**24017058 IAI3D**

**Projet pandas**

**MOHAME BAZBER**

🔸 **INTRODUCTION**

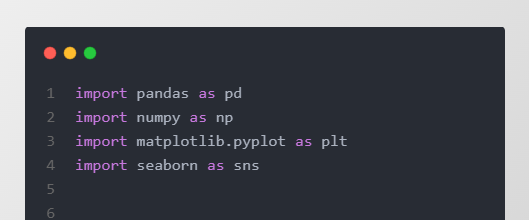
Le présent travail a pour objectif d’explorer, analyser et interpréter les données contenues dans le fichier « Données sur les ventes.csv ». Ce fichier regroupe des informations détaillées sur les ventes réalisées par différentes entités ou employés, notamment les noms des commerciaux, leurs départements, les chiffres de ventes, les dates et les régions concernées.

Afin d’effectuer une analyse efficace, nous avons utilisé la bibliothèque Python Pandas, qui constitue un outil puissant pour la manipulation et l’analyse de données tabulaires. Grâce à Pandas, il est possible de charger rapidement le fichier CSV, de traiter les données manquantes, de filtrer ou regrouper les informations pertinentes, et de générer des statistiques descriptives comme la somme des ventes totales, les ventes moyennes par département, ou encore l’identification des performances minimales et maximales. En complément, des outils de visualisation tels que Matplotlib et Seaborn peuvent être utilisés pour représenter graphiquement les tendances observées.

Cette analyse vise à fournir des indicateurs clés de performance (KPI) permettant d'évaluer l’efficacité commerciale et d’orienter les futures décisions stratégiques de l’entreprise.

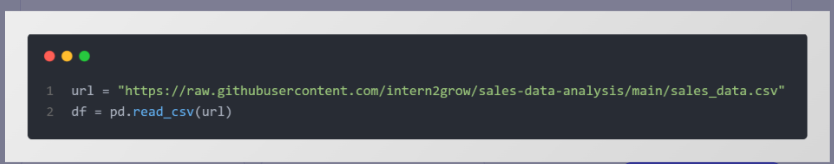
**Importation des bibliothèques**

Cette section est essentielle car elle prépare l’environnement Python à effectuer différentes tâches en important des bibliothèques spécialisées. Chacune joue un rôle fondamental dans le traitement, l’analyse et la visualisation des données :



1. **pandas (importé sous le nom pd)**  
   Cette bibliothèque est incontournable pour la manipulation de données sous forme tabulaire, c’est-à-dire des tableaux avec lignes et colonnes (semblables aux feuilles Excel). Elle fournit la structure de données appelée DataFrame, qui permet de facilement importer, nettoyer, trier, filtrer et analyser des ensembles de données. C’est le cœur du traitement de données dans la plupart des projets de data science.
2. **numpy (importé sous le nom np)**  
   numpy est utilisé pour effectuer des opérations mathématiques et numériques sur des tableaux (arrays). C’est une bibliothèque très performante pour le calcul scientifique. Dans ce script, elle peut être utilisée pour faire des statistiques comme la moyenne, la variance, l’écart-type, ou pour effectuer des opérations mathématiques vectorisées plus rapidement qu’avec des boucles classiques.
3. **matplotlib.pyplot (importé sous le nom plt)**  
   matplotlib est une bibliothèque de visualisation de données. Le module pyplot permet de tracer différents types de graphiques (histogrammes, courbes, diagrammes en barres, etc.). Il fournit un contrôle très précis sur l’apparence des graphiques, comme les couleurs, les étiquettes, les titres, les grilles, etc.
4. **seaborn (importé sous le nom sns)**  
   seaborn est construit sur matplotlib, mais rend la création de graphiques statistiques beaucoup plus simple et esthétique. Elle est souvent utilisée pour représenter les distributions de données, les relations entre variables (corrélations, régressions), ou pour créer des visualisations complexes avec peu de code. Elle est particulièrement utile pour explorer rapidement les données et découvrir des tendances visuelles.

**Chargement des données**

****

Le fichier CSV contenant les données de ventes est chargé depuis GitHub dans une DataFrame appelée df. Cela permet de manipuler facilement les données du fichier dans le reste du code.

