TP5 : SQL requête SELECT dans une table,

1 objectif

Objectif :

- Utiliser l'opérateur « SELECT » qui permet de rechercher (sélectionner) des informations dans une base de données avec des conditions avec une expression logique
- Filtrer et trier le résultat de l'opérateur « SELECT »

1.1 synthèse sur des éléments abordés :

```
SELECT  [nom_table.nom_champs, nom_table.nom_champs2.....]
FROM
nom_table1, nom_table2 ….
WHERE  [conditions, jointure … ]
ORDER BY [nom_table.nom_champs] [DESC]
[LIMIT]  ;
```

• Par exemple :

```
SELECT * FROM catalogue WHERE ref="123abc";     (* : tous les champs)

SELECT Client.NomClient FROM Client WHERE Client.PrivilegeClient=true
ORDER by Client.NomClient;
```

[conditions]SQL

- Les opérateurs de comparaison `=` `<=` `<>` …
- Les opérateurs logiques `AND` `OR` `NOT` (expressions)
- Un champ compris entre 2 valeurs `nom_champ BETWEEN 1 AND 30`
- Une chaine de caractères comme …(comparaison) `nom_champ LIKE '123%'`
- Un champ est vide `nom_champ IS NULL` Utilisation de date "aaaa-mm-jj"
- Pour trier les champs
  - `ORDER BY ETU_Nom [ASC]` « CROISSANT »
  - `ORDER BY ETU_Nom DESC` « DECROISSANT »
- Sélectionner des résultats non redondants (distincts) `SELECT DISTINCT nom_champ …`
- Quand un champ appartient à une liste de valeurs (…) `WHERE nom_champ IN ('valeur1', 'valeur2', ...)`

1.2 remarque

lors de l'écriture des chaînes de texte :

- ' (apostrophe ou single quote, TOUCHE Alt Gr + 4) sert à entourer une valeur de format alphanumérique.
- ` (accent grave ou backtick, TOUCHE Alt Gr + 7) sert à entourer les noms des objets de la BDD (base, table, colonne), surtout lorsque ces noms peuvent être mal interprétés par MySQL : Noms en plusieurs mots ; Mots réservés de MySQL.
- " (guillemet ou double quote, TOUCHE Alt Gr + 3) est une alternative (propre à MySQL) ; elle est déconseillée, car seul ' est normalisé.

2 création de la table (structure) et insertion des enregistrements

2.1 création de la table etudiant

Créer une table pour gérer des étudiants , avec comme nom de la table **Etudiant** et comme colonnes (champs) dans la table :

Etudiant

Etudiant
idEtudiant
nom
adresse
ville
code_postal
telephone
date_naisssance
sexe
boursier
groupe_id
semestre_id
note_ue1
note_ue2

- `idEtudiant` : ce champ (attribut) est de type **entier** , c'est une clé primaire avec un compteur (champ qui utilise l'attribut «AUTO_INCREMENT»).
- `nom` : ce champ (attribut) est de type **chaîne de caractères variable** de taille 20 caractères maximum.
- `adresse` : ce champ (attribut) est de type **chaîne de caractères variable** de taille 40 caractères maximum.
- `ville` : ce champ (attribut)est de type **chaîne de caractères variable** de taille 20 caractères maximum.
- `code_postal` : ce champ (attribut) est de type **entier**
- `telephone` : ce champ (attribut) est de type **chaîne de caractères variable** de taille 13 caractère maximum
- `date_naissance` : ce champ (attribut) est de **date**
- `sexe` : ce champ (attribut) est de type **chaîne de caractères variable** de taille 1 caractère maximum.
- `boursier` : ce champ (attribut) est de type **chaîne de caractères variable** de taille 1 caractère maximum.
- `groupe_id` : ce champ (attribut) est de type **entier**
- `semestre_id` : ce champ (attribut) est de type **entier**
- `note_ue1` : ce champ (attribut) est de type **numeric** qui affiche un numérique avec 2 chiffres après la virgule, la valeur la plus grande est 99,99
- `note_ue2` : ce champ (attribut) est de type **numeric** qui affiche un numérique avec 2 chiffres après la virgule, la valeur la plus grande est 99,99

