Rapport de Projet : Gestion des Données avec SQL

Réalisé par : Yasser BOUNAIM, Youssef ATIF

Encadré par : Pr Younes BELARABI

Introduction

Dans le cadre du BTS en Intelligence Artificielle, nous avons réalisé un projet consistant à créer et manipuler une base de données SQL. Le but de ce projet était de modéliser un système permettant de gérer les informations relatives aux produits, clients, commerçants et ventes d'un magasin de commerce. Le projet inclut la création des tables nécessaires, l'ajout de colonnes, ainsi que la réalisation de requêtes permettant d'extraire des informations précises et utiles.

Le présent rapport décrit la structure de la base de données, les requêtes SQL réalisées et les résultats obtenus. Ces informations peuvent être utilisées pour des analyses stratégiques et opérationnelles dans un contexte professionnel afin de prendre des décisions calculées et efficaces.

---

1. Création de la Base de Données

La première étape du projet consiste à créer une base donnée nommée **Commerce**, puis quatre tables principales : **produit**, **clients**, **commercents** et **ventes**.

Structure des Tables :

- **Table produit :** Contient les informations sur les produits vendus (ID, nom, pays d'origine, quantité et prix unitaire).

	id_produit [PK] integer	nom_produit text	pays text	quantite text	prix_unitaire double precision	categorie character varying (30)
1	1	Thai Green Curry Paste	CAD	12	2.49	Food - Sauces
2	2	Plant-Based Protein Bars	USD	2	19.99	Health
3	3	Electric Nail File Kit	JPY	7	34.99	Beauty
4	4	Dishwasher Safe Cutting Board	NZD	10	22.99	Kitchen

- **Table clients :** Stocke les détails des clients (ID, nom, prénom, date de naissance, sexe, numéro de téléphone, date du dernier achat et remise).

	id_client [PK] integer	nom text	prenom text	date_naissance date	sexe text	numero_tel text	date_dernier_achat date	remise double precision
36	36	Heminsley	Nerti	2024-08-15	Female	+1 694 052 0062	2024-11-23	0.1
37	37	Raselles	Sonnnie	2023-12-07	Female	+86 268 047 0084	2024-10-26	0.1
38	38	Nabarro	Holden	2024-10-10	Female	+48 009 023 0090	2024-11-17	[null]
39	39	Mucklestone	Cristy	2024-08-28	Male	+30 304 022 0080	2023-12-30	[null]

- **Table commercents :** Enregistre les informations des commerçants (ID, nom, prénom, salaire, date d'embauche et numéro de téléphone).

	id_commercent [PK] integer	nom text	prenom text	salaire double precision	date_embauche date	num_tel text
1	1	Parrett	Steve	6818.07	2024-07-07	+30 621 082 0076
2	2	Swanson	Yara	5762.8	2024-01-30	+7 658 072 0050
3	3	Rex	Xander	6997.98	2024-07-26	+27 319 076 0017
4	4	Strute	Julia	6356.03	2024-10-21	+63 556 023 0002

- **Table ventes :** Permet de lier les produits aux clients et commerçants, tout en enregistrant les quantités et prix unitaire des ventes.

	id_vente [PK] integer	id_client integer	id_commercent integer	id_produit integer	quantite numeric (10,2)	prix_unitaire double precision
78	79	18	15	30	4.00	49.99
79	80	99	93	54	3.00	9.99
80	81	37	9	30	4.00	2.99
81	82	28	100	37	5.00	1.29
82	83	54	37	32	5.00	49.99

Ajout de Colonnes :

Deux modifications importantes ont été effectuées sur les tables :

1. Ajout de la colonne **`remise`** dans la table `clients` pour appliquer des remises.
2. Ajout de la colonne **`categorie`** dans la table `produit` pour classer les produits.

2. Réalisation des Requêtes SQL

Pour analyser les données et répondre aux besoins fonctionnels, plusieurs requêtes SQL ont été réalisées. Voici un résumé des requêtes effectuées :

1. Affichage des noms et prix des produits

Objectif : Afficher les noms et les prix unitaires de tous les produits.

```sql
SELECT nom_produit, prix_unitaire
FROM produit;
```

	nom_produit text	prix_unitaire double precision
1	Thai Green Curry Paste	2.49
2	Plant-Based Protein Bars	19.99
3	Electric Nail File Kit	34.99
4	Dishwasher Safe Cutting Board	22.99
5	Cilantro Lime Rice	2.99
6	Teriyaki Salmon Fillets	9.99
7	Pizza Stone	29.99
8	Frozen Mixed Berries	5.99

