

EXO : soit les relations :

Appart (NomIm, NoApp, Superficie, Etage, LivretFoncier)

```
CREATE TABLE Appart
(
  NomIm VARCHAR (20) NOT NULL,
  NoApp INT NOT NULL,
  Superficie INT NOT NULL,
  Etage INT NOT NULL,
  LivretFoncier VARCHAR (10) NOT NULL CHECK (LivretFoncier IN ('OUI', 'NON')),
  PRIMARY KEY (NomIm, NoApp)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Le champ «LivretFoncier » dans la table « Appart » permet de savoir si oui ou non, un appartement dispose d'un livret foncier dans la conservations foncières (pour ceux qui ne le savent pas, un livret foncier est un document délivré par la conservations foncières au propriétaire d'un bien immobilier, afin de lui permettre de prouver son droit de propriété et prouver l'authenticité de la propriété par le cadastre). Nous avons utilisé la commande CHECK pour limiter la plage de valeurs qui peut être placée dans une colonne «LivretFoncier » de la table « Appart » à deux valeurs uniquement (oui ou non).

Ainsi l'insertion suivante dans la table « Appart » génère une erreur :

```
INSERT INTO Appart (NomIm, NoApp, Superficie, Etage, LivretFoncier)
VALUES
('Imb1', 1, 150, 14, 'O');
```

Par contre les insertions suivantes dans la table « Appart » ne génèrent aucune erreur:

```
INSERT INTO Appart (NomIm, NoApp, Superficie, Etage, LivretFoncier)
VALUES
('Imb1', 1, 150, 14, 'OUI'),
('Imb1', 34, 50, 15, 'OUI'),
('Imb1', 51, 200, 2, 'OUI'),
('Imb1', 52, 50, 5, 'NON'),
('Imb2', 1, 250, 1, 'OUI'),
('Imb2', 2, 250, 2, 'OUI');
```

Personne (Nom, Age, Profession)

```
CREATE TABLE Personne
(
  Nom VARCHAR (10) NOT NULL,
  Age INT NOT NULL CHECK (Age>0 AND Age<=110),
  Profession VARCHAR (15) NOT NULL,
```

```
PRIMARY KEY (Nom)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Nous avons limité la tranche d'âge de la colonne « Age » dans la table « Personne » à l'intervalle] 0,110] par la commande **CHECK** qui n'autorise que certaines valeurs pour cette colonne. Ainsi l'insertion suivante dans la table « Personne » génère une erreur :

```
INSERT INTO Personne (Nom, Age, Profession)
VALUES
('Imad', 200, 'Informat');
```

Par contre, les insertions suivantes dans la table « Personne » ne génèrent aucune erreur :

```
INSERT INTO Personne (Nom, Age, Profession)
VALUES
('Houari', 51, 'Informat'),
('Youcef', 34, 'Cadre'),
('Djamila', 23, 'Stagiaire'),
('Mourad', 52, 'Acteur'),
('Hassiba', 34, 'Médecin');
```

Immeuble (**NomIm**, Adresse, NbEtage, Année-Construction, NomGérant)

```
CREATE TABLE Immeuble
(
  NomIm VARCHAR (20) NOT NULL,
  Adresse TEXT NOT NULL,
  NbEtage INT NOT NULL,
  AnnéeConstruction YEAR (4) NOT NULL,
  NomGérant VARCHAR (20) NOT NULL,
  PRIMARY KEY (NomIm)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
INSERT INTO Immeuble (NomIm, Adresse, NbEtage, AnnéeConstruction, NomGérant)
VALUES
('Imb1', 'Alger centre', 15, '1975', 'Hassiba'),
('Imb2', 'Kouba', 20, '1973', 'Houari');
```

Occupant (**#NomImO, #NoAppO, #NomOc**, AnneeArv)

```
CREATE TABLE Occupant
(
  NomImO VARCHAR (20) NOT NULL,
  NoAppO INT NOT NULL,
  NomOc VARCHAR (20) NOT NULL,
  AnneeArv INT NOT NULL CHECK ((AnneeArv >=1990) AND (AnneeArv <=2000)),
)
```

