

Hello DBMS +

Hello veille ...

Friand d'apprendre de nouvelles choses et particulièrement en science des données, vous souhaitez approfondir vos **connaissances sur la donnée**, matière première des métiers liés à l'intelligence artificielle.

Vous vous documentez donc et réalisez une veille complète sur les éléments suivants :

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

Les données sont des faits, ou des observations; elles se présentent sous différentes formes cela peut être des images, des sons, du texte ou des chiffres et elles sont collectés dans le but de tirer des conclusions. Les données peuvent être structurées ou non structurées.

Les données structurées sont organisées en éléments prédéfinis et chaque élément correspond à un concept ou à un élément d'information spécifique.

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

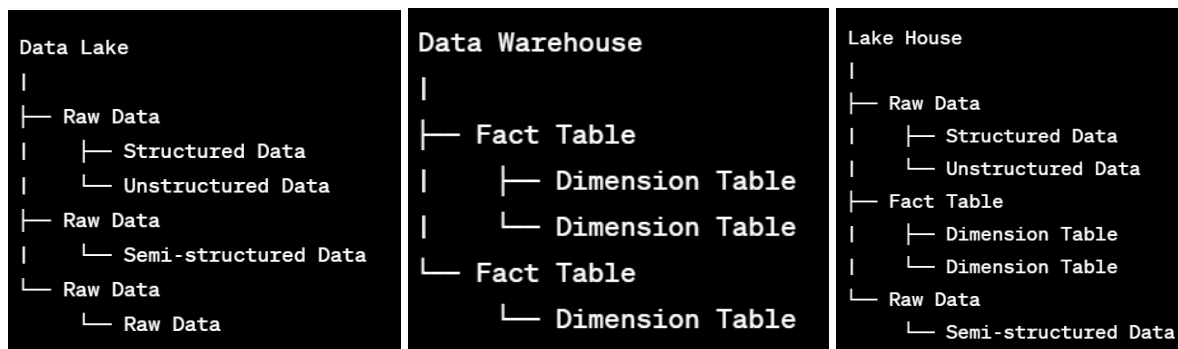
On mesure la qualité des données à travers la propreté de ses caractéristiques. Elles peuvent être internes ou externes à l'entreprise mais cela ne doit pas nous empêcher de citer l'exactitude, la cohérence, la validité, l'actualité, la clarté, l'intégrité ou même la sécurité de la donnée.

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

Les données de Data Lake peuvent être aussi bien structurées, non structurées aussi bien que semi-structurées pour une illustration ultérieure.

Par contre un Data Warehouse possède essentiellement des données traitées et structurées.

Un Data Lake House combine les avantages des deux précédents avec une architecture de gestion de données hybride qui associe la flexibilité du stockage massif de données brutes du Data Lake avec les performances et la structure optimisée du Data Warehouse.



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

Le système de gestion de bases de données appelé aussi SGBD est un logiciel qui sert à stocker, manipuler, gérer et partager les données de manière efficace. Il offre des fonctionnalités pour créer, lire, mettre à jour et supprimer des données mais aussi pour définir et manipuler la structure de bases de données. Les SGBD facilitent la gestion des données en fournissant une interface entre les utilisateurs et les bases de données.

Il existent plusieurs SGDB :

My SQL, Oracle Database, SQL Server, MongoDB, etc.

Voici quelques illustrations des SGBD citées:

| MySQL DB | Oracle Database | Microsoft SQL Server | MongoDB |
|-----------|-----------------|----------------------|----------------|
| Tables | Tables | Tables | Collections |
| Users | Employees | Customers | Products |
| ID Name | ID Name | ID Name | _id Name |
| 1 John | 1 Alice | 1 Emily | 1 Laptop |
| 2 Jane | 2 Bob | 2 David | 2 Smartphone |

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 une base de données où l'information est organisée dans des tableaux à deux dimensions appelés des relations de tables, selon un modèle introduit par Edgar F. Codd en 1960. Selon ce modèle relationnel, une base de données consiste en une ou plusieurs relations.

A contrario, une base de données non relationnelle permet de stocker des données volumineuses. Celles-ci peuvent être regroupées sur plusieurs machines afin de réduire les coûts de maintenance.

La différence entre la base de données relationnelle et celle non relationnelle réside dans la façon de stocker les données. L'une stocke les données dans des tables tandis que l'autre les stockent au format clé-valeur de manière à stocker davantage de quantités.

Exemple d'application : WordPress (CMS); SAP (Système ERP); Oracle Financial Service (Application Financière)

- MongoDB (Base de données de Documents); Redis (Base de données clé-valeur); Apache Cassandra (Base de données de colonnes); Neo4j (Base de données de graphs); Exemple combiné Polyglotte (Application E-Commerce -

Données de Catalogue (Produits) Données de Commande et de Panier)

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

Les clés primaires servent à identifier une ligne de manière unique. Dans une table nous pouvons avoir une seule clé primaire.

Les clés étrangères permettent de gérer les relations entre plusieurs tables et garantissent la cohérence des données.

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

ACID est un acronyme désignant les termes de: Atomicité, Cohérence, Isolation et Durabilité.

Ces quatre principes permettent d'assurer que les transactions de bases de données soient traitées de façon fiable.

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 est une méthode de conception, de développement et de réalisation de projets informatiques. Le but de cette méthode est d'arriver à concevoir un système d'information. Elle est basée sur la séparation des données et des traitements à effectuer en plusieurs modèles conceptuels et physiques.

UML est utilisé dans l'industrie du logiciel pour la conception, la documentation et la communication entre les membres d'une équipe de développement. Il offre une approche visuelle pour décrire les aspects complexes des systèmes logiciels.

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

Structured Query Language est un langage informatique conçu pour explorer les bases de données. Il permet de définir, manipuler et protéger les données de manière simple et schématique.

Les commandes les plus utilisées dans SQL sont: SELECT, INSERT, UPDATE, DELETE, CREATE TABLE, ALTER TABLE, DROP TABLE.

Voici quelques jointures qu'il est possible de faire en SQL:

INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN

Hello SQL...

Job 1 :

Soit la table de données **world** contenant des **informations sur différents pays du monde**, avec des détails tels que la région, la population, la superficie en miles carrés, la densité de la population, le taux de mortalité infantile, le PIB par habitant ainsi que d'autres informations pertinentes. Elle peut être récupérée [ici](#).

1. Modifiez la requête ci-dessus afin d'**afficher la population de "Germany"**.

```
SELECT Population  
FROM world  
WHERE Country = 'Germany'
```

| 123 Population |
|----------------|
| 82 422 299 |

2. Modifiez la requête ci-dessus afin d'afficher le nom et la population des pays **"Sweden"**, **"Norway"** et **"Denmark"**.

```
SELECT Country, Population  
FROM world  
WHERE Country = 'Sweden' OR Country = 'Norway' OR Country = 'Denmark'
```

| | ABC Country ▼ | 123 Population ▼ |
|---|---------------|------------------|
| 1 | Denmark | 5 450 661 |
| 2 | Norway | 4 610 820 |
| 3 | Sweden | 9 016 596 |

3. Créez une requête permettant d'**afficher les pays dont la superficie est supérieure à 200 000 mais inférieure à 300 000**.

```
SELECT Country
FROM world
WHERE "Area (sq. mi.)" > 200000 AND "Area (sq. mi.)" < 300000
```

| | ABC Country ▼ |
|----|----------------|
| 1 | Belarus |
| 2 | Burkina Faso |
| 3 | Ecuador |
| 4 | Gabon |
| 5 | Ghana |
| 6 | Guinea |
| 7 | Guyana |
| 8 | Laos |
| 9 | New Zealand |
| 10 | Oman |
| 11 | Romania |
| 12 | Uganda |
| 13 | United Kingdom |
| 14 | Western Sahara |

Job 2

Considérons la table **world** du job précédent :

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

```

SELECT Country
FROM world
WHERE Country LIKE 'B%'

```

| | ABC Country |
|----|--------------------|
| 7 | Belize |
| 8 | Benin |
| 9 | Bermuda |
| 10 | Bhutan |
| 11 | Bolivia |
| 12 | Bosnia & Herzeg |
| 13 | Botswana |
| 14 | Brazil |
| 15 | British Virgin Is. |
| 16 | Brunei |
| 17 | Bulgaria |
| 18 | Burkina Faso |
| 19 | Burma |
| 20 | Burundi |

2. Créez une requête permettant de trouver les noms de pays commençant par "A".

```

SELECT Country
FROM world
WHERE Country LIKE 'A%'

```

| | ABC Country |
|---|-------------|
| 1 | Albania |
| 2 | Algeria |

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

```

SELECT Country
FROM world
WHERE Country LIKE '%y'

```

| | ABC Country ▼ |
|---|---------------|
| 1 | Germany |
| 2 | Guernsey |
| 3 | Hungary |
| 4 | Italy |
| 5 | Jersey |
| 6 | Norway |
| 7 | Paraguay |
| 8 | Turkey |
| 9 | Uruguay |

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

```

SELECT Country
FROM world
WHERE Country LIKE '%land'

```

| | ABC Country ▼ |
|---|---------------|
| 1 | Finland |
| 2 | Greenland |
| 3 | Iceland |
| 4 | Ireland |
| 5 | New Zealand |
| 6 | Poland |
| 7 | Swaziland |
| 8 | Switzerland |
| 9 | Thailand |

