les commandes SQL les plus utilisées

Commande	Explication et Exemple
SELECT	Permet de sélectionner des données depuis une table.
	Exemple : `SELECT nom, age FROM utilisateurs;` (Sélectionne les colonnes `nom` et `age` de la table `utilisateurs`).
INSERT	Permet d'insérer de nouvelles lignes dans une table.
	Exemple : `INSERT INTO utilisateurs (nom, age) VALUES ('Alice', 25);` (Ajoute une nouvelle ligne avec `nom = 'Alice'` et `age = 25`).
UPDATE	Permet de modifier des données existantes dans une table.
	Exemple : `UPDATE utilisateurs SET age = 26 WHERE nom = 'Alice';` (Met à jour l'âge de l'utilisateur `Alice` à 26).
DELETE	Permet de supprimer des lignes d'une table.
	Exemple : `DELETE FROM utilisateurs WHERE nom = 'Alice';` (Supprime l'utilisateur `Alice` de la table).
CREATE TABLE	Permet de créer une nouvelle table.
	Exemple : `CREATE TABLE utilisateurs (id INT PRIMARY KEY, nom VARCHAR(50) NOT NULL, age INT);` (Crée une table `utilisateurs` avec une clé primaire et une colonne `nom` qui ne peut pas être nulle).
ALTER TABLE	Permet de modifier la structure d'une table (ajouter, supprimer ou modifier des colonnes).
	Exemple : `ALTER TABLE utilisateurs ADD email VARCHAR(100);` (Ajoute une colonne `email` à la table `utilisateurs`).
DROP TABLE	Permet de supprimer une table.
	Exemple : `DROP TABLE utilisateurs;` (Supprime la table `utilisateurs`).
PRIMARY KEY	Permet de définir une colonne comme clé primaire (identifiant unique).
	Exemple : `CREATE TABLE utilisateurs (id INT PRIMARY KEY, nom VARCHAR(50));` (Crée une table avec `id` comme clé primaire).
FOREIGN KEY	Permet de définir une clé étrangère pour créer une relation entre deux tables.
	Exemple : `CREATE TABLE commandes (id INT PRIMARY KEY, utilisateur_id INT, FOREIGN KEY (utilisateur_id) REFERENCES utilisateurs(id));` (Crée une table `commandes` avec une clé étrangère liée à la table `utilisateurs`).
NOT NULL	Permet de s'assurer qu'une colonne ne peut pas contenir de valeurs nulles.
	Exemple : `CREATE TABLE utilisateurs (id INT PRIMARY KEY, nom VARCHAR(50) NOT NULL);` (Crée une table où la colonne `nom` est obligatoire).
JOIN	Permet de combiner des lignes de deux ou plusieurs tables en fonction d'une condition.
	Exemple : `SELECT utilisateurs.nom, commandes.montant FROM utilisateurs JOIN commandes ON utilisateurs.id = commandes.utilisateur_id;` (Combine les tables `utilisateurs` et `commandes`).

LEFT JOIN	Retourne toutes les lignes de la table de gauche, même si aucune correspondance n'existe dans la table de droite.
	Exemple : `SELECT utilisateurs.nom, commandes.montant FROM utilisateurs LEFT JOIN commandes ON utilisateurs.id = commandes.utilisateur_id;` (Retourne tous les utilisateurs, même ceux sans commandes).
RIGHT JOIN	Retourne toutes les lignes de la table de droite, même si aucune correspondance n'existe dans la table de gauche.
	Exemple : `SELECT utilisateurs.nom, commandes.montant FROM utilisateurs RIGHT JOIN commandes ON utilisateurs.id = commandes.utilisateur_id;` (Retourne toutes les commandes, même celles sans utilisateur associé).
WHERE	Permet de filtrer les résultats en fonction d'une condition.
	Exemple : `SELECT * FROM utilisateurs WHERE age > 30;` (Sélectionne les utilisateurs de plus de 30 ans).
GROUP BY	Permet de regrouper les résultats par une ou plusieurs colonnes.
	Exemple : `SELECT age, COUNT(*) FROM utilisateurs GROUP BY age;` (Compte le nombre d'utilisateurs par âge).
ORDER BY	Permet de trier les résultats par une ou plusieurs colonnes.
	Exemple : `SELECT * FROM utilisateurs ORDER BY age DESC;` (Trie les utilisateurs par âge en ordre décroissant).
LIMIT	Permet de limiter le nombre de résultats retournés.
	Exemple : `SELECT * FROM utilisateurs LIMIT 10;` (Retourne les 10 premiers utilisateurs).
COUNT	Permet de compter le nombre de lignes correspondant à une condition.
	Exemple : `SELECT COUNT(*) FROM utilisateurs;` (Compte le nombre total d'utilisateurs).
SUM	Permet de calculer la somme des valeurs d'une colonne.
	Exemple : `SELECT SUM(montant) FROM commandes;` (Calcule la somme totale des montants dans la table `commandes`).
AVG	Permet de calculer la moyenne des valeurs d'une colonne.
	Exemple : `SELECT AVG(age) FROM utilisateurs;` (Calcule l'âge moyen des utilisateurs).
DISTINCT	Permet de sélectionner des valeurs distinctes (sans doublons).
	Exemple : `SELECT DISTINCT age FROM utilisateurs;` (Sélectionne les âges uniques dans la table `utilisateurs`).
UNION	Permet de combiner les résultats de deux requêtes en un seul ensemble de résultats.
	Exemple : `SELECT nom FROM utilisateurs UNION SELECT nom FROM clients;` (Combine les noms des tables `utilisateurs` et `clients` sans doublons).
UNION ALL	Permet de combiner les résultats de deux requêtes en un seul ensemble de résultats, y compris les doublons.
	Exemple : `SELECT nom FROM utilisateurs UNION ALL SELECT nom FROM clients;` (Combine les noms des tables `utilisateurs` et `clients` avec doublons).
HAVING	Permet de filtrer les résultats après un `GROUP BY`.
	Exemple : `SELECT age, COUNT(*) FROM utilisateurs GROUP BY age HAVING COUNT(*) > 5;` (Sélectionne les âges avec plus de 5 utilisateurs).