Note Pays mont modifié job 4

A vérifier gdp

Notes sur le sujet SQL :

Job 8.12 On nous demande de créer une variable departement\_id alors que le premier e du mot n’est pas présent dans les autres tables ou le mot department est présent.

**Veille … ZZZ**

**A. Qu’est ce qu’une donnée ? Sous quelle forme peut-elle se présenter ?**

Une donnée, c'est une information stockée dans un ordinateur, elle peut prendre la forme de texte, des chiffres, de vidéos, des images, des fichiers etc …

**B. Donnez et expliquez les critères de mesure de qualité des données.**

Les critères de mesure de qualité des données sont la précision, la fiabilité et l’exploitabilité.

Il est important de pouvoir accéder à des informations pertinentes rapidement sans nécessiter une modification de la structure de la donnée à chaque requête.

Les données doivent être fiable, il faut donc s’assurer de leur provenance, de leur véracité, et préserver leur intégrité.

Les données doivent également être complètes, idéalement depuis leur récolte mais l’absence de données dans certaines colonnes peut également être interprété par l’analyste et comblé par différentes solutions comme une classification binaire, l’utilisation de moyenne etc …

Il ne faut cependant pas abuser de ces méthodes pour éviter de dénaturer la donnée et admettre qu’elle est inexploitable si les informations manquent.

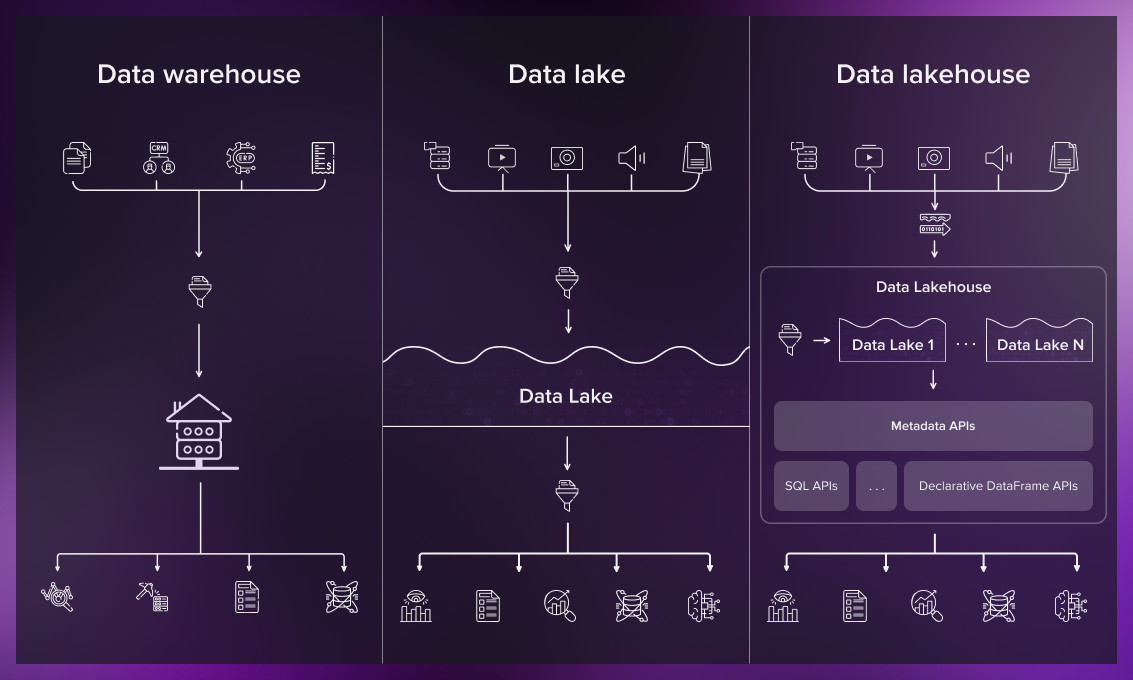
**C. Définissez et comparez les notions de Data Lake, Data Warehouse et Lake House. Illustrez les différences à l’aide de schémas.**

Les notions de Data Lake, Data Warehouse et Lake House sont des architectures de stockage et de gestion de données.

