📚 Résolution ExercicesSQL

> **Création de la note à** `14:21` **le** `2024-11-26`**.**

📝 Prise de Notes - Cours

Voici une proposition d'organisation du document dans Word avec titres et structure claire pour répondre aux questions en SQL :

Exercice 1 : Villes de France

Base de données utilisées

- **Tables :**
  1. `departement` : contient des informations sur les départements de France.
  2. `villes_france_free` : contient des informations détaillées sur les communes françaises (population, superficie, code postal, etc.).

1. Liste des 10 villes les plus peuplées en 2012

Description : Obtenez les 10 villes avec la population la plus élevée en 2012.

Requête SQL :

```sql
SELECT ville_nom, ville_population_2012
FROM villes_france_free
ORDER BY ville_population_2012 DESC
LIMIT 10;
```

2. Liste des 50 villes ayant la plus faible superficie

Description : Sélectionnez les 50 villes avec la plus petite superficie.

Requête SQL :

```sql
SELECT ville_nom, ville_surface
FROM villes_france_free
WHERE ville_surface IS NOT NULL
ORDER BY ville_surface ASC
LIMIT 50;
```

3. Liste des départements d’outre-mer

Description : Récupérez tous les départements dont le numéro commence par "97".

Requête SQL :

```sql
SELECT departement_nom, departement_code
FROM departement
WHERE departement_code LIKE '97%';
```

4. Les 10 villes les plus peuplées en 2012 avec leur département

Description : Affichez les 10 villes les plus peuplées en 2012, accompagnées du nom de leur département.

Requête SQL :

```sql
SELECT v.ville_nom, v.ville_population_2012, d.departement_nom
FROM villes_france_free v
JOIN departement d ON v.ville_departement = d.departement_code
ORDER BY v.ville_population_2012 DESC
LIMIT 10;
```

5. Nombre de communes par département, triées par le plus grand nombre de communes

Description : Calculez le nombre de communes dans chaque département et triez par ordre décroissant.

Requête SQL :

```sql
SELECT d.departement_nom, d.departement_code, COUNT(v.ville_id) AS nombre_communes
FROM departement d
JOIN villes_france_free v ON d.departement_code = v.ville_departement
GROUP BY d.departement_code
ORDER BY nombre_communes DESC;
```

6. Les 10 départements ayant la plus grande superficie totale

Description : Affichez les 10 départements ayant la plus grande superficie totale cumulée des communes.

Requête SQL :

```sql
SELECT d.departement_nom, d.departement_code, SUM(v.ville_surface) AS superficie_totale
FROM departement d
JOIN villes_france_free v ON d.departement_code = v.ville_departement
GROUP BY d.departement_code
ORDER BY superficie_totale DESC
LIMIT 10;
```

7. Nombre de villes dont le nom commence par "Saint"

Description : Comptez toutes les villes dont le nom commence par "Saint".

Requête SQL :

```sql
SELECT COUNT(*) AS nombre_villes
FROM villes_france_free
WHERE ville_nom LIKE 'Saint%';
```

8. Villes ayant un nom existant plusieurs fois, triées par fréquence

Description : Trouvez les villes partageant le même nom, triées par le nombre d'occurrences.

Requête SQL :

```sql
SELECT ville_nom, COUNT(*) AS occurence
FROM villes_france_free
GROUP BY ville_nom
HAVING COUNT(*) > 1
ORDER BY occurence DESC;
```

9. Liste des villes avec une superficie supérieure à la moyenne

Description : Obtenez la liste des villes ayant une superficie plus grande que la moyenne de toutes les communes.

Requête SQL :

```sql
SELECT ville_nom, ville_surface
FROM villes_france_free
WHERE ville_surface > (SELECT AVG(ville_surface) FROM villes_france_free);
```

10. Départements ayant plus de 2 millions d’habitants

Description : Affichez les départements dont la population totale dépasse 2 millions d’habitants.

Requête SQL :

```sql
SELECT d.departement_nom, d.departement_code, SUM(v.ville_population_2012) AS
population_totale
FROM departement d
JOIN villes_france_free v ON d.departement_code = v.ville_departement
GROUP BY d.departement_code
HAVING population_totale > 2000000;
```

11. Remplacement des tirets par un espace pour les villes commençant par "SAINT-"

Description : Modifiez les noms des villes pour remplacer les tirets par des espaces uniquement pour celles qui commencent par "SAINT-".

