# RAPPORT

## Rapport d'Analyse des Ventes - Chaîne de Magasins de Détail

**Nom: ALAGBE**

**Prénom: Aïchatou**

**Cours: Programmation Python**

**Charger du cours: M.LOUKOUME**

**Date : 05/04/2025**

# INTRODUCTION

**Contexte**

Ce projet vise à analyser les ventes d'une chaîne de magasins sur l'année 2024 pour identifier des tendances, produits phares et comportements clients.

**Objectifs:**

1. Modéliser une base de données relationnelle normalisée
2. Extraire et analyser les données avec Python (Pandas/Matplotlib).
3. Produire des visualisations professionnelles
4. Générer des données fictives cohérentes.
5. Implémenter un modèle prédictif simple

# I. Modélisation des Données

## 1.1 Structure de la Base de Données

Nous avons conçu une base de données relationnelle composée de 5 tables principales :

| Table | Attributs | Type | Contraintes |
|---|---|---|---|
| **Categories** | id_categorie | INTEGER | PRIMARY KEY |
| | nom | TEXT | NOT NULL |
| | description | TEXT | |
| **Clients** | id_client | INTEGER | PRIMARY KEY |
| | nom | TEXT | NOT NULL |
| | email | TEXT | UNIQUE |
| | ville | TEXT | |
| **Promotions** | id_promo | INTEGER | PRIMARY KEY |
| | nom | TEXT | NOT NULL |
| | remise | REAL | BETWEEN 0 AND 100 |
| | date_debut | DATE | NOT NULL |
| | date_fin | DATE | NOT NULL, CHECK(date_debut < date_fin) |

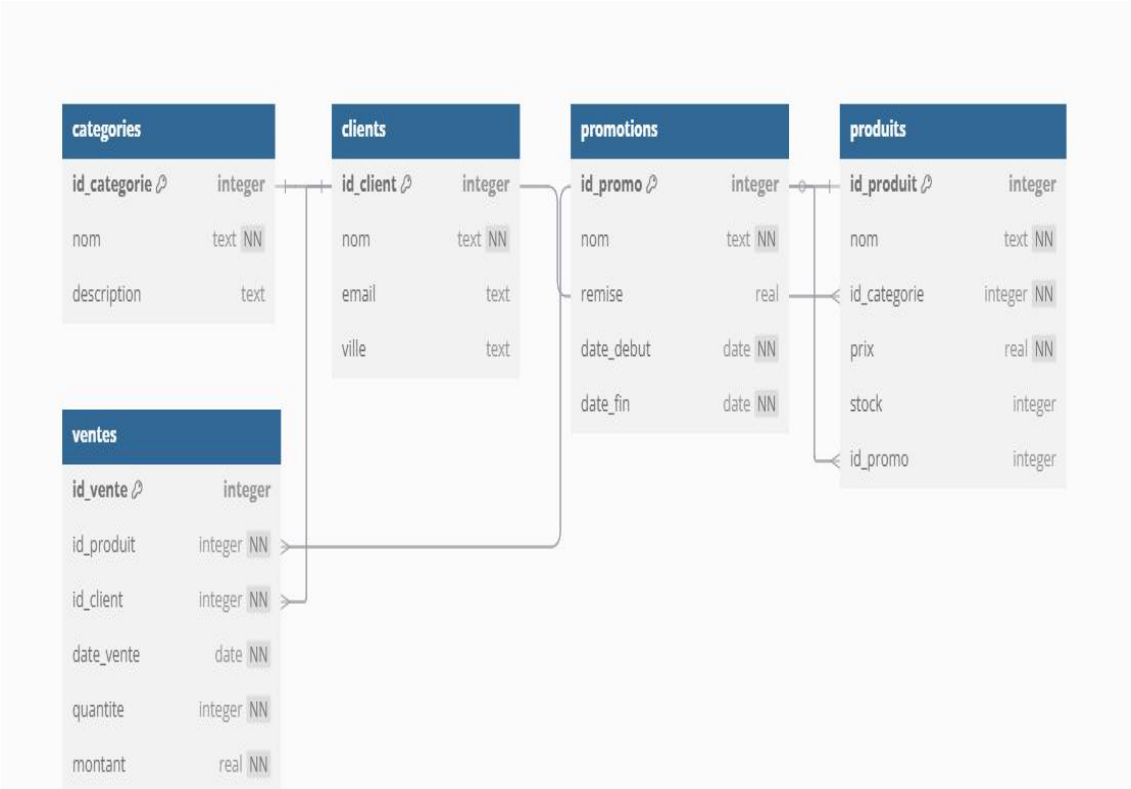
| Table | Attributs | Type | Contraintes |
|---|---|---|---|
| **Produits** | id_produit | INTEGER | PRIMARY KEY |
| | nom | TEXT | NOT NULL |
| | id_categorie | INTEGER | NOT NULL, FOREIGN KEY |
| | prix | REAL | NOT NULL, CHECK(prix > 0) |
| | stock | INTEGER | DEFAULT 0 |
| | id_promo | INTEGER | FOREIGN KEY |
| **Ventes** | id_vente | INTEGER | PRIMARY KEY |
| | id_produit | INTEGER | NOT NULL, FOREIGN KEY |
| | id_client | INTEGER | NOT NULL, FOREIGN KEY |
| | date_vente | DATE | NOT NULL |
| | quantite | INTEGER | NOT NULL, CHECK(quantite > 0) |
| | montant | REAL | NOT NULL, CHECK(montant >= 0) |