Un Data Lake est une infrastructure de stockage qui permet de stocker des données brutes dans leur format natif, sans structure prédéfinie.

Un Data Warehouse est une base de données centralisée et structurée conçue pour stocker des données provenant de différentes sources après avoir exécuté sur elles des tâches ETL (Extract, Transform, Load).

Un Lake House est une approche hybride qui combine les avantages des deux notions précédentes, avec l’organisation propre au Data Lake et la structure adaptée aux tâches d’ETL.

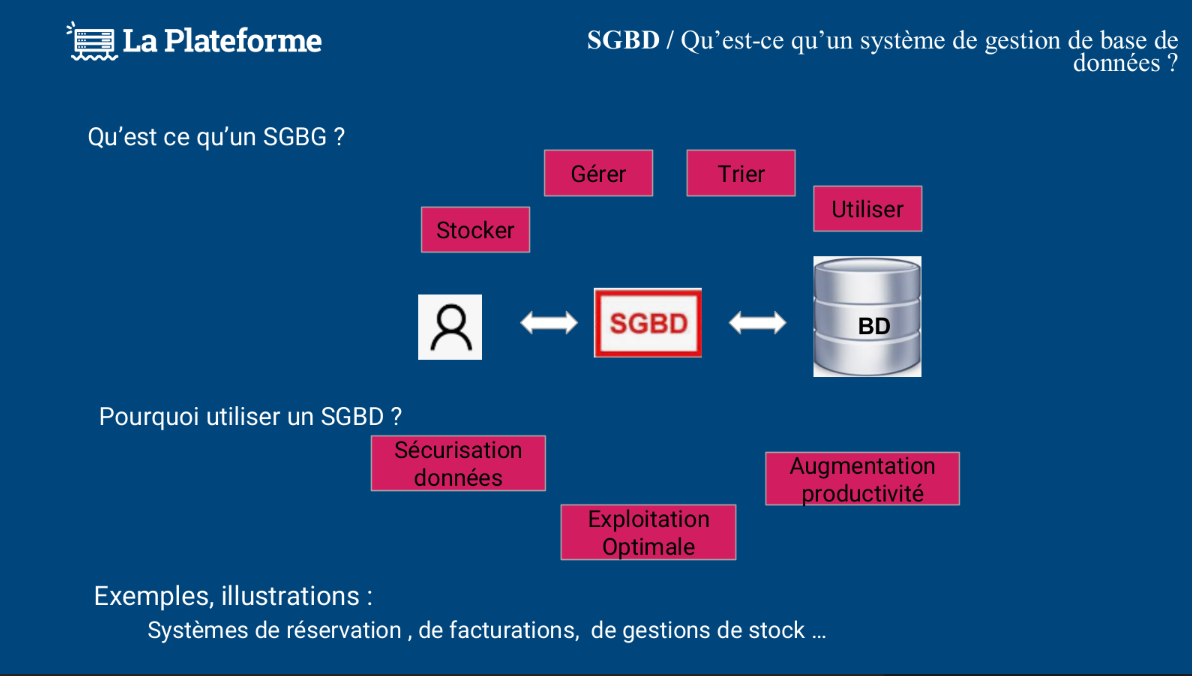


**D. Donnez une définition et des exemples de systèmes de gestion de bases de données avec des illustrations.**

Les SGBD sont des logiciels destinés au stockage, et à la gestion des données de manières structurée.

Ils facilitent leur manipulation et leur mise à jour.

Mysql, Oracle Database et MongoDB comptent parmi les SGBD les plus connus.



**E. Qu’est-ce qu’une base de données relationnelle ? Qu’est-ce qu’une base de données non relationnelle ? Donnez la différence entre les deux avec des exemples d’applications.**

Une base de données relationnelle est normalement composée de plusieurs tables indirectement reliées entre elles par l’utilisation de clé étrangère, colonne de la table contenant des identifiants que l’on retrouve dans la clé primaire d’une autre table. Toutes les tables de base de données ont par défaut une clé primaire qui permet par le biais de requêtes SQL de créer une table temporaire où extraire de nouvelles données composés d’informations provenant des deux tables.

