** Université de Carthage **

**Faculté des Sciences Economiques et de Gestion de Nabeul**

**Spécialité : Ingénierie Economique et Financière**

**Projet Python :**

**Analyse et prévision de l'évolution économique tunisienne à partir des données du Global Economic Monitor**

**Elaboré : Amani krout & Nawras Trabelsi**

**Encadrante : Mrs Fadoua Bouafif**

**Année universitaire :**

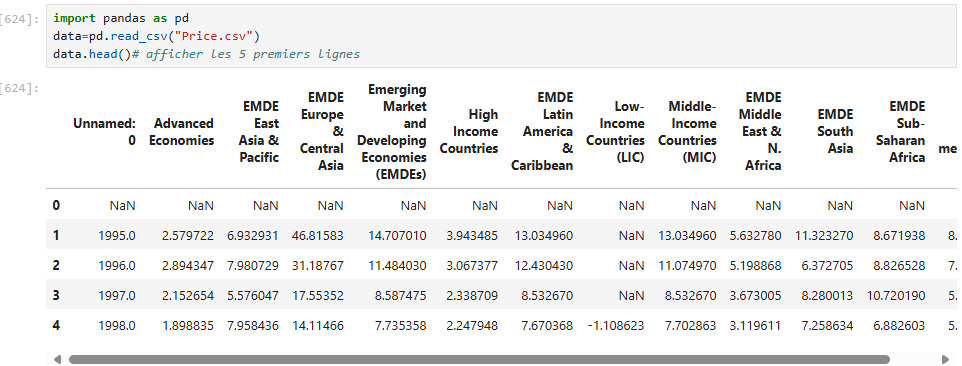
2024/2025

Notre projet d'aide à la décision porte sur l'analyse des indicateurs économiques de différents groupes de pays, tels que définis par leurs caractéristiques économiques et géographiques.

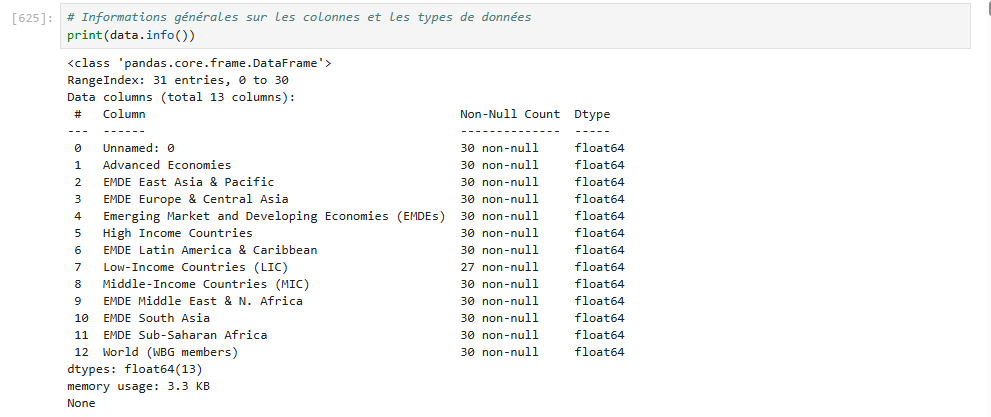
L'indicateur utilisé dans le fichier est l'indice des prix à la consommation (IPC), qui mesure l'évolution des prix des biens et services consommés par les ménages. Cet indicateur permet d'analyser les tendances inflationnistes par région et catégorie économique, offrant une vue d'ensemble des variations du coût de la vie sur plusieurs années.

Le fichier de données utilisé contient 13 colonnes, représentant des ensembles spécifiques de pays :

* **Advanced Economies** regroupe les économies avancées comme les États-Unis, le Japon, ou encore les pays de l'Union européenne.
* **EMDE East Asia & Pacific** couvre les économies émergentes et en développement d'Asie de l'Est et du Pacifique, incluant des pays comme la Chine, l'Indonésie et la Thaïlande.
* **EMDE Europe & Central Asia** rassemble des pays comme la Russie, la Turquie et le Kazakhstan.
* **Emerging Market and Developing Economies (EMDEs)** inclut un large éventail de pays en développement et à marché émergent, tels que le Brésil, l'Inde et l'Afrique du Sud.
* **High Income Countries** se concentre sur les pays à revenu élevé, comme Singapour et les Émirats arabes unis.
* **EMDE Latin America & Caribbean** inclut des pays comme le Brésil, le Mexique et l'Argentine.
* **Low-Income Countries (LIC)** correspond aux pays à faible revenu, comme le Niger, Haïti et le Mozambique.
* **Middle-Income Countries (MIC)** se divise en pays à revenu intermédiaire inférieur, tels que l'Inde et le Kenya, et ceux à revenu intermédiaire supérieur, comme la Chine et le Brésil.
* **EMDE Middle East & N. Africa** concerne des pays tels que la Tunisie, l'Égypte et l'Arabie saoudite.
* **EMDE South Asia** comprend l'Inde, le Bangladesh et le Pakistan.
* **EMDE Sub-Saharan Africa** se concentre sur des pays comme le Nigeria, le Ghana et l'Éthiopie.
* Enfin, **World (WBG members)** fournit des données globales pour l'ensemble des membres de la Banque mondiale.



Ce code Python utilise la librairie pandas (pour la manipulation de données) pour importer des données depuis un fichier CSV nommé "Price.csv"



Ce code Python utilise la bibliothèque pandas pour afficher des informations générales sur un Data Frame appelé data. Plus précisément, il utilise la méthode info ()**:** Il donne une vue d'ensemble de la structure du Data Frame, notamment le nombre de lignes, le nombre de colonnes, les noms des colonnes et les types de données , Identifier les données manquantes , Vérifier les types de données , Optimisation mémoire

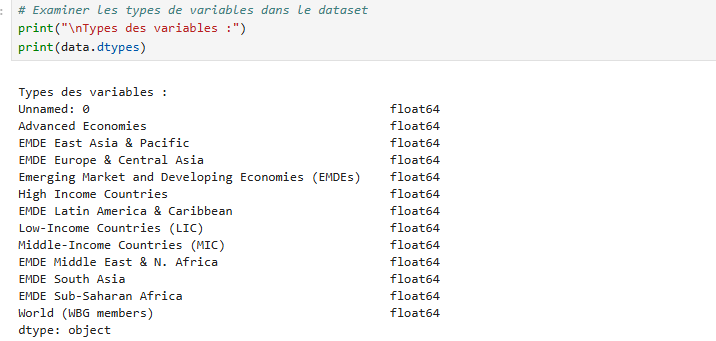
Dans notre data

 Le DataFrame a 31 lignes et 13 colonnes.

 Toutes les colonnes contiennent des nombres à virgule flottante (type float64).

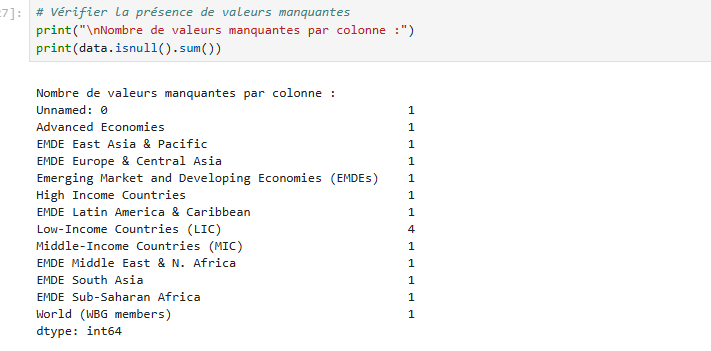
 La plupart des colonnes ont 30 valeurs non nulles, sauf "Low-Income Countries (LIC)" qui en a 27, indiquant qu'il y a 3 valeurs manquantes dans cette colonne.

 Le DataFrame utilise 3.3 KB de mémoire.



Ce code Python pour afficher les types de données de chaque colonne d'un Data Frame

Le résultat de print (data.dtypes) montre que toutes les colonnes du Data Frame data sont de type float64. Cela signifie que toutes les colonnes contiennent des nombres à virgule flottante en 64 bits.



Ce code Python permet de détecter et de quantifier les valeurs manquantes (NaN) dans un Data Frame. C'est une étape importante du nettoyage et de la préparation des données, car les valeurs manquantes peuvent perturber les analyses et les modèles.

Le résultat de print(data.isnull().sum()) montre le nombre de valeurs manquantes pour chaque colonne :

* La plupart des colonnes ont une seule valeur manquante (1).
* La colonne "Low-Income Countries (LIC)" a quatre valeurs manquantes (4).

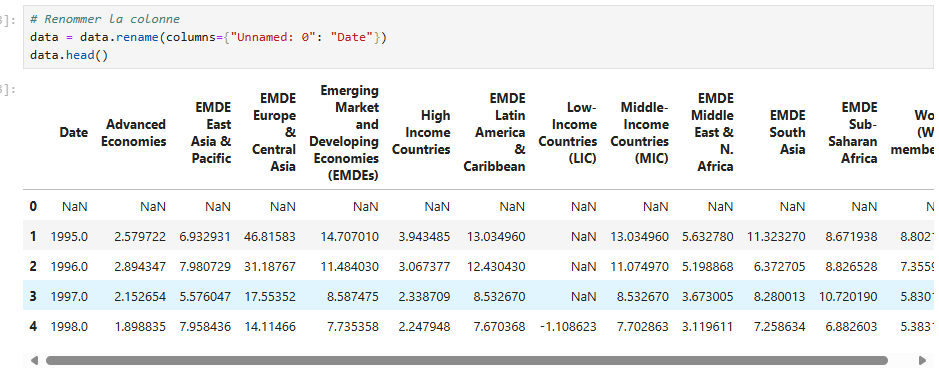
L'information dtype : int64 indique que les valeurs de la série résultante sont des entiers 64 bits.

Remarque

Il est essentiel de détecter et de gérer les valeurs manquantes avant de procéder à des analyses ou à la modélisation. Différentes stratégies existent pour gérer les valeurs manquantes, comme :

* **Suppression des lignes ou des colonnes contenant des valeurs manquantes :** Cette approche est simple, mais elle peut entraîner une perte d'information si les données supprimées sont importantes.
* **Imputation (remplacement des valeurs manquantes) :** On remplace les valeurs manquantes par des valeurs estimées, par exemple la moyenne, la médiane, ou une valeur prédite par un modèle.