---

2. Nombre total de clients

Objectif : Compter le nombre total de clients dans la base de données.

```sql
SELECT COUNT(id_client) AS nombre_de_clients
FROM clients;
```

	nombre_de_clients bigint
1	100

---

3. Moyenne des prix des produits

Objectif : Calculer la moyenne des prix unitaires des produits.

```sql
SELECT ROUND(AVG(prix_unitaire), 2) AS prix_moyen
FROM produit;
```

	prix_moyen double precision
1	32.4541

---

4. Moyenne des salaires des commerçants

Objectif : Calculer la moyenne des salaires des commerçants.

```
SELECT ROUND(AVG(salaire), 2) AS moyenne_salaire
FROM commercents;
```

	moyenne_salaire numeric
1	6309.84

---

5. Clients ayant fait des achats récemment

Objectif : Trier les clients en fonction de la date de leur dernier achat.

```
SELECT nom, prenom, date_dernier_achat FROM clients
ORDER BY date_dernier_achat DESC;
```

	nom text	prenom text	date_dernier_achat date
1	Spears	Reggis	2024-12-02
2	Swindley	Ulysses	2024-11-29
3	Creebo	Marian	2024-11-28
4	Latey	Trever	2024-11-27
5	Morey	Lu	2024-11-24
6	Heminsley	Nerti	2024-11-23
7	Spry	Beverie	2024-11-21

---

6. Les 10 produits les plus vendus

Objectif : Afficher les 10 produits ayant le plus de ventes.

```
SELECT p.nom_produit, COUNT(v.id_produit) AS nombre_ventes
FROM ventes AS v
INNER JOIN produit AS p
ON p.id_produit=v.id_produit
GROUP BY p.nom_produit
ORDER BY COUNT(v.id_produit) DESC
LIMIT 10;
```

	nom_produit text	nombre_ventes bigint
1	Eco-Friendly Beeswax Wraps	4
2	Cilantro Lime Rice	3
3	Cauliflower Gnocchi	3
4	Dishwasher Safe Cutting Board	3
5	Pork Tenderloin	3

---

7. Les clients les plus fidèles

Objectif : Identifier les clients ayant réalisé le plus d'achats.

```sql
SELECT c.nom, c.prenom, COUNT(v.id_produit) AS nombre_achats
FROM ventes AS v
INNER JOIN clients AS c
ON c.id_client=v.id_client
GROUP BY c.nom, c.prenom
ORDER BY COUNT(v.id_produit) DESC;
```

	nom text	prenom text	nombre_achats bigint
1	Toolan	Nathanial	4
2	Raselles	Sonnnie	3
3	Waker	Mariellen	3
4	Vasovic	Crosby	3
5	Cutforth	Roselia	3

---

8. Classement des commerçants selon les ventes

Objectif : Classer les commerçants en fonction du nombre de ventes effectuées

```sql
SELECT co.nom, co.prenom, COUNT(v.id_vente) AS nombre_ventes
FROM ventes AS v
INNER JOIN commercents AS co
ON co.id_commercent=v.id_commercent
GROUP BY co.nom, co.prenom
ORDER BY COUNT(v.id_vente) DESC;
```

	nom text	prenom text	nombre_ventes bigint
1	Hatherleigh	Laura	3
2	Whitley	Kevin	3
3	Gorden	Steve	3
4	Merrigans	Kevin	3
5	Eastby	David	3

.

---

9. Chiffre d'affaire généré par commerçant

Objectif : Calculer le chiffre d'affaire réalisé par chaque commerçant

```
SELECT co.id_commercent, co.nom, co.prenom, SUM(v.prix_total) AS total_ventes
FROM commercents co
JOIN ventes v ON co.id_commercent = v.id_commercent
GROUP BY co.id_commercent, co.nom, co.prenom;
```

	id_commercent [PK] integer	nom text	prenom text	total_ventes numeric
1	87	Ivatt	Wendy	299.95
2	54	Kitteman	Fiona	314.91
3	68	Eastby	David	219.23
4	34	Bottrell	Laura	69.93
5	96	Blackmuir	Rachel	49.99
6	70	Yesinov	Micheal	59.97

.