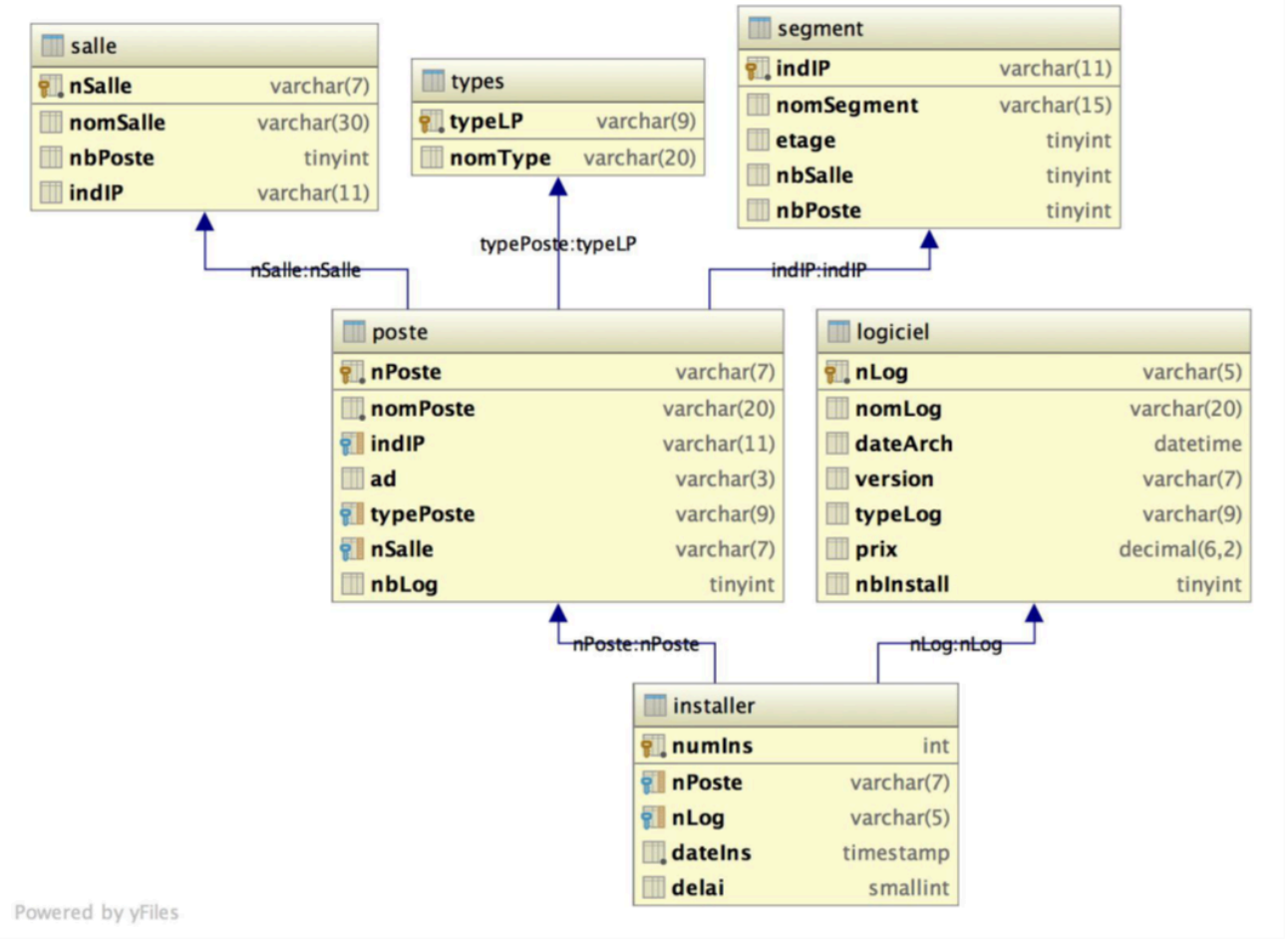
Requête SQL :

```sql
UPDATE villes_france_free
SET ville_nom_reel = REPLACE(ville_nom_reel, 'SAINT-', 'SAINT ')
WHERE ville_nom_reel LIKE 'SAINT-%';
```