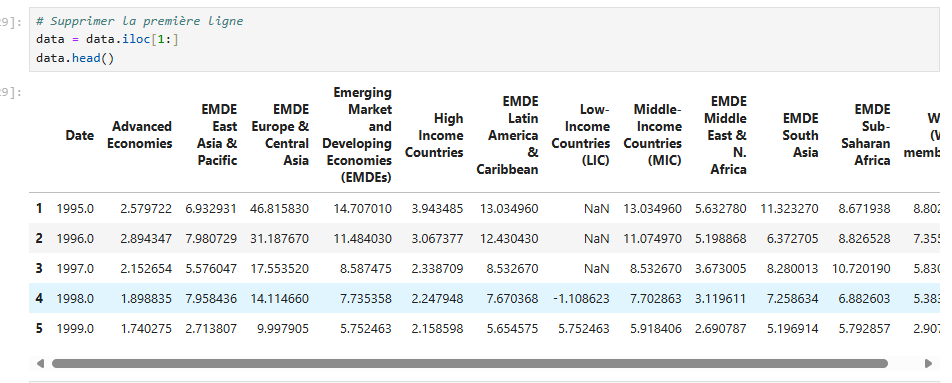
**Nettoyage des données**

****

Ce code Python pour renommer une colonne d'un Data Frame

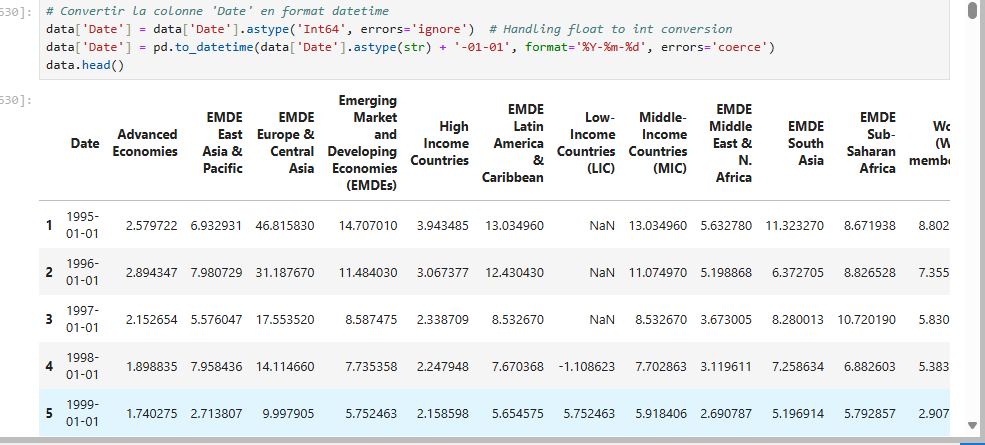
La première colonne du Data Frame, qui s'appelait auparavant "Unnamed: 0", s'appelle maintenant "Date". Les premières lignes affichées par data.Head() montrent également les valeurs de cette colonne, qui semblent correspondre à des années (1995.0, 1996.0, etc.).

Ce code effectue une opération simple mais importante pour la préparation et le nettoyage des données : il renomme une colonne pour la rendre plus informative et plus facile à utiliser dans les analyses ultérieures.



Ce code Python utilise pour supprimer la première ligne d'un Data Frame.

Dans notre cas La première ligne est une ligne qui n'est pas nécessaire pour l'analyse, La nouvelle première ligne a maintenant l'index 1 et contient les données pour l'année 199

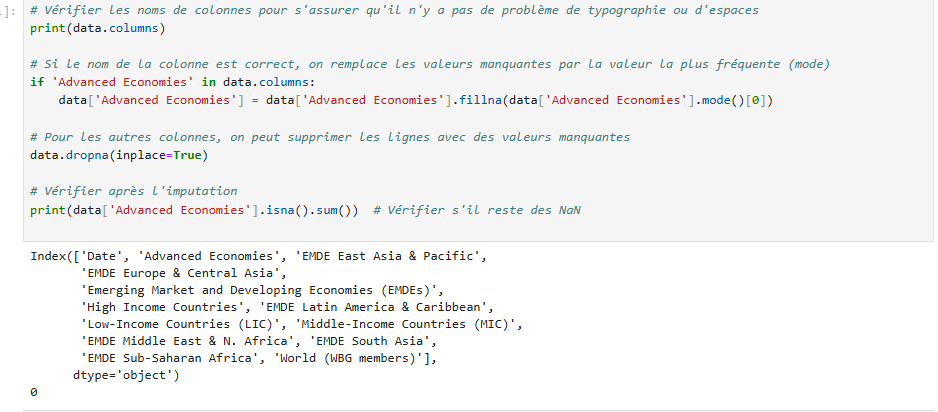


Ce code Python pour convertir la colonne 'Date' d'un Data Frame en type date time

La colonne 'Date' contient maintenant des objets date time au format 'AAAA-01-01’, Première observation '1995' est devenu '1995-01-01'.

**Pourquoi convertir en date time ?**

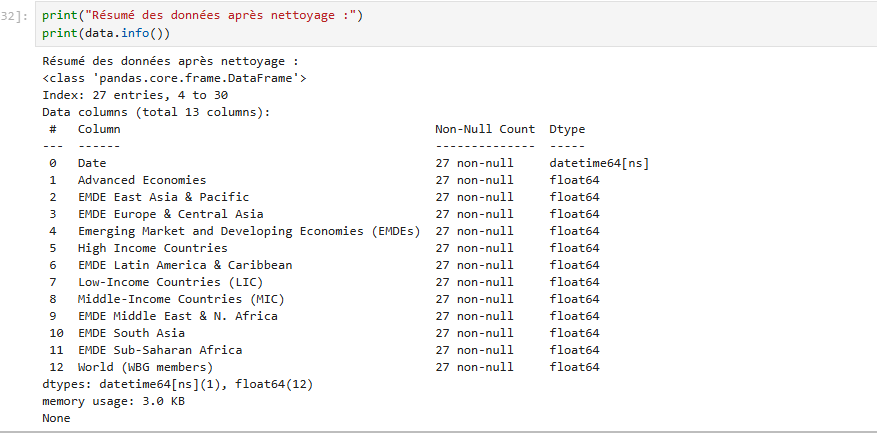
* **Opérations temporelles :** Permet d'effectuer des calculs sur les dates, comme calculer des différences entre deux dates, extraire des années, des mois, des jours, etc.
* **Indexation temporelle :** Permet d'utiliser la colonne 'Date' comme index pour le Data Frame, ce qui facilite les opérations de sélection et de filtrage basées sur le temps.
* **Visualisation :** Les bibliothèques de visualisation comme matplotlib et seaborn gèrent mieux les données de type date time



Ce code Python pour gérer les valeurs manquantes dans un Data Frame , Il met en œuvre une stratégie d'imputation et de suppression pour traiter ces valeurs

En résumé, ce code sert à :

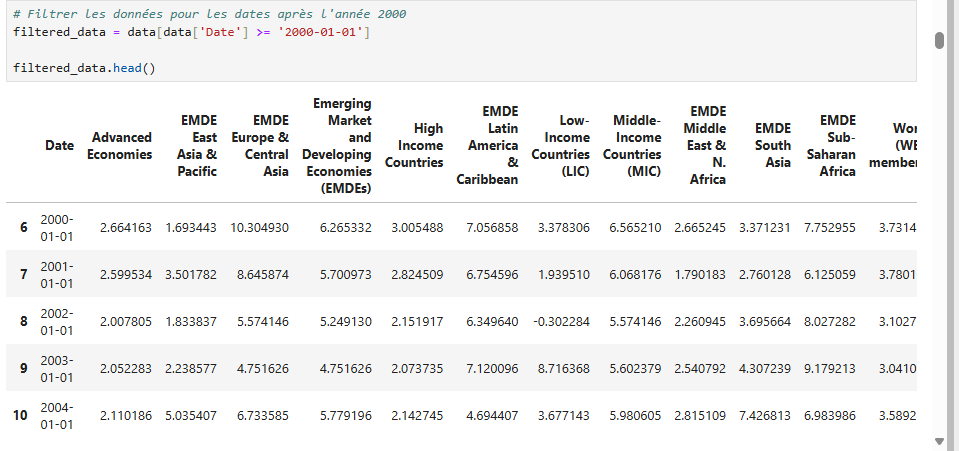
* **Gérer les valeurs manquantes :** Il utilise une stratégie combinée d'imputation et de suppression.
* **Imputation par le mode :** Pour la colonne 'Advanced Economies', les valeurs manquantes sont remplacées par la valeur la plus fréquente.
* **Suppression des lignes :** Pour les autres colonnes, les lignes contenant des valeurs manquantes sont supprimées.
* **Vérification :** Une vérification est effectuée pour s'assurer qu'il ne reste plus de valeurs manquantes dans la colonne 'Advanced Economies' après l'imputation



Ce code Python, affiche un résumé des informations sur un Data Frame nommé data *après* les opérations de nettoyage précédentes (renommage de colonne, suppression de la première ligne, conversion de la colonne 'Date' en date time et gestion des valeurs manquantes).

Le Data Frame contient maintenant 27 lignes et 13 colonnes, toutes les colonnes ont 27 valeurs non nulles, ce qui confirme que les valeurs manquantes ont été gérées.

Pourquoi utiliser data.info () après le nettoyage : vérification documentation de l état...

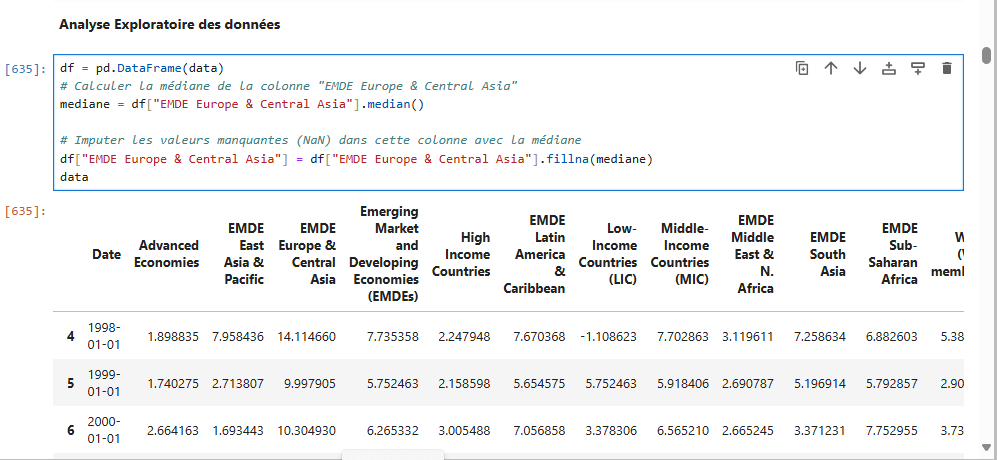


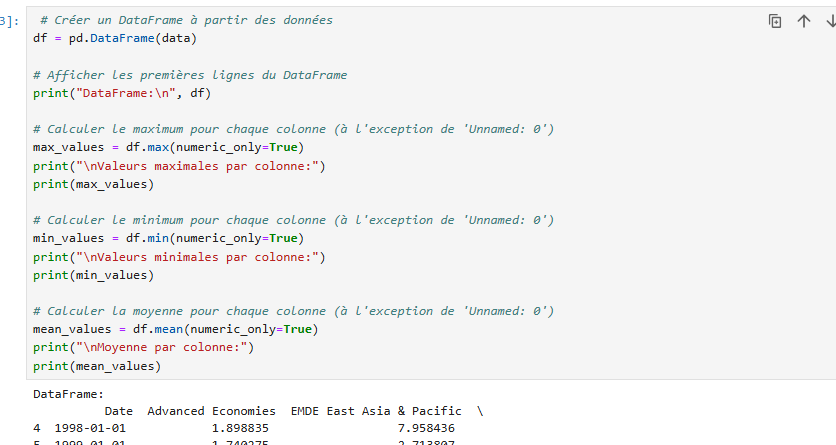
Ce code Python, filtre les données d'un Data Frame nommé data pour ne conserver que les lignes dont la date est postérieure ou égale au 1er janvier 2000

**En résumé, ce code sert à :** **Filtrer les données par date :** Il sélectionne les données d'un Data Frame qui correspondent à une plage de dates spécifiée. Dans ce cas, il conserve les données à partir du 1er janvier 2000.

**Pourquoi filtrer par date ? C’est pour se concentrer sur une période spécifique**

**Analyse Exploratoire des données**





Ce code Python, effectue plusieurs opérations statistiques descriptives sur un Data Frame.

