Base Music - Requêtes - Correction

01 - Liste d'albums : tous les champs, tri par année de sortie croissante

```
SELECT *
FROM albums
ORDER BY albums.alb_annee ASC
```

02 - Liste d'albums : titre album et année de sortie, tri par année de sortie décroissante

```
SELECT albums.alb_titre, albums.alb_annee
FROM albums
ORDER BY albums.alb_annee DESC
```

03 – Liste d'albums : titre album, année de sortie, albums dont le titre commence par la lettre "P", tri par année de sortie croissante

```
SELECT albums.alb_titre, albums.alb_annee
FROM albums
WHERE albums.alb_titre LIKE "P%"
ORDER BY albums.alb_annee ASC
```

04 - Liste d'albums : titre album, année de sortie, albums dont le titre contient le mot "Live", tri par année de sortie croissante

```
SELECT albums.alb_titre, albums.alb_annee
FROM albums
WHERE albums.alb_titre LIKE "%Live%"
ORDER BY albums.alb_annee ASC
```

05 - Liste d'albums : titre album, année de sortie, albums dont le titre se termine par "ions", tri par année de sortie croissante

```
SELECT albums.alb_titre, albums.alb_annee
FROM albums
WHERE albums.alb_titre LIKE "%ions"
ORDER BY albums.alb_annee ASC
```

06 - Liste d'albums : titre album, année de sortie, nom de l'artiste, tri par nom d'artiste croissant

```
SELECT albums.alb_titre, albums.alb_annee, artistes.art_nom
FROM albums, artistes
WHERE albums.art_id = artistes.art_id
ORDER BY artistes.art_nom ASC
```

07 - Liste d'albums : titre album, année de sortie, nom de l'artiste, albums des "Rolling Stones" ou de "Led Zeppelin", tri par année de sortie croissante

```
SELECT albums.alb_titre, albums.alb_annee, artistes.art_nom
FROM albums, artistes
WHERE albums.art_id = artistes.art_id
AND (artistes.art_nom = "Rolling Stones" OR artistes.art_nom = "Led Zeppelin")
ORDER BY albums.alb_annee ASC
```

```
SELECT albums.alb_titre, albums.alb_annee, artistes.art_nom
FROM albums, artistes
WHERE albums.art_id = artistes.art_id
AND artistes.art_nom IN ("Rolling Stones", "Led Zeppelin")
ORDER BY albums.alb_annee ASC
```

08 – Nombre d'albums du groupe "Led Zeppelin"

```
SELECT COUNT(albums.alb_id)
FROM albums, artistes
WHERE albums.art_id = artistes.art_id
AND artistes.art_nom = "Led Zeppelin"
```

09 – Nombre d'albums sortis entre 1970 et 2000 (années incluses)

```
SELECT COUNT(albums.alb_id)
FROM albums
WHERE albums.alb_annee BETWEEN 1970 AND 2000
```

```
SELECT COUNT(albums.alb_id)
FROM albums
WHERE albums.alb_annee >= 1970 AND albums.alb_annee <= 2000
```

10 - Liste d'artistes : nom artiste, libellé de la nationalité, tri par nom artiste croissant

```
SELECT artistes.art_nom, pays.pay_libelle
FROM artistes, pays
WHERE artistes.pay_id = pays.pay_id
ORDER BY artistes.art_nom ASC
```

11 - Liste d'artistes : nom artiste, type d'artiste, artistes anglais seulement (code pays dans la table artiste = ANG), tri par type puis tri par nom artiste croissant

```
SELECT artistes.art_nom, artistes.art_typ
FROM artistes
WHERE artistes.pay_id = "ANG"
ORDER BY artistes.art_typ ASC, artistes.art_nom ASC
```

12 – Nombre d’albums par artiste (nombre et nom de l’artiste), tri par ordre décroissant du nombre d'albums

```
SELECT COUNT(albums.alb_id), artistes.art_nom
FROM albums, artistes
WHERE albums.art_id = artistes.art_id
GROUP BY albums.art_id
ORDER BY COUNT(albums.alb_id) DESC
```

