

UCAO-UUT



Republique Togolaise



[Tp Analyse des ventes]

Étudiant: AGBOGBODO Désiré

Prof : Mr LOUKOUME

Introduction

Ce TP consiste à analyser les ventes d'un magasin en utilisant une base de données SQLite et des outils Python tels que Pandas, NumPy, Matplotlib et Seaborn. L'objectif est de modéliser une base de données, d'y insérer des données fictives, d'effectuer des analyses statistiques et de produire des visualisations graphiques pour en tirer des insights pertinents.

Étape 1 : Modélisation de la base de données

Schéma de la base de données

Nous allons créer une base de données relationnelle avec les tables suivantes :

1. Produits :

- `id_produit` (INTEGER, clé primaire)
- `nom_produit` (TEXT)
- `categorie` (TEXT)
- `prix_unitaire` (REAL)

2. Clients :

- `id_client` (INTEGER, clé primaire)
- `nom_client` (TEXT)
- `email` (TEXT)

3. Ventes :

- `id_vente` (INTEGER, clé primaire)
- `id_produit` (INTEGER, clé étrangère vers Produits)
- `id_client` (INTEGER, clé étrangère vers Clients)
- `date_vente` (TEXT)
- `quantite` (INTEGER)
- `montant_total` (REAL)

Diagramme Entité-Association

- Produits (1) → (N) Ventes (N) ← (1) Clients

Étape 2 : Création de la base de données et insertion des données simulées

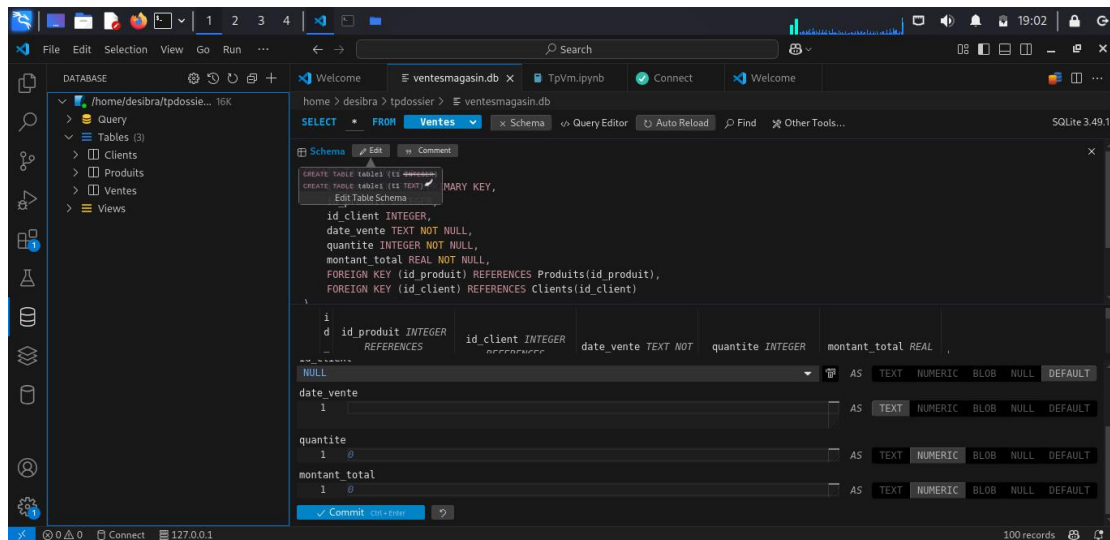
Script SQL pour créer les tables

```
-- Création de la table Produits
CREATE TABLE Produits (
    id_produit INTEGER PRIMARY KEY,
    nom_produit TEXT NOT NULL,
    categorie TEXT,
    prix_unitaire REAL NOT NULL
);

-- Création de la table Clients
CREATE TABLE Clients (
    id_client INTEGER PRIMARY KEY,
    nom_client TEXT NOT NULL,
    email TEXT
);

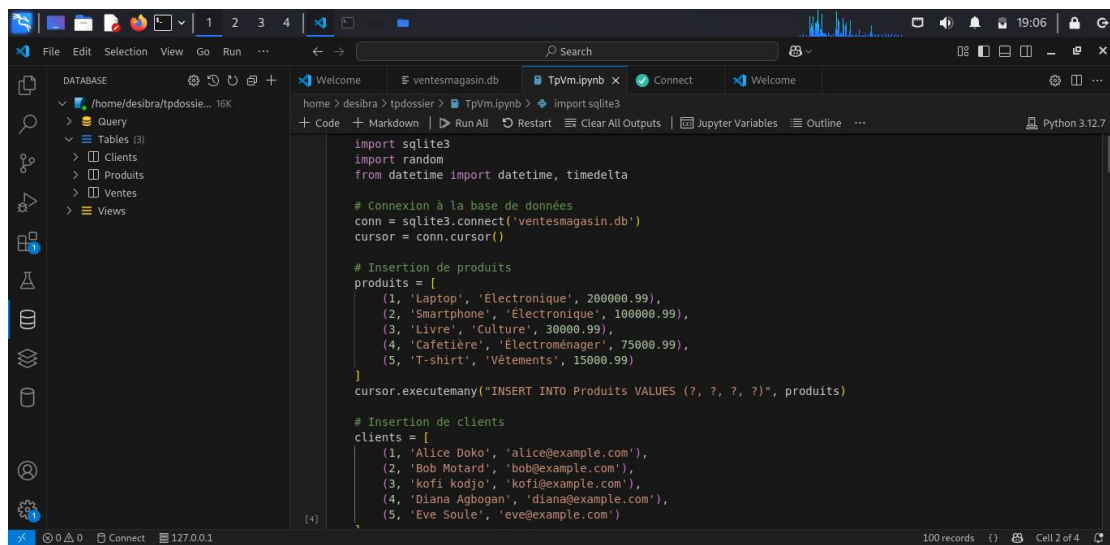
-- Création de la table Ventes
CREATE TABLE Ventes (
    id_vente INTEGER PRIMARY KEY,
    id_produit INTEGER,
    id_client INTEGER,
    date_vente TEXT NOT NULL,
    quantite INTEGER NOT NULL,
    montant_total REAL NOT NULL,
    FOREIGN KEY (id_produit) REFERENCES Produits(id_produit),
    FOREIGN KEY (id_client) REFERENCES Clients(id_client)
);
```