**Data Frame :** Affiche les premières lignes du Data Frame df, avec les colonnes 'Date', 'Advanced Economies' et 'EMDE East Asia & Pacific'.

**Valeurs maximales par colonne :** Affiche les valeurs maximales pour chaque colonne numérique.

**Valeurs minimales par colonne :** Affiche les valeurs minimales pour chaque colonne numérique.

**Moyenne par colonne :** Affiche les moyennes pour chaque colonne numérique.

**Pourquoi calculer le maximum, le minimum et la moyenne sont des opérations fondamentales en analyse exploratoire des données (EDA**) : Elles permettent de : Comprendre la distribution des données, Identifier les valeurs aberrantes (outliers), Détecter les erreurs de saisie, Préparer les données pour la modélisation...

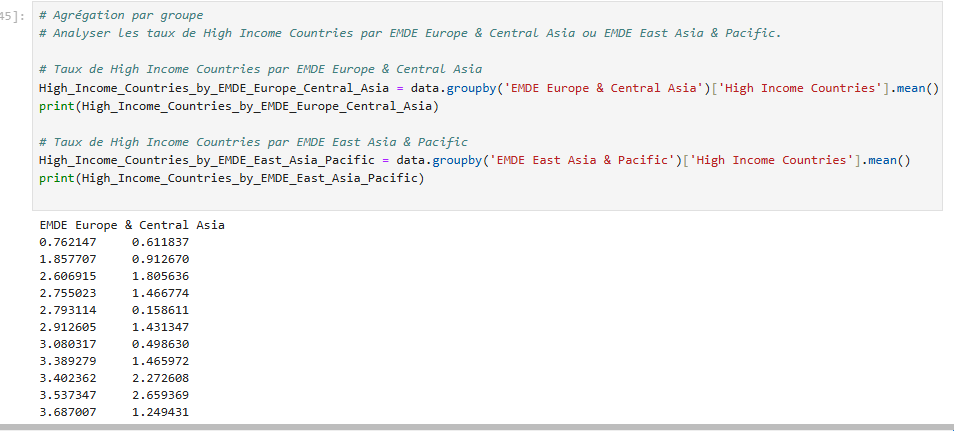


Ce code Python, calcule et affiche le maximum et le minimum des valeurs dans la colonne "High Income Countries" d'un DataFrame

**High Income Countries: 7.920065:** Ceci est la valeur maximale trouvée dans la colonne "High Income Countries".

**High Income Countries : 0.158611:** Ceci est la valeur minimale trouvée dans la colonne "High Income Countries".

La différence entre le maximum et le minimum (7.920065 - 0.158611 = 7.761454) représente l'étendue des valeurs. Cela signifie que les valeurs dans cette colonne varient sur une plage d'environ 7.76 unités.



Ce code Python effectue une agrégation par groupe et calcule la moyenne de la colonne "High Income Countries" pour différents groupes définis par les colonnes "EMDE Europe & Central Asia" et "EMDE East Asia & Pacific".

**En résumé, ce code sert à :**

* **Calculer les moyennes de "High Income Countries" par groupe :** Il permet de comparer les moyennes de "High Income Countries" pour différents groupes définis par les valeurs des colonnes "EMDE Europe & Central Asia" et "EMDE East Asia & Pacific".

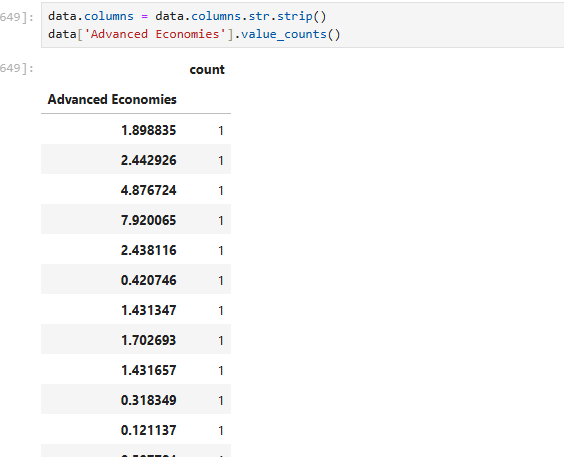
 La première partie affiche les moyennes de "High Income Countries" pour chaque valeur unique présente dans la colonne "EMDE Europe & Central Asia". Par exemple, pour la valeur 0.762147 de "EMDE Europe & Central Asia", la moyenne correspondante de "High Income Countries" est 0.611837.

 La deuxième partie affiche les moyennes de "High Income Countries" pour chaque valeur unique présente dans la colonne "EMDE East Asia & Pacific".

Pourquoi faire une agrégation par groupe ?

**Comparer des sous-groupes :** Elle permet de comparer les statistiques (moyenne, médiane, somme, etc.) d'une variable pour différents groupes

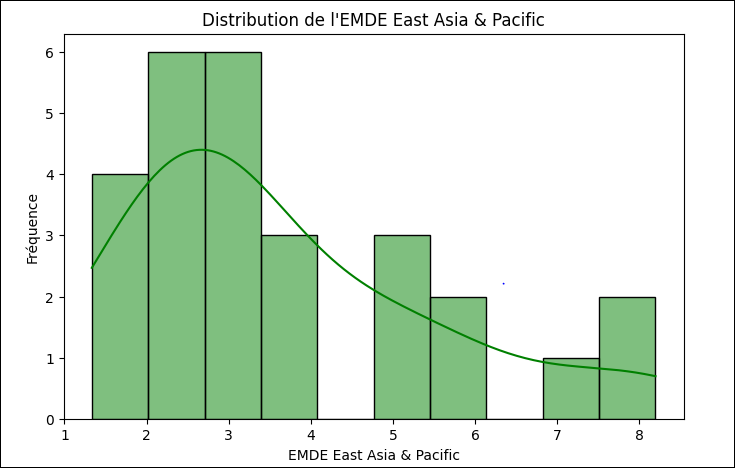
**Identifier des tendances :** Elle peut révéler des tendances ou des différences significatives entre les groupes...



Ce code Python, utilisant la bibliothèque pandas, effectue deux opérations : le nettoyage des noms de colonnes et le comptage des occurrences des valeurs dans la colonne "Advanced Economies".

La méthode value\_counts() est très utile en analyse exploratoire des données (EDA) pour :Identifier les valeurs les plus fréquentes..

Dans le contexte de notre projet, le résultat indique que la colonne "Advanced Economies" pourrait potentiellement servir d'identifiant unique pour chaque observation, car chaque valeur est distincte.

**Visualisation des données**

L'histogramme de la distribution de l'indice des prix à la consommation pour la région **"EMDE East Asia & Pacific"** inclut une courbe KDE (noyau de densité) pour visualiser la distribution sous-jacente

L'histogramme montre la répartition des valeurs de l'indice des prix à la consommation (IPC) pour la région **"EMDE East Asia & Pacific"**.

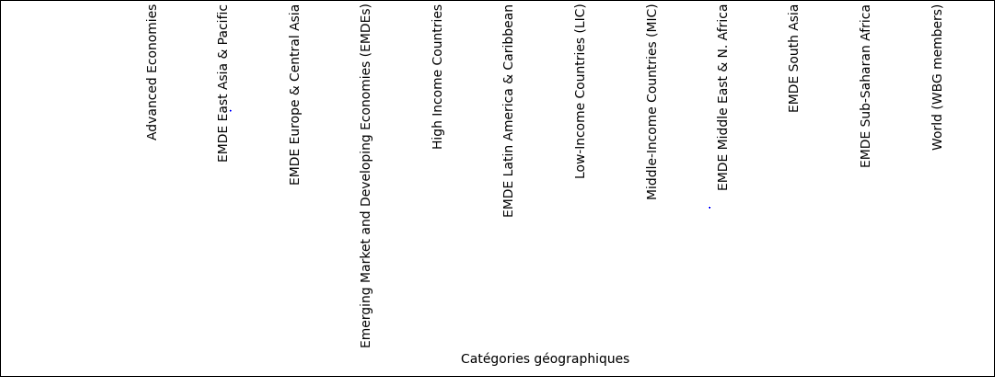
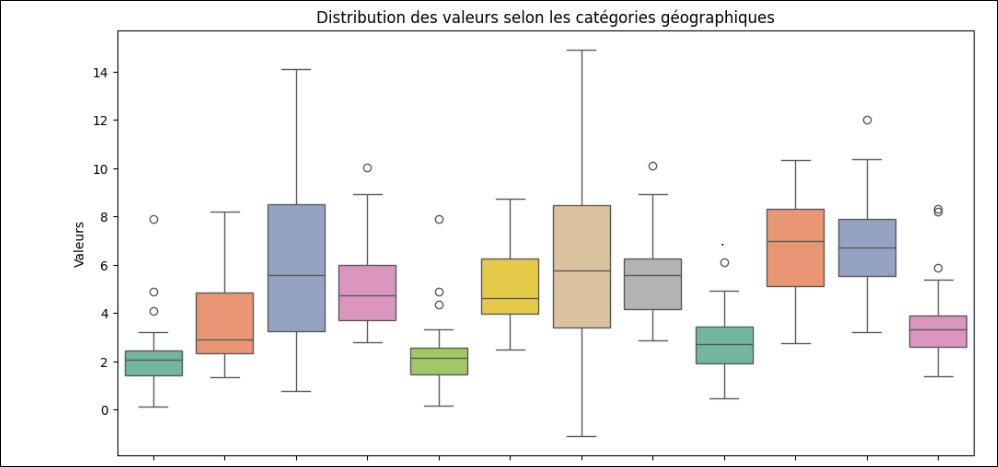
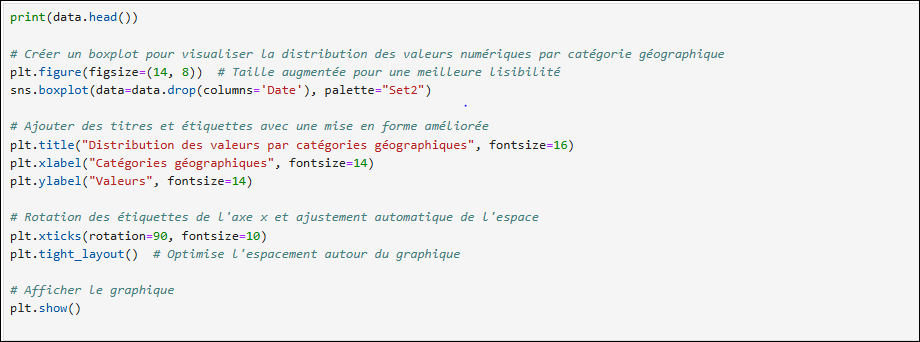
* **Axe horizontal (X)** : Les valeurs de l'IPC dans cette région.
* **Axe vertical (Y)** : La fréquence, c'est-à-dire combien de fois une valeur (ou une plage de valeurs) apparaît dans les données.

**Points principaux :**

1. **Fréquences des valeurs** : La plupart des IPC se concentrent dans une plage spécifique.
2. **Forme de la distribution** : Légèrement asymétrique, ce qui pourrait indiquer des variations dans l'IPC au fil du temps.
3. **Courbe KDE (Kernel Density Estimation)** : Elle lisse les données pour donner une vue générale de la densité des valeurs.

Cela aide à identifier si l'IPC est stable ou s'il varie beaucoup dans cette région

* La majorité des valeurs de l'IPC pour **"EMDE East Asia & Pacific"** se concentrent dans une plage spécifique (autour du ou des pics).
* Si la courbe KDE est étroite, cela indique peu de variation. Si elle est large, l'IPC est plus dispersé dans le temps.
* Une distribution bien centrée et étroite suggère une stabilité de l'IPC dans cette région.
* Une distribution asymétrique ou avec plusieurs pics pourrait indiquer des périodes de forte inflation ou de changements économiques significatifs.