13 - Nombre d'artistes par pays (nombre et libellé du pays), tri par nombre d'artistes décroissant

```
SELECT COUNT(artistes.art_id), pays.pay_libelle
FROM artistes, pays
WHERE artistes.pay_id = pays.pay_id
GROUP BY pays.pay_id
ORDER BY COUNT(artistes.art_id) DESC
```

Avec des aliases :

```
SELECT COUNT(A.art_id) 'Nombre d''artistes', P.pay_libelle 'Pays'
FROM artistes A, pays P
WHERE A.pay_id = P.pay_id
GROUP BY A.pay_id
ORDER BY COUNT(A.art_id) DESC
```

14 - Nombre d'artistes par genre (nombre et libellé du genre), tri par nombre d'artistes décroissant

```
SELECT COUNT(artistes.art_id), genres.gen_libelle
FROM artistes, genres
WHERE artistes.gen_id = genres.gen_id
GROUP BY genres.gen_id
ORDER BY COUNT(artistes.art_id) DESC
```

15 - Nombre d'albums par artiste, artistes ayant sorti au moins 5 albums, ordre décroissant du nombre d'albums

```
SELECT COUNT(albums.alb_id), artistes.art_nom
FROM albums, artistes
WHERE albums.art_id = artistes.art_id
GROUP BY artistes.art_id
HAVING COUNT(albums.alb_id) >= 5
ORDER BY COUNT(albums.alb_id) DESC
```

16 - Prix moyen des albums dans la table albums, tous artistes confondus

```
SELECT AVG(albums.alb_prix)
FROM albums
```

17 - Prix moyen des albums par artiste

```
SELECT AVG(albums.alb_prix), artistes.art_nom
FROM albums, artistes
WHERE albums.art_id = artistes.art_id
GROUP BY artistes.art_id
```

Avec arrondi :

```
SELECT ROUND(AVG(albums.alb_prix),2), artistes.art_nom
FROM albums, artistes
WHERE albums.art_id = artistes.art_id
GROUP BY artistes.art_id;
```

18 - Titre album, prix et nom artiste, des albums les plus chers

```
SELECT albums.alb_titre, albums.alb_prix, artistes.art_nom
FROM albums, artistes
WHERE albums.art_id = artistes.art_id
AND albums.alb_prix = (SELECT MAX(albums.alb_prix) FROM albums)
```

19 - Titre album, prix et nom artiste, des albums qui n'ont jamais été vendus

```
SELECT albums.alb_titre, albums.alb_prix, artistes.art_nom
FROM albums, artistes
WHERE albums.art_id = artistes.art_id
AND albums.alb_id NOT IN (SELECT ventes.alb_id FROM ventes)
```

20 - Palmarès des ventes : quantité d'albums vendus, titre album et nom d'artiste, ordre décroissant des ventes

Avec les quantités vendues uniquement :

```
SELECT COUNT(ventes.alb_id), albums.alb_titre, artistes.art_nom
FROM albums, artistes, ventes
WHERE albums.alb_id = ventes.alb_id
AND artistes.art_id = albums.art_id
GROUP BY ventes.alb_id
ORDER BY COUNT(ventes.alb_id) DESC
```

Avec le montant des ventes en plus :

```
SELECT COUNT(ventes.alb_id) AS 'Quantité vendue', SUM(ventes.ven_prix) AS 'Montant des Ventes',
albums.alb_titre, artistes.art_nom
FROM albums , artistes, ventes
WHERE albums.alb_id = ventes.alb_id
AND artistes.art_id = albums.art_id
GROUP BY ventes.alb_id
ORDER BY COUNT(ventes.alb_id) DESC
```

21 - Montant des ventes par mois

SELECT MONTH(ventes.ven_date), YEAR(ventes.ven_date), SUM(ventes.ven_prix)
FROM ventes
GROUP BY MONTH(ventes.ven_date), YEAR(ventes.ven_date)

22 - Montant des ventes par mois et par client

SELECT MONTH(ventes.ven_date), YEAR(ventes.ven_date), SUM(ventes.ven_prix), clients.cli_nom
FROM ventes, clients
WHERE ventes.cli_id = clients.cli_id
GROUP BY clients.cli_id, MONTH(ventes.ven_date), YEAR(ventes.ven_date)
ORDER BY clients.cli_id, MONTH(ventes.ven_date), YEAR(ventes.ven_date)