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

```

SELECT Country
FROM world
WHERE Country LIKE '%w%'

```


| | ABC Country ▾ |
|----|------------------|
| 1 | Botswana |
| 2 | Kuwait |
| 3 | Malawi |
| 4 | New Caledonia |
| 5 | New Zealand |
| 6 | Norway |
| 7 | Papua New Guin |
| 8 | Rwanda |
| 9 | Swaziland |
| 10 | Sweden |
| 11 | Switzerland |
| 12 | Taiwan |
| 13 | Wallis and Futun |
| 14 | West Bank |
| 15 | Western Sahara |
| 16 | Zimbabwe |

6. Créez une requête permettant de trouver les noms de pays **contenant "oo" ou "ee"**.

```

SELECT Country
FROM world
WHERE Country LIKE '%oo%' OR Country LIKE '%ee%'

```

| | ABC Country ▾ |
|---|---------------|
| 1 | Cameroon |
| 2 | Cook Islands |
| 3 | Greece |
| 4 | Greenland |

7. Créez une requête permettant de trouver les noms de pays **contenant au moins trois fois la lettre a**.

```

SELECT Country
FROM world
WHERE
    (LENGTH(Country) - LENGTH(REPLACE(Country, 'a', ''))) >= 3;

```

| | ABC Country ▼ |
|----|----------------------------|
| 6 | Equatorial Guinea |
| 7 | Guatemala |
| 8 | Jamaica |
| 9 | Kazakhstan |
| 10 | Madagascar |
| 11 | Malaysia |
| 12 | Marshall Islands |
| 13 | Mauritania |
| 14 | Nicaragua |
| 15 | N. Mariana Islands |
| 16 | Panama |
| 17 | Papua New Guinea |
| 18 | Paraguay |
| 19 | Saint Vincent and the Gren |
| 20 | Saudi Arabia |
| 21 | Tanzania |
| 22 | Wallis and Futuna |
| 23 | Western Sahara |

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

```

SELECT Country
FROM world
WHERE Country LIKE 'r%';

```

| | ABC Country |
|----|--------------------|
| 1 | Argentina |
| 2 | Armenia |
| 3 | Aruba |
| 4 | Brazil |
| 5 | British Virgin Is. |
| 6 | Brunei |
| 7 | Croatia |
| 8 | Eritrea |
| 9 | France |
| 10 | French Guiana |
| 11 | French Polynesia |
| 12 | Greece |
| 13 | Greenland |
| 14 | Grenada |
| 15 | Iran |
| 16 | Iraq |
| 17 | Ireland |
| 18 | Sri Lanka |
| 19 | Trinidad & Tobago |
| 20 | Uruguay |

Job 3

Soit la table **students** définie comme suit :

| student_id | first_name | last_name age grade |
|------------|------------|---------------------|
| 1 | Alice | Johnson 22 A+ |
| 2 | Bob | Smith 20 B |

| | | |
|---|---------|---------------|
| 3 | Charlie | Williams 21 C |
| 4 | David | Brown 23 B+ |
| 5 | Eva | Davis 19 A |
| 6 | Frank | Jones 22 C+ |

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

| id | first_name | last_name | age | note |
|----|------------|-----------|-----|------|
| 1 | Alice | Johnson | 22 | A+ |
| 2 | Bob | Smith | 20 | A+ |
| 3 | Charlie | Williams | 21 | C |
| 4 | David | Brawn | 23 | B+ |
| 5 | Eva | Davis | 19 | A |
| 6 | Frank | Jones | 22 | C+ |

```
sqlite> CREATE TABLE students (  
(x1...> student_id INTEGER PRIMARY KEY,  
(x1...> id INTEGER,  
(x1...> first_name TEXT,  
(x1...> last_name TEXT,  
(x1...> age INTEGER,  
(x1...> note REAL  
(x1...> );  
sqlite> INSERT INTO students (id, first_name, last_name, age, note) VALUES  
...> (1, 'Alice', 'Johnson', 22, 'A+'),  
...> (2, 'Bob', 'Smith', 20, 'A+'),  
...> (3, 'Charlie', 'Williams', 21, 'C'),  
...> (4, 'David', 'Brawn', 23, 'B+'),  
...> (5, 'Eva', 'Davis', 19, 'A'),  
...> (6, 'Frank', 'Jones', 22, 'C+');  
sqlite> students.db -cmd  
...> .exit  
...> .save students.db  
...> .quit
```

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;
```

| | id | first_name | last_name | age | note |
|---|----|------------|-----------|-----|------|
| 1 | 1 | Alice | Johnson | 22 | A+ |
| 2 | 3 | Charlie | Williams | 21 | C |
| 3 | 4 | David | Brawn | 23 | B+ |
| 4 | 6 | Frank | Jones | 22 | C+ |

3. Créez une requête permettant de faire **un classement des élèves selon leur note dans un ordre croissant, puis dans un ordre décroissant.**

```
SELECT *
FROM students
ORDER BY CAST(note AS TEXT) ASC;
```

| | 123 id | ABC first_name | ABC last_name | 123 age | 123 note |
|---|--------|----------------|---------------|---------|----------|
| 1 | 5 | Eva | Davis | 19 | A |
| 2 | 1 | Alice | Johnson | 22 | A+ |
| 3 | 2 | Bob | Smith | 20 | A+ |
| 4 | 4 | David | Brawn | 23 | B+ |
| 5 | 3 | Charlie | Williams | 21 | C |
| 6 | 6 | Frank | Jones | 22 | C+ |

```
SELECT *
FROM students
ORDER BY CAST(note AS TEXT) DESC;
```

| | 123 id | ABC first_name | ABC last_name | 123 age | 123 note |
|---|--------|----------------|---------------|---------|----------|
| 1 | 6 | Frank | Jones | 22 | C+ |
| 2 | 3 | Charlie | Williams | 21 | C |
| 3 | 4 | David | Brawn | 23 | B+ |
| 4 | 1 | Alice | Johnson | 22 | A+ |
| 5 | 2 | Bob | Smith | 20 | A+ |
| 6 | 5 | Eva | Davis | 19 | A |

Job 4

Soit la table **nobel** définie comme suit :

| yr | subject | winner |
|------|------------|-----------------------------|
| 1960 | Chemistry | Willard F. Libby |
| 1960 | Literature | Saint-John Perse |
| 1960 | Medicine | Sir Frank Macfarlane Burnet |
| 1960 | Medicine | Peter Madawar |
| ... | | |

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 subject = 'Literature'  
AND yr BETWEEN 1980 AND 1989;
```

5. Créez une requête permettant d'afficher les détails des lauréats du prix de Mathématiques. Combien y en a-t-il ?

```
SELECT *  
FROM nobel  
WHERE subject = 'Mathématique';
```

Nous avons eu un résultat nul car après avoir fait des recherches sur internet pour trouver notre erreur nous nous sommes aperçu que le prix nobel de Mathématiques n'existait pas. A la place nous avons le prix de la médaille Fields et Abel.

Job 5

Considérons la table **world** précédente :

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

```
SELECT Country, Population
FROM world
WHERE Population > (SELECT Population FROM world WHERE Country = 'Russia');
```

| | Country | Population |
|---|---------------|---------------|
| 1 | Bangladesh | 147 365 352 |
| 2 | Brazil | 188 078 227 |
| 3 | China | 1 313 973 713 |
| 4 | India | 1 095 351 995 |
| 5 | Indonesia | 245 452 739 |
| 6 | Pakistan | 165 803 560 |
| 7 | United States | 298 444 215 |

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

```
SELECT Country, "GDP ($ per capita)", Population
FROM world
WHERE Region = 'WESTERN EUROPE'
AND "GDP ($ per capita)" / Population > (SELECT "GDP ($ per capita)" / Population FROM world WHERE Country = 'Italy');
```

| | Country | GDP (\$ per capita) | Population |
|---|------------|---------------------|------------|
| 1 | San Marino | 34 600 | 29 251 |

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 Country, Population
FROM world
WHERE Population >(SELECT Population FROM world WHERE Country = 'United Kingdom')
AND Population <(SELECT Population FROM world WHERE Country = 'Germany')
```

| | ABC Country | 123 Population |
|----|---------------|----------------|
| 1 | h | 147 365 352 |
| 2 | Brazil | 188 078 227 |
| 3 | China | 1 313 973 713 |
| 4 | India | 1 095 351 995 |
| 5 | Indonesia | 245 452 739 |
| 6 | Japan | 127 463 611 |
| 7 | Mexico | 107 449 525 |
| 8 | Nigeria | 131 859 731 |
| 9 | Pakistan | 165 803 560 |
| 10 | Philippines | 89 468 677 |
| 11 | Russia | 142 893 540 |
| 12 | United States | 298 444 215 |
| 13 | Vietnam | 84 402 966 |

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. Exemple :

| name pourcentage |
|------------------|
| Albania 3% |
| Andorra 0% |
| Austria 11% |
| ... |