NB: Ce script a été saisi dans le Query editor du fichier.db du projet qui a été ouvert avec SQLite : open database, ce qui a conduit aux resultat suivant:

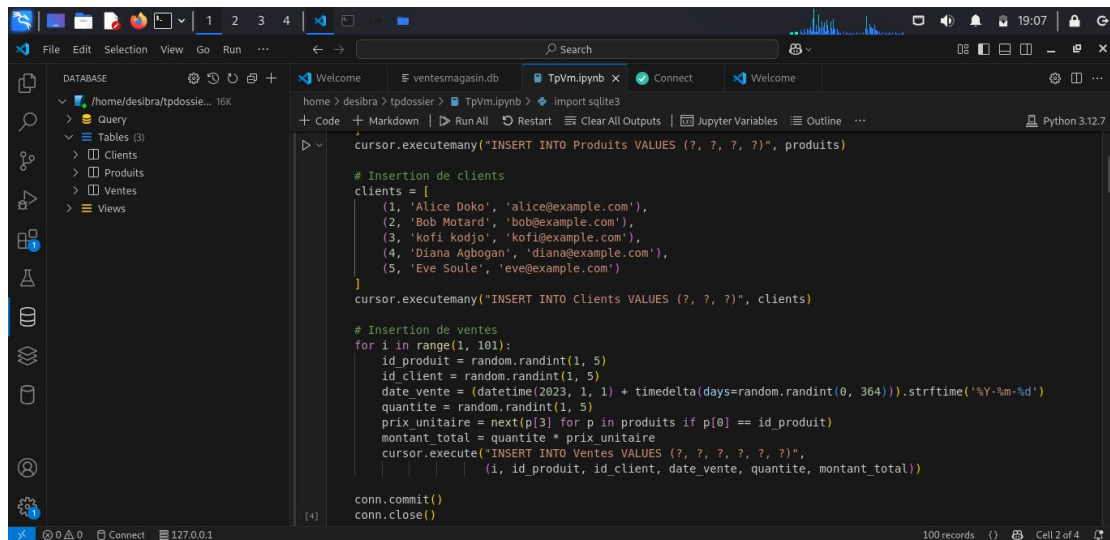


Insertion de données fictives:

Nous allons générer des données aléatoires pour peupler les tables. Voici le script python pour l'insertion de données fictives:



Avec la suite du code en bas....



```
import sqlite3

conn = sqlite3.connect('ventesmagasin.db')
cursor = conn.cursor()

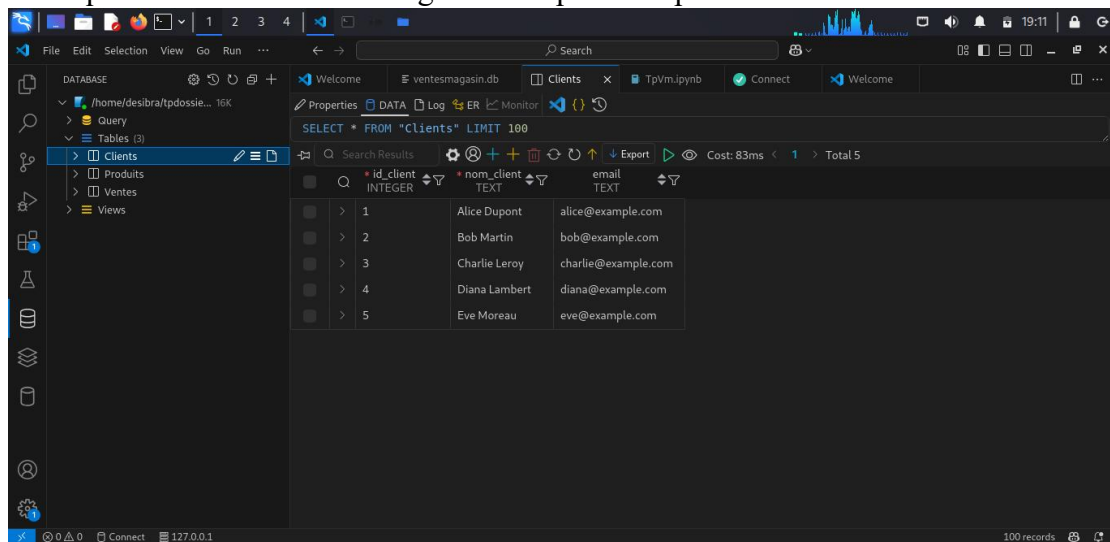
cursor.executemany("INSERT INTO Produits VALUES (?, ?, ?, ?)", produits)

# Insertion de clients
clients = [
    (1, 'Alice Doko', 'alice@example.com'),
    (2, 'Bob Motard', 'bob@example.com'),
    (3, 'kofi kodjo', 'kofi@example.com'),
    (4, 'Diana Agbogun', 'diana@example.com'),
    (5, 'Eve Soule', 'eve@example.com')
]
cursor.executemany("INSERT INTO Clients VALUES (?, ?, ?)", clients)

# Insertion de ventes
for i in range(1, 101):
    id_produit = random.randint(1, 5)
    id_client = random.randint(1, 5)
    date_vente = (datetime(2023, 1, 1) + timedelta(days=random.randint(0, 364))).strftime('%Y-%m-%d')
    quantite = random.randint(1, 5)
    prix_unitaire = next(p[3] for p in produits if p[0] == id_produit)
    montant_total = quantite * prix_unitaire
    cursor.execute("INSERT INTO Ventes VALUES (?, ?, ?, ?, ?)",
        (i, id_produit, id_client, date_vente, montant_total))

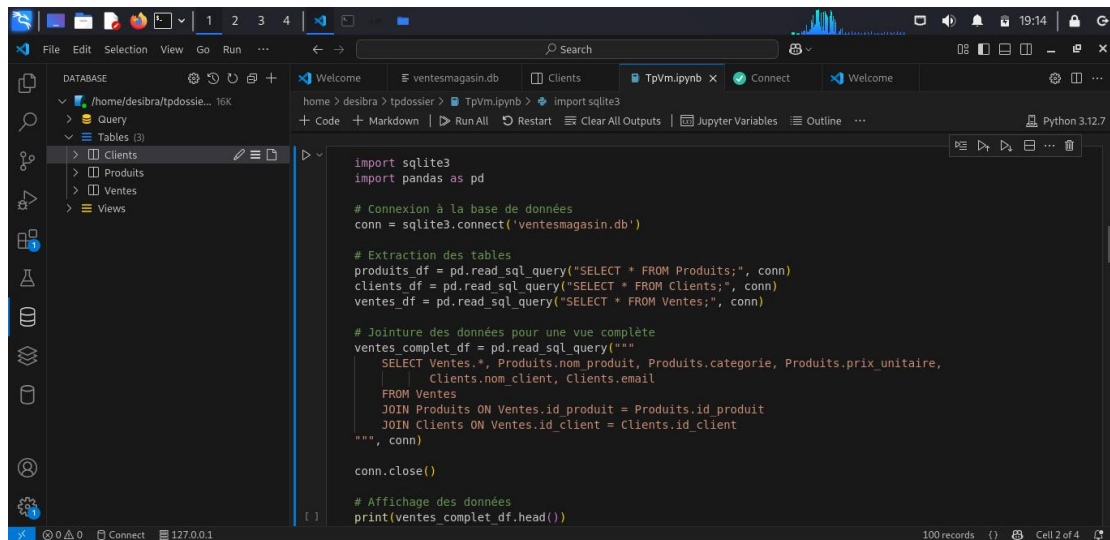
conn.commit()
conn.close()
```

