**TD PL/SQL**

On considère la base de données Gestock suivante concernant la gestion des ventes

Client (**Numcli**, Nom, Ville, sexe)

Facture (**Numfact**, Datefact, #Numcli)

Produit (**Reference**, NomPro, Pu, Qtéstk)

Vente (#**Reference, #Numfact**, Qtév)

Créer les requêtes suivantes:

1- Ecrire un script anonyme PL/SQL qui crée un utilisateur superadmin qui aura tous les droits et comme mot de passe admin

2- Ecrire un script anonyme PL/SQL qui crée un utilisateur Gestockuser qui n’aura que le droit de consulter toutes les tables et de modifier seulement la table client et comme mot de passe admin.

3-Accorder à l’utilisateur superadmin, la possibilité d’accorder les privilèges à d’autres utilisateurs   
4- Supprimer l’utilisateur Gestockuser.

Créer la table vente sachant que la quantité vendue par défaut est 0 et ne peux jamais être négative

## Fonctions stockées

À la fin de chaque année, certains fournisseurs des produits demandent un inventaire sur les produits vendus. Cet inventaire leur est envoyé par e-mail pour analyse et traitement. L'information demandée est la suivante : donner pour chaque produit, sa désignation, son prix de vente actuel, et une mention sur la demande :

* demande "forte" si le nombre d'articles vendus dépasse 15,
* demande "moyenne" si le nombre d'articles vendus est compris entre 11 et 15,
* et demande "faible" sinon.

### TAF

1. Écrire une fonction stockée qui retourne la valeur de demande (forte, moyenne, faible) en fonction d'un numéro de produit.
2. Écrire une requête SELECT qui utilise cette fonction pour présenter la liste des produits avec la demande.

#### [Indice](https://stph.scenari-community.org/nf17/ora/co/oracle13.html)

* Pensez à utiliser l'instruction show errors en fin de block PL/SQL pour avoir en retour les éventuelles erreurs de complication.
* Utilisez la trame suivante pour vous aider.
* SELECT ..., fDemande (...), ... FROM ... WHERE ...

CREATE OR REPLACE FUNCTION fDemande (num\_produit number) RETURN varchar

IS

qte\_vendue NUMBER;

BEGIN

SELECT ... INTO qte\_vendue

FROM ...

WHERE ...

IF ... THEN

RETURN ('forte');

...

ELSE

...

END IF;

END fDemande;

/

SHOW ERRORS

**Curseurs**

Afin d'affiner l'analyse du profil de chaque client, le service marketing souhaite avoir un état pour chaque client avec le nombre de ses factures et son chiffre d'affaire total.

**Question**

Écrivez deux fonctions stockées qui prendront en argument un numéro de client, et retourneront sous forme de number le nombre de factures et le chiffre d'affaire, respectivement.

[**Indice**](https://stph.scenari-community.org/nf17/ora/co/oracle14.html)

CREATE OR REPLACE FUNCTION nbfact(num\_client NUMBER) RETURN NUMBER

...

END nbfact;

/

SHOW ERRORS

CREATE OR REPLACE FUNCTION ca (num\_client NUMBER) RETURN NUMBER

...

END ca;

/

SHOW ERRORS

**Question**

1. Écrivez un programme PL/SQL anonyme qui affiche pour le client numéro 1 le nombre de factures et le chiffre d'affaire.
2. Écrivez un second programme PL/SQL qui affiche pour chaque client son nombre de factures et son chiffre d'affaire sous la forme suivante :

Client 1

10 / 152.23

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

Client 2

5 / 123.34

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

...

1. Transformez le bloc anonyme en procédure. Testez cette procédure.

[**Indice**](https://stph.scenari-community.org/nf17/ora/co/oracle14.html)

* Utiliser le caractère de concaténation || pour afficher plusieurs informations sur une même ligne :

DBMS\_OUTPUT.PUT\_LINE('Client' || client.num);

* Pensez à exécuter le "SET SERVEROUTPUT ON" avant l'exécution de "DBMS\_OUTPUT.PUT\_LINE".
* Utlisez la syntaxe DECLARE ... BEGIN ... END pour déclarer un bloc PL/SQL anonyme.
* Utiliser un curseur sur la table client :

CURSOR c\_client IS SELECT num FROM client;

## Triggers

Un ensemble de règles de gestion est défini afin de gérer le stock des produits. La politique suivante a été admise :

1. après chaque vente, la quantité vendue est soustraite de la quantité disponible indiquée dans la colonne Qtéstk de la table des produits,
2. si cette quantité (disponible) est inférieure à un "seuil" alors insérer un avertissement dans une table de journalisation avec la date du jour, le numéro de produit, et stock restant après mise à jour.

### Question

1. Créer un déclencheur permettant de mettre à jour la quantité disponible dans la table produit à chaque insertion dans la table ligne facture (traitement du point 1).
2. Testez votre trigger TR\_STOCK\_AUDIT en supprimant et en réinsérant les données des tables FACTURE et Vente.
3. Créer la table de journalisation.
4. Modifiez votre déclencheur de telle façon qu'après la mise à jour, il teste si le stock est inférieur à une valeur seuil de 5. Si le stock est inférieur à ce seuil, alors il devra insérer les informations adéquates dans la table de journalisation (traitement du point 2).

#### [Indice](https://stph.scenari-community.org/nf17/ora/co/oracle15.html)

#### CREATE TRIGGER TR\_STOCK\_AUDIT

#### BEFORE INSERT ON ligne\_fact

* Il s'agit d'un trigger qui cherche à faire une modification pour chaque insertion : utilisez un trigger FOR EACH ROW

Utilisez une requête *update* pour effectuer la mise à jour avec une condition sur le numéro de produit.

UPDATE ...

SET ...

WHERE num=:new.produit;