Exercice 2 : Gestion du Parc Informatique

Requêtes SQL pour l'analyse de la base de données

Base de donnée



Légende

Colonne	Commentaire	Type
indIP	trois premiers groupes IP (exemple : 130.120.80)	VARCHAR(11)
nomSegment	nom du segment	VARCHAR(20)
etage	étage du segment	TINYINT(1)
nSalle	numéro de la salle	VARCHAR(7)
nomSalle	nom de la salle	VARCHAR(20)
nbPoste	nombre de postes de travail dans la salle	TINYINT(2)
nPoste	code du poste de travail	VARCHAR(7)
nomPoste	nom du poste de travail	VARCHAR(20)
ad	dernier groupe de chiffres IP (exemple : 11)	VARCHAR(3)
typePoste	type du poste (UNIX, TX, PCWS, PCNT)	VARCHAR(9)
dateIns	date d'installation du logiciel sur le poste	dateTIME
nLog	code du logiciel	VARCHAR(5)
nomLog	nom du logiciel	VARCHAR(20)
dateAch	date d'achat du logiciel	dateTIME
version	version du logiciel	VARCHAR(7)
typeLog	type du logiciel (UNIX, TX, PCWS, PCNT)	VARCHAR(9)
prix	prix du logiciel	DECIMAL(6,2)
numIns	numéro séquentiel des installations	INTEGER(5)
dateIns	date d'installation du logiciel	TIMESTAMP
delai	intervalle entre achat et installation	SMALLINT
typeLP	types des logiciels et des postes	VARCHAR(9)
nomType	noms des types (Terminaux X, PC Windows...)	VARCHAR(20)

Introduction

Ce document présente les requêtes SQL permettant d'extraire des données spécifiques à partir de la base de données du parc informatique d'une entreprise. Les requêtes sont classées en deux catégories :

- **Requêtes monotables et de groupement.**
- **Requêtes multitables et sous-requêtes.**

Chaque requête est accompagnée d'une explication concise et des résultats attendus.

Requêtes Monotables et de Groupement

1. Type de poste du poste `p8`

Requête SQL :

```sql
SELECT typePoste
FROM poste
WHERE nPoste = 'p8';
```

Explication :
Cette requête sélectionne le type de poste en filtrant les données dans la table `poste` pour le poste identifié

par `nPoste = 'p8'`.

2. Noms des logiciels de type `UNIX`

Requête SQL :

```sql
SELECT nomLog
FROM logiciel
WHERE typeLog = 'UNIX';
```

Explication :
On extrait les noms des logiciels de type `UNIX` depuis la table `logiciel` grâce à la condition `typeLog = 'UNIX'`.

3. Détails des postes de type `UNIX` ou `PCWS`

Requête SQL :

```sql
SELECT nomPoste, CONCAT(indIP, '.', ad) AS adresseIP, nSalle
FROM poste
WHERE typePoste IN ('UNIX', 'PCWS');
```

Explication :
Cette requête récupère les noms de poste, adresses IP complètes (concaténation de `indIP` et `ad`), ainsi que les numéros de salle pour les postes dont le type est `UNIX` ou `PCWS`.

4. Détails des postes du segment `130.120.80` triés par salle

Requête SQL :

```sql
SELECT nomPoste, CONCAT(indIP, '.', ad) AS adresseIP, nSalle
FROM poste
WHERE indIP = '130.120.80'
ORDER BY nSalle DESC;
```

Explication :
On récupère les mêmes détails que précédemment, mais uniquement pour les postes situés sur le segment `130.120.80`. Les résultats sont triés par numéro de salle décroissant.

5. Numéros des logiciels installés sur le poste `p6`

Requête SQL :

```sql
SELECT nLog
FROM installer
WHERE nPoste = 'p6';
```

Explication :
On interroge la table `installer` pour lister les logiciels associés au poste `p6`.

6. Numéros des postes hébergeant le logiciel `log1`

Requête SQL :

```sql
SELECT nPoste
FROM installer
WHERE nLog = 'log1';
```

Explication :
On extrait les postes qui hébergent le logiciel identifié par `nLog = 'log1'`.

