ISTA TINGHIR Filière : Développement Digital
Option : Web Full Stuck
Module N° 3 : Gérer les données
Groupe : DDoWFS201

TP

EXERCICE N°1 :

1. Créer une BD nommée ***test***.
2. Créer dans la BD test une table ***clients*** ayant la structure suivante :

idClient	nom	prenom	dateNaissance	adresse
				Tinghir
				Tinghir

N.B : Le champ adresse contient par défaut la valeur ''**Tinghir**''

3. Ajouter à la table ***clients*** les trois enregistrements : (**1, tadlaoui, fardi, 1990/9/20**), (**2, tarzaoui, ahmed, 1994/8/2, Tinjdad**) et (**2, jabraoui, kenza, 1994/8/2, Golmima**).
   a. Quelle est l'adresse du client dont idClient=1 ?
   b. Quelle est l'adresse du client dont idClient=2 ?
   c. Conclure.

4. Créer dans la BD test une table **pareille*Clients*** ayant la même structure que ***clients***. Les données de ***clients*** sont-elles copiées dans **pareille*Clients ?***

5. Supprimer la table **pareille*Clients***

6. Dans la table ***clients :***
   a. Ajouter le champ **email varchar(255);**
   b. Modifier le type du champ **dateNaissance** en **int**;
   c. Ajouter une colonne **sexe VARCHAR(20)** après la colonne de **prenom** et y insérer les valeurs correctes pour les trois enregistrements de la table ***clients*** ;
   d. Modifier le champ **adresse** en **ville VARCHAR(100) NOT NULL**;
   e. Afficher la nouvelle structure de la table **clients**.
   f. Supprimer le champ **email** ajouté au début.
   g. Réafficher la structure de **clients**.

7. Renommer **clients** en **listeClients .**

8. Afficher les clients domiciliés à Tinghir.
9. Afficher tous les clients dans un ordre croissant de leurs identifiants.

10. Les vues sont des objets de base de données qui permettent d'enregistrer une requête particulière sous forme de table afin de pouvoir les utiliser ultérieurement. Créer une **vue** permettant de lister les clients de la ville de Tinjdad.

11. Le client dont **idClient=2** s'appelle en fait **fadili ahmed** et habite à **Boumalne Dades**. Mettre à jour cet enregistrement.

12. Le client dont **idClient=3** ne fait plus partie de la liste des clients. Supprimez-le.

13. Les procédures stockées permettent de stocker un ensemble de requêtes SQL, à exécuter en cas de besoin. Elles peuvent contenir une instruction conditionnelle telle que IF ou CASE ou des boucles ou même une autre procédure stockée ou une fonction.

```
/* Syntaxe pour créer une procédure stockée */

DELIMITER $$
CREATE PROCEDURE nom_procedure ([paramètres])
BEGIN
                    LeCodeSQL
END $$

/* Syntaxe pour exécuter une procédure stockée */

call nom_procedure ([paramètres]);
```

a. Ecrire puis tester une procédure stockée nommée ***afficher ListeClient*** permettant d'afficher le contenu de la table **listeClients.**
b. Ecrire puis tester une procédure stockée nommée ***afficher Client*** permettant d'afficher le client dont l'identifiant est passé en argument**.**
c. Ecrire une procédure stockée nommée ***ajouterClient*** permettant d'ajouter un client passé en argument**.**
d. Faire trois appels différents à ***ajouterClient*** pour ajouter trois clients d'identifiants respectifs 3, 4 et 5. Qu'en est-il à l'ordre des paramètres lors de l'appel de la procédure ? Conclure.
e. Vérifier l'ajout des derniers enregistrements en appelant ***afficher ListeClient*** *puis* ***afficher Client.***
f. Ecrire puis tester une procédure stockée nommée ***NombreClientParVille*** permettant de calculer le nombre de clients domiciliés à une ville passée en argument**.**
g. Lister toutes les procédures stockées existantes.
h. Afficher les procédures stockées relatives à la BD ***test***.
i. Lister toutes les procédures stockées dont le nom respecte le pattern ***afficher.***
j. *Supprimer la* procédure stockée ***afficher Client*** et vérifier l'exécution de cette dernière action.

14. Supprimer la base de données ***test.***

EXERCICE N°2 :

```
USE nom_bd;
DROP FUNCTION IF EXISTS nom_fonction;
DELIMITER $$
CREATE FUNCTION nom_fonction ([parameters])
        RETURNS type_retour
        DéclarationInformative

/* DéclarationInformative=
DETERMINISTIC : La fonction renverra les mêmes valeurs si les mêmes arguments lui sont fournis,
READS SQL DATA : La fonction lira les données de la BD mais ne modifiera pas les données,
MODIFIES SQL DATA : La fonction modifiera les données dans la base de données,
CONTAINS SQL : La fonction aura des instructions SQL mais elles ne lisent et ne modifient les données
*/
        Bloc_Instructions
END $$
DELIMITER ;
```

La facture d'eau est calculée selon un barème à tranches. La consommation mensuelle d'eau est répartie en tranches, les informations concernant les tranches sont données dans le tableau suivant :

Quantité consommée **Qte** en m^3	Prix unitaire Dh/ m^3
< 9	1,29
9 =< **Qte** < 20	4,29
20 <= **Qte** < 36	6,59
Qte >= 36	10,37

1. Déclarer une variable ***consommation*** et lui affecter la valeur 26.
2. En utilisant la structure conditionnelle ***if … else,*** écrire une fonction ***montantAPayer*** qui calcule et affiche le montant d'une consommation.
3. Ecrire une fonction ***montantAPayer2*** En utilisant la structure conditionnelle **case** et *qui fait le même travail que* ***montantAPayer*** .
4. Tester les deux fonctions pour les consommations suivantes : 6 m^3, 15 m^3 et 40 m^3