PRIMARY KEY (NomImO, NoAppO, NomOc),

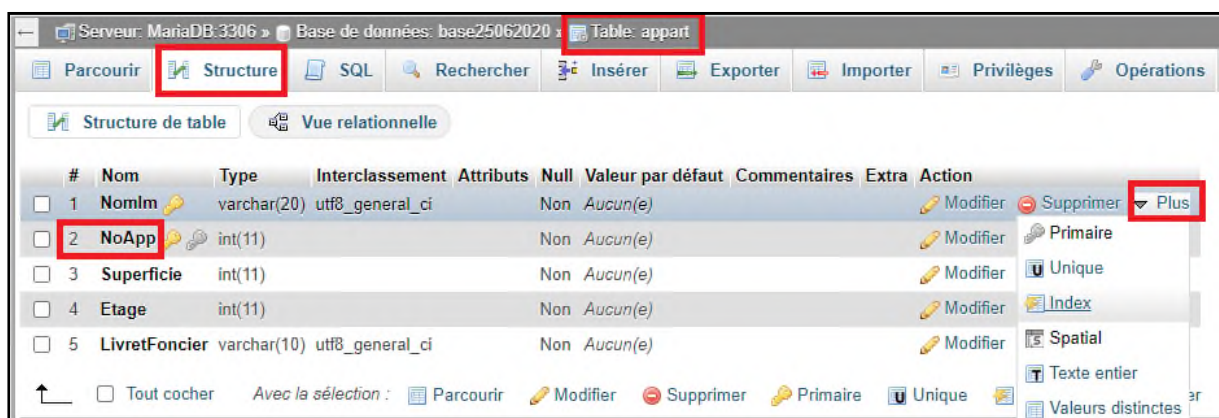
FOREIGN KEY (NomImO) **REFERENCES** Immeuble (NomIm),

FOREIGN KEY (NoAppO) **REFERENCES** Appartement (NoApp)

) **ENGINE=InnoDB DEFAULT CHARSET=utf8;**

Remarque: lors de la création de la table « Occupant » vous aurez certainement une erreur. En effet, la table « Occupant » fait référence à la table « Appartement » et plus particulièrement à l'attribut clé « NoApp ». Cependant la clé primaire de la table « Appartement » est à double colonne (la clé primaire est composée de deux attributs : NomIm et NoApp.), ce qui implique que vous ne pouvez pas référencer uniquement la 2eme colonne (NoApp) de la clé primaire par la clé étrangère «NoAppO» (en d'autre terme, ce problème est due au fait que la colonne référencée n'est pas une clé primaire mais seulement une partie de la clé primaire). J'ai donc ajouté mon propre index (la colonne référencée doit être indexée si elle ne forme pas à elle seule une clé unique) pour la 2eme colonne (NoApp) de la clé primaire dans la table « Appartement » pour corriger l'erreur provoquée lors de la création de la table « Occupant » (en ajoutant des index uniques individuels pour les colonnes qui forment la clé composite, vous pouvez résoudre cette erreur). **Règle à retenir:** les attributs formant la clé primaire doivent être indexés séparément pour pouvoir être référencé par des clés étrangères (s'assurer que chaque colonne à laquelle vous essayez d'ajouter une relation de clé étrangère est indexée. Une fois les deux colonnes de la clé primaire indexées, il est possible de les référencer avec des clés étrangères. Rappelons qu'une clé étrangère est un champ dans une table qui identifie de manière unique une ligne d'une autre table. La table contenant la clé étrangère est appelée la table enfant et la table contenant la clé candidate est appelée la table référencée ou parent).

Pour indexer l'attribut « NoApp » qui forme seulement une partie de la clé primaire de la table « Appartement », allez à la vue globale de la table « Appartement » (dans l'interface de PHPMyAdmin, volet Structure, sélectionnez la colonne « NoApp », option « Plus » et indexez l'attribut référencé « NoApp » :



Nous avons limité l'année d'arrivée d'un occupant à l'intervalle [1990, 2000]. Ainsi les deux insertions suivantes dans la table « Occupant » génèrent des erreurs :

```
INSERT INTO Occupant (NomImO, NoAppO, NomOc, AnneeArv)
VALUES ('Imb1', 1, 'Salim', 1989);
```

```
INSERT INTO Occupant (NomImO, NoAppO, NomOc, AnneeArv)
VALUES ('Imb1', 1, 'Salim', 2010);
```

Par contre, les insertions suivantes dans la table « Occupant » ne génèrent aucune erreur :

```
INSERT INTO Occupant (NomImO, NoAppO, NomOc, AnneeArv)
VALUES
('Imb1', 1, 'Djamila', 1992),
('Imb2', 1, 'Hassiba', 1994),
('Imb2', 2, 'Houari', 1994),
('Imb1', 51, 'Mourad', 1996),
('Imb1', 34, 'Youcef', 1993) ;
```

Donner le code SQL des requêtes suivantes :

1) Donner l'habitant le plus ancien dans l'immeuble 1.

