26 – SQL SELECT FROM WHERE

I – DB Brownser

Il s'agit d'un Système de **Gestion de Bases de Données** (**SGBD**) léger, portable (on peut l'installer sur une clé USB) mais ne gèrant que **SQLite**.

Intro° Télécharger et installer le logiciel DB Brownser si ce n'est pas fait.
Intro° Télécharger depuis le site la base de données contenant films, emplois et personnes.
Intro° Ouvrir ce fichier via DB Browser.

II – Projection : SELECT FROM

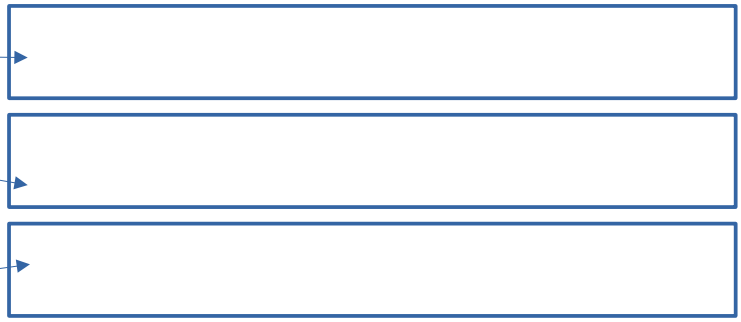
2.1 SQL

SQL veut dire Structured Query Language. Cela veut dire Langage de Requête Structurée.
C'est un **langage déclaratif** qui permet d'interroger la base de données qui fournit alors une réponse.
SQL est donc un langage où nous allons devoir placer des informations à certains endroits particuliers.

2.2 Les tables titre et personne

Déclaration du schéma relationnel de la table **titre** (implémentation SQL).

```
1     CREATE TABLE titre (
2       id INTEGER,
3       tconst VARCHAR(9) UNIQUE,
4       title_type VARCHAR(15),
5       primary_title VARCHAR(40),
6       original_title VARCHAR(40),
7       start_year INTEGER,
8       end_year INTEGER,
9       runtime_minutes INTEGER,
10      genres TEXT,
11      PRIMARY KEY(tconst)
12    );
```



Déclaration du schéma relationnel de la relation **titre**.

```
titre (      id INTEGER, tconst VARCHAR(9) UNIQUE, title_type VARCHAR(15),
             primary_title VARCHAR(40), original_title VARCHAR(40),
             start_year INTEGER, end_year INTEGER, runtime_minutes INTEGER, genres TEXT )
```

Déclaration du schéma relationnel de la table **personne** (implémentation SQL).

```
 1     CREATE TABLE personne (
 2       id INTEGER,
 3       nconst VARCHAR(12) UNIQUE,
 4       name VARCHAR(50),
 5       birth_year INTEGER CHECK(birth_year = 0 OR (birth_year > 1000 AND birth_year < 3000)),
 6       death_year INTEGER CHECK(death_year = 0 OR (death_year > 1000 AND death_year < 3000)),
 7       primary_profession TEXT,
 8       known_for_titles TEXT,
 9       PRIMARY KEY(nconst)
10     );
```

POUR LA SUITE DU COURS, PENSEZ A VOUS RENDRE SUR LE SITE POUR VOIR LES RESULTATS.

2.3 Projection avec SELECT FROM

L'opération de projection consiste à récupérer un sous-ensemble des n-uplets : on ne sélectionne que certains attributs.

Attention, l'ordre fourni à une influence sur la projection. Si on inverse le genre et la durée, on obtient bien ces attributs dans l'ordre fourni.

01° Fournir la requête SQL permettant de récupérer le nom, le numéro d'enregistrement et la date de création.

02° Fournir la requête SQL permettant de récupérer le nom, la date de naissance et la date de mort de la personne.

SQL 1

```
SELECT primary_title, runtime_minutes, genres
FROM titre ;
```

	primary_title	runtime_minutes	genres
1	Bohemios	100	
2	The Fairylogue and Radio-Plays	120	Adventure,Fantasy
3	Lucha por la herencia	92	
4	Cleopatra	100	Drama,History
5	Quo Vadis?	120	Drama,History
6	The Independence of Romania	120	History,War
7	Ana Kadova	92	
8	Atlantis	121	Drama
9	Germinal; or, The Toll of Labor	150	Drama
10	Ingeborg Holm	96	Drama
11	Mala raza	101	
12	Miraklet	100	Drama

```
L'exécution s'est terminée sans erreur.
Résultat : 218289 enregistrements ramenés en 41ms
À la ligne 1 :
SELECT primary_title, runtime_minutes, genres
FROM titre
```

2.4 Clause DISTINCT

Une opération assez commune consiste à trouver les différentes valeurs attribuées à un attribut (voir un résultat à droite)

```
1    SELECT DISTINCT title_type
2    FROM titre;
```

	title_type
1	movie
2	tvMovie
3	tvSeries
4	tvEpisode
5	tvMiniSeries
6	video

III – Condition avec WHERE

La clause **WHERE** permet de restreindre la sélection en précisant des conditions.