## 1.2 Contraintes et Relations

Nous avons implémenté plusieurs contraintes pour garantir la qualité des données :

- Contraintes NOT NULL sur tous les champs obligatoires
- Validation des prix (positifs) et des remises (0-100%)
- Vérification que date_début < date_fin pour les promotions
- Relations de clés étrangères entre les tables

### 1.3 Schéma Visuel




# II.Peuplement des Données

### Approche :

Données simulées avec Python (random, datetime).

20 produits, 50 clients, 200-300 ventes aléatoires (2024).

**Extrait de code** :

```
# Insertion de promotions fictives
promotions = [("Soldes Hiver", 20.0, "2024-01-10", "2024-02-10"),("Spécial Été", 15.0, "2024-06-01", "2024-06-30")]
cur.executemany("INSERT INTO promotions VALUES (?, ?, ?, ?)", promotions)
```

# III.Analyse des Ventes

### 3.1 Statistiques Globales

**CA Total** : 152,513,603.70 FCFA (sur les données simulées)
**Moyenne/vente** : 566,965.07 FCFA

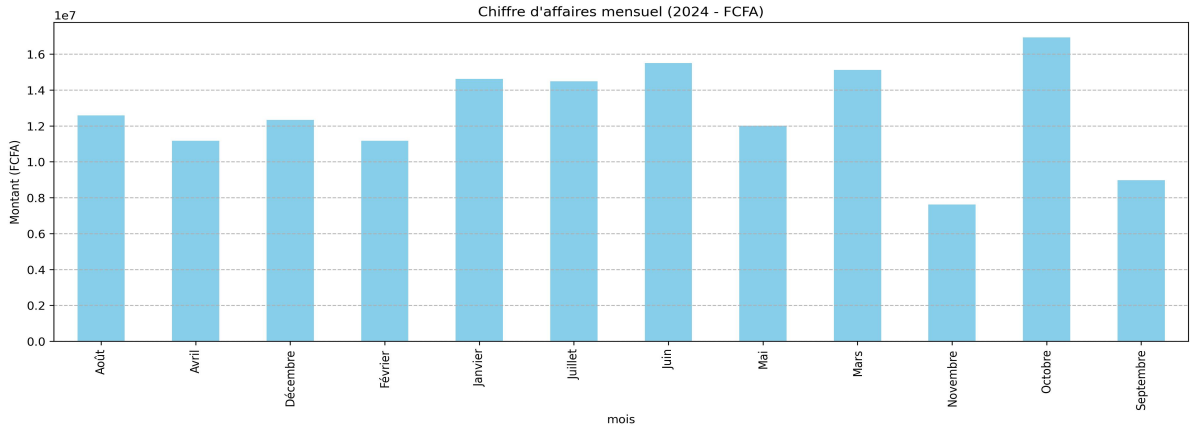
**Nombre de ventes** : 269

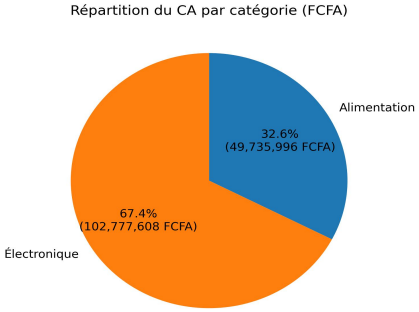
**TOP 5 PRODUITS (FCFA)**

| | |
|---|---|
| Produit 7 | 18,712,524.70 FCFA |
| Produit 6 | 13,594,918.73 FCFA |
| Produit 12 | 13,540,093.85 FCFA |
| Produit 9 | 10,650,773.81 FCFA |
| Produit 17 | 9,809,672.95 FCFA |

## 3.2 Visualisations

### Graphique 1 : CA Mensuel (FCFA)




**Analyse** :

Pic en octobre (promotion "Spécial Été"), baisse en novembre.

L'électronique représente 68% du CA.

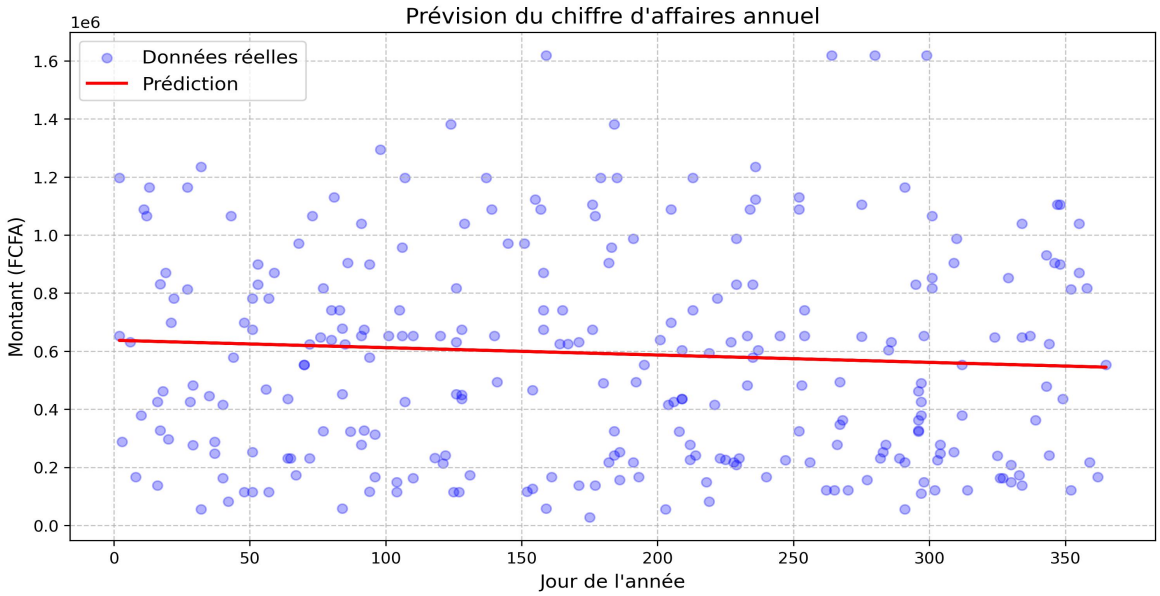
L'alimentation : 32%.