Ex pour une base de données relationnelle : pour une société de location de vélos pourrait comprendre des tables distinctes pour les clients, les vélos, les locations, détails du fournisseur, etc. Les relations entre ces tables permettre par exemple d’étudier les taux de locations des différents vélos proposés selon l’âge ou le sexe du client sans avoir besoin de faire appel aux détails du fournisseur de vélos, ce qui pourrait engendrer des coûts supplémentaires.

Avec une base de données non relationnelle (NoSql), les fiches techniques de chaque modèle de vélo seront difficilement représentables sur la même base de données que les informations concernant les locations de vélos.

Une base de données non relationnelle pourrait en revanche être utilisée pour stocker des informations sur les itinéraires de balades à proximité. Elle pourrait utiliser un modèle de document où chaque document représente un itinéraire avec des champs flexibles, on pourrait imaginer stocker un fichier contenant le chant des oiseaux recensés à proximité de chaque parcours, ce qui serait impossible dans une base de données relationnelle.

Les projets sont parfois commencés en NoSql avant que leur structure ne soit figée sur une architecture de base de données relationnelle.

**F. Définissez les notions de clé étrangère et clé primaire.**

La clé primaire identifie de manière unique chaque enregistrement dans une table, tandis que la clé étrangère établit une relation entre les tables en faisant référence à la clé primaire d'une autre table.

**G. Quelles sont les propriétés ACID ?**

Les propriétés ACID sont un ensemble de caractéristiques qui garantissent la fiabilité des transactions dans une base de données relationnelle.

Atomicity : Opération indivisible des transactions.

Consistency: préserve la cohérence de la requête.

Isolation : Chaque transaction n’interfère pas d’autres possibles transactions simultanées

Durabilité : Persistance des modifications en cas de panne du système

**H. Définissez les méthodes Merise et UML. Quelles sont leur utilité dans le monde de l’informatique ? Donnez des cas précis d’utilisation avec des schémas.**

Merise :

Méthodologie de développement et de gestion de projet informatique, elle suggère une gestion de données en 3 niveaux, conceptuel, logique et physique.

Peut être utilisé comme système de gestion RH dans le cadre des informations relatives aux employés sur un schéma MCD avec des entités type Employé, Département, Salaire etc …

UML :

C’est un langage de modélisation standardisé utilisé pour représenter visuellement la structure, le comportement, et les interactions des systèmes logiciels.

UML peut être utilisé pour modéliser le système d'une application de messagerie instantanée. Un diagramme UML pourrait montrer l'interaction entre les utilisateurs, les messages envoyés, et le transfert de messages.

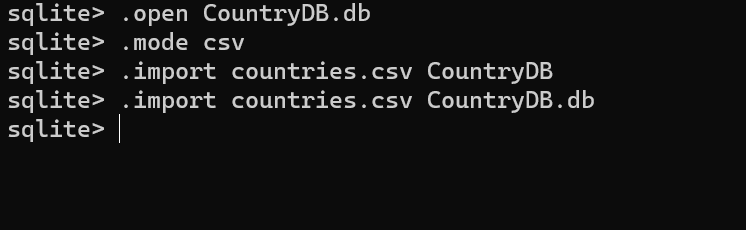
**I. Définissez le langage SQL. Donnez les commandes les plus utilisées de ce langage et les différentes jointures qu’il est possible de faire.**

SQL est un langage de programmation conçu pour la gestion et la manipulation de bases de données relationnelles.

Les commandes les plus utilisées comprennent SELECT et FROM dont l’usage est impératif pour les requêtes de prélèvement de la donnée, ainsi que Insert, Delete pour modifier ou supprimer des données. Puis dans les commandes SELECT ont retrouve très fréquemment les commandes WHERE, GROUP BY, HAVING et ORDER BY qui permettent de conditionner, d’agréger ou de restructurer la donnée lors de requête d’extraction de la donnée.

