

# COMPTE RENDU SQL



DEFRANCE  
BAPTISTE

## Table des matières

<b>Fichiers principaux :</b>	<b>4</b>
Création du schéma :	4
Migration des données :	4
Informations précise :	4
Base principale :	4
Nouvelle base :	4
<b>Fichiers secondaires :</b>	<b>4</b>
Suppression des données :	4
Menu de sélection :	4
Menu de sélection en python :	4
<b>Le projet :</b>	<b>4</b>
Introduction .....	4
Coté professionnel .....	4
<b>SQLITE</b> .....	<b>5</b>
<b>Organisation</b> .....	<b>5</b>
<b>Introduction</b> .....	<b>5</b>
Trello .....	5
VS Code .....	6
Navicat Data.....	6
<b>Quelques notions à maîtriser :</b> .....	<b>7</b>
<b>Les types de clés</b> .....	<b>7</b>
Clés primaires : .....	7
Clés étrangères : .....	7
<b>Les requêtes</b> : .....	<b>8</b>
Requêtes dans le terminal : .....	8
Requêtes dans un fichier .sql :	8
<b>Syntaxe importante</b> :.....	<b>8</b>
INNER JOIN.....	8
IF NOT EXISTS.....	8
DROP.....	8
INSERT INTO.....	9
ATTACH / DETACH.....	9
ALIAS .....	9
<b>Création du schéma</b> .....	<b>9</b>
<b>Introduction</b> .....	<b>9</b>
Mise en place .....	11
<b>Migration de données</b> .....	<b>12</b>
<b>Introduction</b> .....	<b>12</b>
Script SQLITE .....	12

<b>Affichage d'informations .....</b>	<b>13</b>
<b>Introduction .....</b>	<b>13</b>
<b>Première requête .....</b>	<b>13</b>
<b>Deuxième requête.....</b>	<b>14</b>
<b>Troisième requête.....</b>	<b>14</b>
<b>Quatrième requête.....</b>	<b>15</b>
<b>Plus loin .....</b>	<b>17</b>
<b>Suppression.....</b>	<b>17</b>
<b>Bash .....</b>	<b>18</b>
<b>Python .....</b>	<b>18</b>
Introduction .....	18
Menu.....	19
Migration .....	19
Modifier le contenu d'une table .....	20
Informations sur la base de données :	21
Détection des doublons : .....	22
Créer sa propre base de données .....	22
<b>Conclusion .....</b>	<b>23</b>

## Fichiers principaux :

Création du schéma : `schema.sql`

Migration des données : `migration.sql`

Informations précise : `queries.sql`

Base principale : `tpb2.db`

Nouvelle base : `migration.db`

## Fichiers secondaires :

Suppression des données : `drop.sql`

Menu de sélection : `menu.sh`

Menu de sélection en python : `migration.py`

## Le projet :

### Introduction

Ce projet SQL qui nous a été donné contient plusieurs étapes à effectuer. Nous avons à notre disposition un schéma de base de données ainsi que sa base de données correspondante et nous avons un deuxième schéma sans base de données. Le but sera donc de créer ce nouveau schéma avec les tables et les colonnes précisées dans un script SQL. Une fois notre nouveau schéma créé dans un fichier .db, nous devons le remplir avec un script SQL qui servira pour transférer les données existantes de la première base vers notre deuxième base. Pour finir, il faudra créer un troisième script SQL qui permettra d'effectuer des requêtes SQL sur notre nouvelle base et récupérer certaines informations demandées.

### Coté professionnel

Ce projet peut être vu d'un côté professionnel, le but de ce projet va être de permettre à un utilisateur de pouvoir effectuer de nombreuses actions comme la migration, la création de schéma etc... On va donc développer un outil simple d'utilisation permettant d'effectuer des tâches très simplement.

## SQLITE

SQLite est un système de gestion de bases de données relationnelles open-source et léger. Il est écrit en langage C et permet de stocker des données dans des fichiers en format SQL. Il est souvent utilisé pour les applications mobiles et les programmes de petite taille en raison de sa facilité d'utilisation et de sa petite taille de code. Il ne nécessite pas de serveur de base de données externe pour fonctionner, ce qui le rend également populaire pour les applications autonomes.