Occupant (NomImO, NoAppO, NomOc, AnneeArv)

```
SELECT NomOc
FROM Occupant
WHERE NomImO = 'Imb1'
AND AnneeArv = (SELECT MIN(AnneeArv)
                FROM Occupant
                WHERE NomImO = 'Imb1');
```

2) Donner les occupants ordonnés par leur année d'arrivée (Décroissant) et noms (croissant).

Occupant (NomImO, NoAppO, NomOc, AnneeArv)

```
SELECT AnneeArv, NomOc
FROM Occupant
ORDER BY AnneeArv DESC, NomOc ASC;
```

3) Donner le nombre des habitants de chaque immeuble.

Occupant (NomImO, NoAppO, NomOc, AnneeArv)

```
SELECT NomImO, COUNT(NomOc)
FROM Occupant
GROUP BY NomImO;
```

4) Donner le nombre des habitants arrivant en 1994 pour chaque immeuble.

Occupant (NomImO, NoAppO, NomOc, AnneeArv)

```
SELECT NomImO, COUNT(NomOc)
FROM Occupant
WHERE AnneeArv=1994
GROUP BY NomImO;
```

5) Donner l'immeuble qui a plus que deux appartements occupés.

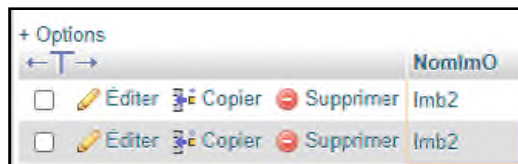
Occupant (NomImO, NoAppO, NomOc, AnneeArv)

```
SELECT NomImO, COUNT(NoAppO)
FROM Occupant
GROUP BY NomImO
HAVING COUNT(NoAppO) > 2;
```

6) Donner les immeubles où tout le monde a emménagé en 1994.

Occupant (NomImO, NoAppO, NomOc, AnneeArv)

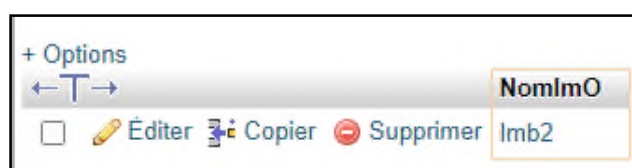
```
SELECT NomImO
FROM Occupant
WHERE NomImO NOT IN (SELECT NomImO
                     FROM Occupant
                     WHERE Anneearv !=1994);
```



+ Options				NomImO
← T →				
<input type="checkbox"/>	✎ Éditer	📄 Copier	🗑 Supprimer	Imb2
<input type="checkbox"/>	✎ Éditer	📄 Copier	🗑 Supprimer	Imb2

Pour éviter qu'un même immeuble apparaisse dans l'affichage :

```
SELECT DISTINCT NomImO
FROM Occupant
WHERE NomImO NOT IN (SELECT NomImO
                     FROM Occupant
                     WHERE Anneearv !=1994);
```



+ Options				NomImO
← T →				
<input type="checkbox"/>	✎ Éditer	📄 Copier	🗑 Supprimer	Imb2

7) Donner les Immeubles (avec les noms des occupants) occupés après 1993 (un immeuble est dit occupé s'il compte au moins un occupant).

