DOCUMENTATION DU PROJET SQL

Projet de fin du cours Analyse des données avec SQL

Auteur : DOH Yawo Marie Victor

Formation : D-CLIC 3 – Parcours Data

Date : Janvier 2026

INTRODUCTION

Ce document a pour objectif d'expliquer en termes simples le projet SQL réalisé, même si vous n'avez aucune connaissance en bases de données. Vous comprendrez ce qui a été fait et pourquoi.

Qu'est-ce qu'une base de données ?

Imaginez une bibliothèque géante où toutes les informations d'une entreprise sont stockées de manière organisée. Une base de données, c'est exactement ça : un système qui permet de stocker, organiser et retrouver facilement des informations.

QU'EST-CE QUE SQL ?

SQL signifie "Structured Query Language" (Langage de Requête Structuré).

C'est un langage informatique qui permet de :

- **Créer** des espaces de stockage (tables)
- **Ajouter** des informations (insertion de données)
- **Rechercher** des informations spécifiques (requêtes)
- **Modifier** ou **Supprimer** des informations
- **Analyser** des données (calculs, statistiques)

Analogie simple : Si une base de données est une bibliothèque, SQL est le système de catalogage qui permet de ranger les livres, de les retrouver et de savoir combien il y en a.

STRUCTURE DU PROJET

Le projet est organisé en **15 sections principales** qui couvrent tous les aspects fondamentaux de SQL :

Vue d'ensemble des sections :

1. **Sections 0-7** : Création des tables et insertion des données

2. **Sections 8-11** : Recherche et filtrage des données
3. **Sections 12-13** : Calculs et statistiques
4. **Sections 14-15** : Relations entre tables et requêtes avancées

EXPLICATION DÉTAILLÉE DES SECTIONS

SECTION 0 : Création de la Table DEPARTEMENTS

Ce qui a été fait : Création d'une table pour stocker les informations sur les départements de l'entreprise.

Pourquoi cette section existe-t-elle ? Cette section n'était pas dans le projet initial, mais elle était nécessaire. Les autres tables font référence aux départements, donc il fallait d'abord créer cette table.

Données insérées :

- Direction (Paris)
- Ressources Humaines (Paris)
- IT (Lyon)
- Marketing (Paris)

Analogie : Avant de ranger des livres par catégorie, il faut d'abord créer les étagères et les étiquettes des catégories.

SECTION 1 : Création de la Table EMPLOYES

Ce qui a été fait : Création d'une table pour stocker les informations des employés de l'entreprise.

Informations stockées pour chaque employé :

- Identifiant unique (ID)
- Prénom et Nom
- Email
- Salaire
- Date d'embauche
- Département d'appartenance
- Manager (responsable hiérarchique)

Données insérées : 5 employés

- Marie Dupont (Direction, 85 000€, pas de manager - c'est la directrice)
- Jean Martin (RH, 72 000€, manager : Marie)
- Sophie Bernard (IT, 68 000€, manager : Marie)
- Pierre Dubois (RH, 55 000€, manager : Jean)
- Claire Moreau (RH, 52 000€, manager : Jean)

Détail technique important :

- `IDENTITY(1,1)` : signifie que l'ID est généré automatiquement (1, 2, 3, 4...) sans qu'on ait besoin de le saisir
- `FOREIGN KEY` : crée un lien avec d'autres tables (département, manager)

SECTION 2 : Création de la Table CLIENTS

Ce qui a été fait : Création d'une table pour gérer la liste des clients de l'entreprise.

Informations stockées :

- Nom du client
- Email
- Ville et Pays (France par défaut)
- Date d'inscription
- Segment (Premium ou Standard)

Données insérées : 4 clients de différents pays

Pourquoi un segment ? Pour catégoriser les clients selon leur importance commerciale.

SECTION 3 : Tables COMMANDES et LIGNES DE COMMANDE

Ce qui a été fait : Création de deux tables liées pour gérer les commandes passées par les clients.

Table COMMANDES : informations générales sur chaque commande

- Quel client a commandé
- Quand
- Montant total
- Statut (En cours, Livré, etc.)

Table LIGNES_COMMANDE : détails de ce qui a été commandé

- Chaque ligne représente un article dans une commande

Analogie : Quand vous commandez sur Amazon, vous avez un numéro de commande (table commandes) et une liste d'articles (table lignes_commande).

SECTION 4 : Création de la Table PROJETS

Ce qui a été fait : Création d'une table pour gérer les projets de l'entreprise.