Cela a conduit à l'enregistrement de ces données dans la base de données comme l'indique la table client de l'image suivant par exemple :



	id_client	nom_client	email
>	1	Alice Dupont	alice@example.com
>	2	Bob Martin	bob@example.com
>	3	Charlie Leroy	charlie@example.com
>	4	Diana Lambert	diana@example.com
>	5	Eve Moreau	eve@example.com

Étape 3 : Extraction des données avec Python



```
import sqlite3
import pandas as pd

# Connexion à la base de données
conn = sqlite3.connect('ventesmagasin.db')

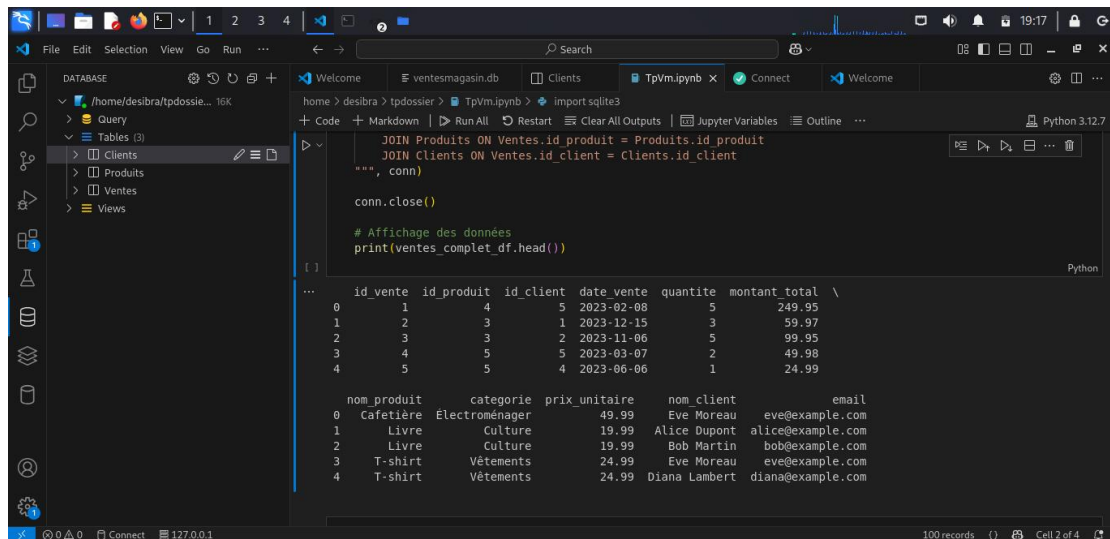
# Extraction des tables
produits_df = pd.read_sql_query("SELECT * FROM Produits;", conn)
clients_df = pd.read_sql_query("SELECT * FROM Clients;", conn)
ventes_df = pd.read_sql_query("SELECT * FROM Ventes;", conn)

# Jointure des données pour une vue complète
ventes_complet_df = pd.read_sql_query("""
    SELECT Ventes.*, Produits.nom_produit, Produits.categorie, Produits.prix_unitaire,
           Clients.nom_client, Clients.email
    FROM Ventes
    JOIN Produits ON Ventes.id_produit = Produits.id_produit
    JOIN Clients ON Ventes.id_client = Clients.id_client
""", conn)

conn.close()

# Affichage des données
print(ventes_complet_df.head())
```