```

SELECT
Country AS name,
printf('%.2f%%', (Population * 100.0 / (SELECT Population FROM world WHERE Country = 'Germany')))) AS percentage
FROM
world
WHERE
Region = 'WESTERN EUROPE';

```

| | name | percentage |
|----|---------------|------------|
| 1 | Andorra | 0.09% |
| 2 | Austria | 9.94% |
| 3 | Belgium | 12.59% |
| 4 | Denmark | 6.61% |
| 5 | Faroe Islands | 0.06% |
| 6 | Finland | 6.35% |
| 7 | France | 73.86% |
| 8 | Germany | 100.00% |
| 9 | Gibraltar | 0.03% |
| 10 | Greece | 12.97% |
| 11 | Guernsey | 0.08% |
| 12 | Iceland | 0.36% |
| 13 | Ireland | 4.93% |
| 14 | Isle of Man | 0.09% |
| 15 | Italy | 70.53% |
| 16 | Jersey | 0.11% |
| 17 | Liechtenstein | 0.04% |
| 18 | Luxembourg | 0.58% |
| 19 | Malta | 0.49% |
| 20 | Monaco | 0.04% |
| 21 | Netherlands | 20.01% |
| 22 | Norway | 5.59% |
| 23 | Portugal | 12.87% |
| 24 | San Marino | 0.04% |
| 25 | Spain | 49.01% |
| 26 | Sweden | 10.94% |
| 27 | Switzerland | 9.13% |
| 28 | United Kingdo | 73.53% |

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, "Area (sq. mi.)"  
FROM world  
GROUP BY Region  
ORDER BY MAX("Area (sq. mi.)")DESC ;
```

| | Country | Region | Area (sq. mi.) | |
|----|--------------|----------------------|----------------|--|
| 1 | Russia | C.W. OF IND. STATES | 17 075 200 | |
| 2 | Canada | NORTHERN AMERICA | 9 984 670 | |
| 3 | China | ASIA (EX. NEAR EAST) | 9 596 960 | |
| 4 | Brazil | LATIN AMER. & CARIB | 8 511 965 | |
| 5 | Australia | OCEANIA | 7 686 850 | |
| 6 | Sudan | SUB-SAHARAN AFRICA | 2 505 810 | |
| 7 | Algeria | NORTHERN AFRICA | 2 381 740 | |
| 8 | Saudi Arabia | NEAR EAST | 1 960 582 | |
| 9 | France | WESTERN EUROPE | 547 030 | |
| 10 | Poland | EASTERN EUROPE | 312 685 | |
| 11 | Lithuania | BALTICS | 65 200 | |

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

```
SELECT Country, Region, Population
FROM world
WHERE Population <= 25000000
ORDER BY Population DESC |
```

| | ABC Country ▾ | ABC Region ▾ | 123 Population ▾ |
|----|---------------|----------------------|------------------|
| 1 | Malaysia | ASIA (EX. NEAR EAST) | 24 385 858 |
| 2 | Korea, North | ASIA (EX. NEAR EAST) | 23 113 019 |
| 3 | Taiwan | ASIA (EX. NEAR EAST) | 23 036 087 |
| 4 | Ghana | SUB-SAHARAN AFRICA | 22 409 572 |
| 5 | Romania | EASTERN EUROPE | 22 303 552 |
| 6 | Yemen | NEAR EAST | 21 456 188 |
| 7 | Australia | OCEANIA | 20 264 082 |
| 8 | Sri Lanka | ASIA (EX. NEAR EAST) | 20 222 240 |
| 9 | Mozambique | SUB-SAHARAN AFRICA | 19 686 505 |
| 10 | Syria | NEAR EAST | 18 881 361 |
| 11 | Madagascar | SUB-SAHARAN AFRICA | 18 595 469 |
| 12 | Cote d'Ivoire | SUB-SAHARAN AFRICA | 17 654 843 |
| 13 | Cameroon | SUB-SAHARAN AFRICA | 17 340 702 |

Job 6

Considérons une nouvelle fois la table **world** précédente :

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

```
SELECT
SUM(Population) AS TotalPopulation
FROM world;
```

| | 123 TotalPopulation ▾ |
|---|-----------------------|
| 1 | 6 524 044 551 |

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

```
SELECT Region,  
SUM(Population) AS TotalPopulation  
FROM world  
GROUP BY Region;
```

| | ABC Region | 123 TotalPopulation |
|----|----------------------|---------------------|
| 1 | ASIA (EX. NEAR EAST) | 3 687 982 236 |
| 2 | BALTICS | 7 184 974 |
| 3 | C.W. OF IND. STATES | 280 081 548 |
| 4 | EASTERN EUROPE | 119 914 717 |
| 5 | LATIN AMER. & CARIB | 561 824 599 |
| 6 | NEAR EAST | 195 068 377 |
| 7 | NORTHERN AFRICA | 161 407 133 |
| 8 | NORTHERN AMERICA | 331 672 307 |
| 9 | OCEANIA | 33 131 662 |
| 10 | SUB-SAHARAN AFRICA | 749 437 000 |
| 11 | WESTERN EUROPE | 396 339 998 |

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

```
SELECT Region,  
SUM("GDP ($ per capita)") AS TotalGDP  
FROM world  
GROUP BY Region;
```

| | ABC Region | 123 TotalGDP |
|----|----------------------|--------------|
| 1 | ASIA (EX. NEAR EAST) | 225 500 |
| 2 | BALTICS | 33 900 |
| 3 | C.W. OF IND. STATES | 48 000 |
| 4 | EASTERN EUROPE | 117 700 |
| 5 | LATIN AMER. & CARIB | 390 700 |
| 6 | NEAR EAST | 167 300 |
| 7 | NORTHERN AFRICA | 27 300 |
| 8 | NORTHERN AMERICA | 130 500 |
| 9 | OCEANIA | 173 200 |
| 10 | SUB-SAHARAN AFRICA | 118 500 |
| 11 | WESTERN EUROPE | 757 300 |

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

```

SELECT Region,
SUM("GDP ($ per capita)") AS TotalGDP
FROM world
WHERE Region IN ('SUB-SAHARAN AFRICA', 'NORTHERN AFRICA')
GROUP BY Region

```

| | ABC Region | 123 TotalGDP |
|---|--------------------|--------------|
| 1 | NORTHERN AFRICA | 27 300 |
| 2 | SUB-SAHARAN AFRICA | 118 500 |

```

SELECT
'Combined Africa' AS Region,
SUM("GDP ($ per capita)") AS TotalGDP
FROM
world
WHERE
Region IN ('SUB-SAHARAN AFRICA', 'NORTHERN AFRICA');

```

| | ABC Region | 123 TotalGDP |
|---|-----------------|--------------|
| 1 | Combined Africa | 145 800 |

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

```
SELECT  
    COUNT(*) AS NumberOfCountries  
FROM  
    world  
WHERE  
    "Area (sq. mi.)" >= 1000000;
```

| | 123 NumberOfCountries ▼ |
|---|-------------------------|
| 1 | 30 |

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

```
SELECT  
    SUM(Population) AS TotalPopulation  
FROM  
    world  
WHERE  
    Country IN ('Estonia', 'Latvia', 'Lithuania');
```

| | 123 TotalPopulation ▼ |
|---|-----------------------|
| 1 | 7 184 974 |

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

```
SELECT Region ,  
       COUNT(*) AS NumberOfCountry  
FROM  
    world  
GROUP BY  
    Region;
```

| | ABC Region | 123 NumberOfCountry |
|----|----------------------|---------------------|
| 1 | ASIA (EX. NEAR EAST) | 28 |
| 2 | BALTICS | 3 |
| 3 | C.W. OF IND. STATES | 12 |
| 4 | EASTERN EUROPE | 12 |
| 5 | LATIN AMER. & CARIB | 45 |
| 6 | NEAR EAST | 16 |
| 7 | NORTHERN AFRICA | 6 |
| 8 | NORTHERN AMERICA | 5 |
| 9 | OCEANIA | 21 |
| 10 | SUB-SAHARAN AFRICA | 51 |
| 11 | WESTERN EUROPE | 28 |

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

```
SELECT Region,  
       SUM(Population) AS TotalPopulation  
FROM  
    world  
GROUP BY  
    Region  
HAVING  
    SUM(Population) >= 100000000;
```

| | ABC Region ▼ | 123 TotalPopulation ▼ |
|---|----------------------|-----------------------|
| 1 | ASIA (EX. NEAR EAST) | 3 687 982 236 |
| 2 | C.W. OF IND. STATES | 280 081 548 |
| 3 | EASTERN EUROPE | 119 914 717 |
| 4 | LATIN AMER. & CARIB | 561 824 599 |
| 5 | NEAR EAST | 195 068 377 |
| 6 | NORTHERN AFRICA | 161 407 133 |
| 7 | NORTHERN AMERICA | 331 672 307 |
| 8 | SUB-SAHARAN AFRICA | 749 437 000 |
| 9 | WESTERN EUROPE | 396 339 998 |

Job 7

Soit la base de données **UEFA EURO 2012** constituée des tables suivantes :

Game

| id | mdate | stadium | team1 | team2 |
|------|--------------|---------------------------|-------|-------|
| 1001 | 8 June 2012 | National Stadium, Warsaw | POL | GRE |
| 1002 | 8 June 2012 | Stadion Miejski (Wroclaw) | RUS | CZE |
| 1003 | 12 June 2012 | Stadion Miejski (Wroclaw) | GRE | CZE |

| | | | | |
|------|--------------|--------------------------|-----|-----|
| 1004 | 12 June 2012 | National Stadium, Warsaw | POL | RUS |
| ... | | | | |