Occupant (NomImO, NoAppO, NomOc, AnneeArv)

```
SELECT NomImO, NomOc
FROM Occupant
WHERE NomImO NOT IN (SELECT NomImO
                      FROM Occupant
                      WHERE AnneeArv <=1993);
```

Rappel (Sous-requêtes avec la clause ANY):

Syntaxe:

<Opérande> <opérateur de comparaison> ANY (<sous-requête>)

Le mot **ANY**, qui doit suivre immédiatement un opérateur de comparaison (l'opérateur de comparaison peut être =, <, >, <>, !=, <=, >=,...etc.), signifie : retourne **VRAI** si la comparaison est **VRAI** pour **UNE** des lignes que la sous-requête. Cette commande s'utilise dans une clause conditionnelle juste après un opérateur conditionnel et juste avant une sous-requête.

Exemple:

```
SELECT s1 FROM t1 WHERE s1 > ANY (SELECT s1 FROM t2);
```

Cette requête peut se traduire de la façon suivante : sélectionner toutes les lignes de table1, où la condition (supérieure) est vérifiée pour n'importe quel résultat de la sous-requête.

Supposons qu'il y ait une ligne dans la table t1 qui contienne {10}. L'expression est **VRAI** si la table t2 contient {21,14, 7} car il y a une valeur de t2, 7, qui est inférieure à 10. Cette expression est **FAUSSE** si la table t2 contient {20,10}.

L'exemple ci-après montre une utilisation de la commande **ANY** dans une requête SQL et qui à le même sens que la commande **IN**:

```
SELECT s1 FROM t1 WHERE s1 = ANY (SELECT s1 FROM t2);
```

Et

```
SELECT s1 FROM t1 WHERE s1 IN (SELECT s1 FROM t2);
```

Ces deux commandes sont identiques.

Ainsi, dans le langage SQL, la commande **ANY** permet de comparer une valeur avec le résultat d'une sous-requête. Il est donc possible de vérifier si cette valeur est "égale", "différente", "supérieur", "supérieur ou égale", "inférieur" ou "inférieur ou égale" pour au moins une des valeurs de la sous-requête.

8) Donner le dernier occupant par année de l'immeuble 1.

Occupant (NomImO, NoAppO, NomOc, AnneeArv)

```
SELECT NomOc
FROM Occupant
WHERE NomImO= "Imb1"
AND AnneeArv >= ALL (SELECT AnneeArv
                     FROM Occupant
                     WHERE NomImO= "Imb1");
```

Ou bien :

```
SELECT NomOc
FROM Occupant
WHERE NomImO= "Imb1"
AND AnneeArv = (SELECT MAX(AnneeArv)
               FROM Occupant
               WHERE NomImO= "Imb1");
```

9) Donner les immeubles où personne n'a emménagé en 1996.

Occupant (NomImO, NoAppO, NomOc, AnneeArv)

```
SELECT DISTINCT NomImO
FROM Occupant
WHERE NomImO NOT IN (SELECT NomImO
                    FROM Occupant
                    WHERE Anneearv = 1996);
```

Rappel (Les Vues):

Dans une base de données MySQL, les données sont stockées dans des tables. Il est possible de rassembler des informations provenant de plusieurs tables en utilisant des jointures entre tables. Il existe d'autres sources de données que les tables dans MySQL: les vues.

Une vue est une sorte de table virtuelle qui ne contient en réalité que le résultat d'une requête SQL (c'est-à-dire dont les données ne sont pas stockées dans une table de la base de données). Une vue s'utilise exactement comme une table dans les requêtes SQL et facilite l'écriture des requêtes complexes. Pour créer une vue, on utilise une requête SQL avec la clause **CREATE VIEW** (à la place de **CREATE TABLE** pour les tables). Les données présentées dans une vue sont définies grâce à une clause **SELECT**.

La syntaxe de création d'une vue en SQL : La création d'une vue se fait grâce à la clause **CREATE VIEW** suivie du nom que l'on attribut à la vue, puis du nom des colonnes dont on désire agrémenter cette vue, puis enfin d'une clause **AS** précédant la sélection. La syntaxe d'une vue ressemble à :

```
CREATE VIEW NomVue (colonne1, colonne2,..., colonneN)
AS (SELECT colonne1, colonne2,....., colonneN
    FROM .....
    WHERE.....
    .....
    .....
    .....);
```

Les vues ainsi créées peuvent être l'objet de nouvelles requêtes en précisant le nom de la vue au lieu d'un nom de table dans une requête. Techniquement les vues ne stockent pas les données qu'elles contiennent mais conservent juste la requête permettant de les créer.