Pour les jointures, les options sont l’INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN. Cette dernière est moins utilisé dans la mesure où elles apporte le plus souvent trop d’information superflue à la requête SQL. Left & Right sont l’inverse l’une de l’autre et la logique humaine tend plus vers la requête Left qui est donc bien plus utilisé et est un outil très puissant. Inner Join renvoie uniquement les lignes où une correspondance est trouvée entre les colonnes des deux tables jointes.

Job 1 :



**1.**

**SELECT** \*

**FROM** world

**WHERE** **TRIM**(Country) = 'Germany'

**2.**

**SELECT** \*

**FROM** world

**WHERE** **TRIM**(Country) **IN** ('Sweden', 'Norway', 'Denmark');

**3.**

**SELECT** \*

**FROM** world

**WHERE** Area > '200000' **AND** Area < '300000'

**Job 2 :**

**1. Créez une requête permettant de trouver les noms de pays commençant par la**

**lettre B.**

**SELECT** \*

**FROM** world

**WHERE** **TRIM**(Country) **LIKE** 'B%'

**2. Créez une requête permettant de trouver les noms de pays commençant par “Al”.**

**SELECT** \*

**FROM** world

**WHERE** **TRIM**(Country) **LIKE** 'Al%'

**3. Créez une requête permettant de trouver les noms de pays finissant par la lettre**

**y.**

**SELECT** \*

**FROM** world

**WHERE** **TRIM**(Country) **LIKE** '%y'

**4. Créez une requête permettant de trouver les noms de pays finissant par “land”.**

**SELECT** \*

**FROM** world

**WHERE** **TRIM**(Country) **LIKE** '%land'

**5. Créez une requête permettant de trouver les noms de pays contenant la lettre w.**

**SELECT** \*

**FROM** world

**WHERE** **TRIM**(Country) **LIKE** '%w%'

**6. Créez une requête permettant de trouver les noms de pays contenant “oo” ou**

**“ee”.**

**SELECT** \*

**FROM** world

**WHERE** **TRIM**(Country) **LIKE** '%oo%' **OR** **TRIM**(Country) **LIKE** '%ee%'

**7. Créez une requête permettant de trouver les noms de pays contenant au moins**

**trois fois la lettre a.**

**SELECT** \*

**FROM** world

**WHERE** **LENGTH**(**REPLACE**(Country, 'a', '')) <= **LENGTH**(Country) - 3;

**SELECT** \*

**FROM** world

**WHERE** (Country **LIKE** '%a%a%a%') **AND** **NOT** (Country **LIKE** '%a%a%a%a%');

**8. Créez une requête permettant de trouver les noms de pays ayant la lettre r**

**comme seconde lettre.**

**SELECT** \*

**FROM** world

**WHERE** **UPPER**(Country) **LIKE** '\_R%';

**Job 3**

**1. Créez une requête permettant d’afficher toutes les colonnes de la table students.**

**SELECT** \*

**FROM** students

**2. Créez une requête permettant de filtrer la table et d’afficher les élèves âgés de**

**strictement plus de 20 ans.**

**SELECT** \*

**FROM** students

**WHERE** age > 20

**3. Créez une requête permettant de faire un classement des élèves selon leur note**

**dans un ordre croissant, puis dans un ordre décroissant.**

La question pose une grosse problématique car A+ et A- seront quel que soit le tri effectuer afficher avant ou après A, il faut donc soit utiliser une condition CASE dans le order by ou bien garder un tableau parallèle associant des valeurs numériques à chaque possibilitées de notes, puis effectuer une requête INNER Join and ORDER BY la colonne de tri numérique.

Croissant

**SELECT** \*

**FROM** students

**WHERE** Ranking **IN** ('A+', 'A', 'A-', 'B+', 'B', 'B-', 'C+', 'C', 'C-', 'D+', 'D', 'D-', 'F')

**ORDER** **BY**

**CASE**

**WHEN** Ranking = 'A+' **THEN** 1

**WHEN** Ranking = 'A' **THEN** 2

**WHEN** Ranking = 'A-' **THEN** 3

**WHEN** Ranking = 'B+' **THEN** 4

**WHEN** Ranking = 'B' **THEN** 5

**WHEN** Ranking = 'B-' **THEN** 6

**WHEN** Ranking = 'C+' **THEN** 7

**WHEN** Ranking = 'C' **THEN** 8

**WHEN** Ranking = 'C-' **THEN** 9

**WHEN** Ranking = 'D+' **THEN** 10

**WHEN** Ranking = 'D' **THEN** 11

**WHEN** Ranking = 'D-' **THEN** 12

**WHEN** Ranking = 'F' **THEN** 13

**END** **ASC**;

Décroissant

Il suffit d’inverser la méthode d’order by sur la dernière ligne avec :