Goal

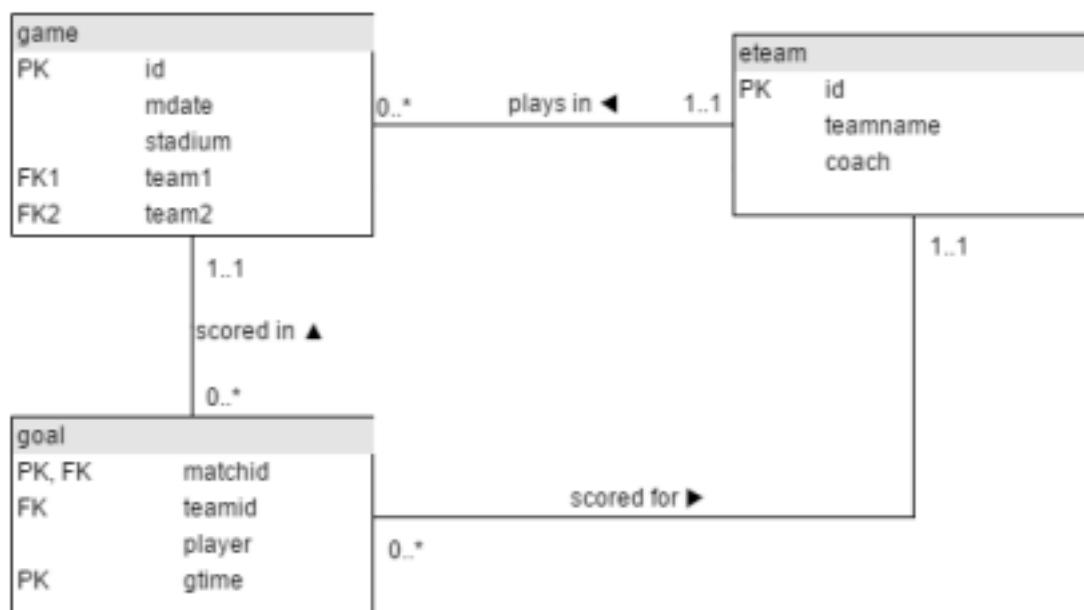
| matchid | teamid | player | gtime |
|---------|--------|--------------------|-------|
| 1001 | POL | Robert Lewandowski | 17 |

| | | | |
|------|-----|----------------------|----|
| 1001 | GRE | Dimitris Salpingidis | 51 |
| 1002 | RUS | Alan Dzagoev | 15 |
| 1002 | RUS | Roman Pavlyuchenko | 82 |
| ... | | | |

Eteam

| id | teamname | coach |
|-----|----------------|------------------|
| POL | Poland | Franciszek Smuda |
| RUS | Russia | Dick Advocaat |
| CZE | Czech Republic | Michal Bilek |
| GRE | Greece | Fernando Santos |
| ... | | |

Modèle relationnel de la base de donnée :



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

Nous avons dans ce schéma relationnel une description de la relation entre la table Game et Goal qui représente un match qui peut avoir plusieurs buts mais un but appartient à un seul match.

Nous avons également une relation entre la table Game et la table Eteam. Si une équipe participe à plusieurs matchs elle sera représentée une seule fois dans la table Eteam dans les colonnes référencées sous team1 et team2.

La relation entre Goal et Game plusieurs buts peuvent appartenir à un seul match mais chaque but est associé à un seul match.

La relation entre Goal et Eteam est que chaque but est associé à une seule équipe.

La relation entre Eteam et Game c'est qu'une équipe peut jouer dans plusieurs matchs mais chaque match est associé à une seule équipe.

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'.

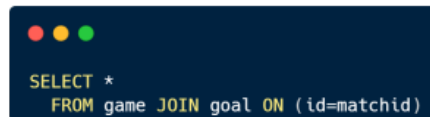
```
SELECT * FROM goal
WHERE player LIKE '%Bender'
```

```
SELECT matching,player
FROM Goal
WHERE teamid = "GER"
```

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

```
SELECT colonnes id, stadium, team1, team2  
FROM Game  
WHERE id = 2012
```

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**.



```
SELECT *  
FROM game JOIN goal ON (id=matchid)
```

```
SELECT player, teamid, stadium, mdate  
FROM Game JOIN Goal ON (id = matching)
```

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

```
SELECT team1, team2, player  
FROM Game JOIN Goal ON (id = matchid)  
WHERE Goal player = "Mario"
```

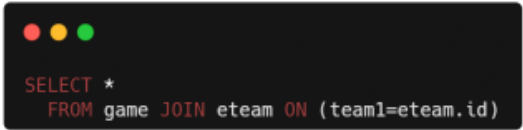
6. Créez une requête permettant de **joindre la table goal et la table eteam sur les clés id - teamid**.

```
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 player, teamid, coach, gtime  
FROM Goal JOIN Eteam ON Goal.teamid = Eteam.id  
WHERE Goal.gtime <= 10;
```

8. La requête suivante permet de joindre la table **game** et la table **etteam** sur les clés **team1 - 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 *  
FROM game JOIN eteam ON (team1=etteam.id)
```

```
SELECT mdate, teamname  
FROM Game JOIN Eteam ON Game.team1 = Eteam.id  
WHERE Eteam.coach = '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 matching, player, gtime  
FROM Goal JOIN Game ON Goal.matching = Game.id  
WHERE Game.stadium = 'National Stadium,Warsaw';
```

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

```
SELECT teamid  
COUNT(*) AS total_goals,  
FROM Goals  
GROUP BY teamid
```

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 Game.stadium, COUNT(Goal.matchid) AS goals_scored  
FROM Game JOIN Goal ON Game.id = Goal.matchid  
GROUP BY Game.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 Game.id AS match_id, Game.mdate AS match_date, COUNT(Goal.matchid) AS goals_scored
FROM Game JOIN Goal ON Game.id = Goal.matchid
WHERE Goal.teamid = 'FRA';
```

Job 8

Soient les tables **Employees** et **Departments** constituant la base de données **SomeCompany** définies comme suit :

1. **Employees:** employee_id (INT, PK), first_name (VARCHAR), last_name (VARCHAR), birthdate (DATE), position (VARCHAR), department_id (INT, FK).
2. **Departments:** department_id (INT, PK), department_name (VARCHAR), department_head (INT, FK), location (VARCHAR).
3. **Projects:** project_id (INT, PK), project_name (VARCHAR), start_date (DATE), end_date (DATE), department_id (INT, FK).

Employees

| employee_id | first_name | last_name | birthdate | position | department_id |
|-------------|------------|-----------|------------|------------------------|---------------|
| 1 | John | Doe | 1990-05-15 | Software Engineer | 1 |
| 2 | Jane | Smith | 1985-08-20 | Project Manager | 2 |
| 3 | Mike | Johnson | 1992-03-10 | Data Analyst | 1 |
| 4 | Emily | Brown | 1988-12-03 | UX Designer | 1 |
| 5 | Alex | Williams | 1995-06-28 | Software Developer | 1 |
| 6 | Sarah | Miller | 1987-09-18 | HR Specialist | 3 |
| 7 | Ethan | Clark | 1991-02-14 | Database Administrator | 1 |

| | | | | | |
|----|---------|--------|------------|-------------------|---|
| 8 | Olivia | Garcia | 1984-07-22 | Marketing Manager | 2 |
| 9 | Emilia | Clark | 1986-01-12 | HR Manager | 3 |
| 10 | Daniel | Taylor | 1993-11-05 | Systems Analyst | 1 |
| 11 | William | Lee | 1994-08-15 | Software Engineer | 1 |
| 12 | Sophia | Baker | 1990-06-25 | IT Manager | 2 |

Departments

| department_id | department_name | department_head | location |
|---------------|-----------------------|-----------------|--------------------|
| 1 | IT | 11 | Headquarters |
| 2 | ProProject Management | 2 | Branch Office West |

| | | | |
|---|-----------------|---|--------------------|
| 3 | Human Resources | 6 | Branch Office East |
|---|-----------------|---|--------------------|

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

```
sqlite> .open SomeCompany.db
sqlite>
```

```
CREATE DATABASE IF NOT EXISTS 'SomeCompany'
```

2. Créez la **table Employees**.

```
CREATE TABLE Employees(
  employee_id INT PRIMARY KEY,
  first_name VARCHAR,
  last_name VARCHAR,
  birthdate DATE,
  position VARCHAR,
  department_id INT
);
```

3. Créez la **table Departments**.

```
CREATE TABLE Departments(
  department_id INT PRIMARY KEY,
  department_name VARCHAR,
  department_head INT,
  location VARCHAR
);
```

```
INSERT INTO Departments (department_id, department_name, department_head, location)
VALUES
(1, 'IT', 11, 'Headquarters'),
(2, 'Project Management', 2, 'Branch Office West'),
(3, 'Human Resources', 6, 'Branch Office East');
```

| | 123 department_id | ABC department_name | 123 department_head | ABC location |
|---|-------------------|---------------------|---------------------|--------------------|
| 1 | 1 | IT | 11 | Headquarters |
| 2 | 2 | Project Management | 2 | Branch Office We |
| 3 | 3 | Human Resources | 6 | Branch Office East |