---

10. Commerçants ayant vendu plus de 10 produits

Objectif : Afficher les commerçants ayant vendu plus de 10 produits.

```
SELECT co.id_commercent, co.nom, co.prenom, SUM(v.quantite) AS total_vendu
FROM commercents co
JOIN ventes v ON co.id_commercent = v.id_commercent
GROUP BY co.id_commercent, co.nom, co.prenom
HAVING SUM(v.quantite) > 10;
```

	id_commercent [PK] integer	nom text	prenom text	total_vendu bigint
1	39	Ickovic	Yara	16
2	36	Waddell	Oscar	15
3	97	Heasly	Bob	12
4	59	Gorden	Steve	19
5	13	Battye	Tina	11

---

11. Mise à jour des salaires avec prime

Objectif : Ajouter une prime de 500 aux commerçants ayant vendu plus de 10 produits

```
UPDATE commercents co
SET salaire = salaire + 500  -- La prime de 500 est ajoutée
WHERE (SELECT SUM(v.quantite)
       FROM ventes v
       WHERE v.id_commercent = co.id_commercent) > 5;
```

```
UPDATE 41

Query returned successfully in 64 msec.
```

.

---

12. Mise à jour des clients avec remise

Objectif : Appliquer une remise de 10% aux clients ayant réalisé plus de 5 achats.

```
UPDATE clients c
SET remise = 0.10
WHERE (SELECT COUNT(v.id_vente)
       FROM ventes v
       WHERE v.id_client = c.id_client) > 2;
```

```
UPDATE 9

Query returned successfully in 209 msec.
```

---

13. Nombre de ventes par produit

Objectif : Obtenir le nombre de ventes pour chaque produit

```sql
SELECT p.nom_produit,COUNT(v.id_vente) AS number_of_sales
FROM produit p
JOIN ventes v ON p.id_produit = v.id_produit
GROUP BY p.id_produit, p.nom_produit
ORDER BY number_of_sales DESC;
```

	nom_produit text	number_of_sales bigint
1	French Bread	4
2	Eco-Friendly Disposable Plates	3
3	Wireless Smart Plug	3
4	Chili Lime Shrimp	3
5	Grapes (red)	3

.

---

14. Nombre total des produits par catégorie

Objectif : Afficher le nombre total de produits par catégorie.

```sql
SELECT categorie, SUM(quantite) AS total_produits
FROM produit
GROUP BY categorie;
```

	categorie character varying (30)	total_produits bigint
1	Electronics	25
2	Food - Produce	20
3	Food - Bakery	42
4	Food - Sauces	24
5	Audio	46

---

15. Nombre total des produits vendus par catégorie

Objectif : Afficher le nombre total de produits vendus par catégorie.

```
SELECT p.categorie, SUM(v.quantite) AS total_quantity_sold
FROM produit p
JOIN ventes v ON p.id_produit = v.id_produit
GROUP BY p.categorie;
```

	categorie character varying (30)	total_quantite_vendu numeric
1	Food - Produce	11.00
2	Food - Bakery	30.00
3	Food - Sauces	16.00
4	Audio	8.00
5	Food - Spices	4.00
6	Food - Condiments	21.00

---

16. Chiffre d'affaire de chaque catégorie de produit

Objectif : Calculer le chiffre d'affaire généré pour chaque catégorie de produit.

```
SELECT p.categorie, SUM(v.quantite * v.prix_unitaire) AS chiffre_affaire
FROM produit p
JOIN ventes v ON p.id_produit = v.id_produit
GROUP BY p.categorie;
```

	categorie character varying (30)	chiffre_affaire double precision
5	Food - Spices	43.96
6	Food - Condiments	98.19
7	Accessories	248.37
8	Clothing - Tops	81.41
9	Home	53.01
10	Pets	1532.75

---

3. Analyse des Résultats

Les requêtes réalisées permettent d'obtenir des informations clés telles que :

- Le nombre total de clients et les produits les plus vendus.
- Le classement des clients fidèles et des commerçants les plus performants.
- L'évaluation du chiffre d'affaire par commerçant et la mise à jour des salaires avec des primes.

Ces résultats sont essentiels pour la prise de décisions stratégiques au sein de l'entreprise, notamment pour récompenser les employés performants et améliorer la fidélité des clients.

