La commande SELECT

# Contexte :

Avoir des téraoctets de données peut sembler inutile si l’on ne sait pas comment les utiliser. Cependant, avec les bons outils et les bonnes compétences, ces données peuvent être exploitées pour fournir des informations précieuses qui peuvent améliorer les performances commerciales et aider à la prise de décisions.

Le rôle des SGBD comme MySQL est d’organiser et de stocker ces données de manière efficace et fiable, tout en fournissant un accès rapide et précis aux données lorsque cela est nécessaire. Les SGBD offrent des fonctionnalités avancées telles que la capacité à écrire des requêtes complexes, à trier et à filtrer les données, à effectuer des analyses, et à générer des rapports personnalisés.

En utilisant les fonctionnalités avancées de MySQL, vous pouvez extraire des informations précieuses de vos données et les utiliser pour améliorer vos performances commerciales. Par exemple, vous pouvez :

Analyser les tendances de vente pour déterminer les produits les plus populaires

Identifier les goulots d’étranglement dans les processus commerciaux et améliorer l’efficacité

Détecter les comportements des clients pour personnaliser les offres et améliorer l’expérience client

Suivre les performances de l’entreprise à l’aide de rapports personnalisés

Il s’agit de la commande SELECT, elle est capable d’extraire des données spécifiques d’une ou plusieurs tables d’une base de données. Elle offre une grande flexibilité pour sélectionner les colonnes à afficher, filtrer les données, trier les résultats et traiter les données sélectionnées à l’aide de fonctions d’agrégation.

En fait dès que vous devez afficher quelque chose s vous devez passer par cet ordre SQL !

## Syntaxe générale de SELECT [MySQL] :

**SELECT** colonne1, colonne2,…

**FRO**M table

**WHERE** condition

**GROUP** **BY** colonne

**HAVING** condition

**ORDER** **BY** colonne ASC/DESC ;

### Explication de chaque clause :

En SQL, les clauses sont des instructions qui sont utilisées pour spécifier des conditions ou des contraintes pour les requêtes de base de données. Les clauses peuvent être utilisées pour sélectionner des données spécifiques, filtrer des données, trier des résultats, regrouper des données, et biens plus encore.

SELECT : spécifie les colonnes que vous souhaitez afficher dans le résultat de la requête. Vous pouvez sélectionner une ou plusieurs colonnes en les séparant par une virgule. Vous pouvez également utiliser des fonctions d’agrégation comme SUM, AVG, COUNT, MAX, MIN, etc. pour calculer des valeurs à partir des données dans les colonnes.

FROM : spécifie la table à partir de laquelle vous souhaitez récupérer les données.

WHERE : spécifie la condition qui doit être remplie pour que les lignes soient incluses dans le résultat de la requête. Vous pouvez utiliser des opérateurs de comparaison (comme =, >, <, <>), des opérateurs logiques (comme AND, OR, NOT) et des fonctions comme LIKE pour spécifier des conditions.

GROUP BY : spécifie les colonnes à utiliser pour regrouper les données. Vous pouvez regrouper les données en fonction des valeurs dans une ou plusieurs colonnes.

HAVING : spécifie la condition qui doit être remplie pour que les groupes soient inclus dans le résultat de la requête. Vous pouvez utiliser des opérateurs de comparaison et des fonctions d’agrégation pour spécifier des conditions.

ORDER BY : spécifie la colonne à utiliser pour trier les données. Vous pouvez trier les données par ordre croissant (ASC) ou décroissant (DESC).

## Illustration

Par exemple, pour récupérer toutes les données d’une table "clients" dans une base de données, vous pouvez utiliser la commande SELECT de la manière suivante :

SELECT \* FROM clients ;

Cette commande sélectionne toutes les colonnes de la table "clients" et récupère toutes les données de la table.

Vous pouvez également utiliser la commande SELECT pour filtrer les données en fonction de certaines conditions. Par exemple, pour récupérer les données des clients dont le pays est "France", vous pouvez utiliser la commande SELECT de la manière suivante :

SELECT \* FROM clients WHERE pays = 'France’ ;

Cette commande sélectionne toutes les colonnes de la table "clients" pour lesquelles le pays est égal à "France".