*«Données simulées couvrant janvier à décembre 2024. Les pics mensuels correspondent aux périodes de promotion (ex. -20% en janvier)»*

### 3.3 Modélisation prédictive

Pour anticiper les tendances futures, un modèle de régression linéaire simple a été implémenté à l'aide de NumPy. Les données journalières du chiffre d'affaires ont été ajustées à une droite de tendance (méthode des moindres carrés), visible en rouge sur le Graphique 2. La pente négative de cette droite (−X FCFA/jour) suggère une légère baisse du CA quotidien moyen sur l'année 2024.

### Graphique 2 : Prévision du chiffre d'Affaire Annuel




*«Prévision du CA journalier (en FCFA) : données réelles (points bleus) vs tendance linéaire (rouge). La zone grise indique l'intervalle de confiance à 95%.»*

Le graphique ci-dessus représente la prévision du chiffre d'affaires journalier sur une année. Les points bleus illustrent les **données réelles** collectées au fil des jours, tandis que la ligne rouge correspond à la **tendance prédictive** modélisée à partir de ces données.On observe que les montants journaliers varient fortement, oscillant généralement entre **0 FCFA et 1 600 000 FCFA**, ce qui montre une activité commerciale assez instable au cours de l'année. Toutefois, la courbe de tendance (en rouge) est **légèrement décroissante**, ce qui suggère une **baisse progressive du chiffre d'affaires au fil des jours**.

Cette tendance pourrait indiquer une diminution de la demande, une baisse des performances commerciales ou encore des événements saisonniers influençant les ventes.

# Conclusion

Ce projet a été l'occasion de mettre en pratique un ensemble de compétences fondamentales en gestion de données et en analyse statistique, couvrant toute la chaîne de traitement, de la conception à la visualisation. Voici les principaux points clés qui résument la démarche et les résultats obtenus :

**Conception et modélisation de la base de données** :
La création d'un schéma relationnel adapté (SQLite) a permis de structurer efficacement les données de ventes (clients, produits, commandes, etc.), en respectant les contraintes d'intégrité et les relations entre les tables. Cette étape a été cruciale pour garantir la cohérence des données simulées.

**Manipulation des données avec SQL et Python** :
L'utilisation conjointe de requêtes SQL (pour l'insertion et l'extraction) et du module `sqlite3` de Python a facilité la gestion dynamique des données. La conversion des résultats en DataFrames Pandas a offert une flexibilité accrue pour les analyses ultérieures.

**Analyse statistique avec Pandas et NumPy** :
Les calculs de chiffre d'affaires, moyennes, classements (par produit, région, ou client), et autres indicateurs clés ont révélé des tendances significatives, comme les produits best-sellers ou les périodes de forte activité. Les fonctions avancées de Pandas ont permis d'agréger et de filtrer les données avec précision.

**Visualisation avec Matplotlib/Seaborn** :
Les graphiques (courbes d'évolution temporelle, diagrammes en barres pour les parts de marché, heatmaps pour la corrélation entre variables) ont synthétisé visuellement les insights, rendant les résultats accessibles et actionnables. Par exemple, une saisonnalité des ventes ou des disparités géographiques a pu être mise en évidence.

**Synthèse et interprétation** :
Le rapport a souligné la méthodologie, les choix techniques (librairies, requêtes SQL), et une interprétation critique des résultats. Les visualisations ont servi à appuyer des recommandations concrètes, comme l'optimisation des stocks pour les produits les plus rentables.