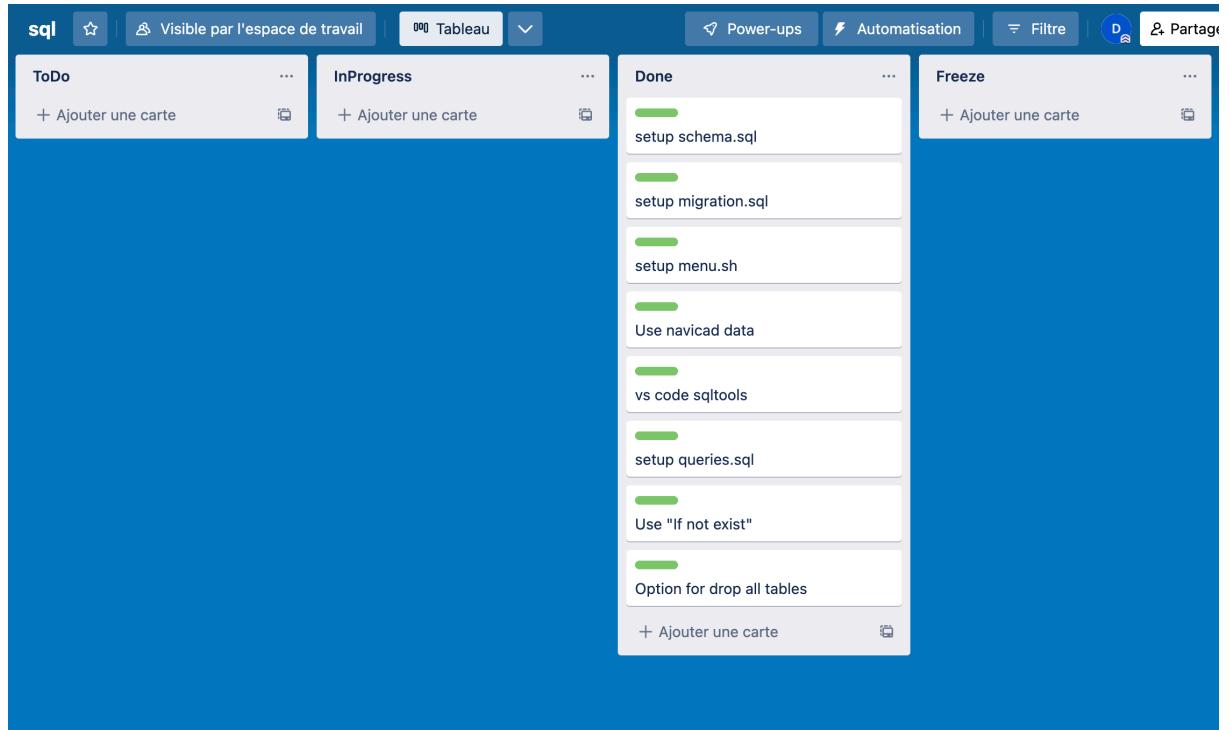
## Organisation

### Introduction

Pour mener à bien ce projet, j'ai décidé d'utiliser plusieurs outils qui me serviront pour optimiser mon travail et être plus productif.

### Trello

Pour organiser les différentes tâches à effectuer en fonction du temps et ainsi respecter les DeadLines, j'utilise Trello avec le diagramme de Gant qui permettra de suivre mon avancé et de voir si je respecte bien la finalisation des différentes tâches en fonction du temps.



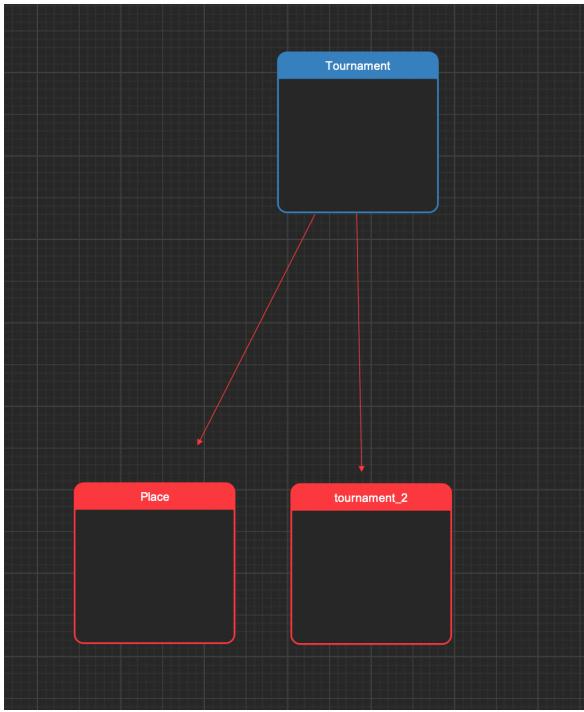
## VS Code

Pour pouvoir commencer à réaliser ce projet, j'ai utilisé l'IDE VS Code et j'ai installé l'extension SQLTools permettant de visualiser mes deux bases de données et ainsi pouvoir exécuter mes requêtes sur la base voulu.

ID Coach	ID Game	License Date	ID Employee Data
1	1	2023-01-09T09:41:0...	17
2	2	2023-02-27T09:41:...	18
3	3	2023-03-23T09:41:...	19
4	4	2023-04-24T09:41:...	20
5	5	2023-05-08T09:41:...	21
6	6	2023-06-05T09:41:...	22

## Navicat Data

Pour pouvoir mieux visualiser mes schémas et pour pouvoir y ajouter des modifications, j'ai utilisé cet outil qui permet de créer des schémas pour des bases de données et ainsi pouvoir le modeler à ma manière. Cela m'a permis de créer un brouillon et de marquer les étapes à effectuer en priorité lors du transfert de données.