Utiliser le script ci-dessous pour insérer des enregistrements

```
-- version 2022

INSERT INTO Etudiant (idEtudiant, nom, adresse, ville, code_postal, telephone, date_naissance, sexe,
        boursier, groupe_id, semestre_id, note_ue1, note_ue2) VALUES
(1, 'BERNARD', '1 rue sous bois', 'Belfort', '90000', '0384545401', '2004-01-01', 'F', 'N', 1, 1, '11.50',
        '10.50'),
(19, 'CHAVEAUX', '5 rue du chasseur', 'Strasbourg', '68000', '0384545419', '2004-01-01', 'H', 'N', 2, 2,
        '10.50', '9.50'),
(20, 'PRETTOT', '8 rue vilapogo', 'Belfort', '90000', '0384545420', '2004-07-12', 'H', 'O', 2, 2, '12.50',
        '8.50'),
(7, 'RIOT', '67 rue pasteur', 'Montbeliard', '25200', '0384545407', '2004-06-06', 'H', 'N', 2, 1, '13.50',
        '12.50'),
(8, 'BOISSENIN', '1 rue sous bois', 'Belfort', '90000', '0384545408', '2004-08-09', 'H', 'O', 2, 1, '15.5',
        '16.50'),
(2, 'PEQUIGNOT', '2 rue de la liberation', 'Valdoie', '90300', '0384545402', '2003-06-06', 'H', 'O', 1, 1,
        '17.50', '9.50'),
(3, 'ZILLIOX', '7 rue du verger', 'Bavilliers', '90120', '0384545403', '2003-01-01', 'H', 'N', 1, 1,
        '11.50','9.25'),
(4, 'MONNIER', '3 rue du boulanger', 'TAILLECOURT', '25400', '0384545404', '2003-02-06', 'H', 'O', 1, 1,
        '9.50', '15.5'),
(5, 'BRISCHOUX', '5 rue du chasseur', 'Belfort', '90000', '0384545405', '2003-05-25', 'H', 'N', 2, 1, '10',
        '9.50'),
(6, 'DUVAL', '8 rue vilapogo', 'Bavilliers', '90120', '0384545406', '2003-03-11', 'H', 'O', 2, 1, '13.50',
        '8.50'),
   (10, 'FAIVRE', '7 rue des vergers  de rioz', 'Valdoie', '90300', '0384545410', '2003-01-05', 'H', 'O',
        3, 1, '8.50', '12.50'),
(17, 'DELANOE', '7 rue du verger', 'Valdoie', '90300', '0384545417', '2003-01-01', 'H', 'N', 1, 2, '12.50',
        '14.50'),
(18, 'BONVALOT', '3 rue du boulanger', 'Belfort', '90000', '0384545418', '2003-01-01', 'H', 'O', 1, 2,
        '10', '11.50'),
(21, 'COULON', '67 rue pasteur', 'Valdoie', '90300', '0384545421', '2003-01-01', 'H', 'N', 2, 2, '17',
        '6.5'),
(23, 'KENDE', '2 rue de la liberation', 'Bavilliers', '90120', '0384545423', '2003-05-31', 'H', 'N', 3, 2,
        '15', '7.5'),
(24, 'KLEIN', '7 rue du verger', 'Montbeliard', '25200', '0384545424', '2003-01-01', 'H', 'N', 3, 2, '13',
        '15'),
```

```
(25, 'VALZER', '3 rue du boulanger', 'Valdoie', '90300', '0384545425', '2003-01-06', 'H', 'O', 3, 2, '11',
        '10'),
(26, 'PY', '5 rue du chasseur', NULL, '90000', '0384545426', '2003-01-10', 'F', 'N', 3, 2, '12.20',
        '7.90'),
(27, 'VERNET', '8 rue vilapogo', NULL, '90120', '0384545427', '2003-02-02', 'H', 'O', 4, 3, '10.30',
        '11.85'),
(28, 'BAILLIT', '67 rue pasteur', NULL, '25200', '0384545428', '2003-01-01', 'H', 'N', 4, 3, '7.90',
        '15.90'),
(16, 'LUZET', '2 rue de la liberation', 'Belfort', '90000', '0384545416', '2002-01-01', 'H', 'O', 1, 2,
        '10.25', '9.25'),
  (22, 'VALOT', '1 rue sous bois', '', '90000', '0384545422', '2002-12-12', 'H', 'O', 2, 2, '10', '9.20'),
(29, 'DUPONT', '8 rue vilapogo', '', '90300', '0384545429', '2002-06-06', 'H', 'O', 4, 3, '13.20',
        '15.50'),
   (11, 'FAIVRE', '3 rue des vergers', 'Cernay', '68000', '0384545411', '2002-01-01', 'F', 'N', 3, 1,
        '5.6', '12.05'),
(12, 'DUCHENNE', '5 rue du chasseur', 'Belfort', '90000', '0384545412', '2002-01-01', 'F', 'O', 3, 1,
        '10.4', '11.3'),
(13, 'BOULANGER', '8 rue vilapogo', 'Belfort', '90000', '0384545413', '2002-01-01', 'F', 'N', 1, 2, '13',
        '9.20'),
   (9, 'FONTAINE-LEGIOT', '2 rue des vergers', 'Mulhouse', '68000', '0384545409', '2001-01-01', 'H', 'N',
        3, 1, '11.25', '12'),
(14, 'MOREAU', '67 rue pasteur', 'Belfort', '90000', '0384545414', '2001-06-01', 'H', 'O', 1, 2, '9',
        '12.50'),
(15, 'RIGOULOT', '1 rue sous bois', 'Valdoie', '90300', '0384545415', '2001-12-12', 'H', 'N', 1, 2, '15',
        '10.50');
```

