somme IS NOT NULL AND choix='e' THEN
v_euro:= v_somme*10.12;
dbms_output.put_line('somme EN EURO est '|| v_euro) ;

ELSIF v_somme IS NOT NULL AND choix='d' THEN
v_euro:= v_somme*12.23;
dbms_output.put_line('somme EN DOLLAR est '||
v_dollar) ;
ELSE dbms_output.put_line('entrer d ou e);


END IF ;
END LOOP;
CLOSE c_somme;
   END;
END;
/
```

4) Fonction qui prend l'ID et retourne le nombre d'articles existants ;

```
DECLARE OR REPLACE FUNCTION
nombre_article(id IN)
 RETURN DOUBLE
IS
BEGIN
DECLARE
CURSOR c_id IS
SELECT COUNT Cd_categorie FROM categorie
WHERE Cd_categorie= id);
v_nombre c_id%TYPE :=0;
BEGIN
OPEN c_id ;
LOOP
FETCH c_id INTO v_nombre ;
   EXIT WHEN v_nombre%NOTFOUND;
dbms_output.put_line('le nombre d'articles existants
pour la categorie '|| id || 'est' || v_nombre);

END LOOP;
   CLOSE c_id;
      END;
   END;
   /
```