## Quelques notions à maîtriser :

Les types de clés :

Clés primaires :

Les clés primaires sont des identifiants qui permettent d'identifier chaque enregistrement. Les clés primaires sont des INT. Une clé primaire ne peut contenir de valeur NULL et doit être auto-incrémenté pour pouvoir avoir un identifiant différent du précédent. Voici la syntaxe à utiliser en sqlite pour créer une clé primaire :

```
IdTournament INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
```

Clés étrangères :

Les clés étrangères sont des champs qui pointent vers une clé primaire d'une autre table. Cela permet de créer une référence entre deux tables avec un côté « mère / fille ». Voici la syntaxe à utiliser en sqlite pour créer une clé étrangère :

```
FOREIGN KEY(IdPlace) REFERENCES Place(IdPlace),
```

Les requêtes :

Il existe deux manières d'effectuer ses requêtes en sqlite :

Requêtes dans le terminal :

Il est possible d'effectuer des requêtes SQL dans notre terminal. En effet, il suffit de préciser le nom de la base de données dans laquelle nous souhaitons effectuer nos requêtes :

```
sqlite3 ma_base.db
```

Une fois cette commande lancée, nous pouvons effectuer nos requêtes.

Requêtes dans un fichier .sql :

Il est possible de mettre ses requêtes dans un fichier .sql puis de l'exécuter dans le terminal pour éviter de les taper à la main dans celui-ci. Pour faire cela, nous créons un fichier .sql, nous mettons nos requêtes à l'intérieur et nous le lançons directement dans le terminal :

```
sqlite3 ma_base.db < mon_script.sql
```

Une fois cette commande lancée, nous avons toutes les requêtes contenues dans notre script qui s'exécuteront dans la base de données précisé.

Syntaxe importante :

INNER JOIN

Le mot clé « INNER JOIN » nous permet de pouvoir créer un lien entre différentes colonnes de notre base de données. Nous utilisons INNER JOIN pour pouvoir sélectionner seulement les résultats où les données des colonnes précisées correspondent.

IF NOT EXISTS

Le mot clef “IF NOT EXISTS” permet de préciser lors d'une requête d'effectuer celle-ci seulement si la table ou la structure en question existe.

DROP

Le mot clef « DROP » permet de pouvoir supprimer tout une table. Il ne s'arrête pas à enlever les données mais bien à supprimer la structure de notre table.

## INSERT INTO

Le mot clef « INSERT INTO » va nous permettre de pouvoir insérer de la donnée dans une ou plusieurs colonnes.

## ATTACH / DETACH

Les mots clefs « ATTACH » et « DETACH » nous permettent de pouvoir faire une liaison entre deux bases .db. En effet, sans ces termes nous ne pouvons pas préciser une table d'une base de données étrangère si celle-ci n'est pas reconnue.