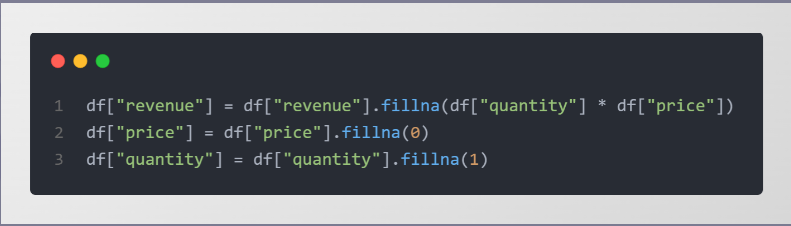
**Exploration et Nettoyage des Données**

Cette phase est cruciale dans tout projet d’analyse de données, car elle permet de comprendre la structure du jeu de données et de le préparer pour les étapes suivantes (visualisation, modélisation, etc.). Voici ce qui est effectué dans ce bloc :



1. **Aperçu du contenu des données :**  
   On utilise la méthode head() pour afficher les premières lignes du DataFrame. Cela donne un aperçu rapide de la forme des données, des types de colonnes, et permet de repérer d’éventuelles anomalies ou valeurs étranges. C’est une étape de diagnostic visuel.
2. **Informations générales avec info() :**  
   Cette méthode fournit un résumé très utile : elle affiche le nombre total de lignes, les noms des colonnes, leur type (entier, flottant, objet, etc.), et surtout le nombre de valeurs non nulles (non manquantes) dans chaque colonne. Cela permet de détecter immédiatement s’il y a des données manquantes à traiter.
3. **Comptage précis des valeurs manquantes :**  
   Avec isnull().sum(), on calcule combien de valeurs sont absentes (nulles) dans chaque colonne. Ce diagnostic est fondamental, car les valeurs manquantes peuvent biaiser les analyses ou bloquer certaines opérations statistiques ou algorithmiques.

* **Traitement des valeurs manquantes :**  
  Une fois que l'on a identifié les lignes où la colonne "Revenu" est vide, on applique une méthode de remplissage (imputation) basée sur une logique métier .  
  le revenu est calculé comme la multiplication de la Quantité vendue par le Prix unitaire.  
  Ainsi, pour chaque ligne où "Revenue" est manquant, on applique automatiquement la formule :



Cela permet non seulement de conserver les lignes précieuses du jeu de données (au lieu de les supprimer), mais aussi de garantir une cohérence logique dans les informations.

**Manipulation des données**

****

Ici, on filtre les ventes pour ne conserver que celles de la catégorie "Électronique", on trie le DataFrame par revenu décroissant pour identifier les meilleures ventes, et on ajoute une colonne appelée "Revenu\_TVA" qui représente le revenu augmenté de la TVA (20 %), ce qui peut être utile pour des analyses financières ou fiscales.

1. Filtrage d’un sous-ensemble de données : electronique = df[df["Catégorie"] == "Électronique"]

Cette ligne utilise le filtrage conditionnel avec Pandas. Elle permet d’extraire uniquement les lignes où la colonne "Catégorie" a pour valeur "Électronique". Cela nous permet de concentrer notre analyse sur un type précis de produit. Le résultat est un nouveau DataFrame appelé electronique contenant uniquement les ventes de cette catégorie. C’est très utile pour faire des comparaisons entre catégories ou analyser les performances spécifiques d’un segment.

1. Tri des données : df\_sorted = df.sort\_values(by="Revenu", ascending=False)

Ici, nous utilisons la méthode sort\_values pour trier les lignes du DataFrame en fonction des valeurs de la colonne "Revenu", par ordre décroissant (ascending=False). Cela nous permet d’identifier rapidement les ventes les plus rentables en haut du tableau. Le tri est essentiel pour extraire les top performances, établir des classements, ou visualiser les extrêmes (meilleures et pires ventes).

1. Création d’une nouvelle colonne : df["Revenu\_TVA"] = df["Revenu"] \* 1.20

Cette ligne ajoute une nouvelle colonne au DataFrame appelée "Revenu\_TVA", qui correspond au revenu TTC (toutes taxes comprises), en considérant une taxe de 20 %. Cela se fait en multipliant chaque valeur de la colonne "Revenu" par 1.20. Ce genre d’enrichissement de données est très courant lorsqu’on prépare un rapport destiné à la comptabilité, à la fiscalité ou à des études de rentabilité.

**Analyse statistique complète**

Cette section permet de faire une analyse statistique complète de la colonne "Revenu" : on obtient les statistiques descriptives de base (moyenne, médiane, minimum, maximum, écart-type, variance) pour comprendre la répartition des revenus. Ensuite, on agrège les revenus par catégorie et par vendeur pour comparer les performances de vente selon ces deux dimensions.

 df["Revenu"].describe()  
→ Cette méthode retourne un résumé statistique global de la colonne "Revenu", incluant :

* le nombre total d’observations (count),
* la moyenne (mean),
* l’écart-type (std) qui mesure la dispersion autour de la moyenne,
* les valeurs minimales (min) et maximales (max),
* ainsi que les quartiles (25%, 50%, 75%) dont la médiane (50%) qui donne une idée de la distribution centrale des revenus.  
  C’est une méthode incontournable pour comprendre rapidement la tendance générale des données.

