TP N° 3-Les Functions/Procedure

1.Print Number Of Product For Each Category

```
create or replace procedure tp3q1

is

nbproduit integer:=0;
nom categorie.nomcateg%type;

cursor cr is
select (select count(p.refprod) from produit p where p.codecateg = c.codecateg ) nb , c.nomcateg from
    categorie c;

begin
open cr;
fetch cr into nbproduit,nom;

while (cr%found) loop
dbms_output.put_line('La Categorie '''||nom||''' possede '||nbproduit||' produits');
fetch cr into nbproduit , nom;
end loop;

exception

when no_data_found then
dbms_output.put_line(sqlcode||sqlerrm);

when others then
dbms_output.put_line(sqlcode||sqlerrm);

end;
/
```

TP3Q1 Procedure

We created a porcedure that takes no parameter since we will print number of product for each category so naturally we will have to use the cursor to loop through the table , to fetch name of category and its number of products we used nested select :

Inner Query

```
select count(p.refprod) from produit p where p.codecateg = c.codecateg
```

This inner query select number of product for each category (p.codecateg = c.codecateg)

Outer Query

```
select (inner query) nb , c.nomcateg from categorie c;
```

This outer query select number of products and name of category

2.Delete Product That Never Got Ordered

```
create or replace procedure tp3q2

is

begin

delete produit p where refprod NOT IN (select dc.refprod from detailcommande dc where dc.refprod = p.refprod);

end;
/
```

TP3Q2 Procedure

We created a procedure that deletes all products that never got ordered using nested select :

Inner Query

```
select dc.refprod from detailcommande dc where dc.refprod = p.refprod
```

This fetch list of product that got ordered at least once

Outer Query

```
delete produit p where refprod NOT IN (inner query);
```

Deletes products that !(got ordered at least once) → products that never got ordered

3.Print Employe Names And Their Revenue

```
create or replace procedure tp3q3

is
cursor cr is
select e.nom , e.prenom ,
(select nvl(sum(dc.prixunit*dc.qte-dc.remise),0) from detailcommande dc where dc.numcom in (
select c.numcom from commande c where c.numemp = e.numemp)) chiffre_affaire from employe e
where (select nvl(sum(dc.prixunit*dc.qte-dc.remise),0) from detailcommande dc where dc.numcom in (
select c.numcom from commande c where c.numemp = e.numemp)) between 8000 and 10000;
nom employe.nom%type;
prenom employe.prenom%type;
ca number;

begin
open cr;
fetch cr into nom,prenom,ca;

while cr%found loop
dbms_output.put_line('nom '||nom||' prenom '||prenom||' chiffre d''affaire '||ca);
fetch cr into nom,prenom,ca;
end loop;

close cr;
end;
/
```

TP4Q1 Trigger

We created a procedure that prints all employes first name , last name and revenue , so we will have to use a cursor to loop through all employees , we used nested select :

Inner Query

```
select nvl(sum(dc.prixunit*dc.qte-dc.remise),0) from detailcommande dc where dc.numcom
in (select c.numcom from commande c where c.numemp = e.numemp)
```

This nested inner query calculate the revenue of an employee if employee doesn't exist in commande table it will return 0 instead of null because of nvl , the secong query filters order made by an employe (c.numep = e.numemp)

Outer Query

```
select e.nom , e.prenom , (inner query) chiffre_affaire from employe e
where (inner query) between 8000 and 10000;
```

It select first name , last name , and revenue for each employe then filter to employe that their revenue between 8000 and 10000

4.Number Of Order Made By An Inputed Client

4.a.Stand Alone Function

```
create or replace function tp3q4 (code in char)
return number
is
nbcom number:=0;

begin
select count(c.codeclient) into nbcom from commande c where c.codeclient = code;
RETURN nbcom;
end;
/
```

TP3Q4 Function

Simple Function that takes as parameter codeclient and then returns number of order made by that client using a simple select query

4.b.Pl/SQL Code That Calls The TP3Q4 Function

```
1
2  declare
3
4  cursor cr is select c.codeclient from client c;
5  code client.codeclient%type;
6  nbcom number;
7
8  begin
9  open cr;
10 fetch cr into code;
11
12 while cr%found loop
13 nbcom :=  tp3q4 (code);
14 dbms_output.put_line(code||' a '||nbcom||' commandes');
15 fetch cr into code;
16 end loop;
17
18 close cr;
19 end;
20 /
```

PL/SQL Code

We made a pl/sql block that uses cursor to loop through all the clients , and inside the while loop for each iteration we will call the tp3q4 function so we can have number of order made by that client and be able to print it

5.Procedure To Insert On Commande Table

```
create or replace procedure tp3q5 (numcom in number , codeclient in varchar2 ,numemp in number,datecom in date
    ,alivavant in varchar2 ,
dateenv in date , nummess in number, port in varchar2 , destinataire varchar2, adrliv in varchar2 ,villeliv in
     varchar2,
regionliv in varchar2,codepostalliv in varchar2 ,paysliv in varchar2)

is
exist number:=0;
pk exception;
fk_mes exception;
fk_client exception;
fk_emp exception;

begin

select count(c.numcom) into exist from commande c where c.numcom = numcom;
if(exist!=0)then
raise pk;
end if;

select count(c.codeclient) into exist from client c where c.codeclient = codeclient;
if(exist=0)then
raise fk_client;
end if;

select count(e.numemp) into exist from employe e where e.numemp = numemp;
if(exist=0)then
raise fk_emp;
end if;

select count(m.nummess) into exist from messager m where m.nummess = nummess;
if(exist=0)then
raise fk_mes;
end if;

insert into commande values (numcom,codeclient,numemp,datecom,alivavant,dateenv, nummess, port, destinataire,
adrliv,villeliv,regionliv,codepostalliv,paysliv);

exception
when pk then
dbms_output.put_line('Primary Key Already Exist');
when fk_client then
dbms_output.put_line('Client Foreign key Doesn''t Exist');
when fk_emp then
dbms_output.put_line('Employe Foreign key Doesn''t Exist');
when fk_mes then
dbms_output.put_line('Messager Foreign key Doesn''t Exist');
when others then
dbms_output.put_line(sqlcode||sqlerrm);
end;
/
```

TP4Q5 Procedure

We created a procedure that takes as parameter values of commande to be able to insert on commande table , we declared exception for each case where insertion is impossible :

- pk : primary key already exist
- fk_client : client foreign key doesn't exist
- fk_mes : messager foreign key doesn't exist
- fk_emp : employee foreign key doesn't exist