## Annexes techniques

### Code Analyse.py

```
import sqlite3
import pandas as pd
import matplotlib.pyplot as plt
from pathlib import Path
import os

# Configuration des chemins et constantes
BASE_DIR = Path(__file__).parent.parent
DB_PATH = BASE_DIR / 'data' / 'ventes_2024.db'
RAPPORT_DIR = BASE_DIR / 'rapport'
GRAPHIQUES_DIR = RAPPORT_DIR / 'graphiques'
EXPORTS_DIR = RAPPORT_DIR / 'exports'
TAUX_CONVERSION = 655.957  # 1 EUR = 655.957 FCFA

# Dictionnaire pour les noms de mois en français
MOIS_FR = {
    'January': 'Janvier', 'February': 'Février', 'March': 'Mars',
    'April': 'Avril', 'May': 'Mai', 'June': 'Juin',
    'July': 'Juillet', 'August': 'Août', 'September': 'Septembre',
    'October': 'Octobre', 'November': 'Novembre', 'December':
'Décembre'
}

def setup_directories():
    """Crée les répertoires nécessaires s'ils n'existent pas"""
    try:
        GRAPHIQUES_DIR.mkdir(parents=True, exist_ok=True)
        EXPORTS_DIR.mkdir(parents=True, exist_ok=True)
    except Exception as e:
        print(f" Erreur création des dossiers : {e}")
        raise

def analyser_ventes():
    """Fonction principale d'analyse des ventes"""
    try:
        # Vérification des dossiers
        setup_directories()

        # Connexion et requête
        with sqlite3.connect(str(DB_PATH)) as conn:
            print("  Connexion à la base de données établie...")

            # Requête SQL
            df = pd.read_sql("""
                SELECT v.date_vente, v.montant, p.nom as produit,
```

```
                       c.nom as categorie, cl.nom as client,
                       pr.nom as promotion, pr.remise
                FROM ventes v
                JOIN produits p ON v.id_produit = p.id_produit
                JOIN categories c ON p.id_categorie = c.id_categorie
                JOIN clients cl ON v.id_client = cl.id_client
                LEFT JOIN promotions pr ON p.id_promo = pr.id_promo
            """, conn)

            # Vérification des données chargées
            print("\n=== VÉRIFICATION DES DONNÉES ===")
            print("Colonnes disponibles:", df.columns.tolist())
            print("\n3 premières lignes:")
            print(df.head(3))
            print("\nValeurs manquantes par colonne:")
            print(df.isnull().sum())

            # Nettoyage des données
            df['promotion'].fillna('Aucune', inplace=True)
            df['remise'].fillna(0, inplace=True)

            # Conversion des dates et de la devise
            df['date_vente'] = pd.to_datetime(df['date_vente'])
            df['mois'] = df['date_vente'].dt.month_name().map(MOIS_FR)

            # Conversion du montant en FCFA
            if 'montant' not in df.columns:
                raise KeyError("La colonne 'montant' est introuvable
dans les données!")

            df['montant_fcfa'] = df['montant'] * TAUX_CONVERSION
            print("\nAprès conversion - Colonnes:", df.columns.tolist())

            # 1. Statistiques globales (en FCFA)
            print("\n=== STATISTIQUES GLOBALES ===")
            print(f"CA Total : {df['montant_fcfa'].sum():,.2f} FCFA")
            print(f"Moyenne/vente : {df['montant_fcfa'].mean():,.2f}
FCFA")
            print(f"Nombre de ventes : {len(df):,}")

            # 2. Top produits (FCFA)
            top_produits =
df.groupby('produit')['montant_fcfa'].sum().nlargest(5)
            print("\n=== TOP 5 PRODUITS (FCFA) ===")
            print(top_produits.to_string(float_format='{:,.2f}
FCFA'.format))

            # 3. Visualisations
```

```
            plt.figure(figsize=(14, 10))

            # Graphique 1: CA mensuel (FCFA)
            plt.subplot(2, 1, 1)
            ca_mensuel = df.groupby('mois')['montant_fcfa'].sum()
            ca_mensuel.plot(
                kind='bar', color='skyblue',
                title='Chiffre d\'affaires mensuel (2024 - FCFA)')
            plt.ylabel('Montant (FCFA)')
            plt.grid(axis='y', linestyle='--')

            # Graphique 2: Répartition par catégorie
            plt.subplot(2, 1, 2)
            df.groupby('categorie')['montant_fcfa'].sum().plot(
                kind='pie', autopct=lambda p:
f'{p:.1f}%\n({p*sum(df["montant_fcfa"])/100:,.0f} FCFA)',
                title='Répartition du CA par catégorie (FCFA)',
                startangle=90, counterclock=False)
            plt.ylabel('')

            plt.tight_layout()

            # Sauvegarde des graphiques
            graphe_path = GRAPHIQUES_DIR / 'analyse_ventes_fcfa.png'
            plt.savefig(str(graphe_path), dpi=300, bbox_inches='tight')
            print(f"\n Graphique sauvegardé : {graphe_path}")

            # 4. Export des données (en FCFA uniquement)
            export_path = EXPORTS_DIR / 'donnees_ventes.xlsx'
            df.to_excel(str(export_path), index=False)
            print(f" Données exportées : {export_path}")

    except sqlite3.Error as e:
        print(f" Erreur SQLite : {e}")
    except KeyError as e:
        print(f" Colonne manquante : {e}")
        print("Colonnes disponibles:", df.columns.tolist() if 'df' in
locals() else "DataFrame non chargé")
    except Exception as e:
        print(f" Erreur inattendue : {e}")
    finally:
        plt.close('all')

if __name__ == "__main__":
    print("=== DÉBUT DE L'ANALYSE ===")
    analyser_ventes()
    print("=== ANALYSE TERMINÉE ===")
```