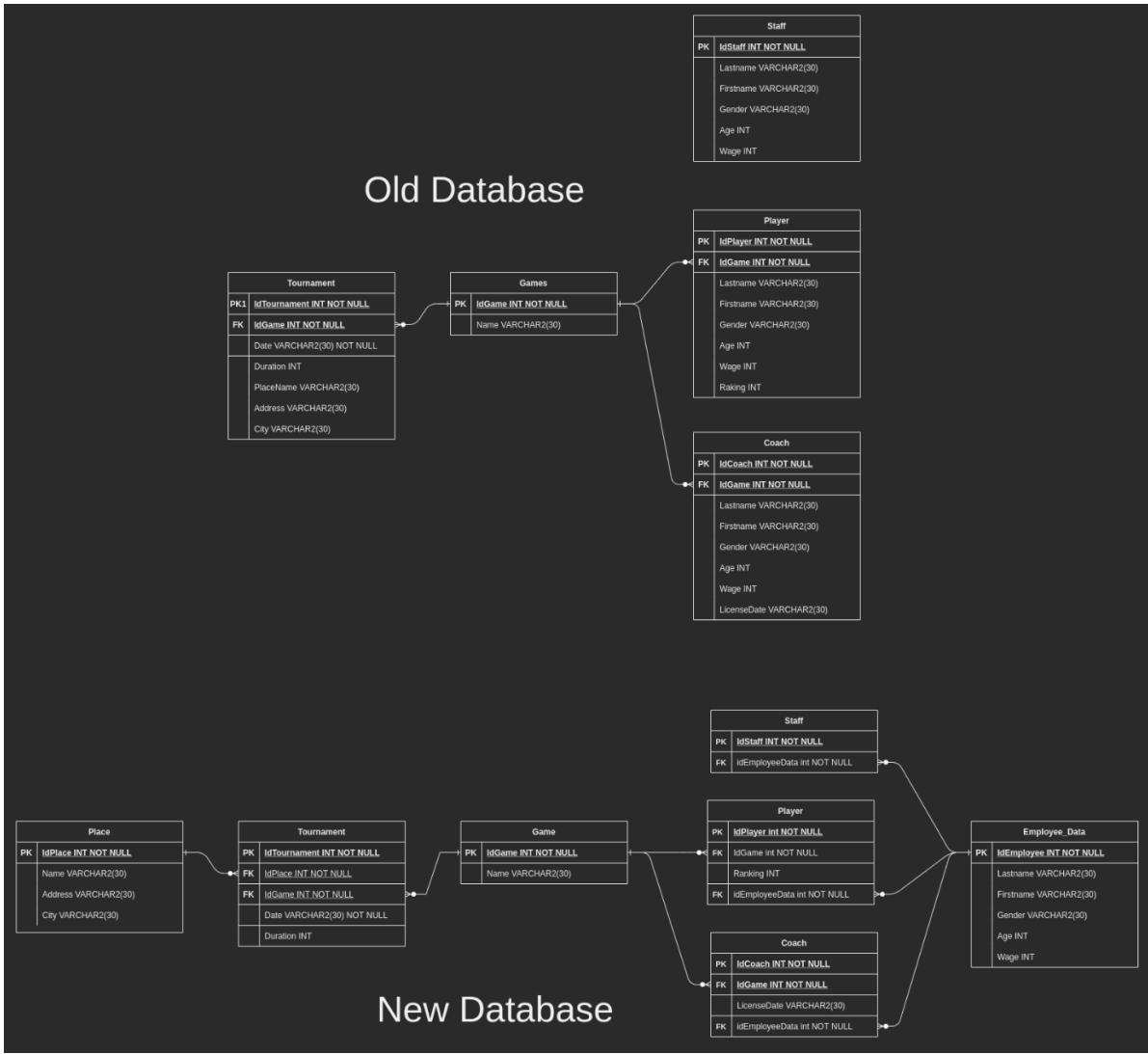
## ALIAS

Le mot clef « AS » faisant référence à ALIAS permet de pouvoir renommer dans les requêtes un nom de structure. Par exemple, nous allons renommer le nom de nos bases de données dans les requêtes pour rendre le code plus lisible et plus compréhensible.

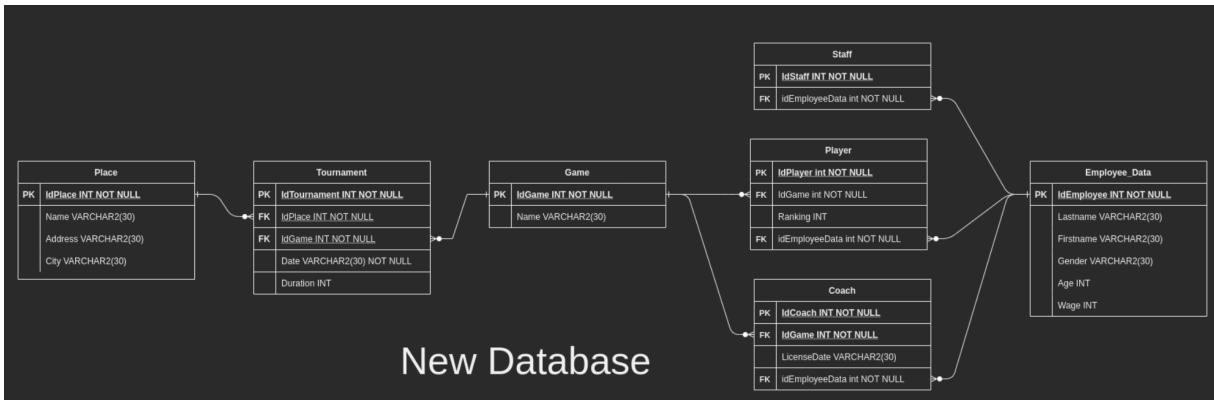
## Création du schéma

### Introduction

Pour rappel, nous avons à notre disposition le schéma de l'ancienne base de données et le schéma de la nouvelle base de données :



Pour la création du deuxième schéma, nous ne nous préoccuperais pas des données. Nous devons juste recréer un schéma avec les informations sur les tables ainsi que les colonnes comme leurs noms ou encore leurs types ainsi que le type des clés.



## Mise en place

Pour cela j'ai créé un script qui se nomme schema.sql qui contiendra la création de toutes ces tables ainsi que des colonnes qui les composent et qui à son exécution créera ce schéma en une seule fois.

```
schema.sql
▶ Run on active connection | Select block
1 CREATE TABLE IF NOT EXISTS Place(
2     IdPlace INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
3     Name VARCHAR2(30),
4     Address VARCHAR2(30),
5     City VARCHAR2(30)
6 );
7
8 CREATE TABLE IF NOT EXISTS Tournament(
9     IdTournament INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
10    IdPlace INTEGER NOT NULL,
11    IdGame INTEGER NOT NULL,
12    Date VARCHAR2(30) NOT NULL,
13    Duration INTEGER,
14    FOREIGN KEY(IdPlace) REFERENCES Place(IdPlace),
15    FOREIGN KEY(IdGame) REFERENCES Game(IdGame)
16 );
17
18 CREATE TABLE IF NOT EXISTS Game(
19     IdGame INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
20     Name VARCHAR2(30)
21 );
22
23 CREATE TABLE IF NOT EXISTS Employee_Data(
24     IdEmployee INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
25     Lastname VARCHAR2(30),
26     Firstname VARCHAR2(30),
27     Gender VARCHAR2(30),
28     Age INTEGER,
```

Comme on peut le voir ici, nous avons toute la création de nos tables ainsi que les colonnes qui leur sont associées en suivant le plan du schéma mis à disposition.

J'ai rajouté le mot clé « IF NOT EXISTS » devant chaque table permettant de la créer uniquement si celle-ci n'est pas déjà présente et ainsi éviter des erreurs de requêtes.

Nous avons maintenant notre schéma de notre nouvelle base qui est finit.