**END DESC**

**Job 4**

**1. Créez une requête permettant d’afficher les prix nobels de 1986.**

**SELECT** \*

**FROM** nobel

WHERE yr = 1986

**2. Créez une requête permettant d’afficher les prix nobels de littérature de 1967.**

**SELECT** \*

**FROM** nobel

**WHERE** yr = 1967 **AND** subject = 'Literature'

**3. Créez une requête permettant d’afficher l’année et le sujet du prix nobel d’Albert**

**Einstein.**

**SELECT** yr, subject

**FROM** nobel

**WHERE** winner = 'Albert Einstein'

**4. Créez une requête permettant d’afficher les détails (année, sujet, lauréat) des**

**lauréats du prix de Littérature de 1980 à 1989 inclus.**

**SELECT** \*

**FROM** nobel

**WHERE** yr >= '1980' **AND** yr <= '1989'

**5. Créez une requête permettant d’afficher les détails des lauréats du prix de**

**Mathématiques. Combien y en a-t-il ?**

N\_Winners devrait afficher Le nombre de lauréats du prix de Mathématiques que j’ai traduit par Mathematics vu que la table est en anglais.

Ce nombre sera 0 vu qu’il n’y en a jamais eu.

**SELECT** \*, **COUNT**(winner) **AS** N\_Winners

**FROM** nobel

**WHERE** subject = 'Mathematics'

**Job 5**

**1. Créez une requête permettant d’afficher les pays dont la population est supérieure à celle de "Russia".**

**SELECT** \*

**FROM** world

**WHERE** Population > (**SELECT** Population **FROM** world **WHERE** **TRIM**(Country) = 'Russia');

**2. Créez une requête permettant d’afficher les pays d'Europe dont le PIB par habitant est supérieur à celui d’ "Italy".**

Il me semble que le PIB et le GDP ne sont pas exactement la même chose mais :

**SELECT** \*

**FROM** world

**WHERE** GDP > (**SELECT** GDP **FROM** world **WHERE** **TRIM**(Country) = 'Italy');

**3. Créez une requête permettant d’afficher les pays dont la population est supérieure à celle du Royaume-Uni mais inférieure à celle de l'Allemagne.**

**SELECT** \*

**FROM** world

**WHERE** Population > (**SELECT** Population **FROM** world **WHERE** **TRIM**(Country) = 'United Kingdom') **AND** Population < (**SELECT** Population **FROM** world **WHERE** **TRIM**(Country) = 'Germany') ;

**4. L'Allemagne (80 millions d'habitants) est le pays le plus peuplé d'Europe.**

**L'Autriche (8,5 millions d'habitants) compte 11% de la population allemande.**

**Créez une requête permettant d’afficher le nom et la population de chaque pays d'Europe, en pourcentage de la population de l'Allemagne.**

**SELECT**

Country,

Population,

**ROUND**((Population \* 100 / 80000000.0), 1) || '%' **AS** pourcentage

**FROM** world

**WHERE** **TRIM**(Region) **LIKE** '%EUROPE%';

**5. Créez une requête permettant de trouver le plus grand pays de chaque continent, en indiquant son continent, son nom et sa superficie.**

**SELECT** Country, Region, **Max**(Population)

**FROM** world

**GROUP** **BY** Region

**6. Créez une requête permettant de trouver les continents où tous les pays ont une population inférieure ou égale à 25 000 000.**