10) Qui n'habite pas un appartement dans un Immeuble qu'il gère lui-même ?

Occupant (NomImO, NoAppO, NomOc, AnneeArv)

Immeuble (NomIm, Adresse, NbEtag, Année-Construction, NomGérant)

```
CREATE VIEW Vue1 (NomOccupant)
AS (SELECT Occupant.NomOc
    FROM Occupant, Immeuble
    WHERE Occupant.NomImO = Immeuble.NomIm
    AND Occupant.NomOc != Immeuble. NomGérant);
```

11) Qui n'habite pas un appartement dans un immeuble géré par Houari ?

Occupant (NomImO, NoAppO, NomOc, AnneeArv)

Immeuble (NomIm, Adresse, NbEtag, Année-Construction, NomGérant)

```
CREATE VIEW Vue2 (NomOccupant)
AS (SELECT NomOc
    FROM Occupant
    WHERE NomImO NOT IN (SELECT NomIm
                        FROM Immeuble
                        WHERE NomGérant='Houari'));
```

Ou bien :

```
CREATE VIEW Vue22 (NomOccupant)
AS (SELECT Occupant.NomOc
    FROM Occupant, Immeuble
    WHERE Occupant.NomImO = Immeuble.NomIm
    AND Immeuble. NomGérant!= 'Houari');
```


12) Qui gère l'immeuble où habite Djamilia ?

Occupant (NomImO, NoAppO, NomOc, AnneeArv)

Immeuble (NomIm, Adresse, NbEtage, Année-Construction, NomGérant)

```
CREATE VIEW Vue3 (NomGérant)
AS (SELECT NomGérant
    FROM Immeuble
    WHERE NomIm IN (SELECT NomImO
                    FROM Occupant
                    WHERE NomOc = 'Djamila'));
```

Ou bien :

```
CREATE VIEW Vue33 (NomGérant)
AS (SELECT Immeuble.NomGérant
    FROM Occupant, Immeuble
    WHERE Occupant.NomImO = Immeuble.NomIm
    AND Occupant.NomOc = 'Djamila');
```

EXO Révisions 1 :

Soit le schéma de la base de données **Cinéma** :

FILM (NumF, Titre, GenreF, AnneeR, DureeF, NomRealisateur)

SeanceProjection (Titre, NomSalle, HeureDébut, Version)

JOUE (NomActeur, Titre, Salaire)

PERSONNE (NumP, NomP, PrénomP, DateNais)

On suppose que tout individu est identifié par son Nom.

Donnez les requêtes SQL permettant de répondre aux questions suivantes :

1/- Donnez la liste de tous les films (avec tous leurs caractéristiques):

```
SELECT *
FROM FILM ;
```

2/- Donnez le titre de tous les films dont la durée dépasse 180 minute :

```
SELECT Titre
FROM FILM
WHERE DureeF > 180 ;
```

3/-Quelle est la durée moyenne des films réalisés par « **Djaafar Kassem** » :

```
SELECT AVG(DureeF)
FROM FILM
WHERE NomRealisateur='Djaafar Kassem';

Ou bien

SELECT AVG(DureeF)
FROM FILM
WHERE NomRealisateur LIKE '%Djaafar%Kassem%';
```

4/- Quelle est la durée totale des films dans lesquels jouent « **Malika Belbey** »

```
SELECT SUM(DureeF)
FROM FILM, JOUE
WHERE (FILM.Titre = JOUE.Titre) AND (JOUE.NomActeur ='Malika Belbey');
```

Ou bien

```
SELECT SUM(DureeF)
FROM FILM
WHERE Titre IN (SELECT Titre
                FROM JOUE
                WHERE NomActeur ='Malika Belbey');
```

5/-Quels sont les titres des films de durée la plus courte ?

```
SELECT Titre
FROM FILM
WHERE DureeF = (SELECT MIN (DureeF)
               FROM FILM);
```

Ou bien

```
SELECT Titre
FROM FILM
WHERE DureeF <= ALL (SELECT DureeF
                    FROM FILM);
```

6/-Pour chaque réalisateur, donner la liste de ses films diffusés en version originale.

```
SELECT NomRealisateur, Titre
FROM FILM
WHERE Titre IN (SELECT Titre
                FROM SeanceProjection
```

```
WHERE Verion='Originale') ;  
GROUP BY NomRealisateur;
```