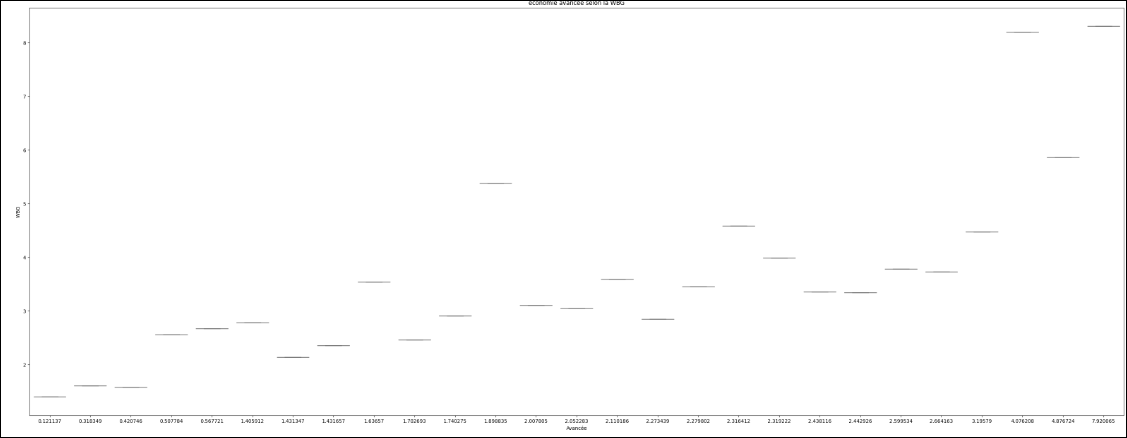
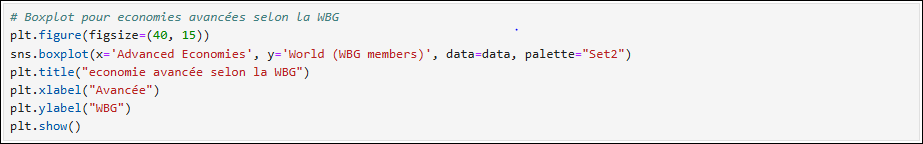
Le boxplot visualise la répartition des valeurs de l'indice des prix à la consommation (IPC) pour chaque catégorie géographique.

Ce graphique permet de comparer rapidement les niveaux et la dispersion de l'IPC entre les régions.

* **Axes :**
  + **Axe X** : Les différentes catégories géographiques (ex. : *Advanced Economies*, *EMDE East Asia & Pacific*).
  + **Axe Y** : Les valeurs de l'IPC associées.
* **Variabilité des valeurs :** **Comparaison entre régions** :
  + Certaines régions montrent des variations plus importantes que d'autres (boîtes plus grandes ou plus dispersées).
  + Par exemple, **"EMDE Europe & Central Asia"** a une boîte très étendue, indiquant une forte dispersion ( variation) des IPC.
  + D'autres, comme **"High Income Countries"**, montrent une variation plus faible (plus stable).
* **Médiane :** Elle est différente pour chaque région, reflétant des niveaux d'IPC typiques variés.
* **Valeurs aberrantes :** Des points individuels en dehors des "moustaches" sont des valeurs extrêmes ou atypiques indiquent des anomalies économiques (événements économiques inhabituels) sont des variations importantes de l'IPC dans certaines régions correspondrent à des crises économiques, à des périodes de forte inflation ou à des événements spécifiques dans ces régions.
* **Interprétation des boîtes :** **Analyse du boxplot :**

**Chaque boîte** :

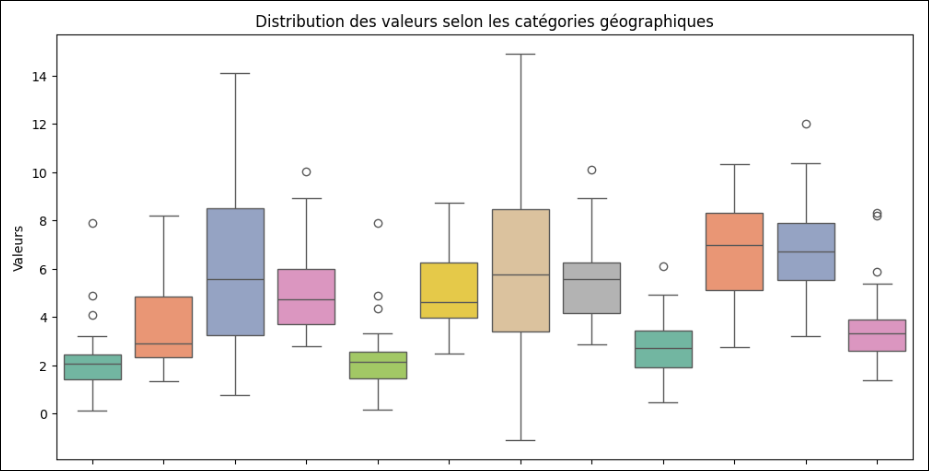
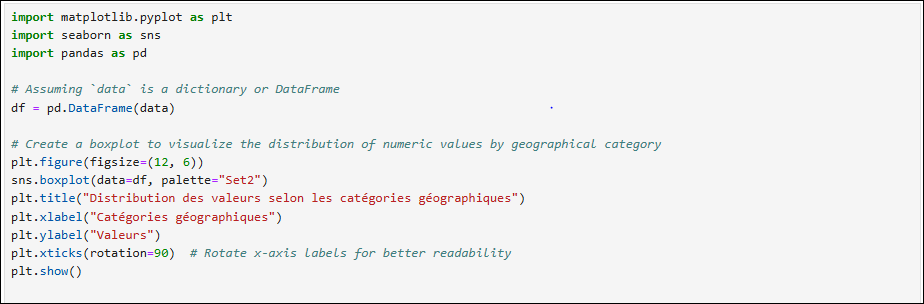
* + Représente la répartition des données pour une région ou un groupe spécifique.
  + Les indices de prix à la consommation varient largement selon les régions, ce qui reflète probablement des différences dans les contextes économiques.
  + La ligne centrale correspond à la médiane (valeur centrale des données).
  + Les extrémités de la boîte indiquent le 1er quartile (Q1) et le 3ème quartile (Q3), montrant où se trouve la majorité des valeurs.
  + Les "moustaches" montrent l'étendue des valeurs sans compter les éventuels points aberrants.



**C**e code permet de visualiser de manière simple et efficace la distribution d'un indicateur économique (une variable numérique) pour un groupe spécifique d'économies, à savoir les économies avancées, en utilisant un boxplot .

on trouve :

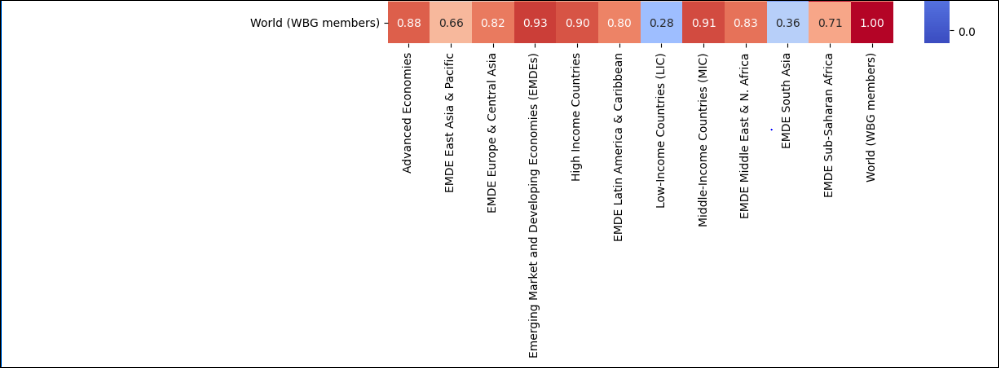
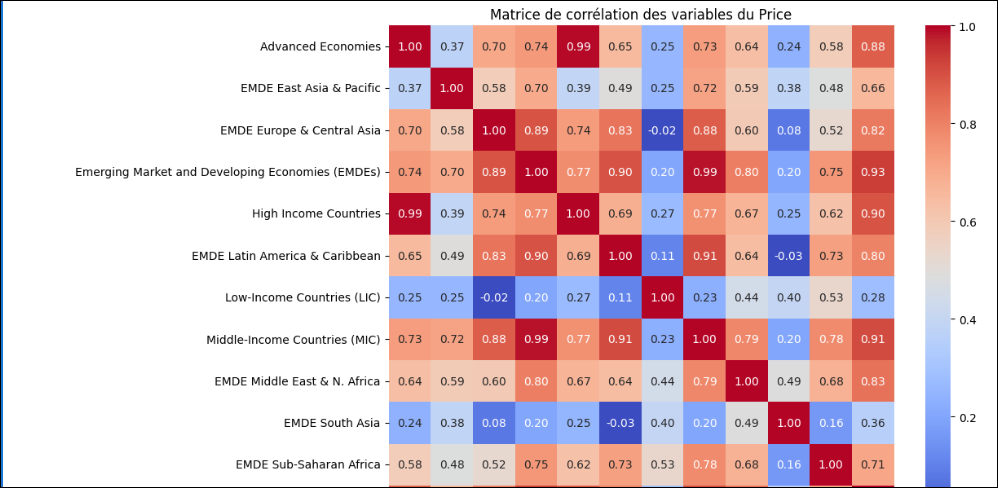
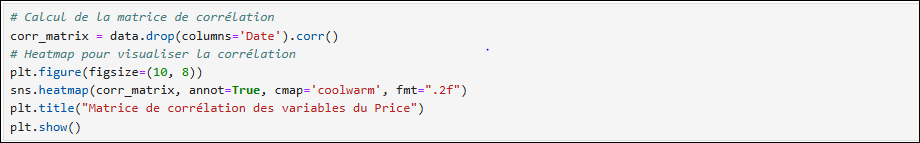
* **La dispersion des valeurs** de l'indicateur économique pour les économies avancées.
* **La présence d'outliers** (économies avec des valeurs très élevées ou très faibles par rapport aux autres).
* **Les différences de distribution** entre les économies avancées et d'autres groupes d'économies (si d'autres colonnes sont ajoutées au boxplot).



L'objectif de ce code est de :

* **Visualiser la distribution** de l'indice des prix à la consommation (ou d'un autre indicateur économique) pour différentes régions géographiques.
* **Comparer les niveaux d'inflation** entre ces régions.
* **Identifier les valeurs atypiques** (outliers) dans les données.

Le code fourni permet de créer un graphique simple pour visualiser la distribution d'un indicateur économique par région.

Ce code Python est conçu pour **analyser les corrélations entre différentes variables** présentes dans un ensemble de données (représenté par le DataFrame data).

**Décomposition du code:**

1. **Calcul de la matrice de corrélation:**
   * corr\_matrix = data.drop(columns='Date').corr(): Cette ligne calcule la matrice de corrélation de toutes les colonnes numériques du DataFrame data, à l'exception de la colonne "Date". La matrice de corrélation est un tableau qui montre la corrélation linéaire entre chaque paire de variables.
2. **Visualisation de la matrice de corrélation:**
   * plt.figure(figsize=(10, 8)): Crée une figure de taille 10x8 pouces pour accueillir le graphique.
   * sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm', fmt=".2f"): Crée une carte thermique (heatmap) pour visualiser la matrice de corrélation.
     + annot=True: Affiche les valeurs de corrélation dans chaque cellule de la carte thermique.
     + cmap='coolwarm': Utilise une palette de couleurs allant du bleu (corrélation négative) au rouge (corrélation positive) pour représenter les corrélations.
     + fmt=".2f": Formatte les valeurs de corrélation avec deux décimales.
   * plt.title("Matrice de corrélation des variables du Price"): Ajoute un titre au graphique.
   * plt.show(): Affiche la carte thermique.

