DQL

MySQL est une base de données optimisée pour faire des requêtes DQL. Elles permettent d'extraire les données des tables. Nous verrons également que ceci est conditionné par une bonne définition des tables et des relations entre celles-ci.

Supposons que l'on ait une table pilots ayant les colonnes suivantes : name, numFlying, ...

Vous pouvez alors sélectionner l'ensemble des données de cette table à l'aide de la commande suivante, l'astérix désignera alors l'ensemble des colonnes :

```
SELECT * FROM pilots;
```

Définition projection

Une projection extrait qu'une ou plusieurs colonnes de la table :

```
-- on sélectionne uniquement la colonne name (projection)
SELECT `names` FROM pilots ;

-- on sélectionne deux colonnes
SELECT name, certificate FROM pilots ;
```

Restriction

Pour faire une restriction on utilise le mot réservé **WHERE**. Il permet de restreindre l'extraction des données d'une table :

```
-- on sélectionne uniquement le produit ayant l'identifiant unique id = 1
SELECT * FROM pilots WHERE certificate='ct-1' ;
```

MySQL lors d'un SELECT produira une pseudo-table, dans laquelle il extrait le résultat.

Structure de la commande SELECT

Il faut respecter l'ordre des clauses de la commande SELECT afin qu'elle puisse être correctement interprétée :

```
SELECT
    [DISTINCT] select_expression,...
    [FROM table_references
      [WHERE where_definition]
```

```
[ORDER BY {nom_de_colonne } [ASC | DESC]]
[LIMIT [offset,] lignes]
```

DISTINCT

DISTINCT permet de ne pas inclure dans l'extraction des données des doublons. Ci-dessous on extrait uniquement les nombres de jours distincts de la table pilots :

```
SELECT DISTINCT num_jobs FROM pilots;
```

Attention si vous tapez la commande suivante cela extrait les couples distincts num_jobs, name :

```
SELECT DISTINCT num_jobs, name FROM pilots;
```

clause WHERE

On peut utiliser les opérateurs suivants : =, <, >, <=, >=, <> (différent) Les opérateurs logiques **AND**, **OR**, **NOT** Concordances de chaînes : **LIKE IS NULL** teste si un champ n'a pas été affecté d'une valeur (NULL). **BETWEEN** teste l'appartenance à un intervalle de valeurs BETWEEN 1 AND 20 **IN** teste l'appartenance à un groupe de valeurs données : price IN (10, 8, 100)

clause ORDER

Ordre ASC ou DESC classe les résultats.

LIMIT start, length

start : offset, length le nombre d'éléments du tableau.

Exercices

- Sélectionnez tous les pilotes de la compagnie AUS
- Sélectionnez les noms des pilotes de la compagnie FRE1 ayant fait plus de 15 heures de vols.
- Sélectionnez les noms des pilotes de la compagnie FRE1 ayant fait plus de 20 heures de vols.
- Sélectionnez les noms des pilotes de la compagnie FRE1 ou AUST ayant fait plus de 20 de vols.

- Sélectionnez les noms des pilotes ayant fait entre 190 et 200 heures de vols.
- Sélectionnez les noms des pilotes qui sont nés après 1978.
- Sélectionnez les noms des pilotes qui sont nés avant 1978.
- Sélectionnez les noms des pilotes qui sont nés entre 1978 et 2000.
- Quels sont les pilotes qui ont un vol programmé après 2020-05-08 ?
- Sélectionnez tous les noms des pilotes qui sont dans les compagnies : AUS et FRE1.
- Sélectionnez tous les des pilotes dont le nom de compagnie contient un A.
- Sélectionnez tous les pilotes dont le nom de la compagnie commence par un F.
- Sélectionnez tous les pilotes dont le nom de la compagnie contient un I suivi d'un caractère.

Exercice ajout d'un bonus

Ajoutez une colonne **bonus** à la table pilots, puis ajoutez le bonus 1000 pour les certificats 'ct-1', 'ct-11', 'ct-12', pour le certificat ct-56 un bonus de 2000 et pour tous les autres 500.

Exercice meilleur bonus

Faites une requête permettant de sélectionner le pilote ayant eu le meilleur bonus. Vous pouvez utiliser la fonction max de MySQL.

Exercice heure de vols

Combien y-a-t-il d'heure de vols distincts dans la table pilotes ?

Exercice heure de vols moyen

Combien de pilotes sont en dessous de la moyenne d'heure de vols ?