2.2 Requête 1 (expression logique) :

Tester la requête suivante :

```sql
-- demo 1
SELECT   Etudiant.nom
         , Etudiant.adresse
         , Etudiant.semestre_id
FROM
  Etudiant
WHERE
  Etudiant.groupe_id =1 AND Etudiant.ville = 'Belfort'
;
```

- Résultat :

```
+-----------+------------------------+-------------+
| nom       | adresse                | semestre_id |
+-----------+------------------------+-------------+
| BERNARD   | 1 rue sous bois        |           1 |
| BOULANGER | 8 rue vilapogo         |           2 |
| MOREAU    | 67 rue pasteur         |           2 |
| LUZET     | 2 rue de la liberation |           2 |
| BONVALOT  | 3 rue du boulanger     |           2 |
+-----------+------------------------+-------------+
```

Sur le même principe :

- Écrire une requête pour sélectionner le nom, le groupe et le champ 'boursier' (si ils sont boursiers ou non) des étudiants du groupe 1, du semestre 1 quand ils sont boursiers.
- Résultat :

```
+-----------+-----------+----------+
| nom       | groupe_id | boursier |
+-----------+-----------+----------+
| PEQUIGNOT |         1 | O        |
| MONNIER   |         1 | O        |
+-----------+-----------+----------+
```

2.3 Requête 2 (expression logique et tri sur plusieurs champs) :

2.3.1 Requête 2-1 :

Tester la requête suivante : Afficher tous les étudiants triés par semestre, puis par groupe, puis par nom dans l'ordre décroissant. Utiliser l'instruction **ORDER BY champ1, champ2, champ3** ; remplacer « champ » par le nom du champ suivi de **ASC** ou **DESC** (par défaut le tri est fait de façon ascendante)

```
SELECT  semestre_id , groupe_id , nom
FROM Etudiant
ORDER BY semestre_id ASC , groupe_id , nom DESC;
```

Utiliser l'instruction ORDER BY « champ1 » après l'instruction WHERE « conditions ». Aidez vous des informations sur : https://www.w3schools.com/SQL/sql_orderby.asp

2.3.2 Requête 2-2 :

Tester la requête suivante :

```
-- -- demo 2
SELECT   nom, ville
FROM
  Etudiant
WHERE
  groupe_id =2
ORDER BY ville DESC, nom ASC    -- [ASC] option
;
```

- Résultat :

```
+-----------+-------------+
| nom       | ville       |
+-----------+-------------+
| COULON    | Valdoie     |
| CHAVEAUX  | Strasbourg  |
| RIOT      | Montbeliard |
| BOISSENIN | Belfort     |
| BRISCHOUX | Belfort     |
| PRETTOT   | Belfort     |
| DUVAL     | Bavilliers  |
| VALOT     |             |
+-----------+-------------+
```

Sur le même principe :

2.3.3 Requête 2-3 :

Écrire une requête pour sélectionner uniquement les étudiants (nom, groupe, ville, semestre) du groupe 1 et du semestre 1 ou du semestre 2 et de la ville de « Belfort ». Ordonner la liste résultat par semestre croissant puis par groupe croissant puis par nom de ville croissante (par ordre alphabétique) puis par nom d'étudiant décroissant.