Informations stockées :

- Nom du projet

- Budget alloué
- Date de début et de fin
- Statut (En cours, Planifié, Terminé)

Projets insérés :

- Refonte Site Web (150 000€)
- App Mobile V2 (200 000€)
- Migration Cloud (300 000€)
- CRM Integration (80 000€)

SECTION 5 : Table AFFECTATIONS

Ce qui a été fait : Création d'une table pour associer les employés aux projets.

Pourquoi cette table ? Un employé peut travailler sur plusieurs projets, et un projet peut avoir plusieurs employés. Cette table fait le lien entre les deux.

Informations stockées :

- Quel employé
- Sur quel projet
- Quel rôle (développeur, chef de projet, etc.)
- Combien d'heures allouées

Clé primaire composée : Combinaison employé + projet pour éviter les doublons.

SECTION 6 & 7 : Tables VENDEURS et VENTES

Ce qui a été fait : Création de tables pour gérer l'équipe commerciale et leurs ventes.

Table VENDEURS : Liste des commerciaux par région **Table VENTES :** Chaque vente réalisée avec son montant et produit

SECTION 8 : FILTRAGE DES DONNÉES

Ce qui a été fait : Apprentissage de techniques pour rechercher des informations spécifiques.

Exemples concrets :

Question 8.1 : "Montrez-moi tous les employés qui gagnent plus de 50 000€"

WHERE salaire > 50000

Résultat : Marie, Jean, Sophie

Question 8.2 : "Montrez-moi les employés qui gagnent plus de 40 000€ ET qui sont dans le département 1"

WHERE salaire > 40000 AND departement_id = 1

Résultat : Marie uniquement

Question 8.4 : "Montrez-moi les employés des départements 1, 2 ou 3"

WHERE departement_id IN (1, 2, 3)

Analogie : C'est comme utiliser les filtres sur un site de e-commerce pour afficher seulement les produits qui vous intéressent.

SECTION 9 : FILTRAGE AVEC LIKE (Recherche de Motifs)

Ce qui a été fait : Recherche d'informations basée sur des portions de texte.

Exemples :

Question 9.1 : "Tous les prénoms qui commencent par 'M'"

WHERE prenom LIKE 'M%'

Résultat : Marie

Question 9.2 : "Tous les noms qui contiennent 'art'"

WHERE nom LIKE '%art%'

Résultat : Martin

Question 9.3 : "Prénoms avec 'a' en 2e position"

WHERE prenom LIKE '_a%'

Résultat : Marie (M-a-rie)

Symboles utilisés :

- % = n'importe quels caractères
- _ = exactement un caractère

Analogie : Comme la recherche Google, mais plus précise.

SECTION 10 : GESTION DES VALEURS NULL

Ce qui a été fait : Identification des données manquantes.

Qu'est-ce que NULL ? NULL signifie "aucune valeur" ou "information manquante". Ce n'est ni zéro, ni vide, c'est "inconnu".

Question 10.1 : "Montrez-moi les employés qui ont un département assigné"

WHERE departement_id IS NOT NULL

Pourquoi c'est important ? Pour vérifier la complétude des données et éviter les erreurs.

SECTION 11 : TRI DES DONNÉES

Ce qui a été fait : Organisation des résultats dans un ordre spécifique.

Question 11.1 : Trier par salaire du plus petit au plus grand

ORDER BY salaire ASC

Résultat : Claire (52k) → Pierre (55k) → Sophie (68k) → Jean (72k) → Marie (85k)

Question 11.2 : Trier par salaire du plus grand au plus petit

ORDER BY salaire DESC

Résultat : Ordre inverse

Question 11.3 : Trier d'abord par département, puis par salaire

ORDER BY departement_id ASC, salaire DESC

Analogie : Comme trier vos emails par date, expéditeur, ou importance.

SECTION 12 : AGRÉGATION (Calculs Statistiques)

Ce qui a été fait : Calculs sur l'ensemble des données.

Question 12.1 : Statistiques globales sur les salaires

COUNT(*) = Nombre total d'employés
AVG(salaire) = Salaire moyen
MIN(salaire) = Salaire minimum
MAX(salaire) = Salaire maximum

Résultats attendus :

- Nombre d'employés : 5
- Salaire moyen : 66 400€
- Salaire min : 52 000€
- Salaire max : 85 000€

Analogie : Comme calculer votre moyenne de notes à l'école.

SECTION 13 : GROUPEMENT (Statistiques par Catégorie)

Ce qui a été fait : Calculs par groupe (département, région, etc.).

Question 13.1 : "Pour chaque département, combien d'employés et quel est le salaire moyen ?"

GROUP BY departement_id

Résultats attendus :

- Département 1 (Direction) : 1 employé, moyenne 85 000€
- Département 2 (RH) : 3 employés, moyenne 59 666€
- Département 3 (IT) : 1 employé, moyenne 68 000€

Question 13.2 : "Quels départements ont un salaire moyen > 45 000€ ?"

