Projet Python
pour la Data Science

Université Hassan II de Casablanca

Ecole Normale Supérieure de l'Enseignement Technique Mohammedia

Encadrant : Pr. Mohammed Qbadou

1. Introduction

Ce projet a pour objectif de pratiquer l'analyse de données sous Python à travers un fichier Excel intitulé 'sales.xlsx'. Il s'agit de nettoyer les données, créer des indicateurs de performance clés (KPI), et représenter les résultats de manière visuelle afin d'en extraire des tendances et des conclusions.

2. Présentation des données

Le jeu de données contient 7 991 lignes et 12 colonnes. Il inclut des informations sur les commandes, les clients, les régions, les produits, les canaux de vente, ainsi que les quantités commandées, les prix, les coûts, etc.

- ### Objectif :

  Lire les données de sales.xlsx, comprendre leur structure, corriger les problèmes éventuels (valeurs manquantes, doublons, types).

- ### Structure du fichier

  - **Nombre de lignes** : 7 991
  - **Nombre de colonnes** : 12
  - **Aucun doublon** ni valeur manquante

Colonnes disponibles :

Colonne	Type	Description (interprétée)
OrderNumber	object	Référence de la commande
OrderDate	datetime64	Date de commande
Ship Date	datetime64	Date d'expédition
Customer Name Index	int64	Identifiant du client
Channel	object	Canal de vente (Wholesale, Distributor, etc.)
Currency Code	object	Devise (USD, NZD, ...)
Warehouse Code	object	Code entrepôt
Delivery Region Index	int64	Région de livraison
Product Description Index	int64	Identifiant du produit
Order Quantity	int64	Quantité commandée
Unit Selling Price	float64	Prix unitaire de vente
Unit Cost	float64	Coût unitaire

- **Code de départ :**

Voici un script Python de base pour cette étape :

```python
import pandas as pd

# Chargement du fichier Excel
df = pd.read_excel("sales.xlsx")

# Aperçu du DataFrame
print("Aperçu des données :")
print(df.head())

# Dimensions du DataFrame
print("\nDimensions :", df.shape)

# Vérifier les types de données
print("\nTypes de données :")
print(df.dtypes)

# Chercher les valeurs manquantes
print("\nValeurs manquantes :")
print(df.isnull().sum())

# Chercher les doublons
print("\nDoublons :", df.duplicated().sum())
```

- **Résultat :**

```
···  Aperçu des données :
      OrderNumber  OrderDate  Ship Date  Customer Name Index      Channel  \
0   SO - 000225 2017-01-01 2017-01-13                   28    Wholesale
1  SO - 0003378 2017-01-01 2017-01-06                    7  Distributor
2  SO - 0003901 2017-01-01 2017-01-05                   12    Wholesale
3  SO - 0005126 2017-01-01 2017-01-17                    5    Wholesale
4  SO - 0005614 2017-01-01 2017-01-07                   27       Export

  Currency Code Warehouse Code  Delivery Region Index  \
0           NZD         AXW291                     71
1           NZD         AXW291                     54
2           NZD         AXW291                     58
3           USD         AXW291                     29
4           NZD         AXW291                     31

   Product Description Index  Order Quantity  Unit Selling Price  Unit Cost
0                         11               6              2499.1   1824.343
1                          7              11              2351.7   1269.918
2                         13               5              1728.6   1019.874
3                          7               6               978.2    684.740
4                          6               7              2338.3   1028.852

Dimensions : (7991, 12)

Types de données :
...
Unit Cost                   0
dtype: int64

Doublons : 0
```

3. Nettoyage et transformation

Les données ne contenaient ni doublons, ni valeurs manquantes. Des colonnes calculées ont été ajoutées : Sales (chiffre d'affaires), Cost (coût), et Profit (bénéfice).

```python
import pandas as pd

# Chargement du fichier Excel
df = pd.read_excel("sales.xlsx")

# Créer les colonnes calculées
df["Sales"] = df["Order Quantity"] * df["Unit Selling Price"]
df["Cost"] = df["Order Quantity"] * df["Unit Cost"]
df["Profit"] = df["Sales"] - df["Cost"]

# Total Sales
total_sales = df["Sales"].sum()

# Total Profit
total_profit = df["Profit"].sum()

# Total Order Quantity
total_orders = df["Order Quantity"].sum()

# Profit Margin %
profit_margin = total_profit / total_sales if total_sales != 0 else 0

# Affichage
print("Indicateurs globaux :")
print(f"Total Sales        : {total_sales:,.2f}")
print(f"Total Profit       : {total_profit:,.2f}")
print(f"Total Orders       : {total_orders:,}")
print(f"Profit Margin (%)  : {profit_margin*100:.2f}%")
```