# Les opérateurs de bases

Dans les exemples que ci-dessous, la commande SELECT est utilisée pour afficher les résultats des opérations ou des comparaisons effectuées avec les opérateurs.

Par exemple, si vous souhaitez additionner deux valeurs, vous pouvez utiliser l’opérateur d’addition (+) et la commande SELECT pour afficher le résultat de l’addition. De même, si vous souhaitez récupérer des enregistrements dans une table en fonction de certains critères, vous pouvez utiliser des opérateurs de comparaison et la commande SELECT pour afficher les résultats correspondants.

## Opérateurs arithmétiques :

* Addition : SELECT 2 + 3 ; **(Résultat : 5)**
* Soustraction : SELECT 5 - 2 ; **(Résultat : 3)**
* Multiplication : SELECT 4 \* 2 ; **(Résultat : 8)**
* Division : SELECT 10/2 ; **(Résultat : 5)**

## Opérateurs de comparaison :

* Égalité : SELECT 2 = 2 ; **(Résultat : 1)**
* Différence : SELECT 3 ! = 2 ; **(Résultat : 1)**
* Infériorité : SELECT 5 < 10 ; **(Résultat : 1)**
* Supériorité : SELECT 10 > 5 ; **(Résultat : 1)**
* Infériorité ou égalité : SELECT 2 <= 2 ; **(Résultat : 1)**
* Supériorité ou égalité : SELECT 5 >= 5 ; **(Résultat : 1)**

## Opérateurs logiques :

* Opérateur AND : SELECT 1 AND 1 ; **(Résultat : 1)**
* Opérateur OR : SELECT 1 OR 0 ; **(Résultat : 1)**
* Opérateur NOT : SELECT NOT 0 ; **(Résultat : 1)**

## Opérateurs d’affectation :

* Affectation : SET x = 5 ;
* Affectation alternative : SET @x : = 5 ;

## Opérateurs de concaténation :

* Concaténation de chaînes de caractères :
* SELECT CONCAT ('Hello', ' World') ; **(Résultat : "Hello World")**

## Opérateurs de sélection :

* Opérateur IN :
  + SELECT \* FROM ma\_table WHERE mon\_champ IN (1, 2, 3);
* Opérateur BETWEEN :
  + SELECT \* FROM ma\_table WHERE mon\_champ BETWEEN 10 AND 20;

# Les wildcards

Les wildcards (ou caractères génériques) sont des caractères spéciaux qui permettent de rechercher des chaînes de caractères qui correspondent à un modèle spécifique. Les wildcards les plus couramment utilisés sont les suivants :

* % (pourcentage) : ce wildcard représente n’importe quelle chaîne de caractères de longueur quelconque (y compris une chaîne vide). Par exemple, la requête SELECT \* FROM ma\_table WHERE mon\_champ LIKE 'John %' ; renvoie tous les enregistrements de la table ma\_table où le champ mon\_champ débute par "John".
* \_ (underscore) : ce wildcard représente un seul caractère quelconque. Par exemple, la requête SELECT \* FROM ma\_table WHERE mon\_champ LIKE 'T\_m’ ; renvoie tous les enregistrements de la table ma\_table où le champ mon\_champ commence par un "T" et se termine par un "m" (avec un seul caractère entre les deux).
* [ ] (crochets) : ce wildcard symbolise un seul caractère parmi ceux spécifiés entre les crochets. Par exemple, la requête SELECT \* FROM ma\_table WHERE mon\_champ LIKE 'J [ae] n’ ; renvoie tous les inscriptions de la table ma\_table où le champ mon\_champ débute par "J", se finalise par "n" et contient soit un "a" soit un "e" entre les deux.
* [^] (crochets inversés) : ce wildcard représente un seul caractère qui ne figure pas parmi ceux spécifiés entre les crochets. Par exemple, la requête SELECT \* FROM ma\_table WHERE mon\_champ LIKE 'J [^ae] n’ ; réfère tous les enregistrements de la table ma\_table où le champ mon\_champ commence par "J", se termine par "n" et ne comporte ni "a" ni "e" entre les deux.