7. Noms et adresses IP complètes des postes de type `TX`

Requête SQL :

```sql
SELECT nomPoste, CONCAT(indIP, '.', ad) AS adresseIP
FROM poste
WHERE typePoste = 'TX';
```

Explication :
Utilisation de la fonction de concaténation pour reconstituer l'adresse IP complète des postes de type `TX`.

8. Nombre de logiciels installés sur chaque poste

Requête SQL :

```sql
SELECT nPoste, COUNT(nLog) AS nbLogiciels
FROM installer
GROUP BY nPoste;
```

Explication :
On compte les occurrences de logiciels installés sur chaque poste en regroupant par `nPoste`.

9. Nombre de postes par salle, trié par ordre croissant

Requête SQL :

```sql
SELECT nSalle, COUNT(nPoste) AS nbPostes
FROM poste
GROUP BY nSalle
ORDER BY nbPostes ASC;
```

Explication :
On regroupe par salle et compte les postes présents, puis on trie par nombre croissant.

10. Nombre d'installations par logiciel

Requête SQL :

```sql
SELECT nLog, COUNT(DISTINCT nPoste) AS nbInstallations
FROM installer
GROUP BY nLog;
```

Explication :
On compte combien de postes différents hébergent chaque logiciel, en évitant les doublons.

11. Moyenne des prix des logiciels de type `UNIX`

Requête SQL :

```sql
SELECT AVG(prix) AS prixMoyen
FROM logiciel
WHERE typeLog = 'UNIX';
```

Explication :
On utilise la fonction `AVG()` pour calculer la moyenne des prix des logiciels de type `UNIX`.

12. Date la plus récente d'achat d'un logiciel

Requête SQL :

```sql
SELECT MAX(dateAch) AS datePlusRecente
FROM logiciel;
```

Explication :
Cette requête utilise la fonction `MAX()` pour trouver la date d'achat la plus récente.

13. Numéros des postes ayant exactement 2 logiciels installés

Requête SQL :

```sql
SELECT nPoste
FROM installer
GROUP BY nPoste
HAVING COUNT(nLog) = 2;
```

Explication :
Après regroupement, on filtre uniquement les postes ayant exactement deux logiciels installés.

Requêtes Multitables et Sous-requêtes

1. Types de postes qui ne sont pas recensés dans le parc informatique

```sql
SELECT t.nomType
FROM types t
WHERE t.typeLP NOT IN (SELECT DISTINCT typePoste FROM poste);
```

2. Types existant à la fois comme types de postes et types de logiciels

```sql
SELECT DISTINCT t.typeLP
FROM types t
WHERE t.typeLP IN (SELECT DISTINCT typePoste FROM poste)
  AND t.typeLP IN (SELECT DISTINCT typeLog FROM logiciel);
```

3. Types de postes n'étant pas des types de logiciels

```sql
SELECT DISTINCT typePoste
FROM poste
```

```sql
WHERE typePoste NOT IN (SELECT typeLog FROM logiciel);
```

4. Adresses IP complètes des postes qui hébergent le logiciel `log6`

A) Produit cartésien :

```sql
SELECT DISTINCT CONCAT(p.indIP, '.', p.ad) AS adresseIP
FROM poste p, installer i
WHERE p.nPoste = i.nPoste AND i.nLog = 'log6';
```

B) Sous-requête :

```sql
SELECT CONCAT(indIP, '.', ad) AS adresseIP
FROM poste
WHERE nPoste IN (
    SELECT nPoste
    FROM installer
    WHERE nLog = 'log6'
);
```

C) JOIN :

```sql
SELECT DISTINCT CONCAT(p.indIP, '.', p.ad) AS adresseIP
FROM poste p
JOIN installer i ON p.nPoste = i.nPoste
WHERE i.nLog = 'log6';
```

5. Adresses IP complètes des postes qui hébergent le logiciel nommé `Oracle 8`

A) Produit cartésien :

```sql
SELECT DISTINCT CONCAT(p.indIP, '.', p.ad) AS adresseIP
FROM poste p, installer i, logiciel l
WHERE p.nPoste = i.nPoste AND i.nLog = l.nLog AND l.nomLog = 'Oracle 8';
```