Nous allons retrouver les opérateurs de comparaisons habituels avec la syntaxe suivante :

- Egalité : = (contrairement à == en Python)
- Différence : <>
- Autres comparaisons : <, <=, >, >=
- Opérateurs logiques : AND, OR, NOT
- Opérateurs arithmétiques : +, -, *, /, %

```
1    SELECT primary_title, start_year, runtime_minutes
2    FROM titre
3    WHERE start_year = 2020;
```

```
1    SELECT primary_title, start_year, runtime_minutes
2    FROM titre
3    WHERE start_year <> 2020 AND runtime_minutes < 120;
```

```
1    SELECT primary_title
2    FROM titre
3    WHERE start_year <> 2020 AND runtime_minute
```

SQL 1

```
1    SELECT primary_title, start_year, runtime_minutes
2    FROM titre
3    WHERE start_year = 2020 ;
4
```

	primary_title	start_year	runtime_minutes
1	I Know This Much Is True	2020	378
2	The Witches	2020	106
3	Lost Girls and Love Hotels	2020	97
4	Road to Red	2020	120
5	Fantasy Island	2020	109
6	Jak najdalej stad	2020	98
7	Kala azar	2020	91
8	Villa Caprice	2020	103
9	A Deadly Legend	2020	97
10	Seeru	2020	123
11	Love Aaj Kal	2020	142
12	My Cousin	2020	104

```
L'exécution s'est terminée sans erreur.
Résultat : 7521 enregistrements ramenés en 135ms
À la ligne 1 :
SELECT primary_title, start_year, runtime_minutes
FROM titre
WHERE start_year = 2020
```

03° Fournir la requête permettant d'obtenir les titres ayant commencé entre entre 2010 et 2012.

04° Fournir la requête permettant d'obtenir les séries ayant commencé après 2018.

05° Fournir la requête permettant d'obtenir les séries finies ayant duré au moins 10 ans. Comme les attributs de date sont des entiers, une date non fournie équivaut à un 0.

3.2 Priorités du AND et du OR

Nous avions vu en 1er que a AND b est équivalent à a . b dans l'algèbre de Bool.
Nous avions vu en 1er que a OR b est équivalent à a + b dans l'algèbre de Bool.
On en déduit alors que le **AND est prioritaire sur le OR**

06° Voici deux requêtes permettant de récupérer les titres qui correspondent à une sortie entre 2010 et 2015. Les deux requêtes sont-elles similaires ?

Sans les parenthèses :

```
1   SELECT primary_title, start_year, end_year
2   FROM titre
3   WHERE start_year >= 2010 AND start_year <= 2015 OR end_year >= 2010 AND end_year <= 2015;
```

Avec les parenthèses :

```
1   SELECT primary_title, start_year, end_year
2   FROM titre
3   WHERE (start_year >= 2010 AND start_year <= 2015) OR (end_year >= 2010 AND end_year <= 2015);
```

IV – Commande ORDER BY

La commande **ORDER BY** permet de trier les lignes dans un résultat d'une requête SQL. On pourra trier les données sur une ou plusieurs colonnes, et par ordre croissant ou decroissant.

Tri simple croissant

```
1      SELECT primary_title, start_year
2      FROM titre
3      WHERE title_type = "movie"
4      ORDER BY start_year;
```

```
1      SELECT primary_title, start_year
2      FROM titre
3      WHERE title_type = "movie"
4      ORDER BY start_year ASC;
```

SQL 1

```
1   SELECT primary_title, start_year
2   FROM titre
3   WHERE title_type = "movie"
4   ORDER BY start_year;
5
```

	primary_title	start_year
1	Jeffries-Sharkey Contest	1899
2	Bohemios	1905
3	The Fairylogue and Radio-Plays	1908
4	L-A-Z-Y	1908
5	Jeffries-Johnson World's Championship ...	1910
6	Arsène Lupin contra Sherlock Holmes	1910
7	Viaje al interior del Perú	1910
8	Lucha por la herencia	1911
9	Defense of Sevastopol	1911
10	Il clarino di Tontolini	1911
11	Cleopatra	1912
12	The Independence of Romania	1912

```
L'exécution s'est terminée sans erreur.
Résultat : 149686 enregistrements ramenés en 659ms
À la ligne 1 :
SELECT primary_title, start_year
FROM titre
WHERE title_type = "movie"
ORDER BY start_year
```

Tri en ordre décroissant

On précise alors le suffixe **DESC**.

```
1      SELECT primary_title, start_year
2      FROM titre
3      WHERE title_type = "movie"
4      ORDER BY start_year DESC;
```

```
1     SELECT primary_title, start_year
2     FROM titre
3     WHERE title_type = "movie" AND start_year <> ''
4     ORDER BY start_year DESC;
```

Tri multiple

Comme vous pouvez le voir ci-dessous, les n-uplets sont triés par date mais apparaissent dans l'ordre d'inscription dans la base de données. Si vous voulez les trier par date puis par ordre alphabétique, il suffit de rajouter l'attribut primary_title après celui de la date.

```
1    SELECT primary_title, start_year
2    FROM titre
3    WHERE title_type = "movie" AND start_year <> ''
4    ORDER BY start_year DESC, primary_title ASC;
```

07° Fournir la requête permettant de recupérer les personnes nées entre 1970 et 1980 en les triant par ordre alphabétique.