Ou bien

```
SELECT NomRealisateur, Titre  
FROM FILM, SeanceProjection  
WHERE (FILM.Titre= SeanceProjection.Titre)  
AND (SeanceProjection.Verion='Originale')  
GROUP BY NomRealisateur;
```

7/-Pour chaque film, donner la liste de ses acteurs.

```
SELECT Titre, NomActeur  
FROM JOUE  
GROUP BY Titre;
```

8/-Donner la liste les films ayant fait jouer moins de dix acteur.

```
SELECT Titre, COUNT(NomActeur)  
FROM JOUE  
GROUP BY Titre  
HAVING COUNT(NomActeur)<10 ;
```

9/-Quels sont les films (identifiant et titre) dans lesquelles l'actrice « **Souhila Mallem** » joue avec « **Youcef Sehaïri** » ?

```
SELECT FILM.NumF, FILM.Titre  
FROM FILM, JOUE  
WHERE (FILM.Titre = JOUE.Titre) AND (JOUE.NomActeur ='Souhila Mallem');  
INTERSECT  
SELECT FILM.NumF, FILM.Titre  
FROM FILM, JOUE  
WHERE (FILM.Titre = JOUE.Titre) AND (JOUE.NomActeur ='Youcef Sehaïri');
```

10/-Donner le nombre de films par genre.

```
SELECT GenreF, COUNT(*)  
FROM FILM  
GROUP BY GenreF ;
```

11/-Trouver le ou les titre (s) et l'année ou les années de réalisation du ou des film (s) le (s) plus long (s).

```
SELECT Titre, AnneeR
FROM FILM
WHERE DureeF = (SELECT MAX (DureeF)
                FROM FILM) ;
```

Ou bien

```
SELECT Titre, AnneeR
FROM FILM
WHERE DureeF >= ALL (SELECT DureeF
                     FROM FILM) ;
```

12/-Trouvez le nom des personnes qui sont ni acteurs ni réalisateur

```
SELECT DISTINCT NomP, PrénomP
FROM PERSONNE
WHERE NomP NOT IN (SELECT NomRealisateur
                   FROM FILM
                   UNION
                   SELECT NomActeur
                   FROM JOUE) ;
```

DISTINCT : compter une seule fois la même valeur.

13/- Trouvez le nom et prénom des réalisateurs qui ont joué dans au moins un de leurs propres films.

```
SELECT DISTINCT PERSONNE.NomP, PERSONNE.PrénomP
FROM PERSONNE, FILM, JOUE
WHERE (PERSONNE.NomP = FILM.NomRealisateur)
AND (FILM.Titre= JOUE.Titre)
AND (FILM.NomRealisateur = JOUE.NomActeur);
```

Ou bien

```
SELECT DISTINCT NomP, PrénomP
FROM PERSONNE
WHERE NomP IN (SELECT NomRealisateur
               FROM FILM
               WHERE (Titre, NomRealisateur) IN (SELECT Titre, NomActeur
                                                  FROM JOUE));
```

14/- Quel est le total des salaires de acteurs du film « **Ouled lahlal** ».

```
SELECT SUM (Salaire)
FROM JOUE
WHERE Titre = 'Ouled lahlal' ;
```

15/- Pour chaque film de « **Djaafar Kassem** » (Titre et Année), donner le total des salaires des acteurs.

```
SELECT JOUE.Titre, SUM(JOUE.Salaire)
FROM JOUE, FILM
WHERE JOUE.Titre = FILM.Titre AND FILM. NomRealisateur ='Djaafar Kassem' ;
GROUP BY JOUE.Titre;
```

Ou bien

```
SELECT Titre, SUM (Salaire)
FROM JOUE
WHERE Titre IN (SELECT Titre
                FROM FILM
                WHERE NomRealisateur ='Djaafar Kassem') ;
GROUP BY Titre;
```

16/- Quels sont les acteurs qui jouent dans plus d'un film.

```
SELECT NomActeur, COUNT (Titre)
FROM JOUE
GROUP BY NomActeur
HAVING COUNT (Titre)>2;
```