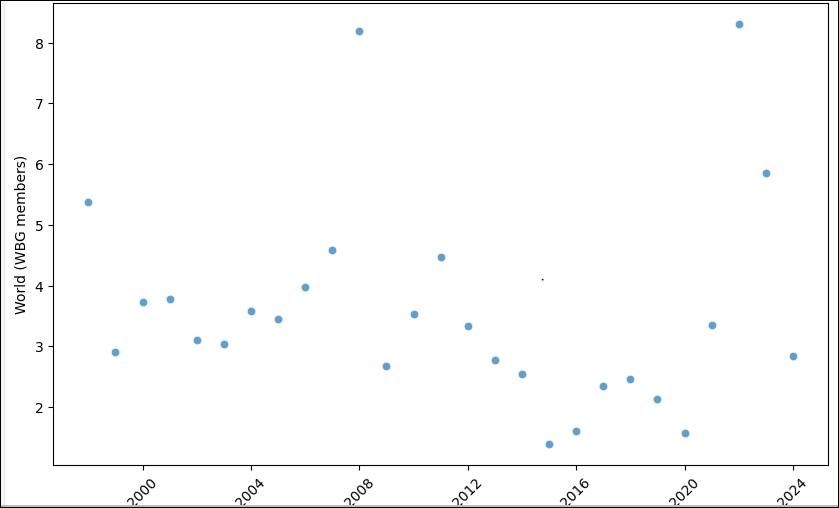
**Utilité du code**

L'objectif principal de ce code est de **comprendre les relations entre les différentes variables** présentes dans un ensemble de données. La matrice de corrélation permet de :

* **Identifier les variables fortement corrélées:** Les variables qui ont une corrélation proche de 1 sont fortement corrélées positivement, tandis que celles qui ont une corrélation proche de -1 sont fortement corrélées négativement.
* **Détecter les colinéarités:** Si deux variables sont fortement corrélées, cela peut poser des problèmes dans les modèles statistiques (multicolinéarité).
* **Sélectionner les variables pertinentes pour une analyse ultérieure:** En identifiant les variables les plus corrélées avec la variable cible, on peut réduire la dimensionnalité des données et améliorer la performance des modèles.

Ce code est un outil précieux pour l'**analyse exploratoire des données**. Il permet de rapidement visualiser les relations entre les différentes variables et d'identifier les variables les plus importantes pour une analyse plus approfondie.



  Le code vise à créer un graphique de type "scatter plot" (nuage de points) pour visualiser la relation entre une variable de date et une variable numérique (probablement un indicateur économique) à l'échelle mondiale.

Ce code Python, en utilisant les bibliothèques Matplotlib et Seaborn, crée un **graphique de dispersion (scatter plot)** à partir d'un ensemble de données contenu dans un DataFrame Pandas nommé data.

**Décomposition du code:**

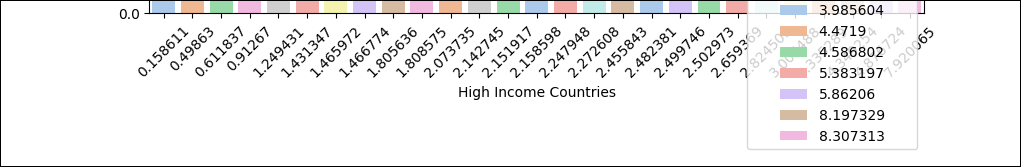
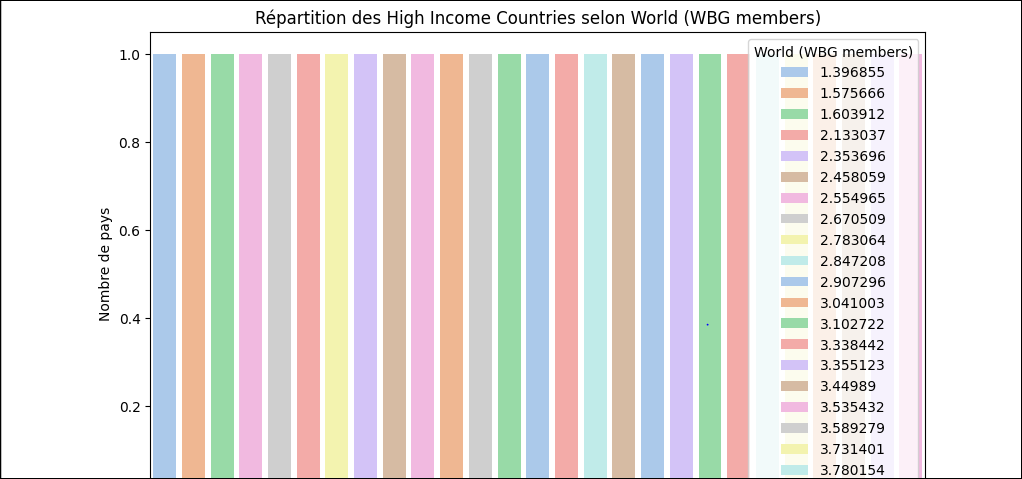
1. **Importation des bibliothèques:**
   * matplotlib.pyplot: Pour la création de graphiques.
   * seaborn: Pour la création de visualisations statistiques, notamment des scatterplots.
2. **Création du graphique:**
   * sns.scatterplot(x='Date', y='World (WBG members)', data=data, alpha=0.7):
     + x='Date': L'axe des abscisses représente la date.
     + y='World (WBG members)': L'axe des ordonnées représente la variable numérique (probablement un indicateur économique mondial).
     + data=data : Indique que les données proviennent du DataFrame nommé data.
     + alpha=0.7: Rend les points légèrement transparents pour éviter la surpopulation si les données sont denses.
3. **Personnalisation du graphique:**
   * plt.title(), plt.xlabel(), plt.ylabel(): Ajoutent respectivement un titre et des étiquettes aux axes.
   * plt.xticks(rotation=45): Incline les étiquettes de l'axe des x de 45 degrés pour une meilleure lisibilité, surtout si les dates sont nombreuses.

**Interprétation du graphique :** Le graphique obtenu permettra de :

* **Visualiser la relation entre la date et la variable "World (WBG members)"**. Cela signifie qu'on pourra observer comment cette variable évolue au fil du temps.
* **Identifier des tendances:** On pourra voir si la variable augmente, diminue, ou si elle présente des cycles (croissance, décroissance, cycles).
* **Détecter des anomalies:** Des points éloignés des autres pourraient indiquer des événements particuliers ou des erreurs dans les données qui ont pu influencer l'évolution de l'indicateur.
* **Comparer l'évolution de cet indicateur** avec d'autres variables (en ajoutant des dimensions au graphique).

Si "World (WBG members)" représente le PIB mondial, le graphique montrera l'évolution du PIB mondial au fil du temps d’où il permettra de visualiser la croissance économique mondiale au fil du temps. On pourra ainsi observer des périodes de forte croissance, de récession, et identifier les événements (crises, pandémies) qui ont pu influencer l'économie mondiale.

Ce code permet d'explorer visuellement la relation entre une variable temporelle (la date) et une variable numérique (un indicateur économique) à l'échelle mondiale. Il est un outil précieux pour l'analyse de séries temporelles et pour identifier les tendances et les cycles économiques.