Cela a conduit aux resultat suivant :



```
JOIN Produits ON Ventes.id_produit = Produits.id_produit
JOIN Clients ON Ventes.id_client = Clients.id_client
""", conn)

conn.close()

# Affichage des données
print(ventes_complet_df.head())
```

	id_vente	id_produit	id_client	date_vente	quantite	montant total	
0	1	4	5	2023-02-08	5	249.95	
1	2	3	1	2023-12-15	3	59.97	
2	3	3	2	2023-11-06	5	99.95	
3	4	5	5	2023-03-07	2	49.98	
4	5	5	4	2023-06-06	1	24.99	

	nom_produit	categorie	prix_unitaire	nom_client	email
0	Cafetière	Electroménager	49.99	Eve Moreau	eve@example.com
1	Livre	Culture	19.99	Alice Dupont	alice@example.com
2	Livre	Culture	19.99	Bob Martin	bob@example.com
3	T-shirt	Vêtements	24.99	Eve Moreau	eve@example.com
4	T-shirt	Vêtements	24.99	Diana Lambert	diana@example.com

Ici on a un script notebook Jupyter qui se connecte à la base de données SQLite. On a utiliser le module standard sqlite3 de Python pour établir la connexion.

import sqlite3 et

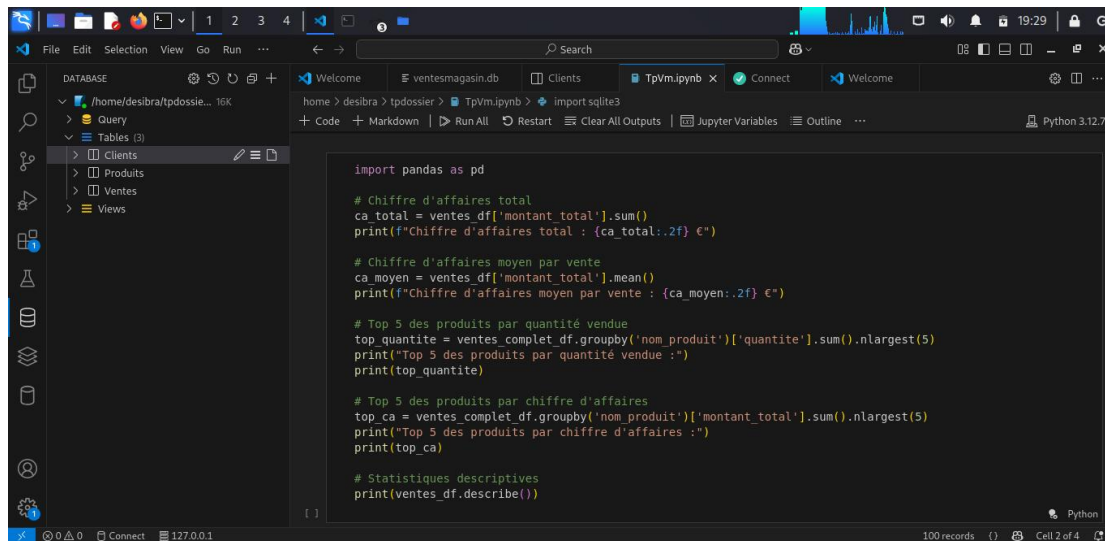
conn = sqlite3.connect('ventesmagasin.db')

nous a permis de nous connecter au fichier de base de données. À l'aide du curseur (cursor = conn.cursor()), on a executé des requêtes **SQL**

SELECT pour extraire les données.

Tous ces differents étapes nous on conduit aux résultats si dessus de l'image.