HAVING AVG(salaire) > 45000

Résultat : Tous les départements

Analogie : Comme calculer la moyenne des notes par matière plutôt que globalement.

SECTION 14 : JOINTURES (Lier des Tables)

Ce qui a été fait : Combinaison d'informations provenant de plusieurs tables.

Question 14.1 : INNER JOIN (jointure interne) "Montrez-moi les employés AVEC leur département"

INNER JOIN departements

Résultat : Seulement les employés qui ont un département assigné

Question 14.2 : LEFT JOIN (jointure à gauche) "Montrez-moi TOUS les employés, même ceux sans département"

LEFT JOIN departements

Résultat : Tous les employés, département = NULL pour ceux qui n'en ont pas

Analogie :

- INNER JOIN = Liste des étudiants inscrits à un cours
- LEFT JOIN = Liste de tous les étudiants, avec leur cours (s'ils en ont un)

Schéma visuel :

INNER JOIN: LEFT JOIN:
[A ∩ B] [A + (A ∩ B)]

SECTION 15 : SOUS-REQUÊTES (Requêtes Imbriquées)

Ce qui a été fait : Utilisation du résultat d'une requête dans une autre requête.

Question 15.1 : "Qui gagne plus que le salaire moyen ?"

WHERE salaire > (SELECT AVG(salaire) FROM employes)

Étapes :

1. Calculer le salaire moyen : 66 400€
2. Chercher qui gagne plus que 66 400€ **Résultat :** Marie (85k), Jean (72k), Sophie (68k)

Question 15.2 : "Qui travaille au département IT ?"

WHERE departement_id = (SELECT id FROM departements WHERE nom = 'IT')

Étapes :

1. Trouver l'ID du département IT : 3
2. Chercher les employés du département 3 **Résultat :** Sophie

Analogie : Comme une poupée russe - une question dans une question.

CONCEPTS CLÉS EXPLIQUÉS

1. IDENTITY(1,1) vs AUTO_INCREMENT

Qu'est-ce que c'est ? Un système qui génère automatiquement des numéros uniques pour identifier chaque ligne.

AUTO_INCREMENT : Utilisé dans MySQL **IDENTITY(1,1)** : Utilisé dans SQL Server

(1,1) signifie :

- Premier chiffre (1) : Commencer à 1

- Deuxième chiffre (1) : Augmenter de 1 à chaque fois

Exemple :

- 1er employé : ID = 1
- 2e employé : ID = 2
- 3e employé : ID = 3

Pourquoi ce changement dans le projet ? Pour être compatible avec SQL Server au lieu de MySQL.

2. CLÉ PRIMAIRE (PRIMARY KEY)

Définition simple : Un identifiant unique pour chaque ligne d'une table. Comme un numéro de sécurité sociale - personne ne peut avoir le même.

Règles :

- Doit être unique
- Ne peut pas être NULL (vide)
- Une seule par table (sauf clé composée)

Exemple : L'ID d'un employé

3. CLÉ ÉTRANGÈRE (FOREIGN KEY)

Définition simple : Un lien vers une autre table. Permet de créer des relations entre les informations.

Exemple : Dans la table `employes`, `departement_id` est une clé étrangère qui pointe vers la table `departements`.

Analogie : Comme une note de bas de page qui renvoie à une autre partie du document.

Avantages :

- Évite les incohérences
- Garantit que le département existe vraiment
- Empêche de supprimer un département qui a des employés

4. CLÉ PRIMAIRE COMPOSÉE

Définition simple : Utilisation de plusieurs colonnes ensemble pour créer un identifiant unique.

Exemple dans le projet : Table `affectations` : combinaison de `employe_id` + `projet_id`

Pourquoi ? Pour éviter qu'un employé soit affecté deux fois au même projet.

Analogie : Comme identifier une salle de classe par "Bâtiment + Étage + Numéro".

5. CONTRAINTES

Qu'est-ce que c'est ? Des règles qui garantissent la qualité des données.