**SELECT** Region

**FROM** world

**GROUP** **BY** **TRIM**(Region)

**HAVING** **MAX**(Population) <= 25000000

**Job 6**

**1. Créez une requête permettant d’afficher la population totale du monde.**

**SELECT** **SUM**(Population)

**FROM** world

**2. Créez une requête permettant d’afficher la population totale de chacun des continents.**

**SELECT** Region, **SUM**(Population) **as** Total\_Population

**FROM** world

**GROUP** **BY** Region

**3. Créez une requête permettant d’afficher le PIB total du continent de chacun des continents.**

**SELECT** Region, **SUM**(GDP) **as** Total\_PIB

**FROM** world

**GROUP** **BY** Region

**4. Créez une requête permettant d’afficher le PIB total du continent africain.**

**SELECT**

**CASE**

**WHEN** **TRIM**(Region) **LIKE** '%AFRICA%' **THEN** 'AFRICA'

**ELSE** **TRIM**(Region)

**END** **AS** Region\_Category,

**SUM**(GDP) **as** Total\_GDP

**FROM** world

**WHERE** **TRIM**(Region) **LIKE** '%AFRICA%'

**5. Créez une requête permettant d’afficher le nombre de pays ayant une superficie supérieure ou égale à 1 000 000m2.**

**SELECT** **COUNT**(Country) **as** Country\_1M²\_Count

**FROM** world

**WHERE** AREA < 1000000

**6. Créez une requête permettant d’afficher la population totale des pays suivants : Estonia, Latvia, Lithuania.**

**SELECT** **SUM**(Population)

**FROM** world

**WHERE** **TRIM**(Country) **IN** ('Estonia', 'Latvia', 'Denmark');

**7. Créez une requête permettant d’afficher le nombre de pays de chaque continent.**

**SELECT** Region, **COUNT**(Country)

**FROM** world

**GROUP** **BY** Region

**8. Créez une requête permettant d’afficher les continents ayant une population totale d’au moins 100 millions d’individus.**

**SELECT** Region, Population

**FROM** world

**GROUP** **BY** **TRIM**(Region)

**HAVING** **MAX**(Population) >= 100000000

**Job 7**

**1. Observez le schéma relationnel de la base de données UEFA EURO 2012ci-dessus. Analysez les cardinalités.**

Eteam = typo ?

Il est possible que nous soyons plusieurs à avoir lu le tableau dans le mauvais sens pour les cardinalités, ce qui rendrait la formulation originale plus cohérente.

Game – goal et eteam – goal paraissent selon moi être à l’envers, dans la mesure où un match devrait être associé à 0 ou plus de goals (1 : 0..N) et chaque goal ne peut être associé qu’à un seul match où il a été marqué (N..1).

C’est également vrai pour eteam – goal, où une équipe peut être associée à plusieurs goals (1 :N) et un goal ne peut être associé qu’à une seule équipe qui l’a marqué (0 :1).

Pour ce qui est de la relation game – eteam elle n’est pas cohérente, plusieurs games devraient pouvoir être associés à plusieurs équipes dans une relation (N : N) tandis qu’une eteam devrait pouvoir être associée à 0 ou plus games (0..N)

**2. La requête ci-dessous permet d’afficher le but marqué par un joueur dont le nom de famille est "Bender". L'astérisque (\*) indique qu'il faut énumérer toutes les colonnes du tableau - une façon d’appeler toutes les colonnes de la table goal (matchid, teamid, player, gtime). Modifiez cette requête afin d’afficher le numéro de match et le nom du joueur pour tous les buts marqués par l'Allemagne. Afin d’identifier les joueurs allemands, vérifiez que : teamid = 'GER'.**

**SELECT** matchid, player

**FROM** goal

**WHERE** **TRIM**(teamid) = 'GER'

**3. Créez une requête permettant d’afficher les colonnes id, stadium, team1, team2 pour le match dont l’id est 1012.**

**SELECT** id, stadium, team1, team2

**FROM** game

**WHERE** id = '1012'

**4. La requête suivante permet de joindre la table game et la table goal sur la colonne id-matchid. Modifiez cette requête afin d’afficher player, teamid, stadium et mdate de chaque but allemand.**

Pas d’Alias pour la première commande.