# Migration de données

## Introduction

Nous avons notre nouvelle base qui est prête à recevoir les données contenues dans l'ancienne base. Nous allons effectuer une migration de données qui déplace les données contenues dans la première base dans les colonnes correspondantes dans notre deuxième base. Nous allons voir que ce n'est pas si simple que cela car nous devons prendre en compte que dans notre nouvelle base, nous avons des tables qui se sont ajoutés, des colonnes qui ont changés de noms etc...

Pour faire cette migration de données, il existe plusieurs moyens afin d'y arriver.

## Script SQLITE

Cette première méthode consiste à créer un script .sql, à se connecter à nos deux bases et de préciser où ira la donnée. Nous devons donc commencer par dire quelles seront les deux bases à utiliser et par la suite commencer à remplir nos colonnes. Pour ce script, j'ai commencé à remplir mes tables qui ne contiennent pas de clés étrangères. Cela est primordial car si nous remplissons d'abord des tables avec des clés étrangères, on ne pourra pas indiquer le chemin des clés primaires qui leurs correspondent car elles n'existeront pas.

On commence donc par remplir nos 3 tables sans clés étrangères :

- Place
- Game
- Employee\_Data

Dans ce dispositif, on voit que l'ancienne table tournament n'est plus pareille sur notre nouvelle base, celle-ci s'est divisé en deux tables : Place et Tournament.

Une fois que nos 3 tables sont remplies, nous pouvons remplir les tables restantes car nous avons toutes les clés primaires dont on a besoin qui sont créées. Pour ce qui est des clés primaires, inutile de les remplir car nous avons faites en sorte de créer les colonnes avec le mode « AUTOINCREMENT » qui les incrémentera automatiquement lorsque les autres colonnes seront remplies.

Attention, certaines tables comportant des clés secondaires ont besoins d'une condition pour créer une liaison entre les différentes tables

Si toutes ces requêtes se sont bien passées, nous avons maintenant notre nouvelle base de données qui est entièrement remplie avec les données de notre première base.

# Affichage d'informations

## Introduction

On nous demande maintenant d'afficher certaines informations en fonction de plusieurs paramètres. Tous ces informations sont tirées de notre nouvelle base de données.

Pour faire ceci, nous allons créer différentes requêtes qui vont rechercher dans la base de données les informations que l'on souhaite récupérer.

- List every tournament for a given game name
- Given a game name, retrieve the average wage of the players
- List all tournament by place
- Get the number of players by gender

## Première requête

Pour la première requête, il faut récupérer toutes les informations de la table tournament avec n'importe quel nom de partie.

Pour faire cela, je suis donc passé par le nom de la partie d'où l'on souhaite récupérer les informations. Le lien qu'à le nom de la partie dans sa table (Game) est l'IdGame. En effet, celui-ci est aussi présent dans la table tournament. On a donc maintenant notre chemin pour récupérer toutes les informations de la table Tournament. On effectue donc la requête en question dans un script .sql :

Pour cet exemple j'ai choisi de prendre le jeu Tekken 7.

```
SELECT * FROM Tournament INNER JOIN Game ON  
Tournament.IdGame = Game.IdGame WHERE Game.Name = 'Tekken 7';
```

IdPlace	IdGame	IdTournament	Duration	Date	Name
abc Filter...	abc Filter...				
2	2	3	12	2022-11-05T09:00:0...	Tekken 7

## Deuxième requête

Pour la deuxième requête, il faut récupérer l'âge moyen des joueurs en fonction d'un nom de jeu.

Pour faire cela, je suis parti de la table Employee\_Data car c'est cette table qui contient les informations sur les salaires de tous les employés. J'ai par la suite grâce à INNER JOIN récupérer seulement les ID des employées qui sont des joueurs en liant IdEmployeeData de la table player avec IdEmployee de la table Employee\_Data. J'ai donc maintenant les ID de tous les joueurs. Je récupère ensuite les IdGame des joueurs en question et comme IdGame est aussi présent dans la table Game je peux les lier entre eux. Il me reste plus qu'à faire une condition en fonction du nom de jeux qui est demandé :

```
SELECT AVG(Employee_Data.Wage) FROM Employee_Data INNER JOIN Player
ON Employee_Data.IdEmployee = Player.IdEmployeeData
INNER JOIN Game ON Player.IdGame = Game.IdGame
WHERE Game.Name = 'Tekken 7';
```

AVG(Employee...
abc Filter...
135000

J'utilise la fonction AVG() qui me permet de calculer une moyenne. Par exemple, pour le jeu Tekken 7, la moyenne des salaires des joueurs qui jouent à ce jeu est de 135 000.

## Troisième requête

Pour la troisième requête, il faut lister tous les tournois en fonction de leur lieu de déroulement.