4. Insérez 6 à 9 nouveaux employés dans la table *Employees*.

```
INSERT INTO Employees (employee_id, first_name, last_name, birthdate, position, department_id)
VALUES
(1, 'John', 'Doe', '1990-05-15', 'Software Engineer', 1),
(2, 'Jane', 'Smith', '1985-08-20', 'Project Manager', 2),
(3, 'Mike', 'Johnson', '1992-03-10', 'Data Analyst', 1),
(4, 'Emily', 'Brown', '1988-12-03', 'UX Designer', 1),
(5, 'Alex', 'Williams', '1995-06-28', 'Software Developer', 1),
(6, 'Sarah', 'Miller', '1987-09-18', 'HR Specialist', 3),
(7, 'Ethan', 'Clark', '1991-02-14', 'Database Administrator', 1),
(8, 'Olivia', 'Garcia', '1984-07-22', 'Marketing Manager', 2),
(9, 'Emilia', 'Clark', '1986-01-12', 'HR Manager', 3),
(10, 'Daniel', 'Taylor', '1993-11-05', 'Systems Analyst', 1),
(11, 'William', 'Lee', '1994-08-15', 'Software Engineer', 1),
(12, 'Sophia', 'Baker', '1990-06-25', 'IT Manager', 2);
```

| | employee_id | first_name | last_name | birthdate | position | department_id |
|----|-------------|------------|-----------|------------|------------------------|---------------|
| 1 | 1 | John | Doe | 1990-05-15 | Software Engineer | 1 |
| 2 | 2 | Jane | Smith | 1985-08-20 | Project Manager | 2 |
| 3 | 3 | Mike | Johnson | 1992-03-10 | Data Analyst | 1 |
| 4 | 4 | Emily | Brown | 1988-12-03 | UX Designer | 1 |
| 5 | 5 | Alex | Williams | 1995-06-28 | Software Developer | 1 |
| 6 | 6 | Sarah | Miller | 1987-09-18 | HR Specialist | 3 |
| 7 | 7 | Ethan | Clark | 1991-02-14 | Database Administrator | 1 |
| 8 | 8 | Olivia | Garcia | 1984-07-22 | Marketing Manager | 2 |
| 9 | 9 | Emilia | Clark | 1986-01-12 | HR Manager | 3 |
| 10 | 10 | Daniel | Taylor | 1993-11-05 | Systems Analyst | 1 |
| 11 | 11 | William | Lee | 1994-08-15 | Software Engineer | 1 |
| 12 | 12 | Sophia | Baker | 1990-06-25 | IT Manager | 2 |

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

```
SELECT last_name, position  
FROM Employees
```

| | last_name | position |
|----|-----------|------------------------|
| 1 | Doe | Software Engineer |
| 2 | Smith | Project Manager |
| 3 | Johnson | Data Analyst |
| 4 | Brown | UX Designer |
| 5 | Williams | Software Developer |
| 6 | Miller | HR Specialist |
| 7 | Clark | Database Administrator |
| 8 | Garcia | Marketing Manager |
| 9 | Clark | HR Manager |
| 10 | Taylor | Systems Analyst |
| 11 | Lee | Software Engineer |
| 12 | Baker | IT Manager |

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

```
UPDATE Employees  
SET position = 'New Job'  
WHERE employee_id = 3;
```

| | employee_id | first_name | last_name | birthdate | position | department_id |
|---|-------------|------------|-----------|------------|-------------------|---------------|
| 1 | 1 | John | Doe | 1990-05-15 | Software Engineer | 1 |
| 2 | 2 | Jane | Smith | 1985-08-20 | Project Manager | 2 |
| 3 | 3 | Mike | Johnson | 1992-03-10 | New Job | 1 |
| 4 | 4 | Emily | Brown | 1988-12-03 | UX Designer | 1 |

7. Supprimez un employé de la table *Employees*.

```
DELETE FROM Employees  
WHERE employee_id = 6;
```

| | 123 employee_id | ABC first_name | ABC last_name | ABC birthdate | ABC position | 123 department_id |
|----|-----------------|----------------|---------------|---------------|------------------------|-------------------|
| 1 | 1 | John | Doe | 1990-05-15 | Software Engineer | 1 |
| 2 | 2 | Jane | Smith | 1985-08-20 | Project Manager | 2 |
| 3 | 3 | Mike | Johnson | 1992-03-10 | New Job | 1 |
| 4 | 4 | Emily | Brown | 1988-12-03 | UX Designer | 1 |
| 5 | 5 | Alex | Williams | 1995-06-28 | Software Developer | 1 |
| 6 | 7 | Ethan | Clark | 1991-02-14 | Database Administrator | 1 |
| 7 | 8 | Olivia | Garcia | 1984-07-22 | Marketing Manager | 2 |
| 8 | 9 | Emilia | Clark | 1986-01-12 | HR Manager | 3 |
| 9 | 10 | Daniel | Taylor | 1993-11-05 | Systems Analyst | 1 |
| 10 | 11 | William | Lee | 1994-08-15 | Software Engineer | 1 |
| 11 | 12 | Sophia | Baker | 1990-06-25 | IT Manager | 2 |

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

```

SELECT
    E.last_name,
    D.department_name,
    D.location
FROM
    Employees E
JOIN
    Departments D ON E.department_id = D.department_id;

```

| | ABC last_name | ABC department_name | ABC location |
|----|---------------|---------------------|--------------------|
| 1 | Doe | IT | Headquarters |
| 2 | Smith | Project Management | Branch Office West |
| 3 | Johnson | IT | Headquarters |
| 4 | Brown | IT | Headquarters |
| 5 | Williams | IT | Headquarters |
| 6 | Clark | IT | Headquarters |
| 7 | Garcia | Project Management | Branch Office West |
| 8 | Clark | Human Resources | Branch Office East |
| 9 | Taylor | IT | Headquarters |
| 10 | Lee | IT | Headquarters |
| 11 | Baker | Project Management | Branch Office West |

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, D.department_name
FROM Employees E JOIN Departments D ON E.department_id = D.department_id
WHERE D.department_name IN ('IT');

```

| | ASC first_name | ASC last_name | ASC department_name |
|---|----------------|---------------|---------------------|
| 1 | John | Doe | IT |
| 2 | Mike | Johnson | IT |
| 3 | Emily | Brown | IT |
| 4 | Alex | Williams | IT |
| 5 | Ethan | Clark | IT |
| 6 | Daniel | Taylor | IT |
| 7 | William | Lee | IT |

```

SELECT E.first_name, E.last_name, D.department_name
FROM Employees E JOIN Departments D ON E.department_id = D.department_id
WHERE D.department_name IN ('Project Management');

```

| | ASC first_name | ASC last_name | ASC department_name |
|---|----------------|---------------|---------------------|
| 1 | Jane | Smith | Project Management |
| 2 | Olivia | Garcia | Project Management |
| 3 | Sophia | Baker | Project Management |

```

SELECT E.first_name, E.last_name, D.department_name
FROM Employees E JOIN Departments D ON E.department_id = D.department_id
WHERE D.department_name IN ('Human Resources');

```

| | ASC first_name | ASC last_name | ASC department_name |
|---|----------------|---------------|---------------------|
| 1 | Emilia | Clark | Human Resources |

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

```

SELECT department_name,
       department_head AS manager
FROM
  Departments
ORDER BY
  department_name;

```

| | asc department_name | 123 manager |
|---|---------------------|-------------|
| 1 | Human Resources | 6 |
| 2 | IT | 11 |
| 3 | Project Management | 2 |

11. Ajoutez **un nouveau département** à la table **Department** (Marketing peut-être?),
ajoutez ou mettez à jour les employés de ce nouveau département.

```

INSERT INTO Departments (department_id, department_name, department_head, location)
VALUES (4, 'Marketing', 7, 'Branch Office Sud');

```

| | 123 department_id | asc department_name | 123 department_head | asc location |
|---|-------------------|---------------------|---------------------|--------------------|
| 1 | 1 | IT | 11 | Headquarters |
| 2 | 2 | Project Management | 2 | Branch Office West |
| 3 | 3 | Human Resources | 6 | Branch Office East |
| 4 | 4 | Marketing | 7 | Branch Office Sud |

```

INSERT INTO Employees (employee_id, first_name, last_name, birthdate, position, department_id)
VALUES
(13, 'Laura', 'Johnson', '1990-09-25', 'Marketing Specialist',
 (SELECT department_id FROM Departments WHERE department_name = 'Marketing'));

```

| | 123 employee_id | ABC first_name | ABC last_name | ABC birthdate | ABC position | 123 department_id |
|----|-----------------|----------------|---------------|---------------|------------------------|-------------------|
| 1 | 1 | John | Doe | 1990-05-15 | Software Engineer | 1 |
| 2 | 2 | Jane | Smith | 1985-08-20 | Project Manager | 2 |
| 3 | 3 | Mike | Johnson | 1992-03-10 | New Job | 1 |
| 4 | 4 | Emily | Brown | 1988-12-03 | UX Designer | 1 |
| 5 | 5 | Alex | Williams | 1995-06-28 | Software Developer | 1 |
| 6 | 7 | Ethan | Clark | 1991-02-14 | Database Administrator | 1 |
| 7 | 8 | Olivia | Garcia | 1984-07-22 | Marketing Manager | 2 |
| 8 | 9 | Emilia | Clark | 1986-01-12 | HR Manager | 3 |
| 9 | 10 | Daniel | Taylor | 1993-11-05 | Systems Analyst | 1 |
| 10 | 11 | William | Lee | 1994-08-15 | Software Engineer | 1 |
| 11 | 12 | Sophia | Baker | 1990-06-25 | IT Manager | 2 |
| 12 | 13 | Laura | Johnson | 1990-09-25 | Marketing Specialist | 4 |

12. Créez **une nouvelle table Project** : project_id (INT, PK), project_name (VARCHAR), start_date (DATE), end_date (DATE), departement_id (INT, FK).