**SELECT** goal.player goal.teamid, game.stadium, game.mdate

**FROM** game

**JOIN** goal **ON** game.id = goal.matchid

**WHERE** goal.teamid = 'GER'

**5. Créez une requête permettant d’afficher team1, team2 et player pour chaque but marqué par un joueur appelé Mario.**

**SELECT** game.team1, game.team2, goal.player

**FROM** game

**JOIN** goal **ON** game.id = goal.matchid

**WHERE** goal.player **LIKE** '%Mario%'

**6. Créez une requête permettant de joindre la table goal et la table eteam sur les clés id - teamid.**

**SELECT** \*

**FROM** goal

**JOIN** eteam **ON** goal.teamid = eteam.id

**7. Créez une requête permettant d’afficher player, teamid, coach, gtime pour tous les buts marqués dans les 10 premières minutes des matchs.**

**SELECT** g.player, g.teamid, e.coach, g.gtime

**FROM** goal g

**JOIN** eteam e **ON** g.teamid = e.id

**WHERE** g.gtime <= 10

**8. La requête suivante permet de joindre la table game et la table eteam sur les clés team1 – et eteam.id. Créez une requête permettant d’afficher les dates des matches ainsi que le nom de l'équipe dont "Fernando Santos" était le coach de l’équipe team1.**

**SELECT** g.mdate, e.teamname

**FROM** game g

**JOIN** eteam e **ON** g.team1 = e.id

**WHERE** e.coach **LIKE** '%Fernando Santos%';

**9. Créez une requête permettant d’afficher la liste des joueurs pour chaque but marqué lors d'un match dont le stade était le “National Stadium, Warsaw”.**

**SELECT** g.player

**FROM** game ga

**JOIN** goal g **ON** ga.id = g.matchid

**WHERE** ga.stadium **LIKE** '%National Stadium, Warsaw%';

**10.Créez une requête permettant d’afficher le nombre total de buts marqués pour chaque équipe de la table goal.**

La condition du HAVING permet de valider

**SELECT** e.teamname, **COUNT**(g.goal) **as** goals\_count

**FROM** goal g

**JOIN** eteam e **ON** e.id = g.teamid

**GROUP** **BY** e.teamname

**HAVING** **COUNT**(g.goal) > 0

**11.Créez une requête permettant d’afficher les stades et le nombre de buts marqués dans chacun des stades de la jointure de game-goal.**

**SELECT** ga.stadium, **COUNT**(g.goal) **as** goals\_count

**FROM** goal g

**JOIN** game ga **ON** ga.id = g.matchid

**GROUP** **BY** ga.stadium

**12.Pour chaque match où l’équipe de France a marqué, créez une requête permettant d’afficher l'id du match, la date du match et le nombre de buts marqués par "FRA".**

**SELECT** ga.id, ga.mdate, **COUNT**(g.goal) **as** score\_count

**FROM** goal g

**JOIN** game ga **ON** ga.id = g.matchid

**JOIN** eteam e **ON** e.id = g.teamid

**WHERE** **TRIM**(e.teamname) = "FRA"

**GROUP** **BY** ga.id

**Job 8**

**1. Créez la base de données SomeCompany à l’aide d’une requête, ajoutez une condition sur l'existence de SomeCompany.**

**CREATE** **DATABASE** SomeCompany**IF** **NOT** **EXISTS** SomeCompany

**2. Créez la table Employees.**

Idéalement on crée dans un premier temps la table Departments pour pouvoir insérer dans la table Employees la notion de foreign key sur department\_id.

**CREATE** **TABLE** Employees(employee\_id **INT** **PRIMARY** **KEY**, first\_name **VARCHAR**(50), last\_name **VARCHAR**(50), birthdate **TEXT**, **position** **VARCHAR**(100), department\_id **INT**)

**3. Créez la table Departments.**

**CREATE** **TABLE** Departments (department\_id **INT** **PRIMARY** **KEY**, department\_name **VARCHAR**(255), department\_head **INT**, location **VARCHAR**(255)**4. Insérez 6 à 9 nouveaux employés dans la table Employees.**

**INSERT** **INTO** Employees **VALUES**(9, 'Emilia', 'Clark', '1986-01-12', 'HR Manager', 3)

Pour 9 employés en d’assurant de récupérer les individus à la tête de départements.