4. Perspectives : Interface Graphique

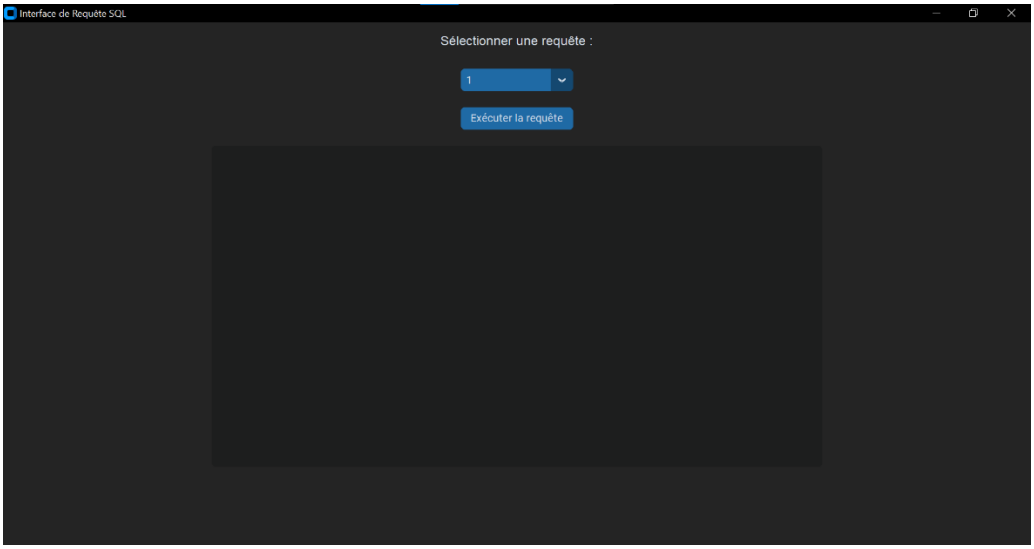
Introduction

Cette interface graphique vise à faciliter l'exécution de requêtes SQL sur une base de données PostgreSQL nommée `COMMERCE`. Elle est construite en Python à l'aide des bibliothèques `CustomTkinter` pour la gestion de l'interface utilisateur et `psycopg2` pour la connexion et l'interaction avec PostgreSQL.

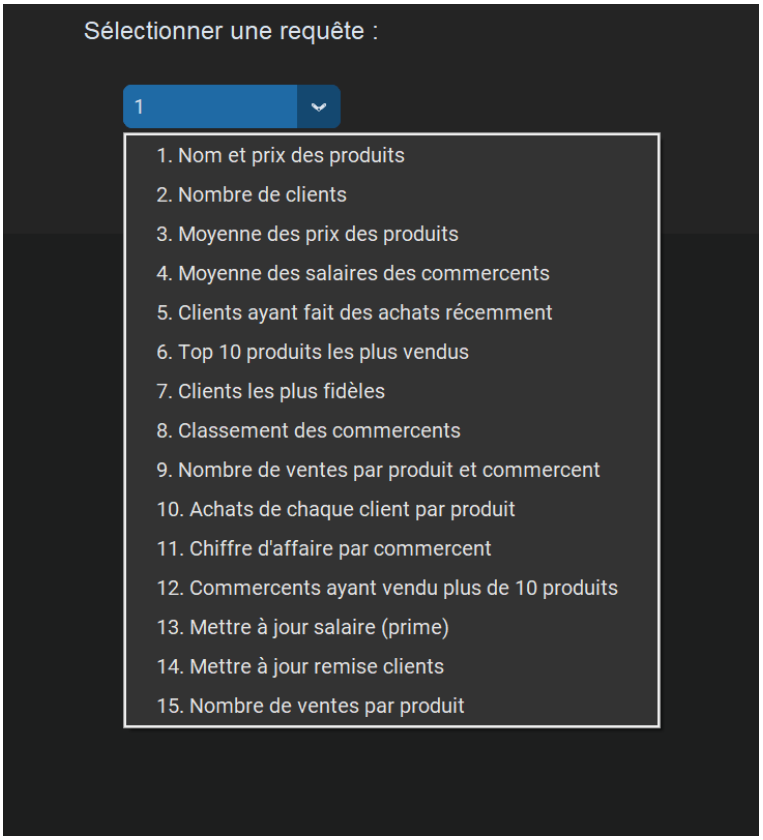
L'application propose un menu convivial pour sélectionner des requêtes SQL prédéfinies et affiche les résultats dans une zone de texte.