17/- Quels sont les salles de cinémas qui programment le plus grand nombre de films.

```
SELECT NomSalle, COUNT (Titre)
FROM SeanceProjection
GROUP BY Nom Salle
HAVING COUNT(Titre) = (SELECT MAX(NbrFilm)
                      FROM (SELECT NomSalle, COUNT (Titre) AS NbrFilm
                            FROM SeanceProjection
                            GROUP BY Nom Salle) AS MaSousRequête);
```

EXO Révisions 2 :

Soit le schéma relationnel suivant:

COUREUR (numLicence, Nom, Prénom, DateNaissance)

RESULTAT (numCourse, numLicence, temps, rang)

Exprimer en SQL les requêtes suivantes:

1/- nom, prénom, temps moyen des coureurs nés avant le 1/1/1970, triés par temps moyen croissant.

```
SELECT Nom, Prénom, AVG (temps)
FROM COUREUR, RESULTAT
WHERE COUREUR.numLicence = RESULTAT.numLicence
AND DateNaissance < '1970-01-01'
GROUP BY Nom, Prénom
SORT BY AVG (temps);
```

2/- numéro de licence et temps des coureurs arrivés dans les 10 premiers.

```
SELECT numLicence, temps
FROM RESULTAT
WHERE rang < 11;
```

EXO Révisions 3 :

On considère le schéma relationnel suivant, modélisant une base de recettes de cuisine
TypesIngrédients (NuméroType, DescriptionType) : contenant par exemple le tuple (2, viande)

Ingrédients (NumIngrédient, NomIngrédient, NumTypeIngrédient, NumTypeMesure) : contenant par exemple le tuple (3, CACAO, 2, 5) le numéro de type de mesure décrit l'unité de mesure utilisée pour cet ingrédient.

Mesures (NumTypeMesure, DescriptionMesure) contenant par exemple le tuple (5, gramme)

IngrédientsRecette(NumRecette, NumIngrédient, NumOrdreIngrédient, Quantité) contenant par exemple le tuple (12, 3, 1, 500)

Recettes (NumRecette, NomRecette, NumTypeRecette, DuréePréparation, NiveauDifficulté) contenant par exemple le tuple (12, tarte aux pommes, 1, 45, 1)

TypeRecettes (NumTypeRecette, DescriptionTypeRecette) contenant par exemple (1, plat principal)

Pour chacune des requêtes ci-dessous, donnez une écriture SQL :

1/-La quantité de sucre incorporée dans la recette 'Gâteau au chocolat'.

```
SELECT IngrédientsRecette.Quantité, Mesures.DescriptionMesure)
FROM Recettes, IngrédientsRecette, Ingrédients, Mesures
```

```
WHERE Mesures.NumTypeMesure = Ingrédients.NumTypeMesure
AND Ingrédients.NumIngrédient = IngrédientsRecette.NumIngrédient
AND IngrédientsRecette.NumRecette = Recettes.NumRecette
AND Recettes.NomRecette = 'Gâteau au chocolat'
AND Ingrédients.NomIngrédient = 'Sucre' ;
```

2/-Liste des ingrédients (nom) en ordre alphabétique qui ne sont utilisés dans aucune recette.

```
SELECT NomIngrédient
FROM Ingrédients
WHERE NumIngrédient NOT IN (SELECT NumIngrédient
                             FROM IngrédientsRecette)
ORDER BY NomIngrédient ;
```

3/-Le nombre de recettes par type de recettes.

```
SELECT TypeRecettes.DescriptionTypeRecette, COUNT(Recettes.NumRecette)
FROM TypeRecettes, Recettes
WHERE TypeRecettes.NumTypeRecette = Recettes.NumTypeRecette
GROUP BY TypeRecettes.DescriptionTypeRecette ;
```

4/-Les recettes qui ont la même durée que la recette de « tarte aux pommes »

```
SELECT NomRecette
FROM Recettes
WHERE DuréePréparation = (SELECT DuréePréparation
                           FROM Recettes
                           WHERE NomRecette = 'tarte aux pommes') ;
```