1 df["Revenu"].var()  
→ Cette méthode calcule la variance, c’est-à-dire la moyenne des carrés des écarts à la moyenne. Elle permet de quantifier la variabilité des revenus.

2 df.groupby("Catégorie")["Revenu"].sum()  
→ Cette instruction regroupe les données par type de produit (colonne "Catégorie") et calcule la somme totale des revenus pour chaque catégorie. Cela permet d’identifier les catégories les plus rentables et d’analyser la contribution relative de chaque type de produit.

3 df.groupby("Catégorie")["Revenu"].mean()  
→ Ici, on calcule la moyenne des revenus par catégorie, ce qui aide à savoir quelles catégories rapportent le plus en moyenne par vente.

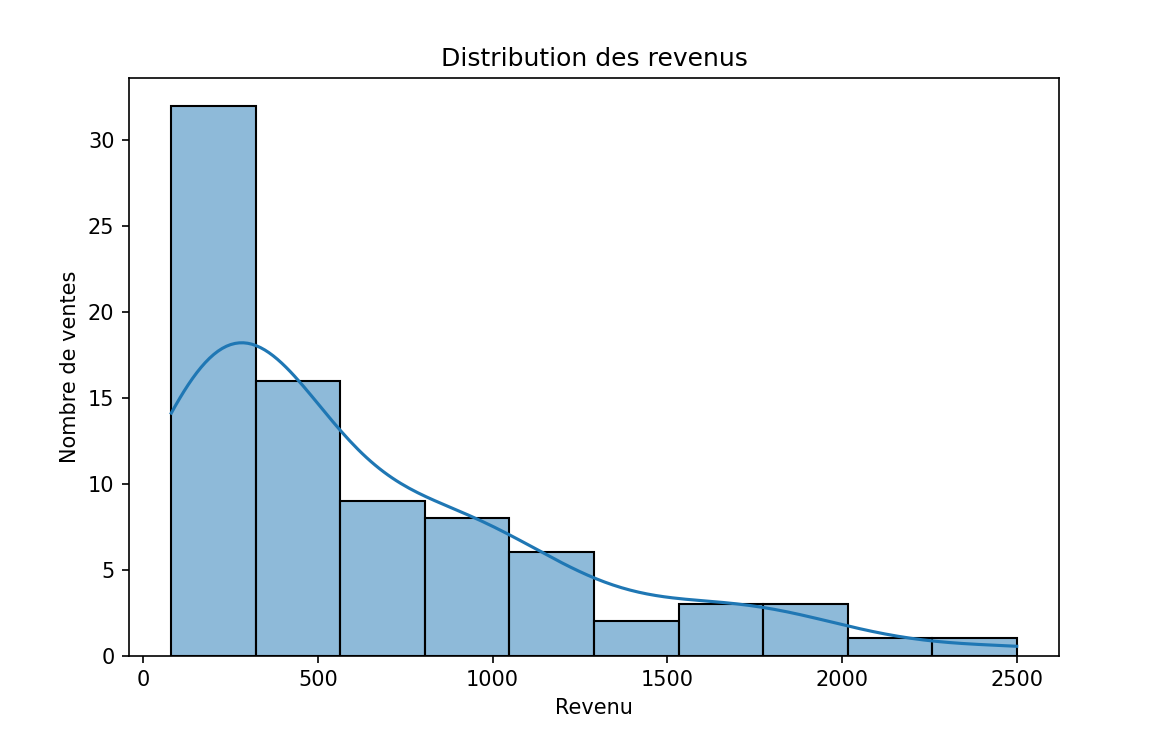
4 df.groupby("Vendeur")["Revenu"].sum()  
→ Cette commande permet de regrouper les revenus par vendeur et d’en calculer le total. Elle est utile pour comparer les performances commerciales individuelles.

5 df.groupby("Vendeur")["Revenu"].mean()  
→ Cette méthode calcule le revenu moyen généré par chaque vendeur, ce qui donne une vision plus nuancée que le revenu total (par exemple, un vendeur peut avoir peu de ventes mais très rentables).

**Visualisation**

cette dernière partie crée des visualisations pour mieux interpréter les résultats. Un histogramme montre la distribution globale des revenus, un boxplot permet de visualiser la répartition des revenus par catégorie (avec médiane, quartiles, et valeurs extrêmes), et deux barplots présentent les revenus totaux par catégorie et par vendeur de manière claire et visuelle, facilitant la comparaison entre les groupes.