08° Fournir la requête permettant de récupérer les films de 2020 (et uniquement les films) en les triant par durée, du plus long au moins long.

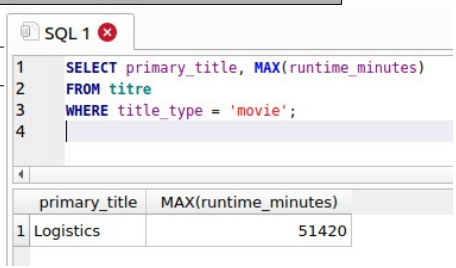
V – Fonctions d'agrégation

5.1 MAX

C'est assez étrange au premier abord : on ne place pas cette recherche de maximum dans le WHERE mais directement dans le SELECT. On ramène alors le nuplet qui correspond à ce maximum. Attention, avec cette façon d'agir, on récupère juste la première entrée en cas d'égalité.

Pour trouver la durée de film la plus longue :

```
1    SELECT primary_title, MAX(runtime_minutes)
2    FROM titre
3    WHERE title_type = 'movie';
```



5.2 MIN

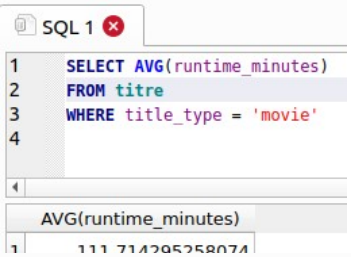
Pour trouver la durée de film la plus courte et le premier puplet correspondant :

```
1    SELECT primary_title, MIN(runtime_minutes)
2    FROM titre
3    WHERE title_type = 'movie';
```

5.3 AVG

Pour trouver la durée moyenne des films :

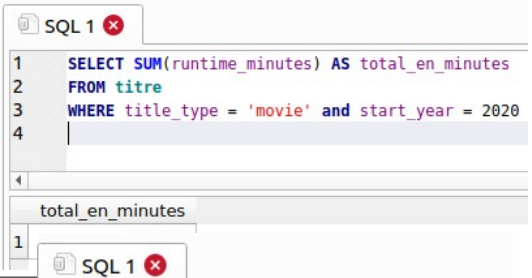
```
1     SELECT AVG(runtime_minutes)
2     FROM titre
3     WHERE title_type = 'movie';
```



5.4 SUM

Pour trouver la durée totale des films produits en 2020 :

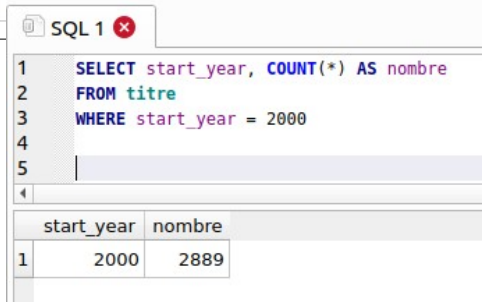
```
1   SELECT SUM(runtime_minutes) AS total_en_minutes
2   FROM titre
3   WHERE title_type = 'movie' and start_year = 2020;
```



5.5 COUNT

Cette fonction permet de compter le nombre de puplet renvoyé par notre requête. Pour l'exemple, voici le nombre de films parus en 2000.

```
1     SELECT start_year, COUNT(*) AS nombre
2     FROM titre
3     WHERE start_year = 2000;
```



09° Fournir la requête pour récupérer la durée moyenne des films de 2010 à 2020.

10° Fournir la requête permettant de compter le nombre de titres de plus de 180 minutes.

Hors programme :

Voici un exemple où on recherche TOUS les puplets dont la durée est la durée minimale.

```
1   SELECT primary_title, runtime_minutes, start_year
2   FROM titre
3   WHERE runtime_minutes = (SELECT MIN(runtime_minutes) AS minimum FROM titre WHERE title_type = 'movie');
```

IV – Like

On utilise LIKE dans la clause WHERE. Cela permet de rechercher un modèle de réponse. Le **caractère %** est le caractère **joker** qui remplace tous les autres : % peut être n'importe quelle chaîne de caractère.

```
1     SELECT *
2     FROM titre
3     WHERE primary_title LIKE '%alien %' or '%aliens%';
```

11° Fournir la requête permettant de récupérer les puplets de films dont le genre contient le genre "Sci-Fi".

12° Fournir la requête permettant de récupérer les puplets de films dont le genre contient le genre "Fantasy" du plus récent au plus ancien et dans l'ordre alphabétique.