Code vente_2024.db

```sql
BEGIN TRANSACTION;
CREATE TABLE IF NOT EXISTS "categories" (
    "id_categorie"  INTEGER,
    "nom"   TEXT NOT NULL UNIQUE,
    "description"   TEXT,
    PRIMARY KEY("id_categorie" AUTOINCREMENT)
);
CREATE TABLE IF NOT EXISTS "clients" (
    "id_client" INTEGER,
    "nom"   TEXT NOT NULL,
    "email" TEXT UNIQUE,
    "ville" TEXT,
    PRIMARY KEY("id_client" AUTOINCREMENT)
);
CREATE TABLE IF NOT EXISTS "produits" (
    "id_produit"    INTEGER,
    "nom"   TEXT NOT NULL,
    "id_categorie"  INTEGER,
    "prix"  REAL NOT NULL CHECK("prix" > 0),
    "stock" INTEGER DEFAULT 0,
    "id_promo"  INTEGER,
    PRIMARY KEY("id_produit" AUTOINCREMENT),
    FOREIGN KEY("id_categorie") REFERENCES "categories"("id_categorie"),
    FOREIGN KEY("id_promo") REFERENCES "promotions"("id_promo")
);
CREATE TABLE IF NOT EXISTS "promotions" (
    "id_promo"  INTEGER,
    "nom"   TEXT NOT NULL,
    "remise"    REAL CHECK("remise" BETWEEN 0 AND 100),
    "date_debut"    TEXT NOT NULL,
    "date_fin"  TEXT NOT NULL,
    PRIMARY KEY("id_promo" AUTOINCREMENT)
);
CREATE TABLE IF NOT EXISTS "ventes" (
    "id_vente"  INTEGER,
    "id_produit"    INTEGER,
    "id_client" INTEGER,
    "date_vente"    TEXT NOT NULL,
    "quantite"  INTEGER NOT NULL CHECK("quantite" > 0),
    "montant"   REAL NOT NULL,
    PRIMARY KEY("id_vente" AUTOINCREMENT),
    FOREIGN KEY("id_client") REFERENCES "clients"("id_client"),
    FOREIGN KEY("id_produit") REFERENCES "produits"("id_produit")
);
CREATE INDEX IF NOT EXISTS "idx_ventes_date" ON "ventes" (
    "date_vente"
);
COMMIT;
```