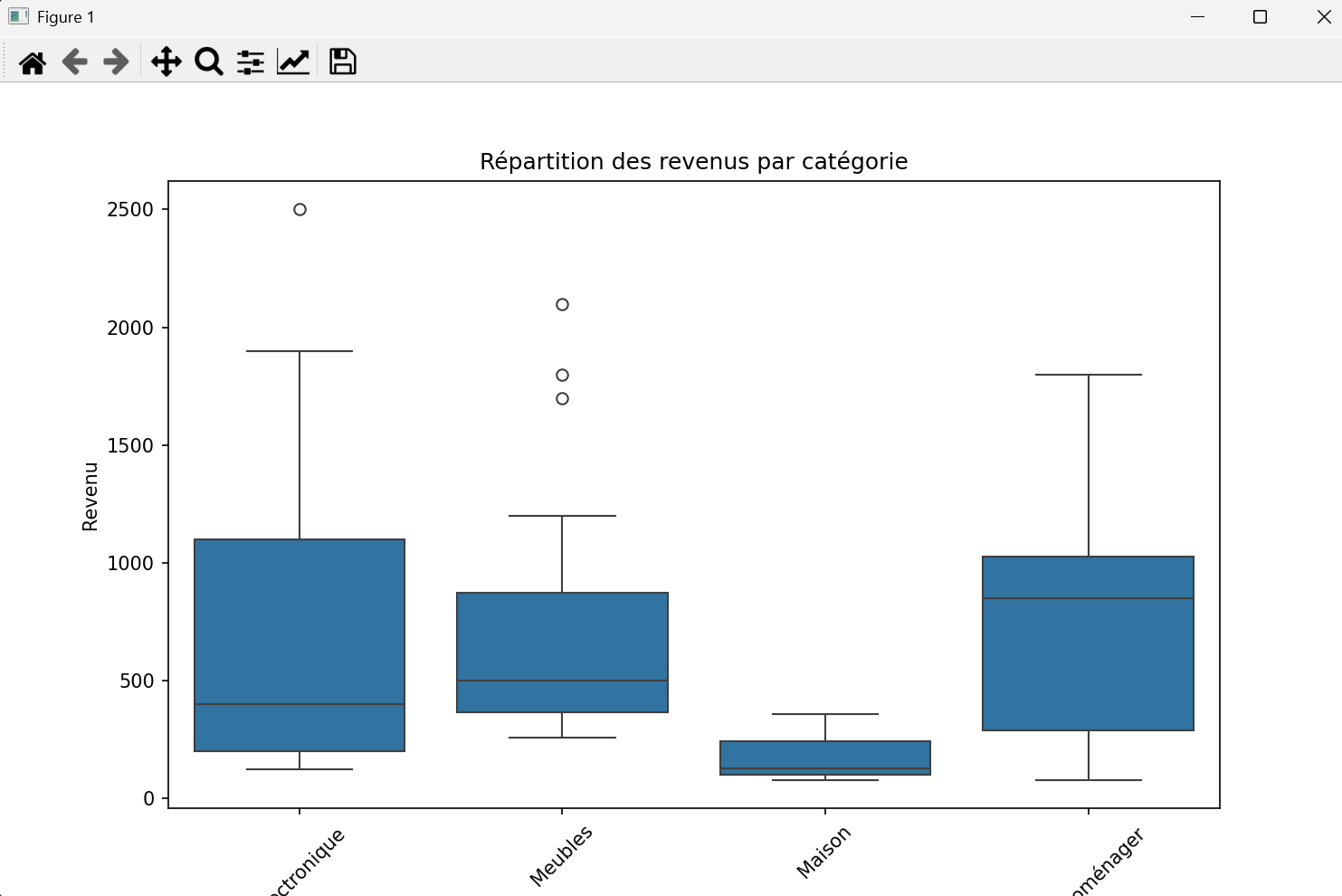




1. Histogramme avec courbe KDE (Kernel Density Estimate)

* Objectif : Visualiser la distribution des revenus de toutes les ventes.
* Ce graphique nous montre comment les revenus sont répartis : s’il y a des pics (revenus fréquents), une concentration autour d’une moyenne ou une asymétrie (revenus élevés ou très faibles).
* La courbe KDE (en bleu) représente une estimation lissée de la distribution pour mieux voir la tendance générale.