Résultat :

```
+-----------+-----------+-------------+-------------+
| nom       | groupe_id | ville       | semestre_id |
+-----------+-----------+-------------+-------------+
| ZILLIOX   |         1 | Bavilliers  |           1 |
| BERNARD   |         1 | Belfort     |           1 |
| MONNIER   |         1 | TAILLECOURT |           1 |
| PEQUIGNOT |         1 | Valdoie     |           1 |
| MOREAU    |         1 | Belfort     |           2 |
| LUZET     |         1 | Belfort     |           2 |
```

```
| BOULANGER |          1 | Belfort     |           2 |
| BONVALOT  |          1 | Belfort     |           2 |
| PRETTOT   |          2 | Belfort     |           2 |
+-----------+------------+-------------+-------------+
```

2.3.4 Requête 2-4 :

Symboles de comparaison et opérateurs logiques

	norme ISO	MySQL	Oracle	PostgreSQL	SQLserver
!=	<>	<> !=	<> !=	<> !=	<> !=
&&	AND	AND &&	AND	AND	AND
||	OR	OR ||	OR	OR	OR
!	NOT	NOT !	NOT	NOT	NOT

Dans la requête ci-dessous, simplifier l'expression dans le WHERE:

```
SELECT nom, groupe_id, semestre_id, ville , code_postal
FROM Etudiant
WHERE
  ( (groupe_id <> 2 AND semestre_id <> 2 AND code_postal BETWEEN 90000 AND 90999)
  OR (semestre_id<> 2   AND code_postal BETWEEN 90000 AND 90999) )
  AND ville != 'Belfort' AND ville !='' AND ville IS NOT NULL
ORDER BY semestre_id, groupe_id, ville ,nom;
```

```
+-----------+-----------+-------------+------------+-------------+
| nom       | groupe_id | semestre_id | ville      | code_postal |
+-----------+-----------+-------------+------------+-------------+
| ZILLIOX   |         1 |           1 | Bavilliers |       90120 |
| PEQUIGNOT |         1 |           1 | Valdoie    |       90300 |
| DUVAL     |         2 |           1 | Bavilliers |       90120 |
| FAIVRE    |         3 |           1 | Valdoie    |       90300 |
+-----------+-----------+-------------+------------+-------------+
```

2.3.5 Requête 2-5 :

Écrire une requête pour sélectionner uniquement les étudiants (nom, groupe, ville, semestre, boursier) qui sont boursiers ou qui habitent la ville de belfort et qui sont tous du semestre 2 (trier les enregistrements par groupe puis ville puis nom de façon ascendante).

- Résultat :

```
+-----------+-----------+---------+-------------+----------+
| nom       | groupe_id | ville   | semestre_id | boursier |
+-----------+-----------+---------+-------------+----------+
| BONVALOT  |         1 | Belfort |           2 | O        |
| BOULANGER |         1 | Belfort |           2 | N        |
| LUZET     |         1 | Belfort |           2 | O        |
| MOREAU    |         1 | Belfort |           2 | O        |
| VALOT     |         2 |         |           2 | O        |
| PRETTOT   |         2 | Belfort |           2 | O        |
| VALZER    |         3 | Valdoie |           2 | O        |
+-----------+-----------+---------+-------------+----------+
```

2.4 Requête 3 (utilisation de DISTINCT)

Requête 3-1 : Tester la requête suivante avec et sans le mot clé `distinct` :

```
SELECT DISTINCT code_postal, ville
FROM Etudiant;
```

Requête 3-2 : Écrire la requête pour afficher les villes distinctes de tous les étudiants dont le code postal commence par 90 et dont le nom de ville n'est pas NULL et n'est pas une chaîne vide. Utiliser le mot clé `DISTINCT` .