B) Sous-requête :

```sql
SELECT CONCAT(indIP, '.', ad) AS adresseIP
FROM poste
WHERE nPoste IN (
    SELECT nPoste
    FROM installer
    WHERE nLog IN (
        SELECT nLog
        FROM logiciel
        WHERE nomLog = 'Oracle 8'
    )
);
```

C) JOIN :

```sql
SELECT DISTINCT CONCAT(p.indIP, '.', p.ad) AS adresseIP
FROM poste p
JOIN installer i ON p.nPoste = i.nPoste
JOIN logiciel l ON i.nLog = l.nLog
WHERE l.nomLog = 'Oracle 8';
```

6. Segments ayant exactement trois postes de type `TX`

A) Produit cartésien :

```sql
SELECT s.nomSegment
FROM segment s, poste p
WHERE s.indIP = p.indIP AND p.typePoste = 'TX'
GROUP BY s.nomSegment
HAVING COUNT(p.nPoste) = 3;
```

B) Sous-requête :

```sql
SELECT nomSegment
FROM segment
WHERE indIP IN (
    SELECT indIP
    FROM poste
    WHERE typePoste = 'TX'
    GROUP BY indIP
    HAVING COUNT(nPoste) = 3
);
```

C) JOIN :

```sql
SELECT s.nomSegment
FROM segment s
JOIN poste p ON s.indIP = p.indIP
WHERE p.typePoste = 'TX'
GROUP BY s.nomSegment
HAVING COUNT(p.nPoste) = 3;
```

7. Noms des salles contenant au moins un poste hébergeant le logiciel `Oracle 6`

A) Produit cartésien :

```sql
SELECT DISTINCT s.nomSalle
FROM salle s, poste p, installer i, logiciel l
WHERE s.nSalle = p.nSalle AND p.nPoste = i.nPoste AND i.nLog = l.nLog AND l.nomLog =
'Oracle 6';
```

B) Sous-requête :

```sql
SELECT nomSalle
FROM salle
WHERE nSalle IN (
    SELECT nSalle
```

```
    FROM poste
    WHERE nPoste IN (
        SELECT nPoste
        FROM installer
        WHERE nLog IN (
            SELECT nLog
            FROM logiciel
            WHERE nomLog = 'Oracle 6'
        )
    )
);
```

C) JOIN :

```
SELECT DISTINCT s.nomSalle
FROM salle s
JOIN poste p ON s.nSalle = p.nSalle
JOIN installer i ON p.nPoste = i.nPoste
JOIN logiciel l ON i.nLog = l.nLog
WHERE l.nomLog = 'Oracle 6';
```

8. Installations (nom segment, nom salle, adresse IP complète, nom logiciel, date d'installation)

Requête :

```
SELECT s.nomSegment, sa.nomSalle, CONCAT(p.indIP, '.', p.ad) AS adresseIP, l.nomLog,
i.dateIns
FROM segment s
JOIN salle sa ON s.indIP = sa.indIP
JOIN poste p ON sa.nSalle = p.nSalle
JOIN installer i ON p.nPoste = i.nPoste
JOIN logiciel l ON i.nLog = l.nLog
ORDER BY s.nomSegment, sa.nomSalle, adresseIP;
```

Procédure Stockée

```
+-------------------------------------------------+
| Resultat 1 exo 1                                |
+-------------------------------------------------+
| Derniere installation en salle : numérodeSalle  |
+-------------------------------------------------+

+-------------------------------------------------------------------------+
| Resultat 2 exo 1                                                        |
+-------------------------------------------------------------------------+
| Poste : numéroPoste Logiciel : nomLogiciel en date du dateInstallation  |
+-------------------------------------------------------------------------+
```

Exemple de bloc MySQL :

```
DELIMITER //

CREATE PROCEDURE DerniereInstallation()
BEGIN
```

```
    SELECT i.nPoste, l.nomLog, i.dateIns
    FROM installer i
    JOIN logiciel l ON i.nLog = l.nLog
    ORDER BY i.dateIns DESC
    LIMIT 1;
END //

DELIMITER ;
```

Exercice 3 : Gestion d'une base de données de vente

Contexte

Nous travaillons sur une base de données comprenant trois tables principales :

- **Client** : informations sur les clients (prénom, nom, email, etc.).
- **Commande** : détails des commandes effectuées.
- **Commande_ligne** : détail des produits associés aux commandes.