Étape 4 : Analyse statistique avec Pandas et NumPy



```
import pandas as pd

# Chiffre d'affaires total
ca_total = ventes_df['montant_total'].sum()
print(f"Chiffre d'affaires total : {ca_total:.2f} €")

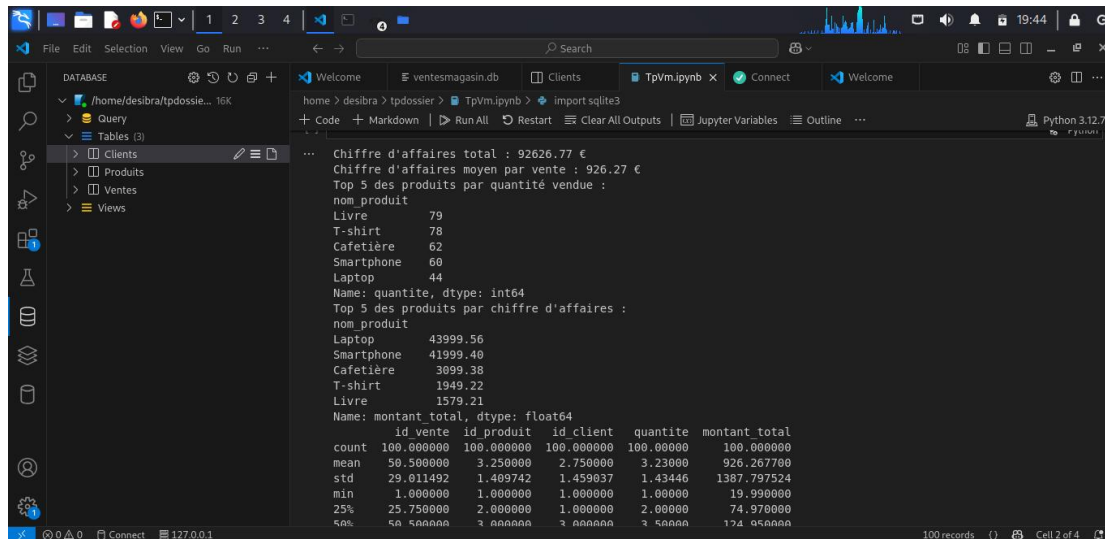
# Chiffre d'affaires moyen par vente
ca_moyen = ventes_df['montant_total'].mean()
print(f"Chiffre d'affaires moyen par vente : {ca_moyen:.2f} €")

# Top 5 des produits par quantité vendue
top_quantite = ventes_complet_df.groupby('nom_produit')['quantite'].sum().nlargest(5)
print("Top 5 des produits par quantité vendue :")
print(top_quantite)

# Top 5 des produits par chiffre d'affaires
top_ca = ventes_complet_df.groupby('nom_produit')['montant_total'].sum().nlargest(5)
print("Top 5 des produits par chiffre d'affaires :")
print(top_ca)

# Statistiques descriptives
print(ventes_df.describe())
```

Ici nous avons effectué une **analyse exploratoire et statistique** pour en extraire des informations pertinentes sur les ventes du magasin. Pour cela on a utilisé les fonctionnalités de **Pandas** et **NumPy** pour les calculs numériques si nécessaire. Ce qui inclut des calculs comme le **Chiffre d'affaires total**, le **chiffre moyen d'affaires**, les **produits plus vendus** etc... Comme l'indique le résultat sur l'image suivante:

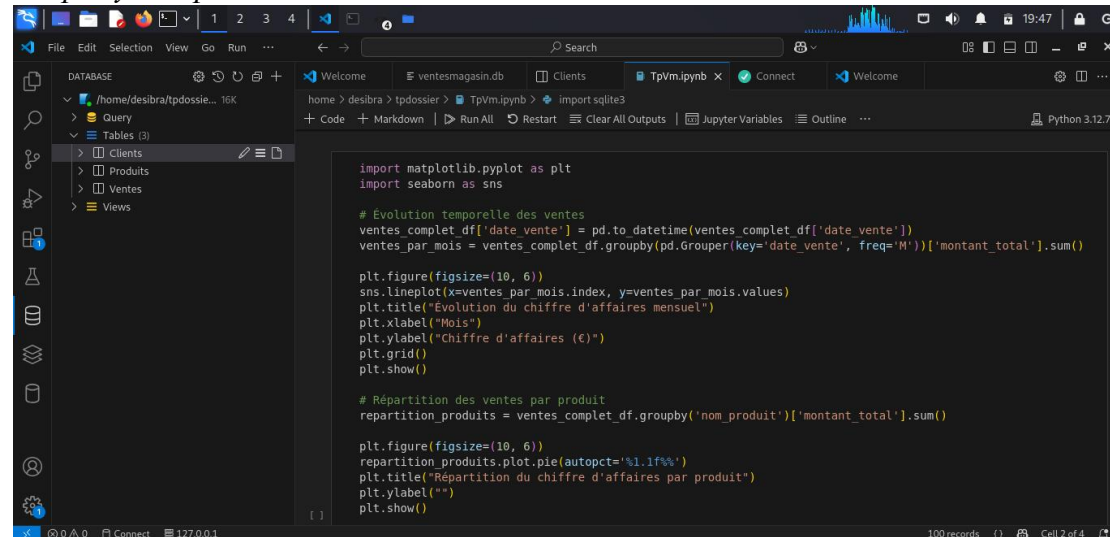


```
... Chiffre d'affaires total : 92626.77 €
Chiffre d'affaires moyen par vente : 926.27 €
Top 5 des produits par quantité vendue :
nom_produit
Livre      79
T-shirt    78
Cafetière  62
Smartphone 60
Laptop     44
Name: quantite, dtype: int64
Top 5 des produits par chiffre d'affaires :
nom_produit
Laptop      43999.56
Smartphone  41999.40
Cafetière   3099.38
T-shirt     1949.22
Livre       1579.21
Name: montant_total, dtype: float64
```

	id_vente	id_produit	id_client	quantite	montant_total
count	100.000000	100.000000	100.000000	100.000000	100.000000
mean	50.500000	3.250000	2.750000	3.230000	926.267700
std	29.011492	1.409742	1.459037	1.43446	1387.797524
min	1.000000	1.000000	1.000000	1.000000	19.990000
25%	25.750000	2.000000	1.000000	2.000000	74.970000
50%	50.500000	3.250000	2.750000	3.230000	926.267700