- Aidez vous des informations sur : https://www.w3schools.com/sql/sql_distinct.asp
- Résultat :

```
+------------+
| ville      |
+------------+
| Belfort    |
| Valdoie    |
| Bavilliers |
+------------+
```

2.5 Requête 4 (IN : un champ appartient à une liste de valeurs)

Écrire une requête pour sélectionner les étudiants et les villes dont la ville appartient à cette liste ('strasbourg','mulhouse','cernay'). Utiliser l'instruction `IN`

- Aidez vous des informations sur : https://www.w3schools.com/SQL/sql_in.asp
- Résultat :

```
+-----------------+------------+
| nom             | ville      |
+-----------------+------------+
| FONTAINE-LEGIOT | Mulhouse   |
| FAIVRE          | Cernay     |
| CHAVEAUX        | Strasbourg |
+-----------------+------------+
```

2.6 Requête 5 (utilisation de NULL)

Écrire la requête pour sélectionner le nom des étudiants dont le champ de la ville est vide (Utiliser l'instruction `IS NULL`) et dont le sexe est « H ».

- Aidez vous des informations sur : https://www.w3schools.com/SQL/sql_null_values.asp
  Remarque : NULL est un cas particulier, on n'écrit jamais ~~=NULL~~ ou ~~!=NULL~~

  Il faut utiliser `IS NULL` ou `IS NOT NULL` car c'est une logique 3 états (VRAI, FAUX, non renseigné (NULL))

- Résultat :

```
+---------+-------+
| nom     | ville |
+---------+-------+
| VERNET  | NULL  |
| BAILLIT | NULL  |
+---------+-------+
```

Faire la même requête mais en rajoutant les étudiants dont le champ de la ville est une chaîne de caractères vide

- Résultat :

```
+---------+-------+
| nom     | ville |
+---------+-------+
| VALOT   |       |
| VERNET  | NULL  |
| BAILLIT | NULL  |
| DUPONT  |       |
+---------+-------+
```

2.7 Requête 6 (utilisation de LIKE)

Requête pour afficher le Nom des étudiants et leur ville. Le nom des étudiants commence par « B » ou « D » et ils habitent une ville qui possède la lettre « o » (triés par nom).

```
SELECT nom,  ville
FROM Etudiant
WHERE  (nom LIKE 'B%' OR nom LIKE 'D%' )
AND  ville LIKE  '%o%'
ORDER BY nom;
```

- Aidez vous des informations sur : https://www.w3schools.com/sql/sql_like.asp
- Résultat :

```
+-----------+---------+
| nom       | ville   |
+-----------+---------+
| BERNARD   | Belfort |
| BOISSENIN | Belfort |
| BONVALOT  | Belfort |
| BOULANGER | Belfort |
| BRISCHOUX | Belfort |
| DELANOE   | Valdoie |
| DUCHENNE  | Belfort |
+-----------+---------+
```

Écrire une requête pour afficher le nom la ville et l'adresse des étudiants dont l'adresse est composée du mot « verger » et dont la ville est composée des lettres « er » ou des lettres « a » suivies de « l ».

- Résultat :

```
+---------+------------+----------------------------+
| nom     | ville      | adresse                    |
+---------+------------+----------------------------+
| DELANOE | Valdoie    | 7 rue du verger            |
| FAIVRE  | Cernay     | 3 rue des vergers          |
| FAIVRE  | Valdoie    | 7 rue des vergers  de rioz |
| ZILLIOX | Bavilliers | 7 rue du verger            |
+---------+------------+----------------------------+
```

2.8 Requête 7 (requête avec des dates, fonctions `now()`, `day()`, `month()`, `year()`)

Écrire une requête pour afficher le nom et la date de naissance des étudiants dont la date de naissance est comprise entre le `"2002-1-1"` And `"2002-12-31"` , qui sont des hommes et qui habitent Belfort ou Montbéliard (utiliser le code postal 90000 (Belfort) et 25200(Montbéliard)). Proposer une deuxième solution avec comme condition (dans le « WHERE ») : l'année est égale à `"2002"` . (trier l'affichage par date de naissance)

- Résultat :

```
+-------+----------------+
| nom   | date_naissance |
+-------+----------------+
| LUZET | 2002-01-01     |
| VALOT | 2002-12-12     |
+-------+----------------+
* selon les mises à jour, votre résultat peut être différent
```

Aidez vous des informations sur https://www.w3schools.com/sql/sql_ref_mysql.asp. Rechercher la fonction « YEAR ».