Chaque question concerne des requêtes SQL à exécuter sur ces tables pour effectuer diverses analyses et mises à jour.

1. Obtenir l'utilisateur ayant le prénom "Muriel" et le mot de passe SHA1 ("test11")

```sql
SELECT *
FROM client
WHERE prenom = 'Muriel'
  AND password = SHA1('test11');
```

2. Liste des produits présents dans plusieurs commandes

```sql
SELECT nom
FROM commande_ligne
GROUP BY nom
HAVING COUNT(DISTINCT commande_id) > 1;
```

3. Liste des produits dans plusieurs commandes avec leurs identifiants associés

```sql
SELECT nom, GROUP_CONCAT(commande_id) AS commandes_associees
FROM commande_ligne
GROUP BY nom
HAVING COUNT(DISTINCT commande_id) > 1;
```

4. Enregistrer le prix total dans chaque ligne de commande

```sql
UPDATE commande_ligne
SET prix_total = quantite * prix_unitaire;
```

5. Montant total de chaque commande avec date et client associé

```sql
SELECT c.id AS commande_id, c.date_achat, cl.prenom, cl.nom, SUM(clig.prix_total) AS
montant_total
FROM commande c
JOIN commande_ligne clig ON c.id = clig.commande_id
JOIN client cl ON c.client_id = cl.id
GROUP BY c.id, c.date_achat, cl.prenom, cl.nom;
```

6. Enregistrer le montant total dans "cache_prix_total"

```sql
UPDATE commande c
JOIN (
    SELECT commande_id, SUM(prix_total) AS montant_total
    FROM commande_ligne
    GROUP BY commande_id
) clig ON c.id = clig.commande_id
SET c.cache_prix_total = clig.montant_total;
```

7. Montant global de toutes les commandes par mois

```sql
SELECT DATE_FORMAT(date_achat, '%Y-%m') AS mois, SUM(clig.prix_total) AS montant_mensuel
FROM commande c
JOIN commande_ligne clig ON c.id = clig.commande_id
GROUP BY DATE_FORMAT(date_achat, '%Y-%m');
```

8. Les 10 clients ayant effectué les plus grands montants

```sql
SELECT cl.prenom, cl.nom, SUM(clig.prix_total) AS montant_total
FROM client cl
JOIN commande c ON cl.id = c.client_id
JOIN commande_ligne clig ON c.id = clig.commande_id
GROUP BY cl.id, cl.prenom, cl.nom
ORDER BY montant_total DESC
LIMIT 10;
```

9. Montant total des commandes pour chaque date

```sql
SELECT c.date_achat, SUM(clig.prix_total) AS montant_total
FROM commande c
JOIN commande_ligne clig ON c.id = clig.commande_id
GROUP BY c.date_achat;
```

10. Ajouter une colonne "category" à la table "commande"

```sql
ALTER TABLE commande ADD COLUMN category INT;
```

11. Mettre à jour la catégorie de chaque commande

```sql
UPDATE commande
SET category = CASE
    WHEN cache_prix_total < 200 THEN 1
    WHEN cache_prix_total BETWEEN 200 AND 500 THEN 2
    WHEN cache_prix_total BETWEEN 500 AND 1000 THEN 3
    WHEN cache_prix_total > 1000 THEN 4
END;
```

12. Créer une table "commande_category"

```sql
CREATE TABLE commande_category (
    id INT AUTO_INCREMENT PRIMARY KEY,
    description VARCHAR(255)
);
```

13. Insérer les descriptifs des catégories

```sql
INSERT INTO commande_category (description) VALUES
('Moins de 200€'),
('Entre 200€ et 500€'),
('Entre 500€ et 1000€'),
('Supérieur à 1000€');
```

14. Supprimer les commandes avant le 1er février 2019

```sql
DELETE clig
FROM commande_ligne clig
JOIN commande c ON clig.commande_id = c.id
WHERE c.date_achat < '2019-02-01';

DELETE FROM commande WHERE date_achat < '2019-02-01';
```