23 - Montant des ventes par mois et par artiste

SELECT MONTH(ventes.ven_date), YEAR(ventes.ven_date), SUM(ventes.ven_prix), artistes.art_nom
FROM ventes, artistes, albums
WHERE ventes.alb_id = albums.alb_id
AND albums.art_id = artistes.art_id
GROUP BY artistes.art_id, MONTH(ventes.ven_date), YEAR(ventes.ven_date)
ORDER BY artistes.art_id, MONTH(ventes.ven_date), YEAR(ventes.ven_date)

24 - Montant des ventes par mois, par artiste et par album

SELECT MONTH(ventes.ven_date), YEAR(ventes.ven_date), SUM(ventes.ven_prix), artistes.art_nom, albums.alb_titre
FROM ventes, artistes, albums
WHERE ventes.alb_id = albums.alb_id
AND albums.art_id = artistes.art_id
GROUP BY artistes.art_id, albums.alb_id, MONTH(ventes.ven_date), YEAR(ventes.ven_date)
ORDER BY artistes.art_id, albums.alb_id, MONTH(ventes.ven_date), YEAR(ventes.ven_date)

25 - Insertion d'un album de « Coldplay » qui n'existe pas encore dans la base

INSERT INTO albums VALUES (51, 'Nouvel album', 2023, 12, 6)

INSERT INTO albums (alb_id, alb_titre, alb_prix) VALUES (52, 'Nouvel album', 15)
Ne fonctionne pas à cause de la contrainte liée à la foreign key sur l'artiste. L'artiste devient obligatoire.

INSERT INTO albums (alb_id, art_id, alb_prix) VALUES (52, 1, 15)

Suppression :
DELETE FROM Ventes

26 - Augmentation de 10% du prix de tous les albums

```
UPDATE albums
SET albums.alb_prix = albums.alb_prix * 1.1
```

27 - Baisse de 5% du prix des albums dont le prix est supérieur ou égal à 15 euros

```
UPDATE albums
SET albums.alb_prix = albums.alb_prix * 0.95
WHERE albums.alb_prix >= 15
```

28 – Créez la requête qui affiche les noms des membres, l'instrument dont ils jouent, les groupes auxquels ils ont appartenu, les années de début et fin de participation à chaque groupe et le nombre d'années de présence de chaque membre dans un groupe (date fin - date début)

```
SELECT membres.mem_nom AS 'Nom musicien', instruments.ins_libelle AS 'Instrument',
artistes.art_nom AS 'Nom Groupe', historique.his_debut AS 'Début', historique.his_fin AS 'Fin',
historique.his_fin - historique.his_debut AS 'Années de présence'
FROM artistes, membres, instruments, historique
WHERE artistes.art_typ = "G"
AND artistes.art_id = historique.art_id
AND historique.mem_id = membres.mem_id
AND membres.ins_id = instruments.ins_id
ORDER BY membres.mem_nom, artistes.art_nom, historique.his_debut
```

29 – Créez la requête qui retrace la carrière d'un membre de groupe : nom du membre, nom du groupe auquel il a contribué, les années de début et fin de participation au groupe et le nombre d'années de présence de chaque membre dans un groupe (date fin - date début), tri par nom du membre puis année début croissante.

```
SELECT membres.mem_nom AS 'Nom musicien', artistes.art_nom AS 'Nom Groupe',
historique.his_debut AS 'Début', historique.his_fin AS 'Fin', historique.his_fin - historique.his_debut AS
'Années de présence'
FROM membres, artistes, historique
WHERE artistes.art_id = historique.art_id
AND historique.mem_id = membres.mem_id
ORDER BY membres.mem_nom, historique.his_debut
```

30 – quels sont les groupes qui ont eu au moins 3 membres, tri par ordre alphabétique du nom de groupe ?

```
SELECT artistes.art_nom AS 'Nom Groupe', COUNT(membres.mem_id) AS 'Nombre de membres'
FROM artistes, membres, historique
WHERE artistes.art_id = historique.art_id
AND historique.mem_id = membres.mem_id
GROUP BY artistes.art_id
HAVING COUNT(membres.mem_id) >= 3
ORDER BY artistes.art_nom
```