Pour faire cela, on récupère le IdPlace dans la table tournament et comme celle-ci existe aussi dans la table Place, on peut donc faire cette requête :

```
SELECT * FROM Tournament INNER JOIN Place ON
Tournament.IdPlace = Place.IdPlace;
```

IdPlace	IdGame	IdTournament	Duration	Date	Name	Address	City
1	1	1	7	2022-12-01T09:00:00	home	1 Rue cassée	BrequinVille
5	1	2	7	2022-12-01T09:00:00	home	1 Rue cassée	BrequinVille
2	2	3	12	2022-11-05T09:00:00	Stadium	3 Rue des champs	ParisVille
3	4	4	2	2022-10-10T09:00:00	Arenes	12 Parsec	YouVille
4	3	5	3	2022-09-15T09:00:00	Stade	33 Rue du stade	StadeVille
1	1	6	2	2022-08-20T09:00:00	home	1 Rue cassée	BrequinVille
5	1	7	2	2022-08-20T09:00:00	home	1 Rue cassée	BrequinVille

J'ai aussi créé une requête qui permet de lister les tournois en fonction de leur lieu de déroulement. Par exemple j'ai décidé de trier les tournois en fonction du lieu qui se nomme « Arenes » :

```
SELECT * FROM Tournament INNER JOIN Place ON
Tournament.IdPlace = Place.IdPlace
WHERE Place.Name = 'Arenes'
```

IdPlace	IdGame	IdTournament	Duration	Date	Name	Address	City
3	4	4	2	2022-10-10T09:00:00	Arenes	12 Parsec	YouVille

## Quatrième requête

Pour la quatrième requête, il faut lister le nombre de joueurs en fonction de leur genre.

On sait à l'avance qu'il existe dans notre base 3 différents types de genres :

- Male
- Female
- Other

On peut donc s'attendre à retrouver 3 résultats, le nombre de joueurs homme, le nombre de joueur femme et le nombre de joueurs avec comme genre other.

Pour ce faire, on récupère comme fais plus haut les Id des joueurs dans notre table Employee\_Data et on va utiliser le terme GROUP BY en précisant que l'on veux grouper notre résultat par le genre pour avoir le nombre de joueurs en utilisant par-dessus la fonction COUNT() :

```
SELECT COUNT(Employee_Data.IdEmployee) FROM Employee_Data INNER JOIN Player  
ON Employee_Data.IdEmployee = Player.IdEmployeeData  
GROUP BY Employee_Data.Gender
```