**Les wildcards sont généralement utilisés avec la commande SELECT et la clause WHERE pour effectuer des recherches de données plus complexes et précises.**

|  |  |
| --- | --- |
| Compétences | Pratiquer les requêtes SQL |
| Objectifs | Résoudre et répondre aux questions qui suivent |
| Durée estimée | 45 à 60 minutes |
| Répertoire de travail |  |
| Fichiers sources |  |
| À produire |  |
| Exercices | [5.1 Sélection sur une table 7](#_Toc127978594)  [5.2 Sélection sur une table 2 7](#_Toc127978595)  [5.3 Sélection avec where 7](#_Toc127978596)  [5.4 Sélection sur une colonne 8](#_Toc127978597)  [5.5 Sélection sur une table avec projection qui élimine les doublons à l’affichage 8](#_Toc127978598)  [5.6 Sélection sur une table avec double restrictions et projection 8](#_Toc127978599)  [5.7 Sélection sur une table avec restriction de type wildcard 9](#_Toc127978600)  [5.8 Intersection d'une relation avec un ensemble énuméré 9](#_Toc127978601)  [5.9 Sélection avec projection et restriction 9](#_Toc127978602)  [5.10 Sélection avec projection et restriction 10](#_Toc127978603)  [5.11 Sélection avec fonctions 10](#_Toc127978604)  [6.1 Trouver le nom des employés ayant travaillé sur tous les projets : 10](#_Toc127978605)  [6.2 Trouver les clients ayant commandé tous les produits : 11](#_Toc127978606)  [6.3 Trouver les produits dont la quantité commandée est supérieure à la moyenne de toutes les quantités commandées : 11](#_Toc127978607) |

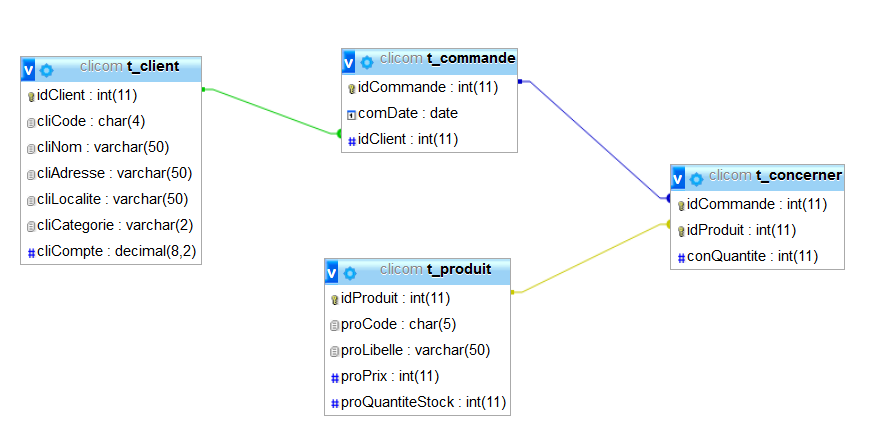
# Exercice pratique

Pour répondre aux questions suivantes, veuillez utiliser le modèle de données fourni et exprimer vos solutions par des requêtes SQL. Vous pouvez créer la base de données en exécutant le fichier "i-164-dbClicom.sql" à l’aide de MySQL.

**Ce fichier contient toutes les instructions indispensables pour construire la base de données, ainsi que quelques enregistrements de données pour vous aider à démontrer vos réponses.**

N’hésitez pas à ajouter d’autres données pour affiner vos vérifications. Les questions posées nécessitent une bonne connaissance de la syntaxe SELECT et des quelques fonctions, alors assurez-vous de bien comprendre les concepts avant de répondre.

## MLD



## Prérequis

Afin de réaliser les exercices ci-dessous, il est indispensable de maîtriser les bases du langage SQL, notamment les différentes commandes, clauses et opérateurs.

