```
# Import des librairies nécessaires
import pandas as pd, numpy as np, matplotlib.pyplot as plt, seaborn as sns
import ast
```

```
# Montage du drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remoun
```

```
# Installation du module pour lecture des xlsb
!pip install pyxlsb
```

Afficher la sortie masquée

Fichier All Publish VideoGame 2022

```
# Import du fichier "Liste de tous les jeux avec note des joueurs"
df_listejeux = pd.read_excel('/content/drive/MyDrive/Datasets/UOI Games/All_publish_VideoGame_2022.xlsb', engine='pyxlsb')
```

```
# Installation du module de profiling
!pip install ydata-profiling
```

Afficher la sortie masquée

```
# Import de Profile Report
from ydata_profiling import ProfileReport
```

```
# Production du rapport de Profiling du df_listejeux
profile = ProfileReport(df_listejeux)
# profile.to_notebook_iframe() # commenter / désactiver pour accélérer le rechargement du notebook
```

```
df_listejeux.info()
```

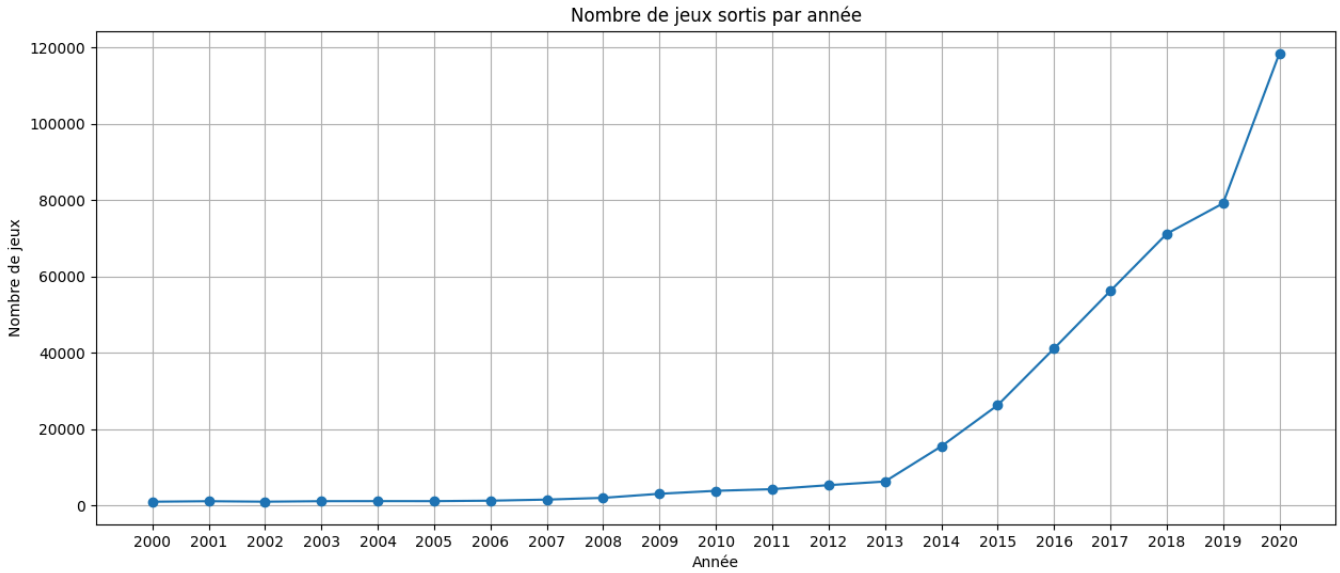
Afficher la sortie masquée

Commencez à coder ou à générer avec l'IA.