```
CREATE TABLE Projects(
  project_id INT PRIMARY KEY,
  project_name VARCHAR,
  start_date DATE,
  end_date DATE,
  departement_id INT
);
```

Ajoutez des observations à cette nouvelle table, analysez la productivité des départements en IT et du nouveau département créé précédemment.

Analyse du nombre total de projets pour le département Marketing:

```
SELECT COUNT(project_id) AS total_projects
FROM Projects
WHERE department_id = (SELECT department_id FROM Departments WHERE department_name = 'Marketing');
```

| | 123 total_projects |
|---|--------------------|
| 1 | 1 |

Analyse du nombre total d'employés pour le département Marketing :

```

SELECT COUNT(employee_id) AS total_employees
FROM Employees
WHERE department_id = (SELECT department_id FROM Departments WHERE department_name = 'Marketing');

```

| | 123 total_employees |
|---|---------------------|
| 1 | 1 |

Calcul du nombre moyen de projets par employés pour le département Marketing:

```

SELECT COUNT(P.project_id) / COUNT(E.employee_id) AS productivity
FROM Projects P
LEFT JOIN Employees E ON P.department_id = E.department_id
WHERE P.department_id = (SELECT department_id FROM Departments WHERE department_name = 'Marketing');

```

| | 123 productivity |
|---|------------------|
| 1 | 1 |

Mise à jour de la table Projects avec la colonne project_name qui correspond à la colonne department_name pour faire des comparaisons:

```

UPDATE Projects
SET project_name =
CASE
WHEN project_id = 1 THEN 'Marketing'
WHEN project_id = 2 THEN 'IT'
WHEN project_id = 3 THEN 'Project Management'
WHEN project_id = 4 THEN 'Human Resources'
WHEN project_id = 5 THEN 'Marketing'
ELSE project_name
END;

```

| | 123 project_id | ABC project_name | ABC start_date | ABC end_date | 123 department_id |
|---|----------------|--------------------|----------------|--------------|-------------------|
| 1 | 1 | Marketing | 2023-01-01 | 2023-12-31 | 1 |
| 2 | 2 | IT | 2023-02-01 | 2024-01-31 | 2 |
| 3 | 3 | Project Management | 2023-03-01 | 2024-02-29 | 3 |
| 4 | 4 | Human Resources | 2023-04-01 | 2024-03-31 | 4 |
| 5 | 5 | Marketing | 2023-05-01 | 2024-04-30 | 5 |

Analyse de la productivité des départements en IT et Marketing

```

SELECT
    D.department_name,
    COUNT(P.project_id) AS nombre_de_projets,
    COUNT(E.employee_id) AS nombre_d_employes,
    COUNT(P.project_id) * 1.0 / COUNT(E.employee_id) AS productivite
FROM
    Departments D
LEFT JOIN
    Projects P ON D.department_id = P.department_id
LEFT JOIN
    Employees E ON D.department_id = E.department_id
WHERE
    D.department_name IN ('IT', 'Marketing')
GROUP BY
    D.department_name;

```

| | department_name | nombre_de_projets | nombre_d_employes | productivite |
|---|-----------------|-------------------|-------------------|--------------|
| 1 | IT | 7 | 7 | 1 |
| 2 | Marketing | 1 | 1 | 1 |

Job 9

Considérons une dernière fois la table **world**. Il existe plusieurs colonnes de cette table que nous n'avons pas pu analyser. **Étudiez au moins 6 autres variables de world**, à l'aide de différentes fonctions et commandes SQL, afin d'**obtenir des insights pertinents** (Literacy, Net migration, Birthrate, Deathrate, Infant mortality, Arable, Crops, ...).

- Calcul du pourcentage de pays ayant une Literacy supérieure à 80%:

```

SELECT COUNT(*) AS count_high_literacy
FROM world
WHERE "Literacy (%)" > 80;

```

| | count_high_literacy |
|---|---------------------|
| 1 | 227 |

- Calcul de la migration nette totale:

```

SELECT SUM("Net migration") AS total_net_migration
FROM world

```


| | 123 total_net_migration |
|---|-------------------------|
| 1 | 16 |

- Calcul du taux de natalité moyen:

```
SELECT AVG(Birthrate) AS average_birthrate
FROM world
```

| | 123 average_birthrate |
|---|-----------------------|
| 1 | 21,3083700441 |

- Pays qui ont un taux de mortalité infantile supérieur à 50 pour 1000 naissances:

```
SELECT Country , "Infant mortality (per 1000 births)"
FROM world
WHERE "Infant mortality (per 1000 births)" > 50;
```

| | asc Country | 123 Infant mortality (per 1000 births) |
|---|----------------|--|
| 1 | Afghanistan | 163,07 |
| 2 | Albania | 21,52 |
| 3 | American Samoa | 9,27 |
| 4 | Andorra | 4,05 |
| 5 | Angola | 191,19 |
| 6 | Anguilla | 21,03 |

Le tableau ne représente pas la totalité des résultats

- Calcul de la superficie totale Arable:

```
SELECT SUM ("Arable (%)") AS total_arable_area
FROM world;
```

| | 123 total_arable_area |
|---|-----------------------|
| 1 | 2 998 |

- Pays ou la superficie cultivée est supérieure à 20% de la superficie totale:

```
SELECT Country, "Crops (%)"
FROM world
WHERE "Crops (%)" > 20;
```

| | ABC Country | 123 Crops (%) |
|---|-------------------|---------------|
| 1 | Afghanistan | 0,22 |
| 2 | Albania | 4,42 |
| 3 | Algeria | 0,25 |
| 4 | Angola | 0,24 |
| 5 | Antigua & Barbuda | 4,55 |

Le tableau ne représente pas la totalité des résultats

Big job : Calculateur d'Empreinte Carbone

L'empreinte carbone (ou le contenu carbone) d'une **activité humaine** est une mesure des **émissions de effet de serre** d'origine anthropique, c'est-à-dire lui être imputées.

Soucieux de l'environnement et de chère planète, vous avez **outil**

pour comprendre cette afin de la

diminuer et **minimiser son**

vous lancez dans le développement d'un

calculateur d'empreinte carbone, visant à

aider à l'évaluation et à la compensation de

l'empreinte carbone, en particulier dans le

contexte de la production d'énergie électrique.



gaz à

qui peuvent

notre

eu l'idée d'un

empreinte

impact. Vous

1. Vous récupérez les données d'intérêt [ici](#). Vous avez à votre disposition un dataset

composé de deux tables, **Country** et **World**. Elles recensent le pourcentage d'utilisation de différentes sources d'énergie (charbon, gaz, pétrole, nucléaire, ...) en 2015 pour la production d'électricité par pays dans la table *Country*, puis par région du monde dans la table *World*.

2. Créez la base de données **CarbonFootprint**, puis les tables **Country** et **World**.

```
sqlite> .open CarbonFootprint.db
sqlite> .mode csv
sqlite> .import carbon-footprint-data.csv CarbonFootprint.db
carbon-footprint-data.csv:26: expected 1 columns but found 2 - extras ignored
carbon-footprint-data.csv:28: expected 1 columns but found 2 - extras ignored
carbon-footprint-data.csv:29: expected 1 columns but found 2 - extras ignored
carbon-footprint-data.csv:40: expected 1 columns but found 2 - extras ignored
carbon-footprint-data.csv:59: expected 1 columns but found 2 - extras ignored
carbon-footprint-data.csv:69: expected 1 columns but found 2 - extras ignored
carbon-footprint-data.csv:70: expected 1 columns but found 2 - extras ignored
carbon-footprint-data.csv:79: expected 1 columns but found 2 - extras ignored
carbon-footprint-data.csv:138: expected 1 columns but found 2 - extras ignored
carbon-footprint-data.csv:140: expected 1 columns but found 2 - extras ignored
sqlite>
```

Table Country

| | ABC Country ▼ | ABC Coal ▼ | ABC Gas ▼ | ABC Oil ▼ | ABC Hydro ▼ | ABC Renewable ▼ | ABC Nuclear ▼ |
|-----|----------------------|------------|-----------|-----------|-------------|-----------------|---------------|
| 115 | South Sudan | 0.0 | 0.0 | 99.6 | 0.0 | 0.4 | 0.0 |
| 116 | Spain | 16.5 | 17.2 | 5.1 | 14.2 | 25.9 | 20.8 |
| 117 | Sri Lanka | 25.7 | 0.0 | 35.1 | 36.5 | 2.7 | 0.0 |
| 118 | Sudan | 0.0 | 0.0 | 21.7 | 78.3 | 0.0 | 0.0 |
| 119 | Suriname | 0.0 | 0.0 | 37.7 | 62.3 | 0.0 | 0.0 |
| 120 | Sweden | 0.6 | 0.3 | 0.2 | 41.5 | 14.3 | 42.3 |
| 121 | Switzerland | 0.0 | 0.7 | 0.1 | 54.3 | 3.8 | 39.3 |
| 122 | Syrian Arab Republic | 0.0 | 64.4 | 21.8 | 13.8 | 0.0 | 0.0 |
| 123 | Tajikistan | 0.0 | 2.9 | 0.0 | 97.1 | 0.0 | 0.0 |
| 124 | Tanzania | 0.0 | 42.2 | 15.5 | 41.6 | 0.6 | 0.0 |
| 125 | Thailand | 21.6 | 68.3 | 1.0 | 3.2 | 5.9 | 0.0 |
| 126 | Togo | 0.0 | 0.0 | 12.0 | 84.5 | 3.5 | 0.0 |
| 127 | Trinidad and Tobago | 0.0 | 99.8 | 0.2 | 0.0 | 0.0 | 0.0 |

Table World

| | ABC Country ▼ | 123 Coal ▼ | 123 Gas ▼ | 123 Oil ▼ | 123 Hydro ▼ | 123 Renewable ▼ | 123 Nuclear ▼ |
|---|----------------------------|------------|-----------|-----------|-------------|-----------------|---------------|
| 1 | World | 40,7 | 21,6 | 4,1 | 16,2 | 6 | 10,6 |
| 2 | East Asia & Pacific | 60,6 | 13,5 | 2,2 | 15 | 4,2 | 3,8 |
| 3 | Europe & Central | 24,1 | 24,3 | 1,3 | 16,6 | 10,5 | 22,4 |
| 4 | Latin America & Caribbean | 6,5 | 26 | 10,6 | 46,5 | 6,4 | 1,9 |
| 5 | Middle East & North Afrika | 3,4 | 64,1 | 28,8 | 2,6 | 0,4 | 0,3 |
| 6 | North America | 35,7 | 24,6 | 1 | 12,9 | 6,6 | 18,9 |
| 7 | South Asia | 65,7 | 9,1 | 5,2 | 11,6 | 4,6 | 2,8 |
| 8 | Sub-Saharan Africa | 51,4 | 8,6 | 4,3 | 21,2 | 1,7 | 3 |

3. Utilisez des requêtes SQL afin d'**analyser les données recueillies** et tirez un maximum d'informations sur les émissions en carbone. **Qu'est ce que vous observez ?** Notez ces observations pour la suite.

Émissions totales par pays