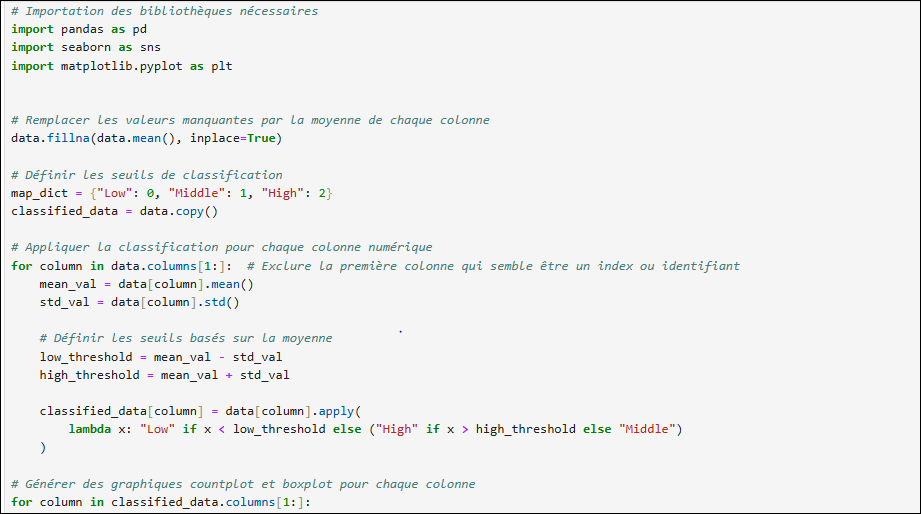
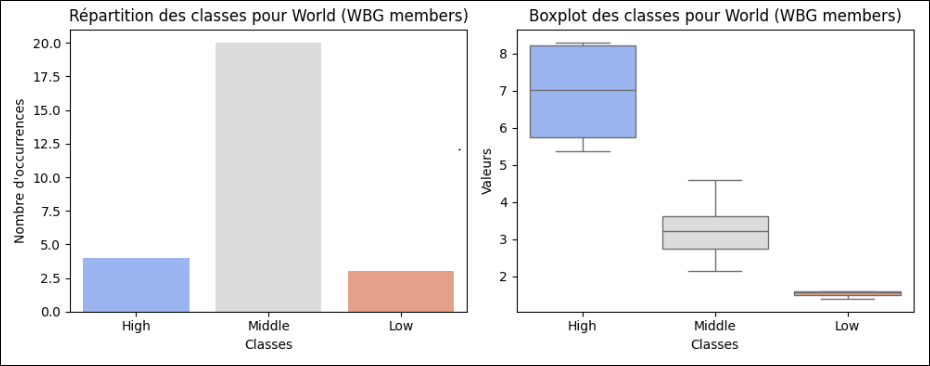
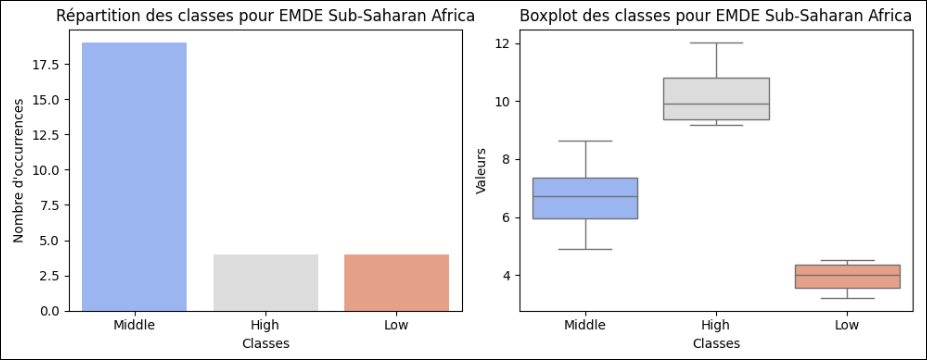
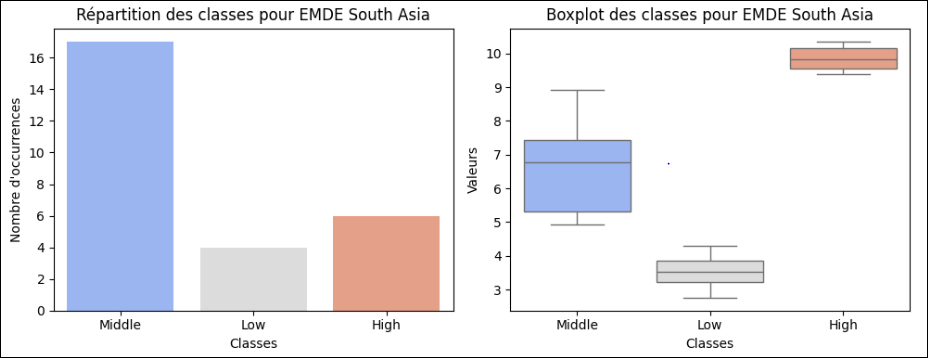
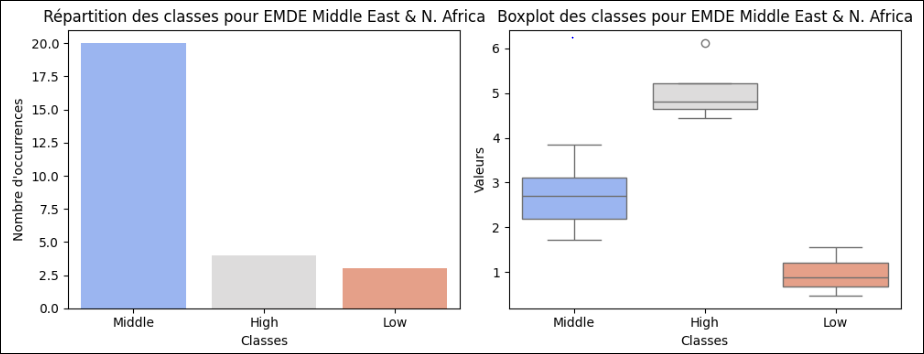
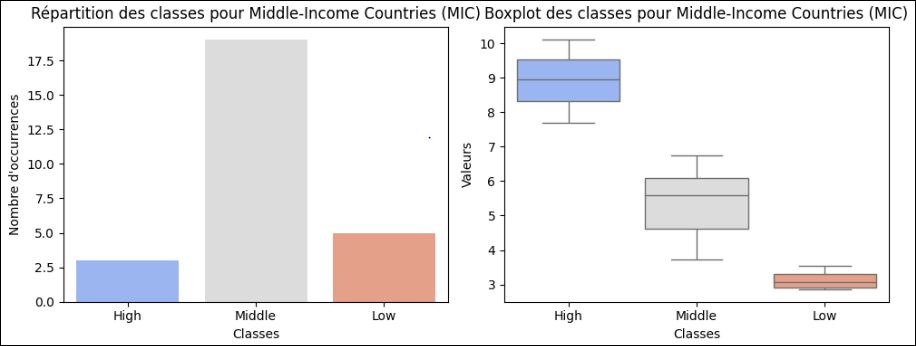
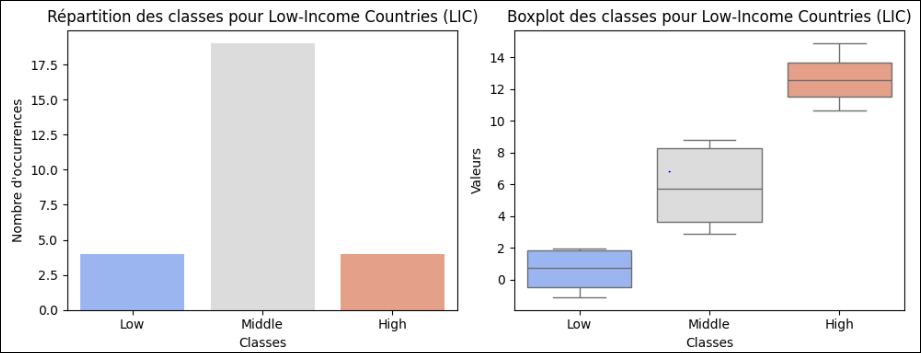
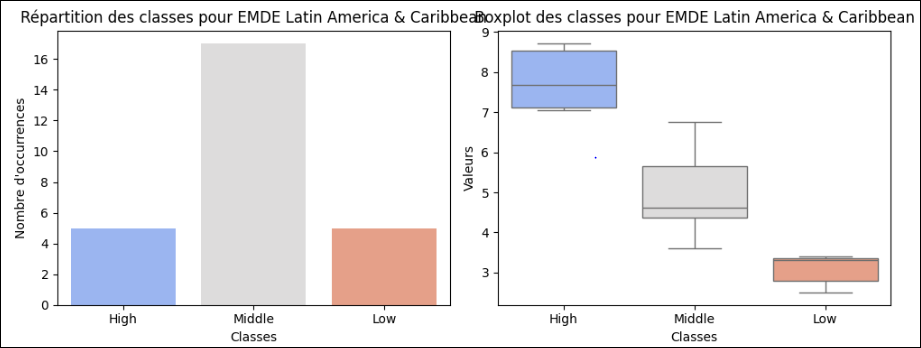
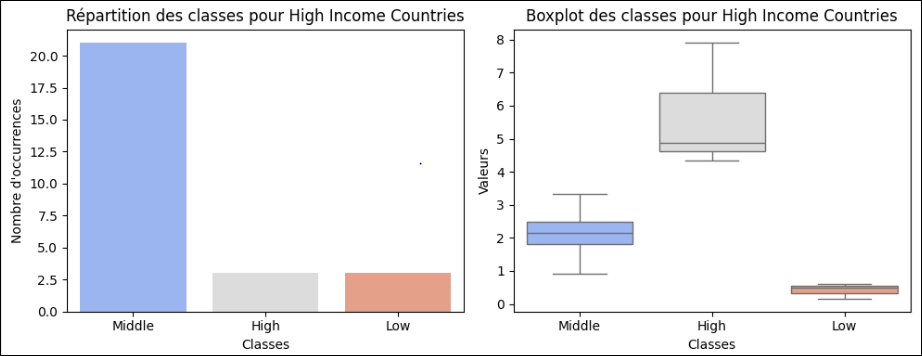
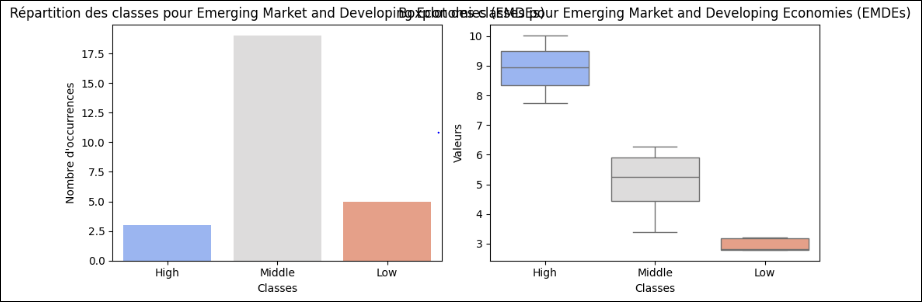
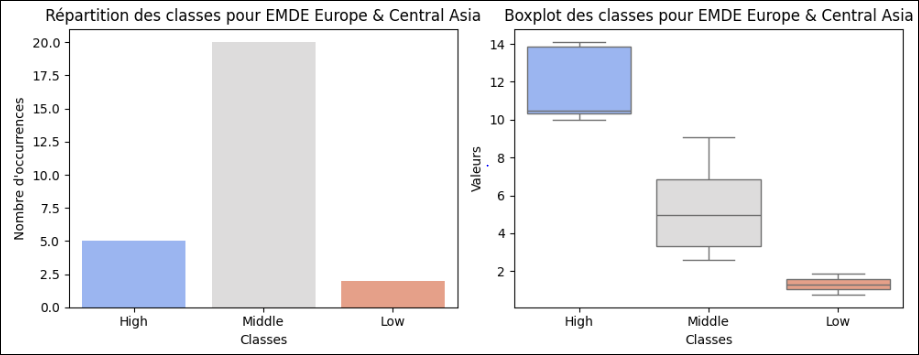
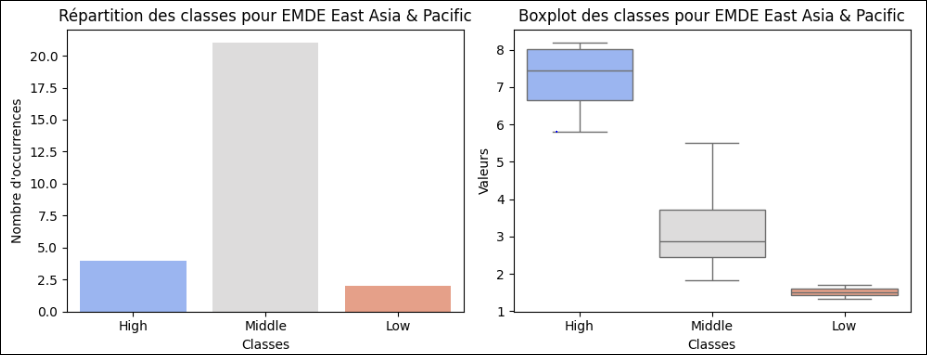
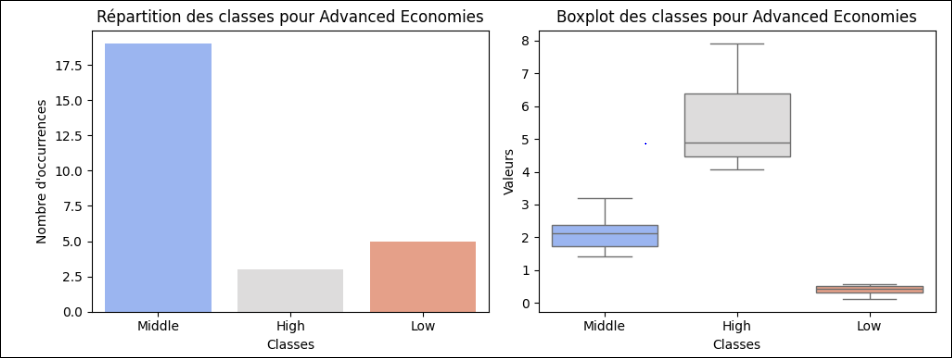
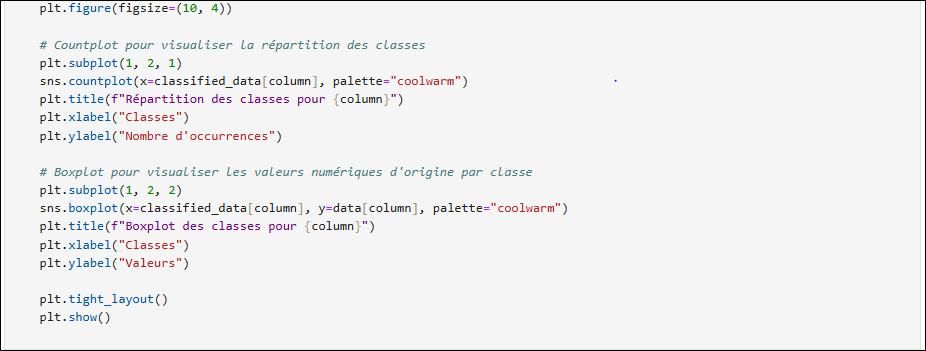
Le code vise à créer un diagramme en barres (countplot) à l'aide de la bibliothèque Seaborn pour visualiser la répartition des pays à revenu élevé (catégorie "High Income Countries") en fonction d'une autre variable catégorielle, "World (WBG members)".Interprétation des résultats

**Interprétation:**

* **Si "World (WBG members)" représente les régions du monde:** Le graphique montrera comment les pays à revenu élevé sont répartis entre les différentes régions (Amérique du Nord, Europe, Asie, etc.).
* **Si "World (WBG members)" représente des groupes d'âge:** Le graphique montrera comment les pays à revenu élevé sont répartis dans différentes tranches d'âge.