Étape 5 : Visualisation des données avec Matplotlib et Seaborn

Script Python pour les visualisations :



```
import matplotlib.pyplot as plt
import seaborn as sns

# Évolution temporelle des ventes
ventes_complet_df['date_vente'] = pd.to_datetime(ventes_complet_df['date_vente'])
ventes_par_mois = ventes_complet_df.groupby(pd.Grouper(key='date_vente', freq='M'))['montant_total'].sum()

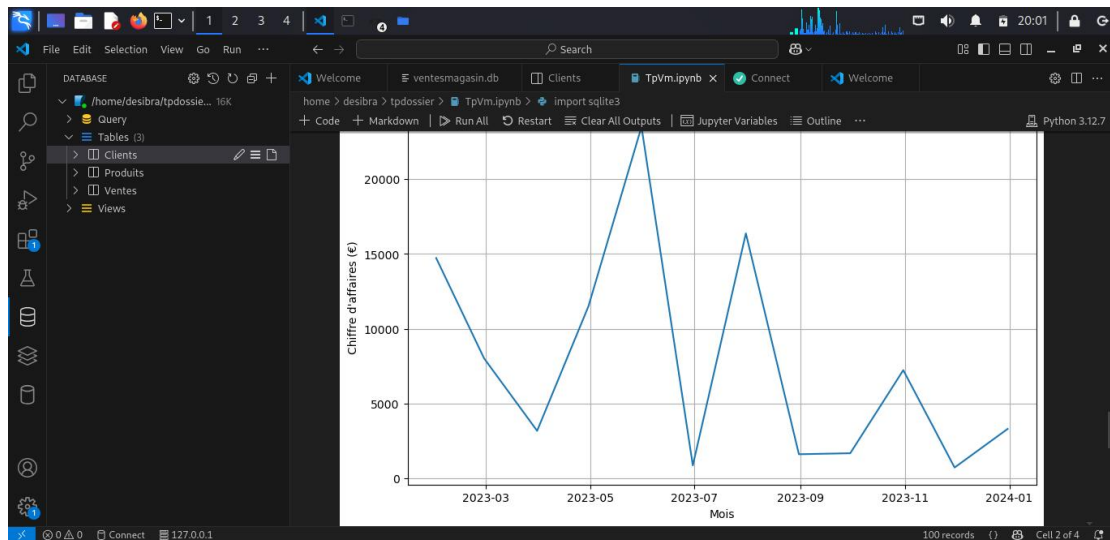
plt.figure(figsize=(10, 6))
sns.lineplot(x=ventes_par_mois.index, y=ventes_par_mois.values)
plt.title("Evolution du chiffre d'affaires mensuel")
plt.xlabel("Mois")
plt.ylabel("Chiffre d'affaires (€)")
plt.grid()
plt.show()

# Répartition des ventes par produit
repartition_produits = ventes_complet_df.groupby('nom_produit')['montant_total'].sum()

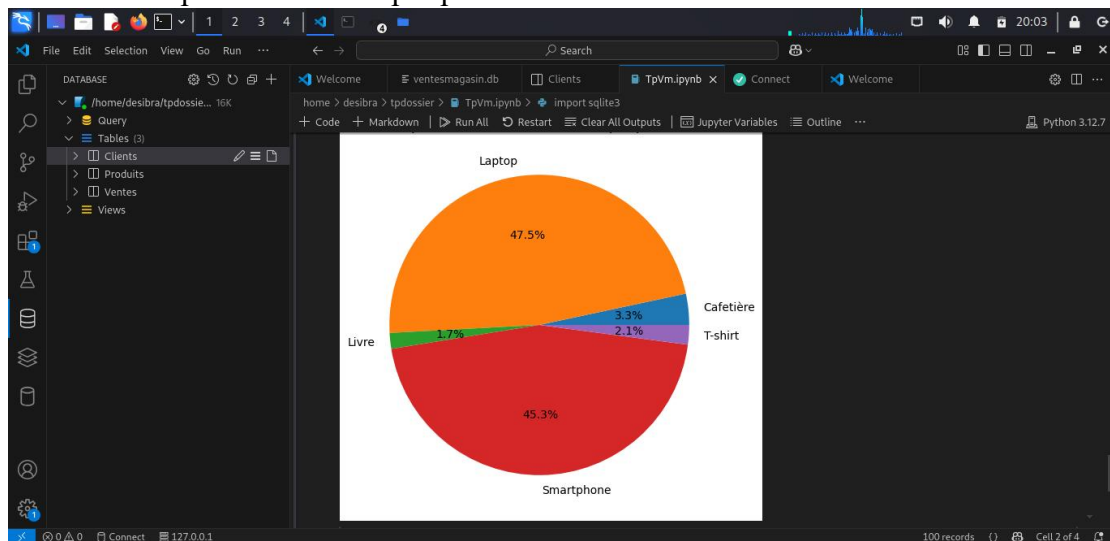
plt.figure(figsize=(10, 6))
repartition_produits.plot.pie(autopct='%1.1f%%')
plt.title("Répartition du chiffre d'affaires par produit")
plt.ylabel("")
plt.show()
```

Ici on présente les résultats sous forme **graphique** afin de visualiser les tendances et les comparaisons de manière éloquente. Pour cela on a utilisé les bibliothèques **Matplotlib** (pour des graphiques basiques personnalisables) et **Seaborn** (pour bénéficier de styles et graphiques statistiques avancés). On a réalisé deux visualisations :

L'évolution temporelle des ventes : Cela donne un aperçu de l'évolution du chiffre d'affaires au cours du temps. On a tracé un graphique en courbe où l'axe des abscisses correspond au temps (les mois) et l'axe des ordonnées correspond au chiffre d'affaires total du magasin sur la période temporelle correspondante comme l'indique l'image suivante :



Répartition des ventes par produit : Cela représente la part de chaque produit dans les ventes totales. Cela permet de visualiser quels produits contribuent le plus au chiffre d'affaires global. Le **diagramme en secteurs sur l'image suivant permet de bien montrer** les parts de marché par produit :



Rapport

1. **Introduction** : Contexte et objectifs du TP.
2. **Modélisation** : Schéma de la base de données et diagramme E/A.
3. **Création de la base**: Scripts SQL et Python pour la génération des données.
4. **Extraction** : Méthode d'extraction et jointure des données.
5. **Analyse** : Résultats des calculs statistiques (tableaux et commentaires).
6. **Visualisation** : Graphiques avec interprétation.

Conclusion:

Ce TP permet de maîtriser la manipulation de bases de données relationnelles et l'analyse de données avec Python. Les résultats obtenus (tendances des ventes, produits phares) sont utiles pour prendre des décisions commerciales éclairées.