COUNT(Employee_Data.IdEmployee)
abc Filter...
4
5
3

## Plus loin

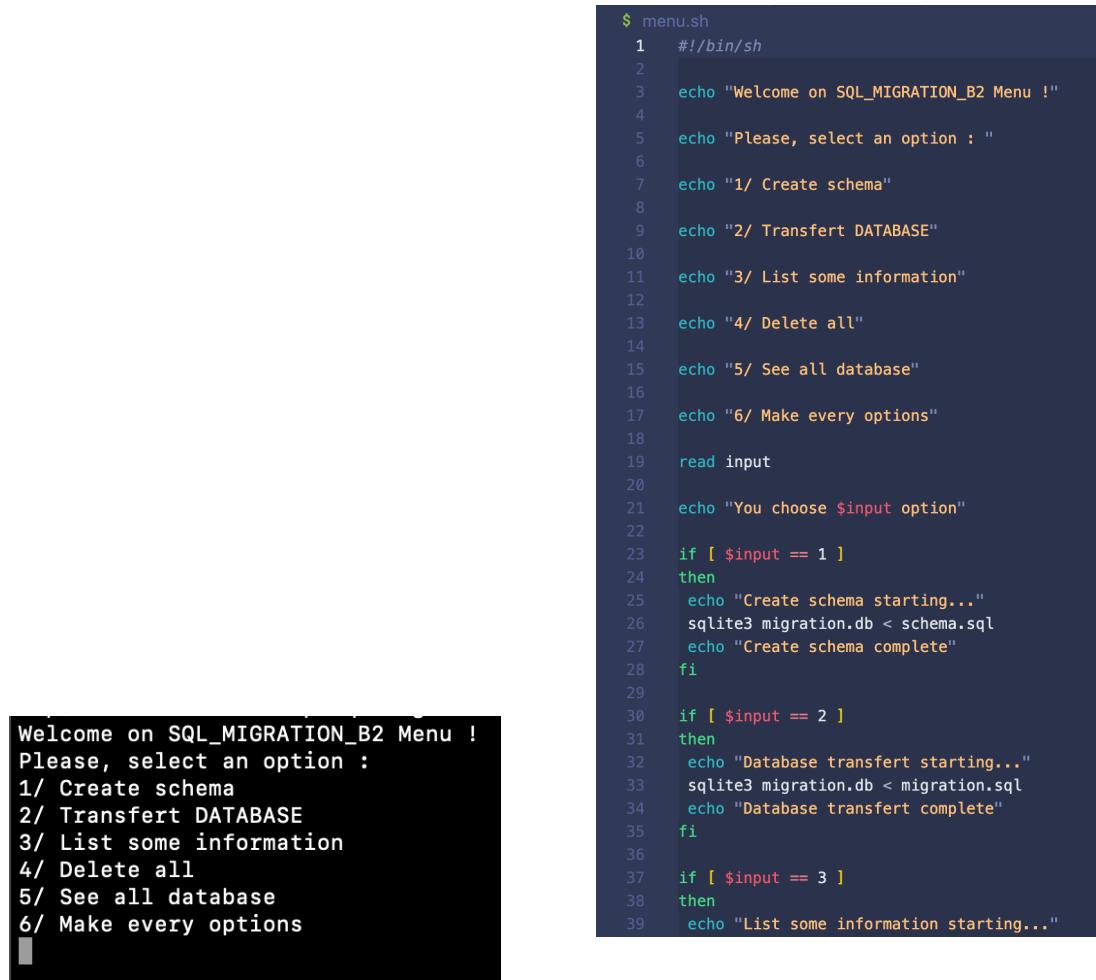
### Suppression

Pour pouvoir rendre ce projet plus poussé dans les actions possible, j'ai créé un script SQL qui permet à son lancement de pouvoir supprimer toutes les tables de ma base et de pouvoir recréer au propre mon schéma et ainsi pouvoir effectuer ma migration :

```
DROP TABLE IF EXISTS Game;  
  
DROP TABLE IF EXISTS Tournament;  
  
DROP TABLE IF EXISTS Place;  
  
DROP TABLE IF EXISTS Employee_Data;  
  
DROP TABLE IF EXISTS Staff;  
  
DROP TABLE IF EXISTS Player;  
  
DROP TABLE IF EXISTS Coach;
```

## Bash

Pour lancer ces scripts j'ai créé un script en bash avec plusieurs options permettant à un utilisateur de pouvoir effectuer les taches qu'il souhaite grâce à ce script :



```
$ menu.sh
1  #!/bin/sh
2
3 echo "Welcome on SQL_MIGRATION_B2 Menu !"
4
5 echo "Please, select an option : "
6
7 echo "1/ Create schema"
8
9 echo "2/ Transfert DATABASE"
10
11 echo "3/ List some information"
12
13 echo "4/ Delete all"
14
15 echo "5/ See all database"
16
17 echo "6/ Make every options"
18
19 read input
20
21 echo "You choose $input option"
22
23 if [ $input == 1 ]
24 then
25   echo "Create schema starting..."
26   sqlite3 migration.db < schema.sql
27   echo "Create schema complete"
28 fi
29
30 if [ $input == 2 ]
31 then
32   echo "Database transfert starting..."
33   sqlite3 migration.db < migration.sql
34   echo "Database transfert complete"
35 fi
36
37 if [ $input == 3 ]
38 then
39   echo "List some information starting..."
```

```
Welcome on SQL_MIGRATION_B2 Menu !
Please, select an option :
1/ Create schema
2/ Transfert DATABASE
3/ List some information
4/ Delete all
5/ See all database
6/ Make every options
```

## Python

### Introduction

Pour pouvoir avoir une structure complètement modifiable, j'ai intégré mes requêtes SQL précédemment créée à un script python utilisant la bibliothèque sqlite3. Grâce à ce script j'ai pu rendre un outil complet pour un utilisateur qui souhaite manier ses bases de données.

## Menu

Pour cela, j'ai créé un menu en python qui permet de choisir d'effectuer plusieurs actions :

```
def menu():
    print("Welcome on SQL_MIGRATION_B2 Menu !")
    print("Please, select an option : ")
    print("1/ Create schema")
    print("2/ Transfert DATABASE")
    print("3/ List some information")
    print("4/ Delete all")
    print("5/ See all database")
    print("6/ Add a table in the schema")
    print("7/ Create a database")
    print("8/ Show Duplicates")
    choix = int(input("Please select an option : "))
    if choix == 1:
        db1 = input("Please, choose the name of your DB : ")
        print("The name of your database is "+db1)
        schema(db1)
    if choix == 2:
        db1 = input("Please, choose the name of your 1st DB : ")
        print("The name of your 1st database is "+db1)
        db2 = input("Please, choose the name of your 2st DB : ")
        print("The name of your 1st database is "+db2)
        migration(db1, db2)
    if choix == 3:
        db1 = input("Please, choose the name of your DB : ")
        print("The name of your database is "+db1)
        show_information(db1)
    if choix == 4:
        db1 = input("Please, choose the name of your DB : ")
        print("The name of your database is "+db1)
        delete(db1)
    if choix == 5:
        db = input("Please, choose the name of your DB : ")
        print("The name of your database is "+db)
        information_database(db)
    if choix == 6:
        db1 = input("Please, choose the name of your DB : ")
        print("The name of your database is "+db1)
        name_table = input("Please select the name of you new table : ")
        row1 = input("Please select the name of your 1st row : ")
        type_row1 = input("Please select the type of your 1st row : ")
        row2 = input("Please select the name of your 2st row : ")
        type_row2 = input("Please select the type of your 2nd row : ")
        row3 = input("Please select the name of your 3rd row : ")
        type_row3 = input("Please select the type of your 3rd row : ")
        row4 = input("Please select the name of your 4th row : ")
        type_row4 = input("Please select the type of your 4th row : ")
        add_table(db1, name_table, row1, type_row1, row2, type_row2, row3, type_row3, row4, type_row4)
    if choix == 7:
        db1 = input("Please, choose the name of your DB : ")
        create_database(db1)
    if choix == 8:
        db = input("Please, choose the name of your DB : ")
        row1 = input("Please select the name of the table : ")
        show_duplicates[db, row1]
```

Voici ce que j'ai pu faire :