Pour les plus rapides

2.9 Requête 8 (utilisation de LIMIT)

La requête ci-dessous retourne les 10 enregistrements de la position 6 à 15

```
SELECT nom
FROM Etudiant
```

```
ORDER BY nom
LIMIT 5,10 ;
```

(https://dev.mysql.com/doc/refman/5.7/en/select.html) Tester et utiliser la lien ci-dessous pour https://www.w3schools.com/SQL/sql_top.asp

Dans une page Web, on désire sélectionner les 6 premiers étudiants triés par Groupe (descendant : plus grand au plus petit) puis par Nom (pour ordre alphabétique). Écrire la requête, utiliser l'instruction **`LIMIT`** pour limiter le nombre d'enregistrements sélectionnés.

- Résultat :

```
+----------+-----------+
| nom      | groupe_id |
+----------+-----------+
| BAILLIT  |         4 |
| DUPONT   |         4 |
| VERNET   |         4 |
| DUCHENNE |         3 |
| FAIVRE   |         3 |
| FAIVRE   |         3 |
+----------+-----------+
```

sélectionner les 4 suivants

- Résultat :

```
+-----------------+-----------+
| nom             | groupe_id |
+-----------------+-----------+
| FONTAINE-LEGIOT |         3 |
| KENDE           |         3 |
| KLEIN           |         3 |
| PY              |         3 |
+-----------------+-----------+
```

Le mot clé **`LIMIT`** n'est pas standard

2.10 Requête 9 (expressions régulières)

Aidez vous des informations la documentation en annexe (suivre le lien) pour les expressions régulières ou sur : https://dev.mysql.com/doc/refman/5.7/en/regexp.html et https://www.w3schools.com/SQL/sql_like.asp Avec la documentation ci dessus (cas de LIKE puis de REGEXP):

- Rechercher une expression régulière qui commence par un chiffre, suivi du chiffre "5", et de 3 chiffres (format D5DDD, D appartient à [0-9]).
- Rechercher une expression régulière qui commence par un caractère, suivi du caractère "a" ou "e" puis de n'importe quel caractère.

- Écrire une requête pour sélectionner le nom des étudiants dont le nom a comme 2eme caractère "a" ou "e" et dont le code postal a comme 2eme chiffre "5" en utilisant LIKE (champ LIKE « expression en remplaçant _ pour un caractère et % pour une chaîne de caractères»)
- Écrire une requête pour sélectionner le nom des étudiants dont le nom a comme 2eme caractère "a" ou "e" et dont le code postal a comme 2eme chiffre "5" en utilisant REGEXP (champ REGEXP « expression régulière »), (LIKER idem).

Remarque : LIKE est plus rapide que REGEXP, voir article http://billauer.co.il/blog/2020/12/mysql-index-pattern-matching-performance/

- Résultat :

```
+---------+-------------+
| nom     | code_postal |
+---------+-------------+
| BAILLIT |       25200 |
+---------+-------------+
```

Pour comprendre les expressions régulières, voici 1 site web intéressant : regex101

2.11 Requête 10

Écrire une requête pour sélectionner les étudiants qui ont plus de 20 ans après le 30 septembre de la fin de l'année universitaire (mois de naissance inférieur à 10 si c'est l'année de leur 20 ans). Ces étudiants devaient payer une sécurité sociale si ils n'étaient pas boursiers . Ordonner les étudiants du plus jeune au plus vieux.

commencer par tester :

```
SELECT day(now());
SELECT month(now());
SELECT year(now());
```

idée : plus de 20 ans entre aujourd'hui et leur date de naissance ou 20 ans et …..

- Résultat :

```
+-----------------+----------------+
| nom             | date_naissance |
+-----------------+----------------+
| FAIVRE          | 2002-01-01     |
| BOULANGER       | 2002-01-01     |
| RIGOULOT        | 2001-12-12     |
| FONTAINE-LEGIOT | 2001-01-01     |
+-----------------+----------------+

* selon les mises à jour, votre résultat peut être différent
```

Si les dates ne sont pas correctes, utiliser l'instruction SQL suivante pour les mettre à jour **`UPDATE Etudiant SET date_naissance=DATE_ADD(date_naissance,INTERVAL 1 YEAR);`**