Vues

1.1 Créez une vue cli_view1 de la table client qui ne présente que le nom et l'adresse mail, tri par nom croissant

CREATE VIEW cli_view1 AS
SELECT clients.cli_nom, clients.cli_email FROM clients ORDER BY clients.cli_nom

1.2 Créez une autre vue cli_view2 identique à la précédente mais qui en plus renomme les colonnes d'origine

CREATE VIEW cli_view2 (nom, email) AS
SELECT clients.cli_nom, clients.cli_email FROM clients ORDER BY clients.cli_nom

1.3 Créez une vue de la table albums qui ne présente que le titre et le prix (en renommant les colonnes d'origine) des albums dont le prix est inférieur ou égal à 15 euros, tri par prix décroissant

CREATE VIEW alb_view1 (titre, prix) AS
SELECT albums.alb_titre, albums.alb_prix
FROM albums
WHERE albums.alb_prix <= 15
ORDER BY albums.alb_prix DESC

1.4 Créez une vue qui présente les informations suivantes :

- Nom de l'artiste
- Libellé complet du pays de l'artiste
- Libellé complet du genre de l'artiste
- Tri par nom d'artiste croissant
- Renommez les colonnes d'origine

CREATE VIEW art_view1(Nom, Pays, Genre) AS
SELECT artistes.art_nom, pays.pay_libelle, genres.gen_libelle
FROM artistes, pays, genres
WHERE artistes.pay_id = pays.pay_id
AND artistes.gen_id = genres.gen_id
ORDER BY artistes.art_nom ASC

Comment voir les vues ?

*SELECT * FROM information_schema.views*

Triggers

2.1 Ecrivez le trigger qui modifiera la colonne cli_ca de la table clients **à chaque création d'une nouvelle vente.**

```
DROP TRIGGER IF EXISTS upca;
DELIMITER //
CREATE TRIGGER upca
AFTER INSERT ON ventes
FOR EACH ROW
UPDATE clients
SET clients.cli_ca = clients.cli_ca + new.ven_prix
WHERE clients.cli_id = new.cli_id
//
DELIMITER ;
```

2.2 Ecrivez le trigger qui modifiera cette colonne à **chaque suppression d'une vente**.

```
DROP TRIGGER IF EXISTS upcadel;
DELIMITER //
CREATE TRIGGER upcadel
AFTER DELETE ON ventes
FOR EACH ROW
UPDATE clients
SET clients.cli_ca = clients.cli_ca - old.ven_prix
where clients.cli_id = old.cli_id
//
DELIMITER ;
```

2.3 Ecrivez le trigger qui modifiera cette colonne à **chaque modification d'une vente**.

```
DROP TRIGGER IF EXISTS upcamod;
DELIMITER //
CREATE TRIGGER upcamod
AFTER UPDATE ON ventes
FOR EACH ROW
UPDATE clients
SET clients.cli_ca = clients.cli_ca - old.ven_prix + new.ven_prix
WHERE clients.cli_id = new.cli_id
//
DELIMITER ;
```

Comment voir les triggers ?

```
SHOW TRIGGERS
```

Ou

```
SELECT trigger_schema, trigger_name, action_statement
FROM information_schema.triggers
```

Délimiteurs

Ce que l'on appelle délimiteur, c'est (par défaut), le caractère **;**

C'est le caractère qui permet de délimiter les instructions.

Il est possible de définir le délimiteur manuellement, pour que **;** ne signifie plus qu'une instruction se termine. Ainsi le caractère **;** pourra être utilisé à l'intérieur d'une instruction, et donc **dans le corps d'une procédure stockée.**

Pour changer le délimiteur, il suffit d'utiliser cette commande :
DELIMITER **|**

À partir de maintenant, vous devrez utiliser le caractère **|** pour signaler la fin d'une instruction.
; ne sera plus compris comme tel par votre session.

DELIMITER n'agit que **pour la session courante**.

Vous pouvez utiliser le ou les caractères de votre choix comme délimiteur, à condition de choisir quelque chose qui ne risque pas d'être utilisé dans une instruction. Il faut donc éviter les lettres, les chiffres, le @ qui sert pour les variables utilisateurs et le \ qui sert à échapper les caractères spéciaux.

Les deux délimiteurs suivants sont les plus couramment utilisés :
DELIMITER **//**
DELIMITER **|**