Nombre de jeux sortis par année

```
# Nombre de jeux par année
sorties_par_annee = df_listejeux['Annee'].value_counts().sort_index()
```

```
# Création du graphique
plt.figure(figsize=(15,6))
plt.plot(sorties_par_annee.index, sorties_par_annee.values, marker='o')
plt.title("Nombre de jeux sortis par année")
plt.xlabel("Année")
plt.ylabel("Nombre de jeux")
plt.xticks(ticks=sorties_par_annee.index) # Défini les ticks de l'axe x en tant que nombres entiers (années)
plt.grid(True)
plt.show()
```

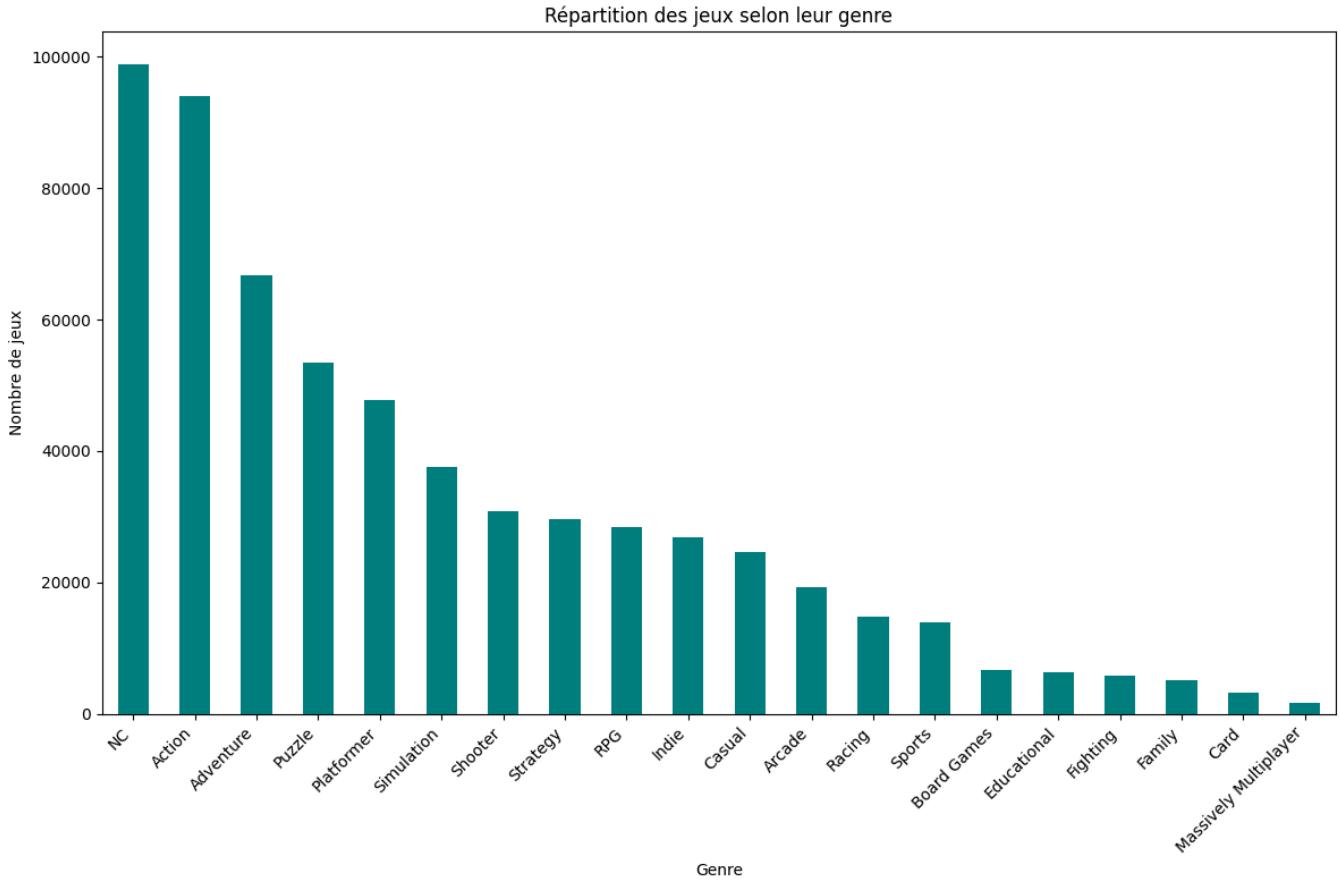


Répartition des jeux selon leur genres

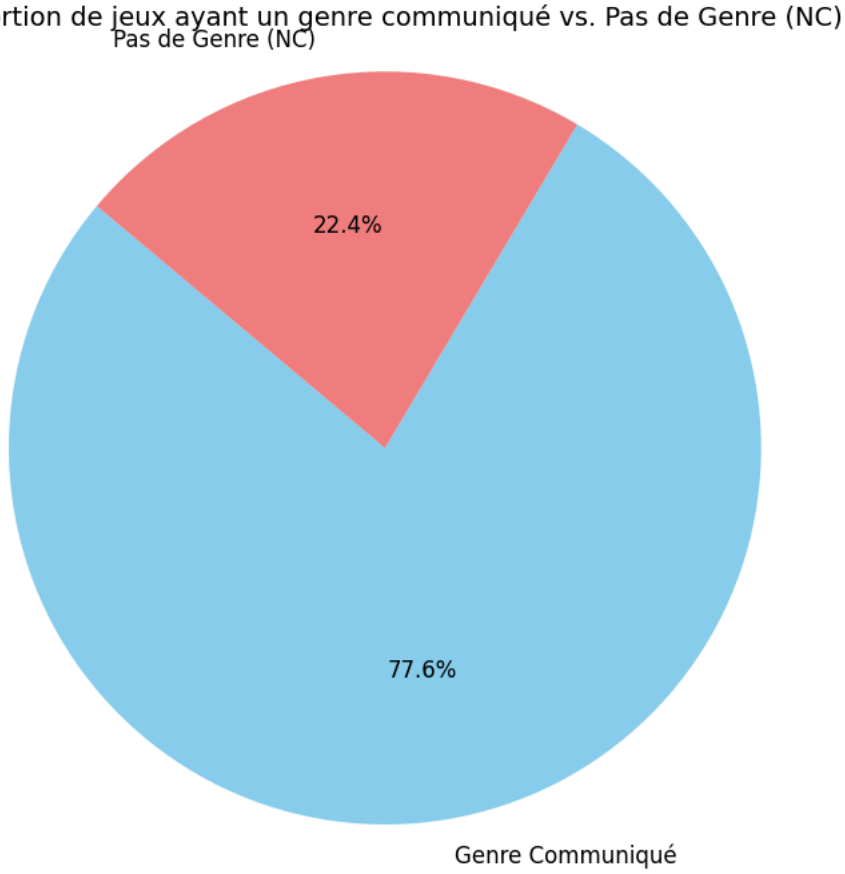
```
# Remplacer à la fois les NaN et les chaînes vides par "NC"
df_listejeux['genres'] = df_listejeux['genres'].replace(['', None], 'NC')

# Vérification après remplacement
df_listejeux['genres'].isna().sum(), (df_listejeux['genres'] == 'NC').sum()
```

```
(0, 98902)
```

```
# Séparer les genres et les mettre dans une liste
# On utilise .str.split() pour séparer les genres en plusieurs éléments dans une liste
genres_list = df_listejeux['genres'].dropna().str.split('\|\|', expand=False)


# genres_list.head(10)


# Compter la fréquence de chaque genre
# On utilise la fonction explode pour aplatir la liste des genres
genres_flat = genres_list.explode()
genre_counts = genres_flat.value_counts()


# Création du graphique
plt.figure(figsize=(12,8))
genre_counts.plot(kind='bar', color='teal')

# Ajouter un titre et des étiquettes
plt.title("Répartition des jeux selon leur genre")
plt.xlabel("Genre")
plt.ylabel("Nombre de jeux")
plt.xticks(rotation=90)

# Incliner les labels des genres pour une meilleure lisibilité
plt.xticks(rotation=45, ha="right")

# Afficher le graphique
plt.tight_layout()  # Pour éviter que les labels se chevauchent
plt.show()
```