Ce code permet de visualiser la répartition d'une catégorie de pays (les pays à revenu élevé) en fonction d'une autre variable catégorielle. Le graphique obtenu donne une idée de la distribution de ces pays et peut aider à identifier des tendances ou des différences significatives.

**Classification selon la moyenne de l'economies de chaque région**

 Ce code Python vise à **analyser et visualiser des données catégorisées** à partir d'un DataFrame (table de données). Il effectue les étapes suivantes :

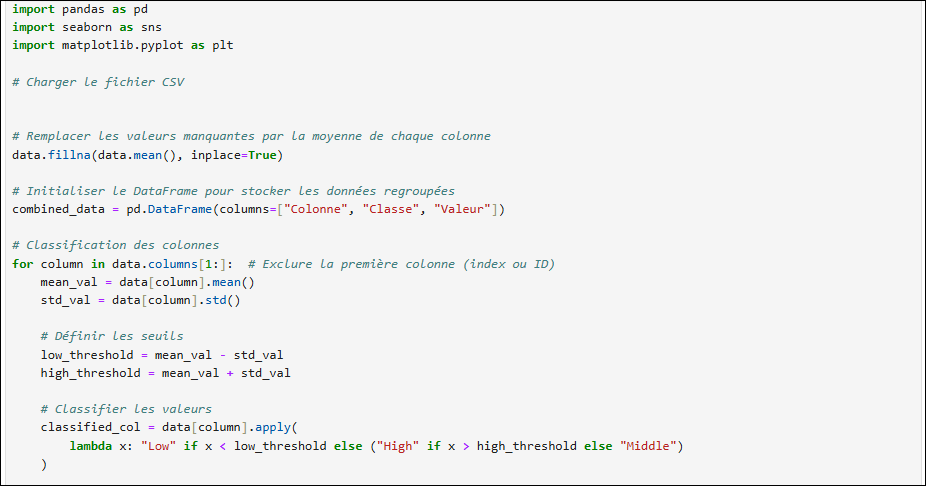
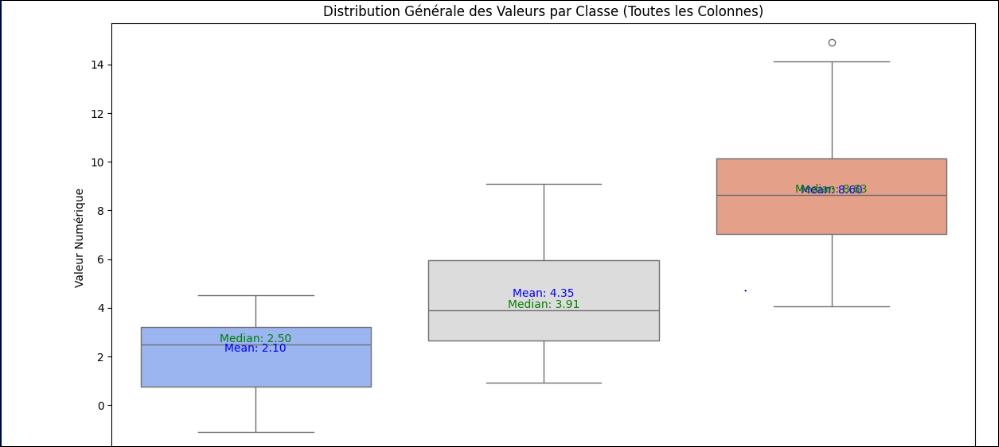
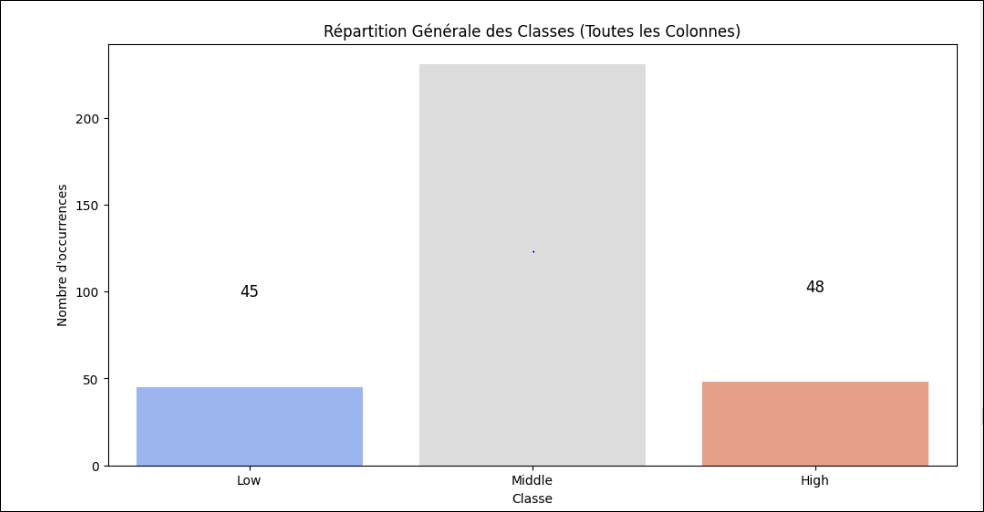
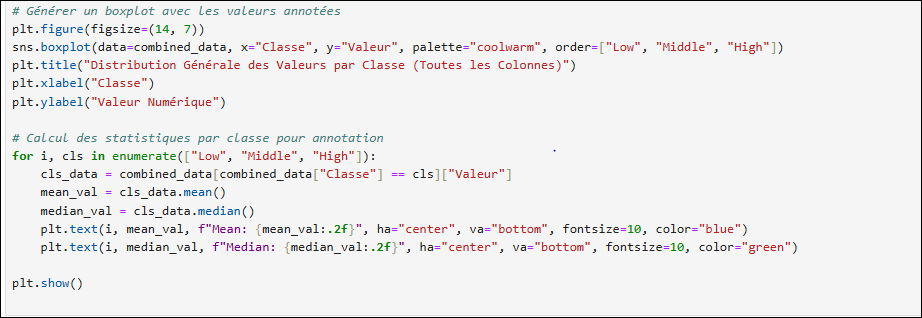
1. **Traitement des données manquantes:** Remplace les valeurs manquantes par la moyenne de chaque colonne.
2. **Classification des données:** Crée de nouvelles colonnes en catégorisant les valeurs numériques en trois classes : "Low", "Middle" et "High", en fonction de leur écart à la moyenne.
3. **Visualisation:** Utilise Seaborn pour créer des graphiques :
   * **Countplot:** Pour visualiser la distribution des catégories pour chaque variable.
   * **Boxplot:** Pour comparer la distribution des valeurs numériques originales pour chaque catégorie.

**Interprétation :**

En exécutant ce code, on obtient une série de graphiques pour chaque colonne numérique de DataFrame. Ces graphiques permettent de :

* **Visualiser la distribution** des valeurs dans chaque catégorie.
* **Identifier les valeurs aberrantes** (outliers).
* **Comparer les distributions** entre les différentes catégories.
* **Identifier les tendances** (par exemple, si les valeurs élevées sont plus fréquentes dans certaines catégories).

Ce code fournit un cadre pour analyser et visualiser des données numériques en les catégorisant en fonction de leur valeur par rapport à une moyenne. Il permet d'identifier des tendances et des différences entre les différentes catégories.

  Ce code Python vise à **analyser et visualiser la répartition de données**, en fournissant une **vue d'ensemble** sur la distribution des classes et des valeurs numériques.

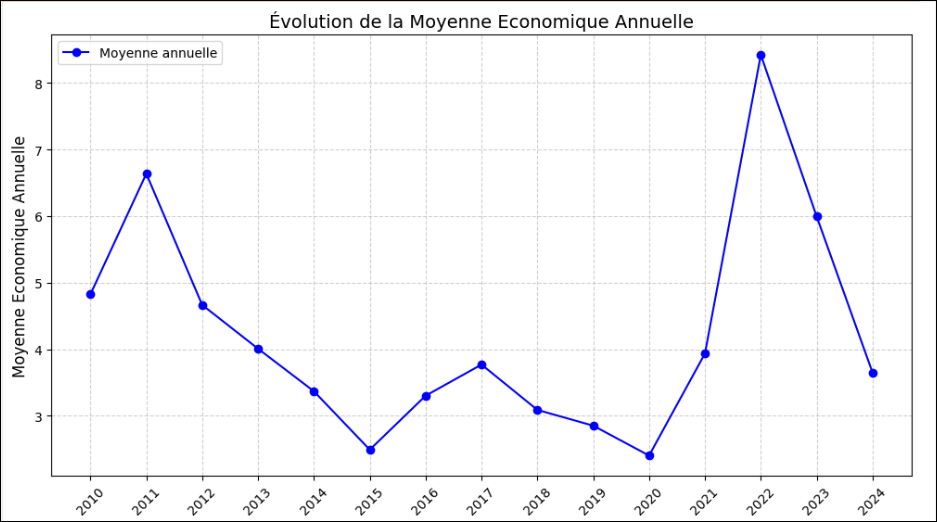
**Interprétation :**

Ces visualisations permettent d'avoir une vue d'ensemble de la répartition des données :

* **Répartition des classes:** Le countplot indique le nombre d'observations dans chaque catégorie ("Low", "Middle", "High") pour l'ensemble des colonnes numériques de votre jeu de données.
* **Distribution des valeurs par classe:** Le boxplot permet de comparer la distribution des valeurs numériques d'origine entre les différentes classes. On peut ainsi observer si certaines classes ont tendance à avoir des valeurs plus élevées ou plus faibles.
* **Valeurs moyennes et médianes:** Les annotations sur le boxplot indiquent la moyenne et la médiane des valeurs pour chaque classe, ce qui permet de résumer la distribution des valeurs.

Ce code fournit un point de départ pour l'analyse exploratoire de données catégorisées. Il permet d'obtenir une vue d'ensemble de la répartition des classes et des valeurs numériques.

**Evolution annuelle de la moyenne économique mondiale**

Le code vise à :

1. **Charger un fichier CSV** contenant des données économiques.
2. **Nettoyer les données:** Vérifier la présence de la colonne "Date" et s'assurer de son format.
3. **Filtrer les données:** Exclure les données des années 1995 à 2009.
4. **Calculer une moyenne annuelle:** Pour chaque ligne (potentiellement représentant différents pays ou régions), calculer la moyenne des valeurs numériques.
5. **Visualiser l'évolution de cette moyenne annuelle** au fil du temps à l'aide d'un graphique linéaire.