## Migration

J'ai pu adapter mes différentes actions pour que l'utilisateur puisse choisir le nom de ses bases de données. En effet, s'il décide de changer le nom des bases le script python permet

que les actions puissent fonctionner. Pour cela je demande avec un input de choisir le nom de la DB et j'utilise cette variable associée dans mes requêtes SQL :

```
db1 = input("Please, choose the name of your 1st DB : ")
print("The name of your 1st database is "+db1)
db2 = input("Please, choose the name of your 2st DB : ")
print("The name of your 1st database is "+db2)
migration(db1, db2)
```

```
def migration(db1, db2):
    conn = sqlite3.connect(db1)
    c = conn.cursor()
    c.executescript('''

ATTACH DATABASE '{}' AS source_db;
ATTACH DATABASE '{}' AS cible_db;
```

## Modifier le contenu d'une table

J'ai pu faire comme avec le script en Bash, ou encore avec mes scripts SQL, créer mon schéma avec mon script python en choisissant la base de données dans laquelle se créera le schéma.

J'ai aussi fait en sorte de pouvoir modifier ce schéma :

- Ajouter une table dans le schéma. Si l'utilisateur souhaite ajouter une table dans le schéma, j'ai fait un exemple avec la possibilité de remplir jusqu'à 4 colonnes.  
L'utilisateur a le choix de choisir le nom de la table, le nom et le type des 4 colonnes :

```
db1 = input("Please, choose the name of your DB : ")
print("The name of your database is "+db1)
name_table = input("Please select the name of your new table : ")
row1 = input("Please select the name of your 1st row : ")
type_row1 = input("Please select the type of your 1st row : ")
row2 = input("Please select the name of your 2nd row : ")
type_row2 = input("Please select the type of your 2nd row : ")
row3 = input("Please select the name of your 3rd row : ")
type_row3 = input("Please select the type of your 3rd row : ")
row4 = input("Please select the name of your 4th row : ")
type_row4 = input("Please select the type of your 4th row : ")
add_table(db1, name_table, row1, type_row1, row2, type_row2, row3, type_row3, row4, type_row4)
```

```

def add_table(db1, name_table, row1, type_row1, row2, type_row2, row3, type_row3, row4, type_row4):
    conn = sqlite3.connect(db1)
    c = conn.cursor()
    c.executescript('''CREATE TABLE IF NOT EXISTS {}(
        {} {},
        {} {},
        {} {},
        {} {}
    );'''.format(name_table, row1, type_row1, row2, type_row2, row3, type_row3, row4, type_row4))
    conn.commit()
    c.close()
    conn.close()

```

Ici je demande avec des input toutes les informations sur les tables ainsi que les colonnes et j'utilise ces variables dans ma requête.

Informations sur la base de données :

Pour aller plus loin dans mon intégration du SQL dans mon script python, j'ai créé une fonctionnalité permettant à l'utilisateur de pouvoir avoir des informations sur une base de données en lui demandant quelle est le nom de la base dans laquelle il veut récupérer ses informations. Les informations qui lui seront présentées seront :

- Le nom des tables
- Le nombres de tables de la base

```

def information_database(db):
    conn = sqlite3.connect('{}'.format(db))
    cursor = conn.cursor()
    cursor.execute("SELECT name FROM sqlite_master WHERE type='table';")
    tables = cursor.fetchall()
    print("Number of tables:", len(tables)-1)
    for table in tables:
        if table[0] != 'sqlite_sequence':
            print("Table name:", table[0])
    conn.close()

```

```
Please, choose the name of your DB : migration.db
The name of your database is migration.db
Number of tables: 7
Table name: Place
Table name: Tournament
Table name: Game
Table name: Employee_Data
Table name: Staff
Table name: Player
Table name: Coach
```

Détection des doublons :

Pour détecter les doublons présents dans ma base de données, j'ai fait une fonction `show_duplicates()` qui permet avec le nom de la base de données choisit par l'utilisateur ainsi que le nom de la table de détecter si il y a des doublons :

```
def show_duplicates(db, table):
    conn = sqlite3.connect('{}'.format(db))
    cursor = conn.cursor()
    table_name = "{}".format(table)
    cursor.execute("SELECT * FROM {}".format(table_name))
    rows = cursor.fetchall()
    unique_values = set()
    duplicates = set()
    for row in rows:
        if row in unique_values:
            duplicates.add(row)
        else:
            unique_values.add(row)
    print("Les doublons dans la table {} sont :{}".format(table_name), duplicates)
    conn.close()
```

Créer sa propre base de données

J'ai aussi mis en place une fonctionnalité permettant de pouvoir créer une table vide qui par la suite pourra être utilisé pour créer un schéma ou encore migrer des données dans celle-ci :

```
def create_database(db1):
    myFile = open(db1+".db", "w+")
    myFile.close()
    menu()
```

## Conclusion

J'ai donc créé un outil permettant d'effectuer de nombreuses actions sur ma base de données et permettre d'automatiser plein d'actions pour rendre le travail de l'utilisateur beaucoup plus simple.

J'ai donc pu adapter mon schéma de sortie en fonction des besoins de l'utilisateur, lui fournir des informations sur sa base de données, modifier le contenu d'une base existante en rajoutant des tables et des colonnes et vérifier s'il existe des doublons dans mes tables.