**5. Récupérez le nom et le poste de tous les employés.**

**CREATE** **TABLE** Employees(employee\_id, first\_name, last\_name, birthdate, **position**, department\_id)

6. Mettez à jour le poste d’un employé dans la table Employees.

**SELECT** (first\_name || ' ' || last\_name) **AS** name, **position**

**FROM** Employees;

7. Supprimez un employé de la table Employees.

**DELETE** **FROM** Employees **WHERE** employee\_id = 6;

**8. Affichez le nom, le département et le bureau de chaque employé.**

Dans un premier temps, il faut ajouter les informations des bureaux trois bureaux à la table Departments :

**INSERT** **INTO** Departments **VALUES**(1, 'IT', 11, 'Headquarters');

…

Ensuite, on effectue la requête SQL

**SELECT** (e.first\_name || ' ' || e.last\_name) **AS** name, d.department\_name, d.location

**FROM** Employees e

**JOIN** Departments d **ON** e.department\_id = d.department\_id;

**9. Affichez, à l’aide d’un filtre, les membres de l’équipe IT, puis le management, puis les ressources humaines.**

**SELECT** (e.first\_name || ' ' || e.last\_name) **AS** name, d.department\_name

**FROM** Employees e

**JOIN** Departments d **ON** e.department\_id = d.department\_id

**ORDER** **BY** d.department\_id ;

**10. Affichez les départements de SomeCompany dans l’ordre alphabétique, avec les managers respectifs de chaque département.**

**SELECT** d.department\_name, (e.first\_name || ' ' || e.last\_name) **AS** manager

**FROM** Employees e

**JOIN** Departments d **ON** e.department\_id = d.department\_id

**WHERE** e.employee\_id **IN** (**SELECT** department\_head **FROM** Departments)

**ORDER** **BY** d.department\_id;

**11.Ajoutez un nouveau département à la table Department (Marketing peut-être?),**

**INSERT** **INTO** Departments **VALUES**(4, 'Marketing', 8, 'Branch Office London');

J’ai choisi Olivia Garcia, dont le poste est Marketing Manager pour être la nouvelle directrice du département Marketing, son id de département a été édité de 2 à 4.

**12. Créez une nouvelle table Project : project\_id (INT, PK), project\_name (VARCHAR), departement\_id (INT, FK). Ajoutez des observations à cette nouvelle table, analysez la productivité des départements en IT et du nouveau département créé précédemment.**

VARCHAR de project\_name pourrait être plus bas, peut-être autour de 60

**CREATE** **TABLE** Project(project\_id **INTEGER** **PRIMARY** **KEY**,

project\_name **VARCHAR**(255),

start\_date **TEXT**,

end\_date **TEXT**,

departement\_id **INT**,

**FOREIGN** **KEY** (departement\_id) **REFERENCES** Departments(department\_id));

Observations :

**INSERT** **INTO** Project **VALUES**(5, 'Small Project 2', '2023-12-1', '2022-12-6', 1);

1|Mid Project 1|2023-12-14|In\_Progress|4

2|Small Project 2|2023-12-1|2022-12-6|1

3|Mid Project 3|2023-12-17|In\_Progress|4

4|small Project 1|2023-12-17|In\_Progress|1

5|Mid Project 2|2022-8-3|2022-10-16|1

6|Big Project 2|2023-04-02|In Progress|4

7|Big Project 1|2023-12-1|In Progress|4

… + entrées pour Project Management et HR.

Il est difficile d’évaluer la productivité du nouveaux service marketing récemment crée en 2022 mais on remarque que les projets de grande ampleur commencent à s’accumuler et ils n’ont encore finit aucun projet. Il est urgent d’opérer un recrutement conséquent ou de procéder à une restructuration du service, avant qu’il ne ternisse l’image de marque de la société ou ne devienne un gouffre financier. Les équipes du département IT semble elles gérer efficacement leur projet et les terminer rapidement. Il pourrait être intéressant de leur demander, ainsi qu’aux autres départements du soutient le temps que le nouveau département prenne ses marques.

**Job 9**

**Big job : Calculateur d’Empreinte Carbone**