```
SELECT Country,
       SUM(Coal + Gas + Oil + Hydro + Renewable + Nuclear) AS TotalEmissions
FROM Country
GROUP BY Country;
```

| | ABC Country ▼ | 123 TotalEmissions ▼ |
|---|---------------|----------------------|
| 1 | Albania | 100 |
| 2 | Algeria | 100 |
| 3 | Angola | 100 |
| 4 | Argentina | 100 |
| 5 | Armenia | 100 |
| 6 | Australia | 100 |
| 7 | Austria | 99 |
| 8 | Azerbaijan | 99,8 |

Émissions totales mondiales

```
SELECT SUM(Coal + Gas + Oil + Hydro + Renewable + Nuclear) AS TotalWorldEmissions
FROM World;
```

| | 123 TotalWorldEmissions ▼ |
|---|---------------------------|
| 1 | 784,1 |

Pays avec les émissions les plus élevées

```
SELECT Country,  
       SUM(Coal + Gas + Oil + Hydro + Renewable + Nuclear) AS TotalEmissions  
FROM Country  
GROUP BY Country  
ORDER BY TotalEmissions DESC  
LIMIT 1;
```

| | ABC Country ▾ | 123 TotalEmissions ▾ |
|---|---------------|----------------------|
| 1 | Uzbekistan | 100,1 |

Moyenne des émissions par source d'énergie