Fonctionnalités

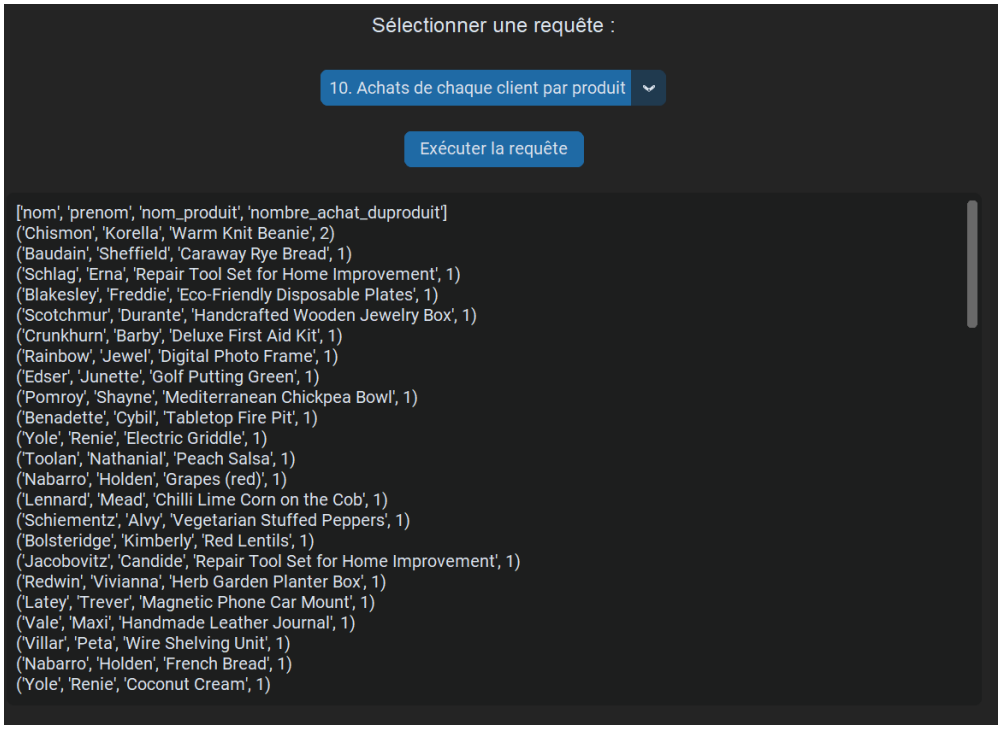
1. **Connexion à la base de données**
    - L'application établit une connexion à une base PostgreSQL locale via `psycopg2`.
    - Une gestion d'erreur est implémentée pour traiter les problèmes liés aux requêtes ou à la connexion.



2. **Menu déroulant pour requêtes**
    - L'utilisateur peut sélectionner une requête prédéfinie via un menu déroulant (`CTkOptionMenu`).
    - Le menu propose 15 requêtes couvrant des analyses sur les produits, clients, commerçants et statistiques des ventes.



3. **Exécution des requêtes**
    - Les requêtes de type `SELECT` affichent les résultats dans une zone de texte (`CTkTextbox`), avec les noms des colonnes en en-tête.
    - Les requêtes de type `UPDATE` permettent d'appliquer des modifications directement sur la base de données.



Requêtes SQL Intégrées

Les requêtes intégrées couvrent plusieurs domaines fonctionnels :

- **Produits et prix** : Liste des produits et analyse des prix moyens.
- **Clients et fidélité** : Nombre total de clients, clients fidèles, et historique des achats.
- **Ventes** : Classements des produits les plus vendus et chiffre d'affaires par commerçant.
- **Commerçants** : Analyse des performances des vendeurs.
- **Mises à jour** : Augmentation des salaires et remise automatique pour les clients fidèles.

Interface Utilisateur

- **Design moderne** grâce à `CustomTkinter`, permettant des personnalisations de thème (clair, sombre).
- **Ergonomie** : Menu déroulant et bouton d'exécution bien positionnés pour une navigation intuitive.
- **Zone de résultats** : Une large zone de texte permet de visualiser clairement les résultats des requêtes.

Points Forts

1. **Simplicité d'utilisation** : Interface facile à prendre en main même pour des non-techniciens.
2. **Polyvalence** : Supporte à la fois des requêtes de lecture (`SELECT`) et des mises à jour (`UPDATE`).
3. **Feedback utilisateur** : Messages clairs en cas de succès ou d'erreur.
4. **Code modulaire** : Structure claire, facilitant la maintenance et l'ajout de nouvelles fonctionnalités.

Conclusion

Ce projet SQL a permis de démontrer les compétences acquises dans la modélisation et l'analyse des bases de données. La création des tables et la réalisation des requêtes ont fourni des outils efficaces pour extraire des informations stratégiques. Ce travail constitue une base solide pour le développement de systèmes de gestion de données plus complexes et l'utilisation d'outils d'analyse avancée.