Si vous ne vous sentez pas à l’aise avec ces notions, nous vous recommandons vivement de revoir attentivement les chapitres antérieurs de ce document, qui traitent de ces concepts de manière détaillée.

Cela vous permettra de progresser plus facilement dans votre apprentissage et de mieux comprendre les exercices proposés.

# Questions

## Sélection sur une table

|  |  |
| --- | --- |
|  | Affichez les caractéristiques de tous les produits ! |
| SQL | SELECT \* FROM produits; |
| Rép |  |

## Sélection sur une table 2

|  |  |
| --- | --- |
|  | Sélectionnez les noms de tous les clients |
| SQL | SELECT cliNom FROM t\_client; |
| Rép |  |

## Sélection avec where

|  |  |
| --- | --- |
|  | Sélectionnez les clients dont le solde de compte est négatif |
| SQL | SELECT \* FROM t\_client WHERE cliCompte < 0; |
| Rép |  |

## Sélection sur une colonne

|  |  |
| --- | --- |
|  | Sélectionnez les codes de tous les clients dont la catégorie est B1 |
| SQL | SELECT cliCode FROM t\_client WHERE cliCategorie = 'B1'; |
| Rép |  |

## Sélection sur une table avec projection qui élimine les doublons à l’affichage

|  |  |
| --- | --- |
|  | Affichez la liste des localités dans lesquelles il existe au moins un client ! |
| SQL | SELECT DISTINCT cliLocalite FROM t\_client; |
| Rép |  |

## Sélection sur une table avec doubles restrictions et projection

|  |  |
| --- | --- |
|  | Affichez le code, le nom et la localité des clients de catégorie C1 n’habitant pas à Toulouse ! |
| SQL | SELECT cliCode, cliNom, cliLocalite  FROM t\_client  WHERE cliCategorie = 'C1'  AND cliLocalite != 'Toulouse'; |
| Rép |  |

## Sélection sur une table avec restriction de type wildcard

|  |  |
| --- | --- |
|  | Affichez les caractéristiques des produits en acier ! |
| SQL | SELECT \* FROM produits WHERE materiau = 'acier'; |
| Rép |  |

## Intersection d’une relation avec un ensemble énuméré

|  |  |
| --- | --- |
|  | Donnez le code, le nom et le compte des clients de Poitiers et de Bruxelles dont le compte est positif ! |
| SQL | SELECT cliCode, cliNom, cliCompte  FROM t\_client  WHERE cliLocalite IN ('Poitiers', 'Bruxelles')  AND cliCompte > 0; |
| Rép |  |

## Sélection avec projection et restriction

|  |  |
| --- | --- |
|  | Quelles catégories de clients trouve-t-on à Toulouse ? |
| SQL | SELECT DISTINCT cliCategorie  FROM t\_client  WHERE cliLocalite = 'Toulouse'; |
| Rép |  |

## Sélection avec projection et restriction

|  |  |
| --- | --- |
|  | Affichez le code, le nom et la localité des clients dont le nom précède alphabétiquement la localité où ils résident ! |
| SQL | SELECT cliCode, cliNom, cliLocalite  FROM t\_client  WHERE cliNom < cliLocalite; |
| Rép |  |

## Sélection avec fonctions

|  |  |
| --- | --- |
|  | Affichez le total, le minimum, la moyenne et le maximum des comptes des clients ! |
| SQL | SELECT  SUM(cliCompte) AS Total\_Compte,  MIN(cliCompte) AS Min\_Compte,  AVG(cliCompte) AS Moyenne\_Compte,  MAX(cliCompte) AS Max\_Compte  FROM t\_client; |
| Rép |  |

# Questions défis

## Trouver le nom des employés ayant travaillé sur tous les projets :

|  |
| --- |
|  |

## Trouver les clients ayant commandé tous les produits :

|  |
| --- |
|  |

## Trouver les produits dont la quantité commandée est supérieure à la moyenne de toutes les quantités commandées :

|  |
| --- |
|  |

### Remarques

Dans un contexte réel, il est probable que l’utilisation de jointures serait plus appropriée pour résoudre ces problèmes.