4. Création de la table de dates

Une table de dates a été générée à partir de la colonne OrderDate, contenant les colonnes Year, Month, Quarter, Day, etc. Elle permet d'effectuer des comparaisons temporelles.

```python
date_range = pd.date_range(start=df["OrderDate"].min(), end=df["OrderDate"].max(), freq='D')
date_table = pd.DataFrame({"Date": date_range})

date_table["Year"] = date_table["Date"].dt.year
date_table["Month"] = date_table["Date"].dt.month
date_table["Month Name"] = date_table["Date"].dt.strftime('%B')
date_table["Quarter"] = date_table["Date"].dt.quarter
date_table["Day"] = date_table["Date"].dt.day
date_table["Day Name"] = date_table["Date"].dt.strftime('%A')
date_table["Week"] = date_table["Date"].dt.isocalendar().week

# Renommer la colonne "Date" de la table de dates
date_table_renamed = date_table.rename(columns={"Date": "OrderDate"})

# Fusion des deux DataFrames sur la colonne OrderDate
df_merged = pd.merge(df, date_table_renamed, on="OrderDate", how="left")

# Afficher un aperçu du résultat
print(df_merged[["OrderDate", "Year", "Month", "Month Name", "Quarter", "Day Name"]].head())
```

5. Calculs des indicateurs de performance (KPIs)

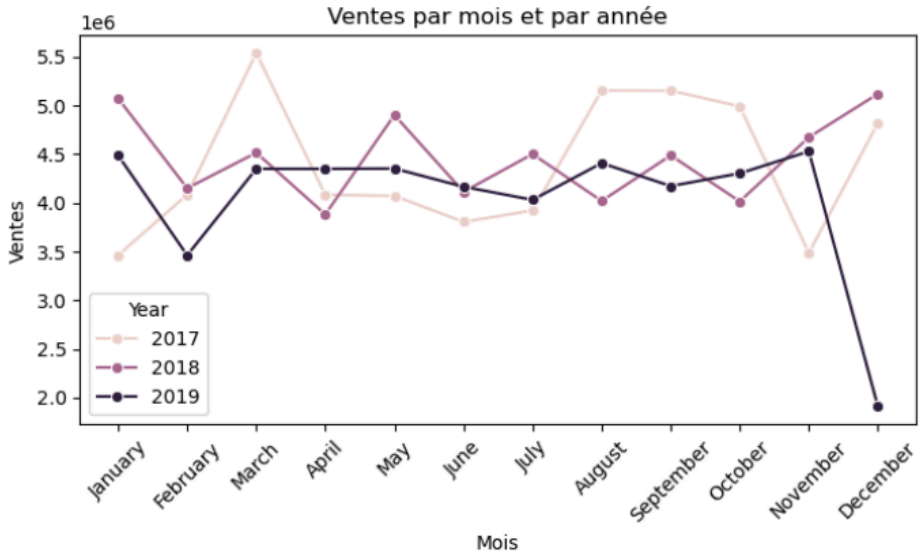
- Total Sales, Profit et Quantité commandée
- Comparaisons YOY (année précédente)
- Marge bénéficiaire et variation %

```
···   Indicateurs globaux :
      Total Sales        : 154,573,140.60
      Total Profit       : 57,789,142.91
      Total Orders       : 67,579
      Profit Margin (%)  : 37.39%

      Comparaison avec l'année précédente :
         Year  Total Sales  Total Profit  Total Order Quantity  Total Sales PY  \
      0  2017   52580534.7   19677772.16                 23052             NaN
      1  2018   53463661.7   19789189.55                 23153      52580534.7
      2  2019   48528944.2   18322181.20                 21374      53463661.7

         Sales Var  Sales Var %  Total Profit PY  Profit Var  Profit Var %  \
      0        NaN          NaN              NaN         NaN           NaN
      1   883127.0         1.68      19677772.16   111417.38          0.57
      2 -4934717.5        -9.23      19789189.55 -1467008.35         -7.41

         Order Qty PY  Order Qty Var  Order Qty Var %
      0           NaN            NaN              NaN
      1       23052.0          101.0             0.44
      2       23153.0        -1779.0            -7.68
```