5/- Expliquez ce que présente chacune des requêtes suivantes :

```
SELECT Recettes. NomRecette
FROM Recettes, IngrédientsRecette, Ingrédients
WHERE Recettes. NumRecette = IngrédientsRecette. NumRecette
AND IngrédientsRecette.NumIngrédient = Ingrédients.NumIngrédient
AND Ingrédients.NomIngrédient = 'sucre'
AND IngrédientsRecette.Quantité = (SELECT MAX(IngrédientsRecette.quantité)
                                   FROM IngrédientsRecette, Ingrédients
                                   WHERE Ingrédients.NumIngrédient =
IngrédientsRecette. NumIngrédient
AND Ingrédients.NomIngrédient = 'sucre') ;
```

Réponse : le nom des recettes qui utilise le plus de sucre.

```

SELECT COUNT(*)
FROM Recettes
WHERE NumTypeRecette = (SELECT NumTypeRecette
                        FROM TypeRecettes
                        WHERE DescriptionTypeRecette = 'plat principal') ;

```

Réponse : c'est le nombre des recettes qui sont considérées comme un plat principal.

EXO Révisions 4 :

Soit le schéma relationnel suivant:

Ville (CodeVille, NomVille, CodePostale, RecetteImpot)

Client (CodeClient, NomClient, PrénomClient, DateNaissance, Fax, #CodeVille)

1/- Liste des villes par ordre alphabétique croissant des villes.

```

SELECT *
FROM Ville
ORDER BY ville ASC;

```

2/- Liste des codes postaux et villes triée par code postal croissant.

```

SELECT CodePostale, NomVille
FROM Ville
ORDER BY CodePostale ASC;

```

3/- Liste des clients par ordre décroissant des noms.

```

SELECT NomClient, PrénomClient
FROM Client
ORDER BY NomClient DESC;

```

4/- Liste des prénoms, noms et date de naissance des clients de plus de 40 ans.

```

SELECT NomClient, PrénomClient
FROM Client
WHERE (YEAR (CURDATE()) - YEAR(DateNaissance)) > 40 ;

```

5/- Liste des villes (ville) par ordre croissant pour lesquelles il n'y a aucun client.

```

SELECT NomVille
FROM Ville

```



```
WHERE CodeVille NOT IN (SELECT CodeVille FROM Client)
ORDER BY NomVille ASC;
```

6/- Le nombre de clients par code de ville

```
SELECT CodeVille, COUNT(NomClient)
FROM Client
GROUPE BY CodeVille ;
```

7/- Donner le nombre de clients qui possède un fax :

```
SELECT COUNT(Fax)
FROM Client ;
```

8/- Donner le pourcentage de clients qui possède un fax :

```
SELECT COUNT(Fax)* 100 /COUNT(CodeClient)
FROM Client ;
```

9/- Quelles sont les villes qui récoltent une recette d'impôt plus grande que celle récoltée par la ville de Bejaïa.

```
SELECT CodeVille, NomVille
FROM Ville
WHERE RecetteImpot > (SELECT RecetteImpot
                      FROM Ville
                      WHERE NomVille= 'Bejaïa') ;
```

10/- Quelle est la ville qui récolte le moins d'impôts.

```
SELECT CodeVille, NomVille
FROM Ville
WHERE RecetteImpot = (SELECT MIN (RecetteImpot)
                     FROM Ville) ;
```

11/- Donnez la moyenne des recettes d'impôts pour l'ensemble des villes.

```
SELECT AVG(RecetteImpot)
FROM Ville ;
```

12/- quelles sont les villes qui ont une recette d'impôts supérieur à la moyenne

```
SELECT CodeVille, NomVille
FROM Ville
WHERE RecetteImpot > (SELECT AVG(RecetteImpot)
                      FROM Ville) ;
```

EXO Révisions 5 :

Soit la relations :

GUIDE-RESTAURANT (NOM-REST, RUE, SPECIALITE, PRIX-MOY)

1/-donnez la liste des prix moyens des repas par spécialité de restaurant :

```
SELECT SPECIALITE, NOM-REST, PRIX-MOY
FROM GUIDE-RESTAURANT
GROUP BY SPECIALITE;
```

2/- donnez les restaurant ayant le prix moyen des repas les plus élevé

```
SELECT NOM-REST
FROM GUIDE-RESTAURANT
WHERE PRIX-MOY = (SELECT MAX (PRIX-MOY)
                 FROM GUIDE-RESTAURANT) ;
```

**BON COURAGE
ET
A BIENTOT INCHALAH**