**Décomposition du code**

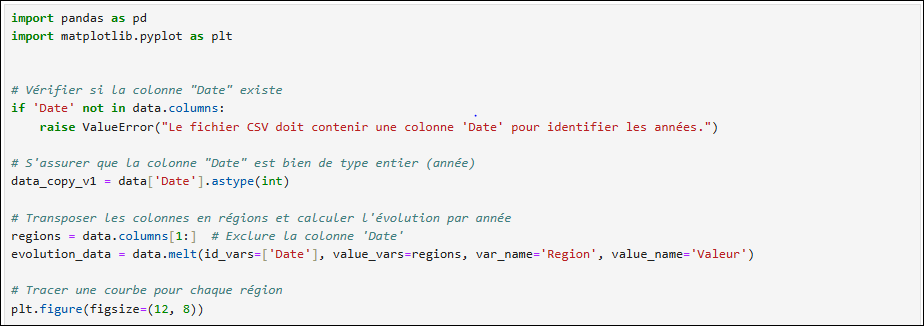
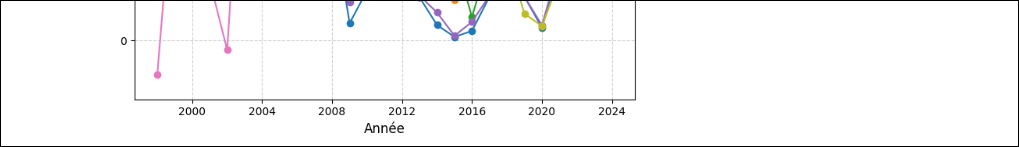
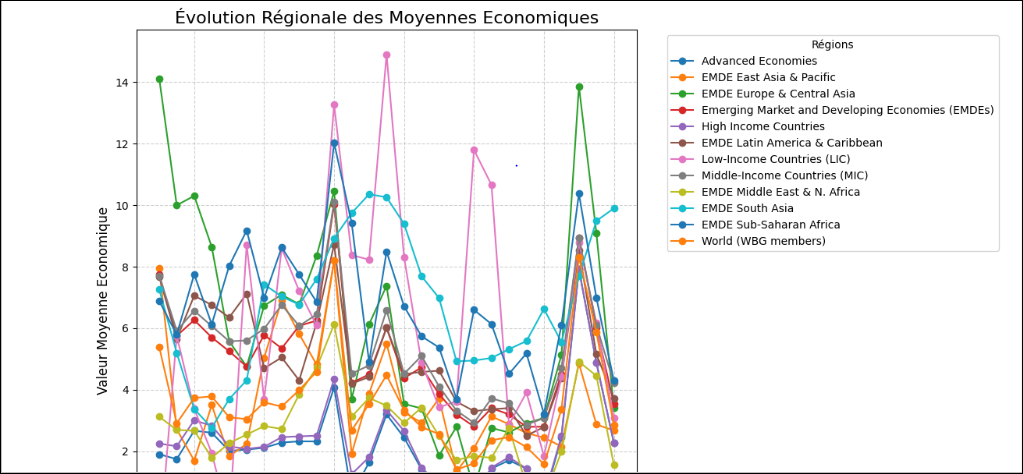
1. **Importation des bibliothèques:**
   * pandas: Pour la manipulation des données.
   * matplotlib.pyplot: Pour la création de graphiques.
2. **Vérification de la colonne "Date":**
   * Assure que la colonne "Date" existe et est bien présente dans le DataFrame.
3. **Formatage de la colonne "Date":**
   * Convertit la colonne "Date" en format année pour faciliter les calculs et les visualisations.
4. **Filtrage des données:**
   * Élimine les lignes correspondant aux années 1995 à 2009.
5. **Calcul de la moyenne annuelle:**
   * Calcule la moyenne de toutes les colonnes numériques (sauf la colonne "Date") pour chaque ligne.
   * Ajoute une nouvelle colonne "Moyenne\_Annuelle" au DataFrame.
6. **Visualisation:**
   * Crée un graphique linéaire montrant l'évolution de la moyenne annuelle au fil du temps.
   * Personnalise le graphique avec un titre, des étiquettes d'axes, une grille et une légende.

**Interprétation :**

Le graphique obtenu permettra de visualiser l'évolution de l'indicateur économique moyen (calculé sur toutes les colonnes numériques) au fil des années, en excluant la période 1995-2009.

Ce code fournit une base solide pour l'analyse de données temporelles. Il permet de visualiser l'évolution d'un indicateur économique moyen au fil du temps et d'identifier les tendances principales.

**Évolution annuelle des moyennes économique par régions**

Le code Python vise à :

1. **Charger un fichier CSV** contenant des données économiques par région et par année.
2. **Nettoyer les données:** Vérifier la présence de la colonne "Date" et s'assurer qu'elle est de type entier.
3. **Transformer les données:** Mettre les données en format long pour faciliter la visualisation par région.
4. **Calculer les moyennes annuelles:** Calculer la moyenne de chaque région pour chaque année.
5. **Visualiser l'évolution des moyennes:** Créer un graphique linéaire montrant l'évolution des moyennes économiques pour chaque région au fil du temps.

**Décomposition du code**

1. **Importation des bibliothèques:**
   * pandas: Pour la manipulation des données.
   * matplotlib.pyplot: Pour la création de graphiques.
2. **Vérification de la colonne "Date":**
   * Assure que la colonne "Date" existe et est de type entier (année).
3. **Transformation des données:**
   * Utilise la fonction melt de pandas pour transformer le DataFrame en un format long, ce qui facilite la création de graphiques par région.
4. **Calcul des moyennes annuelles:**
   * Groupe les données par année et calcule la moyenne pour chaque région.
5. **Visualisation:**
   * Crée un graphique linéaire multi-lignes, chaque ligne représentant une région.
   * Personnalise le graphique avec un titre, des étiquettes d'axes, une légende et une grille.

**Interprétation :**

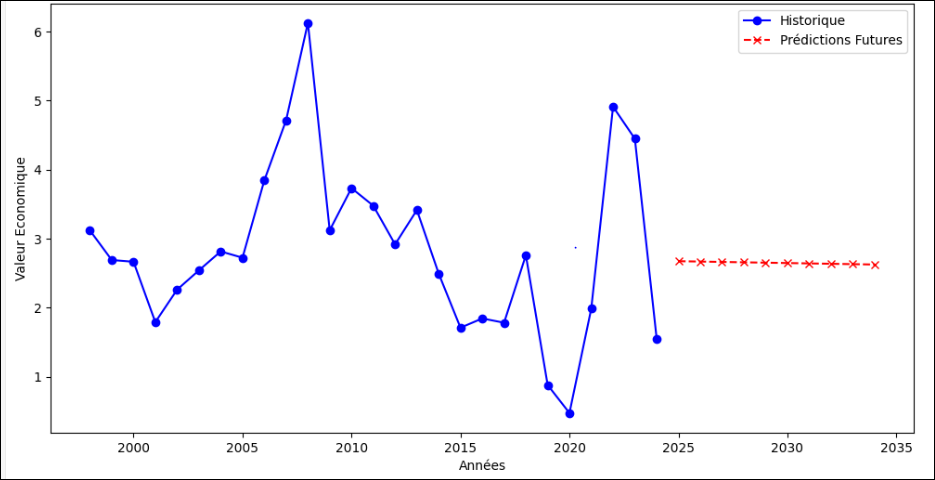
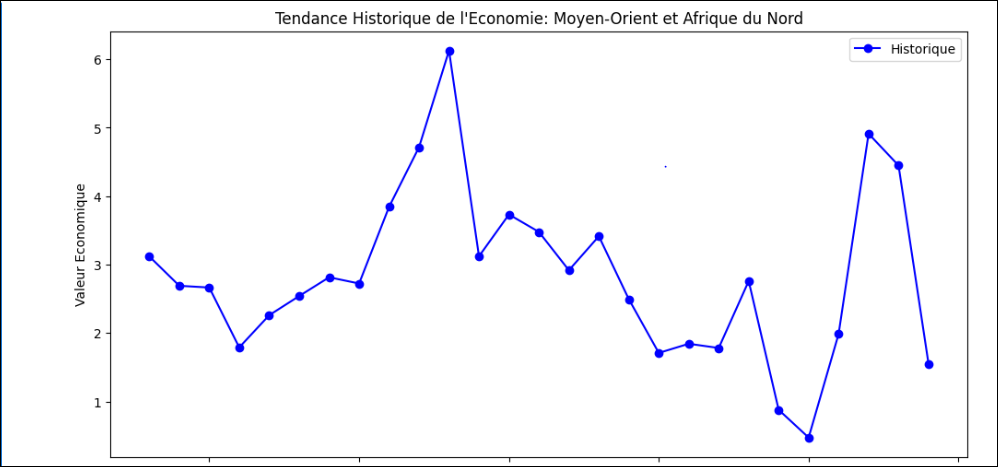
Le graphique généré par ce code permet de :

* **Comparer l'évolution des moyennes économiques** entre différentes régions au fil du temps.
* **Identifier les tendances:** Croissance, décroissance, stabilité.
* **Détecter des points d'inflexion:** Identifier les années où les tendances se sont inversées.
* **Comparer les performances économiques** des différentes régions.

Ce code est un outil puissant pour analyser des données économiques temporelles et visualiser les tendances à long terme. Il permet de comparer les performances de différentes régions et d'identifier les périodes de croissance ou de récession.

**Modélisation et Prédiction de l'évolution économique de la région Moyen Orient & Afrique du nord (Tunisie) dans 10 ans**



  Ce code vise à :

1. **Charger et préparer des données économiques:** Le code commence par charger un fichier CSV contenant des données économiques, probablement sur le Moyen-Orient et l'Afrique du Nord. Il nettoie les données en vérifiant la présence et le format de la colonne "Date".
2. **Visualiser la tendance historique:** Un graphique linéaire est créé pour afficher l'évolution de l'indicateur économique principal (ici, "EMDE Middle East & N. Africa") au fil du temps.
3. **Effectuer une prédiction:** Un modèle de régression linéaire est entraîné sur les données historiques pour prédire les valeurs futures de l'indicateur.
4. **Évaluer le modèle:** Les performances du modèle sont évaluées à l'aide de métriques d'erreur comme l'erreur absolue moyenne (MAE), l'erreur quadratique moyenne (MSE) et le coefficient de détermination (R²).

**Étapes clés du code :**

1. **Importation des bibliothèques:**
   * pandas: Pour la manipulation des données.
   * matplotlib.pyplot: Pour la création de graphiques.
   * sklearn: Pour les modèles de machine learning (ici, la régression linéaire).
2. **Préparation des données:**
   * **Nettoyage:** Vérification de la présence de la colonne "Date" et conversion en format numérique (année).
   * **Transformation:** Création d'un DataFrame spécifique pour l'analyse de la région cible.
   * **Division des données:** Séparation des données en ensembles d'entraînement et de test pour évaluer le modèle.
3. **Modélisation:**
   * **Régression linéaire:** Un modèle de régression linéaire simple est utilisé pour prédire les valeurs futures de l'indicateur économique en fonction du temps.
4. **Évaluation du modèle:**
   * Calcul des métriques d'erreur pour évaluer la performance du modèle.
5. **Visualisation:**
   * **Tendance historique:** Un graphique linéaire affiche l'évolution de l'indicateur au fil du temps.
   * **Prédictions futures:** Les prédictions du modèle sont ajoutées au graphique pour visualiser la tendance future.

Ce code fournit une base solide pour l'analyse de séries temporelles économiques. Il permet de visualiser les tendances passées et de faire des prévisions pour l'avenir. Cependant, il est important de garder à l'esprit que les prévisions économiques sont toujours soumises à un degré d'incertitude et que les modèles doivent être régulièrement mis à jour et affinés.