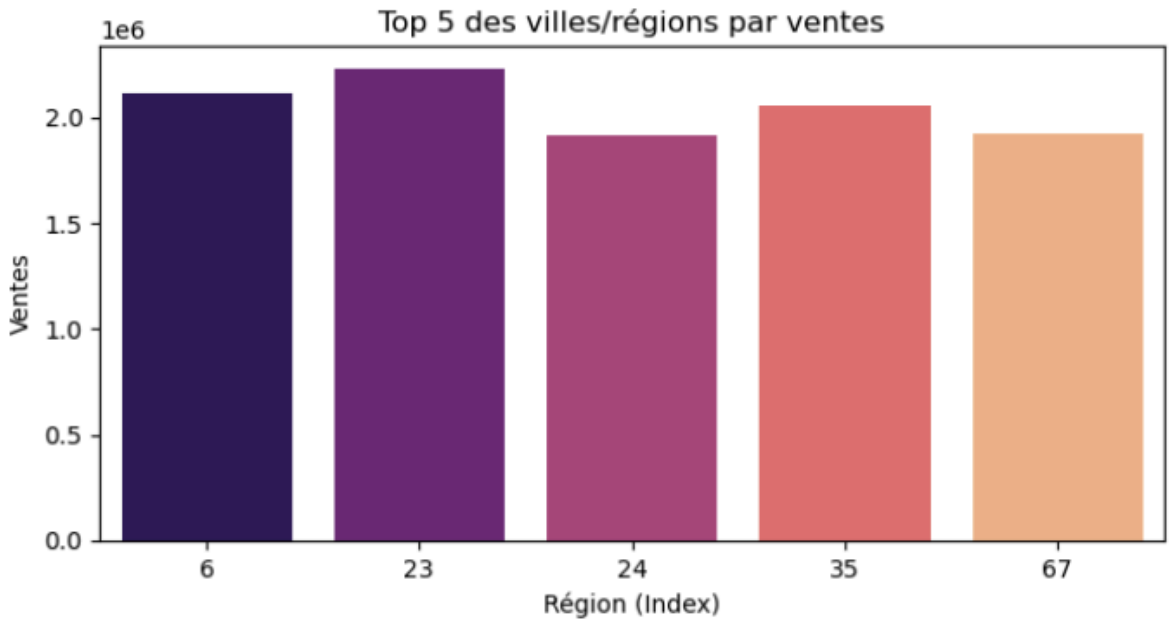
6. Visualisations

Les visualisations ont permis de mieux comprendre les données :

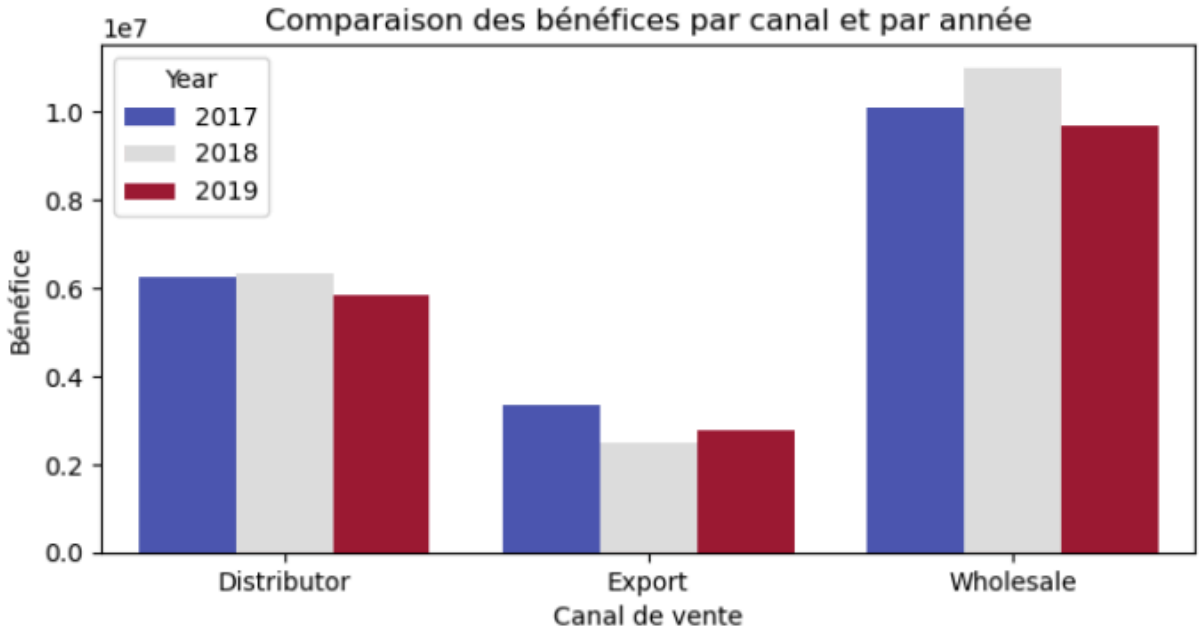
- Ventes par produit et par mois :



- Top 5 des régions et clients :



- Comparaisons des bénéfices par canal :



- Analyse des 5 derniers clients :

7. Conclusion

Ce projet a constitué une immersion concrète dans le cycle complet d'analyse de données, depuis le nettoyage et l'enrichissement des données brutes jusqu'à la génération d'insights stratégiques. À travers l'exploitation d'outils tels que Pandas, Seaborn et Matplotlib, des compétences clés en manipulation de données, visualisation dynamique et calcul de métriques (KPIs) ont été renforcées. Les analyses réalisées – incluant l'évolution temporelle des ventes, la rentabilité par canal, et la performance des clients – ont mis en lumière des tendances exploitables pour optimiser les stratégies commerciales. Ces résultats démontrent l'importance d'une approche data-driven pour prioriser les actions business et allouer les ressources efficacement.

Étudiant : Ayoub El Hallaoui

Date de remise : 20 avril 2025