Types de contraintes utilisées :

NOT NULL : Le champ doit obligatoirement avoir une valeur

prenom VARCHAR(50) NOT NULL

Impossible d'ajouter un employé sans prénom

UNIQUE : Aucune duplication autorisée

email VARCHAR(100) UNIQUE

Deux employés ne peuvent pas avoir le même email

DEFAULT : Valeur automatique si rien n'est spécifié

pays VARCHAR(50) DEFAULT 'France'

Si on ne précise pas le pays, ce sera "France"

6. TYPES DE DONNÉES

VARCHAR(n) : Texte de longueur variable (max n caractères)

- Exemple : `VARCHAR(50)` peut contenir de 0 à 50 caractères
- Utilisé pour : noms, emails, descriptions

INT : Nombre entier

- Exemple : 1, 25, 1000
- Utilisé pour : âges, quantités, identifiants

DECIMAL(p,s) : Nombre avec décimales

- p = nombre total de chiffres
- s = nombre de chiffres après la virgule
- Exemple : `DECIMAL(10,2)` pour les salaires (12345678.90)

DATE : Date au format AAAA-MM-JJ

- Exemple : 2024-01-15
- Utilisé pour : dates d'embauche, dates de commande

7. OPÉRATEURS DE COMPARAISON

= : Égal à **> :** Supérieur à **< :** Inférieur à **>= :** Supérieur ou égal à **<= :** Inférieur ou égal à **!= ou <>** : Différent de

Opérateurs logiques : AND : Les deux conditions doivent être vraies **OR :** Au moins une condition doit être vraie **IN :** Appartient à une liste de valeurs

8. FONCTIONS D'AGRÉGATION

COUNT(*) : Compte le nombre de lignes **SUM(colonne)** : Somme des valeurs **AVG(colonne)** : Moyenne des valeurs **MIN(colonne)** : Valeur minimale **MAX(colonne)** : Valeur maximale

Exemple pratique :

```
SELECT COUNT(*) FROM employes;  -- Résultat : 5
SELECT AVG(salaire) FROM employes;  -- Résultat : 66400
```

GLOSSAIRE DES TERMES TECHNIQUES

Base de données : Système organisé pour stocker et gérer des informations

Table : Structure qui organise les données en lignes et colonnes (comme un tableau Excel)

Colonne : Catégorie d'information (prénom, salaire, etc.)

Ligne : Un enregistrement complet (un employé, un client, etc.)

Requête : Instruction SQL pour demander ou modifier des données

Schéma : Structure globale de la base de données (ensemble des tables)

Index : Système d'accélération pour retrouver rapidement des informations

Contrainte : Règle qui garantit la validité des données

Transaction : Ensemble d'opérations qui doivent réussir ensemble ou échouer ensemble

NULL : Absence de valeur (ni zéro, ni vide, mais "inconnu")

Alias : Nom temporaire donné à une table ou colonne (AS)

Clause WHERE : Partie de la requête qui filtre les résultats

Clause ORDER BY : Partie de la requête qui trie les résultats

Clause GROUP BY : Partie de la requête qui regroupe les résultats

Clause HAVING : Filtre appliqué après un GROUP BY

INNER JOIN : Jointure qui ne retourne que les correspondances

LEFT JOIN : Jointure qui retourne toutes les lignes de gauche + correspondances

RIGHT JOIN : Jointure qui retourne toutes les lignes de droite + correspondances

FULL JOIN : Jointure qui retourne tout

BONNES PRATIQUES APPLIQUÉES

1. Nommage Cohérent

- Noms de tables au pluriel : `employes`, `clients`, `projets`
- Noms de colonnes au singulier et descriptifs : `date_embauche`, `salaire`
- Utilisation du snake_case : `departement_id` au lieu de `departementId`

2. Structure Logique

- Tables créées dans l'ordre des dépendances
- Départements avant employés (car employés référencent départements)

3. Intégrité Référentielle

- Utilisation systématique des clés étrangères
- Évite les données orphelines (employé sans département existant)

4. Documentation

- Commentaires clairs avant chaque section
- Séparation visuelle des parties avec des lignes de séparation

5. Sécurité des Données

- Contraintes NOT NULL sur les champs critiques
- Contraintes UNIQUE sur les emails
- Valeurs par défaut appropriées

CONCLUSION

Ce projet démontre une compréhension complète des fondamentaux de SQL :

✅ **Création de structures** (CREATE TABLE) ✅ **Insertion de données** (INSERT INTO) ✅ **Consultation** (SELECT) ✅ **Filtrage** (WHERE, LIKE, IN) ✅ **Tri** (ORDER BY) ✅ **Agrégation** (COUNT, AVG, MIN, MAX) ✅ **Groupement** (GROUP BY, HAVING) ✅ **Jointures** (INNER JOIN, LEFT JOIN) ✅ **Sous-requêtes** (SELECT dans SELECT)

Le code est **propre**, **bien commenté**, **structuré** et **fonctionnel**.

Tous les concepts essentiels pour manipuler et analyser des données avec SQL sont couverts et appliqués de manière pratique.

Fichier créé par : DOH Yawo Marie Victor

Date : Janvier 2026

Formation : D-CLIC 3 – Parcours Data