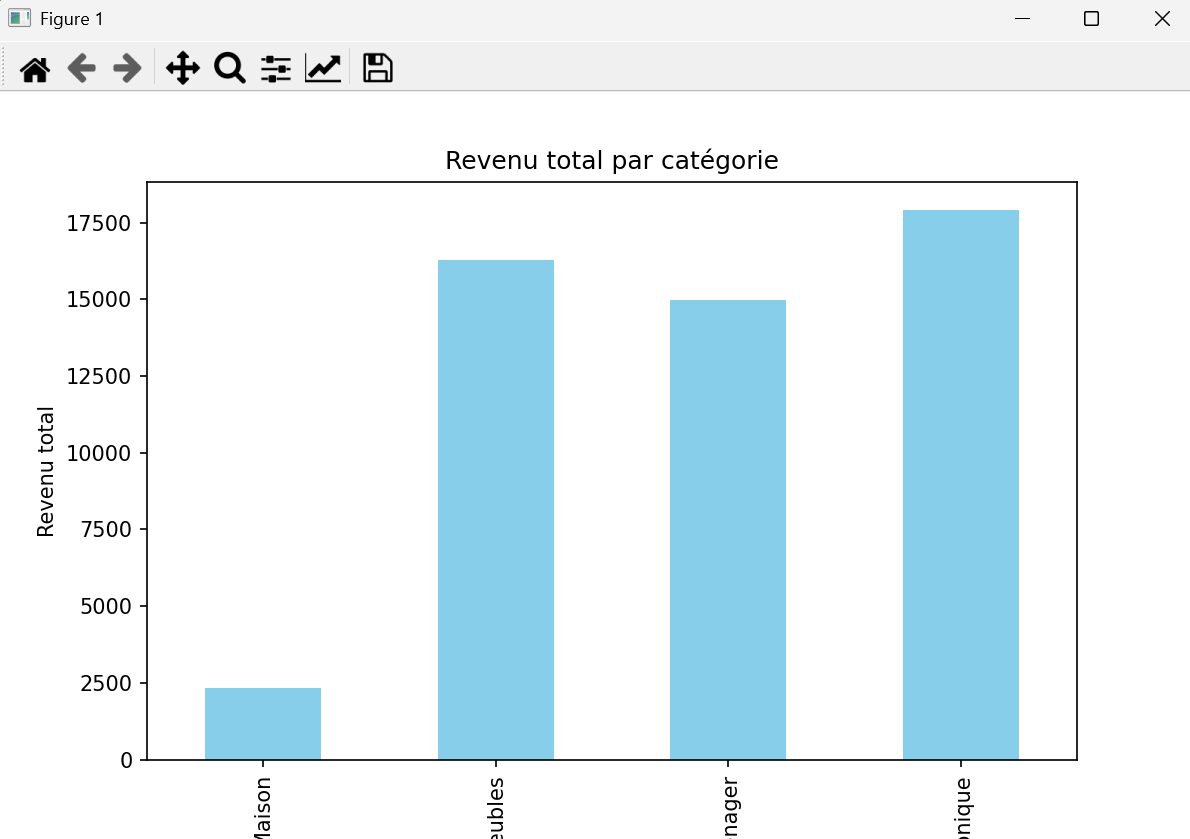




2. Boxplot (boîte à moustaches) des revenus par catégorie

* Objectif : Comparer la répartition des revenus entre les différentes catégories de produits.
* Chaque boîte montre :
  + La médiane (ligne au milieu de la boîte),
  + Le 1er et 3e quartile (les bords de la boîte),
  + Les « moustaches » (valeurs extrêmes non considérées comme aberrantes),
  + Les points isolés (valeurs atypiques).
* Utile pour repérer les catégories avec des revenus plus élevés, plus dispersés, ou des anomalies.





3. Diagramme en barres du revenu total par catégorie

* Objectif : Visualiser le revenu cumulé généré par chaque catégorie de produit.
* On voit ici directement quelles catégories rapportent le plus (ou le moins) en termes de chiffre d’affaires total.
* C’est un graphique simple mais très utile pour identifier les catégories les plus rentables dans l’ensemble des ventes.

**Conclusion** :

En conclusion, ce projet démontre l’importance de l’utilisation de **Pandas** pour l'exploration, la manipulation et l’analyse des données de vente. Grâce à des techniques telles que le filtrage, le tri, et la création de nouvelles colonnes, nous avons pu préparer et nettoyer les données efficacement. Les visualisations, telles que l'histogramme, le boxplot et le diagramme en barres, ont permis de mieux comprendre la distribution des revenus et de comparer les performances des différentes catégories de produits. Ces outils d’analyse offrent des perspectives précieuses pour optimiser les stratégies de vente et améliorer la prise de décision commerciale.