```
SELECT AVG(Coal) AS AvgCoal,  
       AVG(Gas) AS AvgGas,  
       AVG(Oil) AS AvgOil,  
       AVG(Hydro) AS AvgHydro,  
       AVG(Renewable) AS AvgRenewable,  
       AVG(Nuclear) AS AvgNuclear  
FROM Country;
```

| | 123 AvgCoal ▾ | 123 AvgGas ▾ | 123 AvgOil ▾ | 123 AvgHydro ▾ | 123 AvgRenewable ▾ | 123 AvgNuclear ▾ |
|---|---------------|---------------|---------------|----------------|--------------------|------------------|
| 1 | 17,7964539007 | 26,1496453901 | 16,6177304965 | 27,7978723404 | 6,2397163121 | 5,1397163121 |

4. Créez une application Flask où vous présenterez le **contexte de ce mini projet et les observations faites précédemment**.

Notre analyse est faite sur une base de données de 141 pays et 8 régions. Nous avons un jeu de données concernant différentes émissions de charbon, gaz, Petrol, électricité, solaire et nucléaire et nous avons fait des comparaison selon les pays qui consomment le plus et les différentes sources.

Pensez à afficher un aperçu de votre jeu de données **CarbonFootprint**.

Calculateur d'Empreinte Carbone

Country

| Country | Coal | Gas | Oil | Hydro | Renewable | Nuclear |
|-----------|------|------|------|-------|-----------|---------|
| Albania | 0.0 | 0.0 | 0.0 | 100.0 | 0.0 | 0.0 |
| Algeria | 0.0 | 97.8 | 1.8 | 0.4 | 0.0 | 0.0 |
| Angola | 0.0 | 0.0 | 46.8 | 53.2 | 0.0 | 0.0 |
| Argentina | 2.9 | 47.7 | 13.8 | 29.0 | 2.5 | 4.1 |
| Armenia | 0.0 | 42.4 | 0.0 | 25.7 | 0.1 | 31.8 |

World

| Country | Coal | Gas | Oil | Hydro | Renewable | Nuclear |
|----------------------------|------|------|------|-------|-----------|---------|
| World | 40.7 | 21.6 | 4.1 | 16.2 | 6.0 | 10.6 |
| East Asia & Pacific | 60.6 | 13.5 | 2.2 | 15.0 | 4.2 | 3.8 |
| Europe & Central | 24.1 | 24.3 | 1.3 | 16.6 | 10.5 | 22.4 |
| Latin America & Caribbean | 6.5 | 26.0 | 10.6 | 46.5 | 6.4 | 1.9 |
| Middle East & North Afrika | 3.4 | 64.1 | 28.8 | 2.6 | 0.4 | 0.3 |

5. Le tableau suivant montre **les émissions de CO2 de différentes sources de production d'électricité** d'après une étude réalisée par le Groupe d'experts intergouvernemental sur l'évolution du climat datée de 2014. Par exemple, pour l'électricité produite à partir du charbon, les émissions de CO2 par kilowattheure varient de 740 grammes (au minimum) à 910 grammes (au maximum), avec une médiane de 820 grammes. Calculez le **pourcentage de contribution des différentes sources du tableau aux émissions totales de CO2** lors de la production d'électricité **pour tous les pays de Country**.

Indice : Contribution du charbon aux émissions totales de CO2 d'un pays = Pourcentage d'utilisation du charbon du pays x Emission de gCO2 par kWh du charbon.

| Source | Min de gCO2/kWh | Médiane de gCO2/kWh | Max de gCO2/kWh |
|--------------|-----------------|---------------------|-----------------|
| Charbon | 740 | 820 | 910 |
| Gaze naturel | 410 | 490 | 650 |

| | | | |
|------------------------|-----|-----|------|
| Pétrole | 620 | 740 | 890 |
| Hydro | 1 | 24 | 2200 |
| Renouvelable (Solaire) | 26 | 41 | 60 |
| Nucléaire | 3.7 | 12 | 110 |

Contribution des Sources aux Émissions de CO2

| Source | Valeur |
|--------------|--------|
| Charbon | 205.76 |
| Gaz | 180.67 |
| Huile | 173.39 |
| Hydro | 9.41 |
| Renouvelable | 3.61 |
| Nucléaire | 0.87 |

6. Modifiez votre application Flask afin de pouvoir **filtrer vos données selon un pays (ou une région du monde) sélectionnable depuis une selection box.**

Contribution des Sources aux Émissions de CO2

Sélectionner un pays :

| Source | Valeur |
|-----------|--------|
| Coal | 0.24 |
| Gas | 2.34 |
| Oil | 1.02 |
| Hydro | 0.07 |
| Renewable | 0.01 |
| Nuclear | 0.00 |

7. Créez un tableau montrant, pour chaque pays sélectionné depuis la **selection** box créée précédemment, **le pourcentage d'utilisation de différentes ressources**, **l'émission médiane en gCO2kWh de ces ressources** et la **contribution spécifique de ses ressources aux émissions totales de CO2** lors de la production d'électricité.

Par exemple, si l'on sélectionne l'Albanie depuis la selection box, le tableau doit afficher les éléments suivants :

| Source de production | % d'utilisation Médiane de gCO2/kWh | Contribution en émission gCO2/kWh |
|----------------------|-------------------------------------|-----------------------------------|
| Charbon | 0 820 | 0% x 820 = 0 |
| Gaze Naturel | 0 490 | 0% x 490 = 0 |
| Pétrole | 0 740 | 0% x 740 = 0 |
| Hydro | 100 24 | 100% x 24 = 24 |
| Renouvelable | 0 41 | 0 % x 41 = 0 |
| Nucléaire | 0 12 | 0 % x 12 = 0 |

Contribution des Sources aux Émissions de CO2

Sélectionner un pays :

| Source de production | % d'utilisation | Médiane de gCO2/kWh | Contribution en émission gCO2/kWh |
|----------------------|-----------------|---------------------|-----------------------------------|
| Coal | 0 | 820 | 0% x 820 = 0 |
| Gas | 0 | 490 | 0% x 490 = 0 |
| Oil | 0 | 740 | 0% x 740 = 0 |
| Hydro | 100 | 24 | 100% x 24 = 24 |
| Renewable | 0 | 41 | 0% x 41 = 0 |
| Nuclear | 0 | 12 | 0% x 12 = 0 |

| Sélectionner un pays : <input type="text" value="Bolivia"/> <input type="button" value="Filtrer"/> | | | |
|--|-----------------|---------------------|-----------------------------------|
| Source de production | % d'utilisation | Médiane de gCO2/kWh | Contribution en émission gCO2/kWh |
| Coal | 0 | 820 | 0% x 820 = 0 |
| Gas | 100 | 490 | 100% x 490 = 490 |
| Oil | 100 | 740 | 100% x 740 = 740 |
| Hydro | 100 | 24 | 100% x 24 = 24 |
| Renewable | 100 | 41 | 100% x 41 = 41 |
| Nuclear | 0 | 12 | 0% x 12 = 0 |

8. Calculez et affichez l'**émission totale des différentes sources** d'un pays sélectionné : **émissions totales = émission de charbon + émission de gaze + ... + émission de nucléaire. Par exemple : 0 + 0 + 0 + 24 +... + 0 = 24 gCO2/kWh.**

| Sélectionner un pays : <input type="text" value="Belarus"/> <input type="button" value="Filtrer"/> | | | |
|--|-----------------|---------------------|-----------------------------------|
| Source de production | % d'utilisation | Médiane de gCO2/kWh | Contribution en émission gCO2/kWh |
| Coal | 100 | 820 | 100% x 820 = 820 |
| Gas | 100 | 490 | 100% x 490 = 490 |
| Oil | 100 | 740 | 100% x 740 = 740 |
| Hydro | 100 | 24 | 100% x 24 = 24 |
| Renewable | 100 | 41 | 100% x 41 = 41 |
| Nuclear | 0 | 12 | 0% x 12 = 0 |

Émission totale : 99.90 gCO2/kWh

Emission totale CO2 par Pays sélectionné
 Résultat Émissions totale CO2 par Pays sélectionné : 99.90 KgCO2/kWh

9. Calculez et affichez l'**émission totale annuelle pour un pays** (toujours depuis une selection box). On définit la formule permettant de calculer cette valeur comme suit :

Émissions annuelles totales de CO2 = Émissions totales en kgCO2/kWh x nombre d'heures dans une année x consommation électrique, où la consommation électrique en kw doit être spécifiée par l'utilisateur.

Afin d'y voir pour clair, faisons ce calcul pour l'Albanie, on a :

1. Émissions totales de l'Albanie en kgCO2/kWh : **0,024 (kgCO2/kwh)**
2. Nombre d'heures dans une année : **24 x 365**

3. Puissance électrique consommée de manière continue par l'Albanie
(donnée choisie par l'utilisateur) : **1 (kw)**

On calcule donc à l'aide de **la formule précédente**, les émissions en CO2 de l'Albanie durant une année pour la production d'électricité consommée de 1 (kw) par heure, par jour : **0,024 x 24 x 365 x 1 = 210,24 (kg of CO2)**.

Calcul des émissions annuelles pour un Pays

Sélectionner un pays : Consommation électrique par heure (kW) :

Cette partie du code je n'ai pas réussi à la finir . Le résultat ne s'affiche pas je n'arrive pas à le récupérer. Je me pencherais dessus plus tard car nous devons rendre le devoir ce soir.

10. Pour finir, en sachant qu'**un arbre absorbe environ 25 kg de CO2 par an**, affichez le **nombre nécessaire d'arbres à planter afin d'absorber le CO2 engendré par un pays** (sélectionné depuis la selection box, et oui) durant la production d'électricité.

Contribution des Sources aux Émissions de CO2

Sélectionner un pays :

| Source de production | % d'utilisation | Médiane de gCO2/kWh | Contribution en émission gCO2/kWh |
|----------------------|-----------------|---------------------|-----------------------------------|
| Coal | 0 | 820 | 0% x 820 = 0 |
| Gas | 0 | 490 | 0% x 490 = 0 |
| Oil | 0 | 740 | 0% x 740 = 0 |
| Hydro | 100 | 24 | 100% x 24 = 24 |
| Renewable | 0 | 41 | 0% x 41 = 0 |
| Nuclear | 0 | 12 | 0% x 12 = 0 |

Emission totale CO2 par Pays sélectionné

Résultat Émissions totale CO2 par Pays sélectionné : 100.00 KgCO2/kWh

Nombre d'arbres nécessaires pour absorber le CO2

Nombre d'arbres nécessaires : 4 arbres

(BONUS) Et encore du SQL...

Après avoir fini les 11 jobs précédents, vous souhaitez **encore vous exercer sur SQL**, vous êtes bien courageux ! Vous réalisez donc les jobs facultatifs suivants en ayant



pour but de **parfaire vos connaissances**.

Job Video Games Sales

Soit l'ensemble de données **Video Games Sales** contenant une liste de jeux vidéo vendus à plus de 100 000 exemplaires, scrappé depuis le site internet **vgchartz.com** (spécialisé dans la compilation de ventes de jeux vidéo). Récupérez le jeu de données depuis **[ce lien](#)** et **analysez le** en répondant aux questions suivantes :

1. Quelles sont **les valeurs minimales et maximales** pour chaque colonne numérique (Year, NA_Sales, EU_Sales, JP_Sales, Other_Sales) ?
2. Quelles sont **les ventes totales en Amérique du Nord, en Europe, au Japon** et dans **d'autres régions** ?
3. Quelle est **la répartition des ventes** dans **chaque région** ?
4. **Combien de jeux ont été publiés** chaque année ?
5. Quel est **le jeu ayant les ventes totales les plus élevées**, et **sur quelle plateforme** a-t-il été **le plus vendu** ?
6. Quelle **plateforme** a **le plus grand total de ventes** ?
7. Quel genre a **la plus grande moyenne de ventes** ?
8. **Comment évoluent les ventes** au fil des années ?
9. Quelle est **la part de marché de chaque éditeur en termes de ventes totales** ?
10. Qui sont **les principaux éditeurs** en termes de ventes ?

11. Quels sont **les jeux avec les meilleures ventes** dans chaque genre de jeu ?

12. Voyez-vous **une autre problématique à analyser** pour ce jeu de données ?

Job Google Play Store Apps

Soit l'ensemble de données **Google Play Store**,
constitué de 10

000 applications du Play Store scrappées dans le but
afin

d'**analyser le marché d'applications sur Android**.

Récupérez le

jeu de données depuis **ce lien** et **analysez le** en répondant aux
questions suivantes :



1. Quelle est la **note moyenne de toutes les applications** du
Play Store ?

2. Quelle est la **note la plus élevée et la plus basse** du jeu de données ? 3.

Quelle est la **proportion d'applications gratuites et d'applications payantes** ?

4. Quelle est la **moyenne du nombre d'avis pour les applications gratuites et
payantes** ?

5. **Combien d'applications** y a-t-il dans chaque catégorie ?

6. Quelle est la **note moyenne pour chaque catégorie** ?

7. Quelle **catégorie d'applications a la moyenne la plus élevée** d'avis ?

8. Quelle est **la taille moyenne des applications** dans **chaque catégorie** ?

9. Semble-il y avoir une **corrélation entre la taille de l'application et le nombre
d'installations** ?

10. **Combien d'applications existent** pour chaque évaluation du contenu ?

11. Quelle est **la note moyenne pour chaque évaluation** du contenu ?

12. Quelle **évaluation du contenu a le plus grand nombre d'installations** ?
13. Quels **sont les genres les plus courants** dans le jeu de données ?
14. **Combien d'applications** appartiennent à **plusieurs genres** ?
15. Semble-il y avoir **une corrélation entre le fait d'avoir plusieurs genres et des notes plus élevées** ?
16. Quel est **le prix moyen des applications payantes dans chaque catégorie** ?
17. Combien d'**applications gratuites existent dans chaque catégorie** ?
18. Quelles sont les **5 meilleures applications** avec le **plus grand nombre d'installations** ?
19. Quelles sont les **5 meilleures applications gratuites** avec les **notes les plus élevées** ?
20. Semble-il y avoir **une corrélation entre la taille d'une application et sa note** ?
21. Les applications **plus petites sont-elles plus susceptibles d'être gratuites** ?
22. Semble-il y avoir **une corrélation entre le prix d'une application et sa note** ?

Compétences visées

- Systèmes de gestion de base de données
- SQL
- Flask

Rendu

Votre travail devra être sauvegardé dans un repository sur github appelé **hello-DBMS**. Ce repository devra contenir les éléments suivants :

- **La veille scientifique** réalisée dans votre fichier **README** (en plus du contexte du

projet, comme à votre habitude).

- Les **scripts SQL** des différents jobs de la section Hello SQL. Faites en sorte de nommer chaque script selon le job (job1.sql, job2.sql,...).
- Le **script Python de l'application Flask du big job** ainsi que les **scripts HTML et CSS** (l'application doit être un minimum plaisante à regarder).

Base de connaissances

- [Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2020, with forecasts from 2021 to 2025](#)
- [Qu'est-ce qu'un système de gestion de base de données](#)
- [MySQL - The Basics // Learn SQL in 23 Easy Steps](#)
- [SQL – Tout savoir sur le langage de programmation des bases de données](#)
- [SQL.sh - Apprendre le SQL](#)
- [Practice SQL](#)
- [SQL Cheatsheet](#)
- [NoSQL : Tout comprendre sur les bases de données non relationnelles](#)
- [7 Database Paradigms](#)