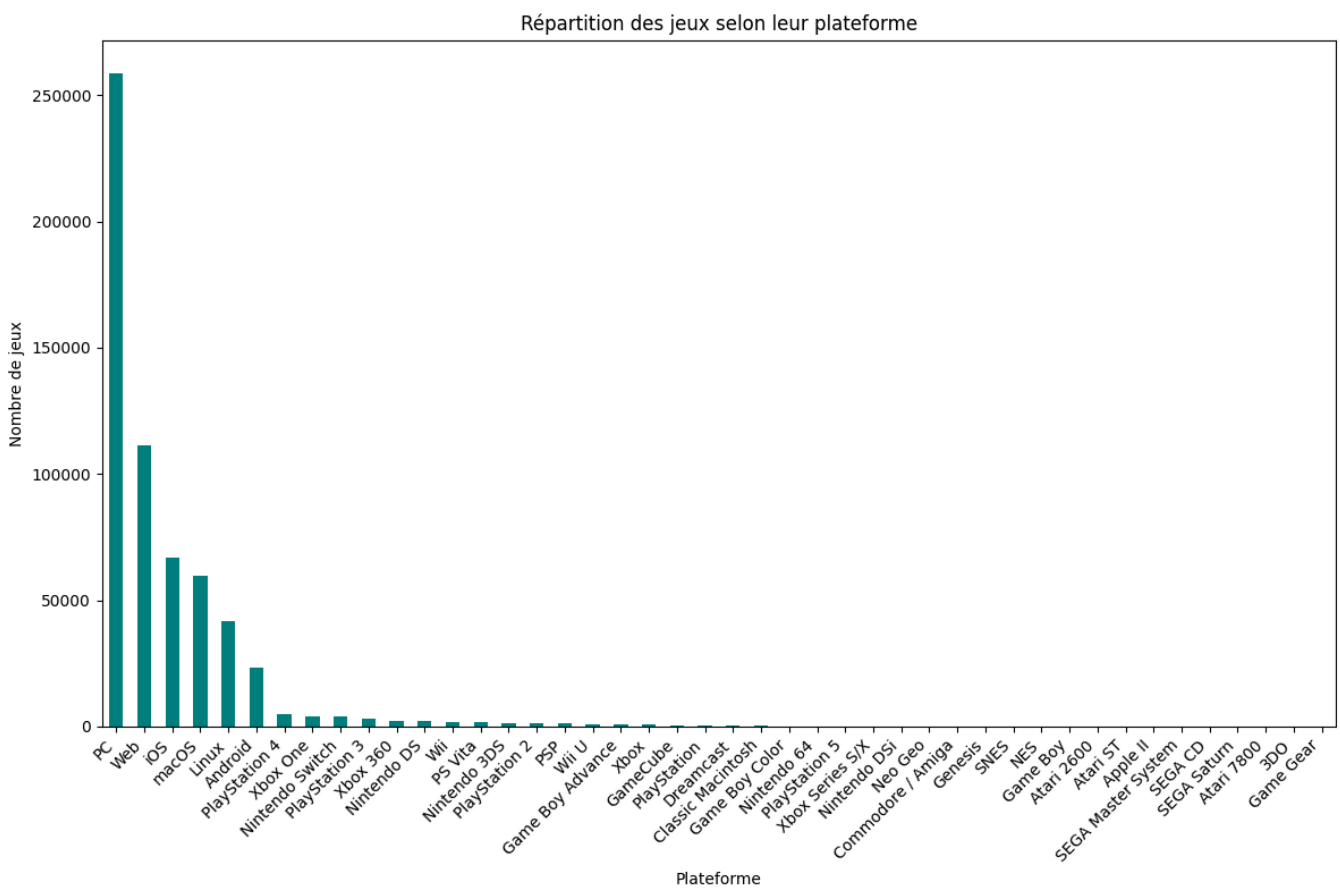



```
# Calculer la proportion de jeux avec et sans genre communiqué
genre_counts = df_listejeux['genres'].value_counts()
proportion_nc = genre_counts.get('NC', 0)
proportion_with_genre = genre_counts.sum() - proportion_nc

# Création du graphique en camembert
labels = ['Genre Communiqué', 'Pas de Genre (NC)']
sizes = [proportion_with_genre, proportion_nc]
colors = ['skyblue', 'lightcoral']

# Création du graphique
plt.figure(figsize=(8, 8))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140, colors=colors, textprops={'fontsize': 12})
plt.title("Proportion de jeux ayant un genre communiqué vs. Pas de Genre (NC)", fontsize=14)

# Affichage
plt.axis('equal')  # Pour s'assurer que le graphique est un cercle
plt.show()
```

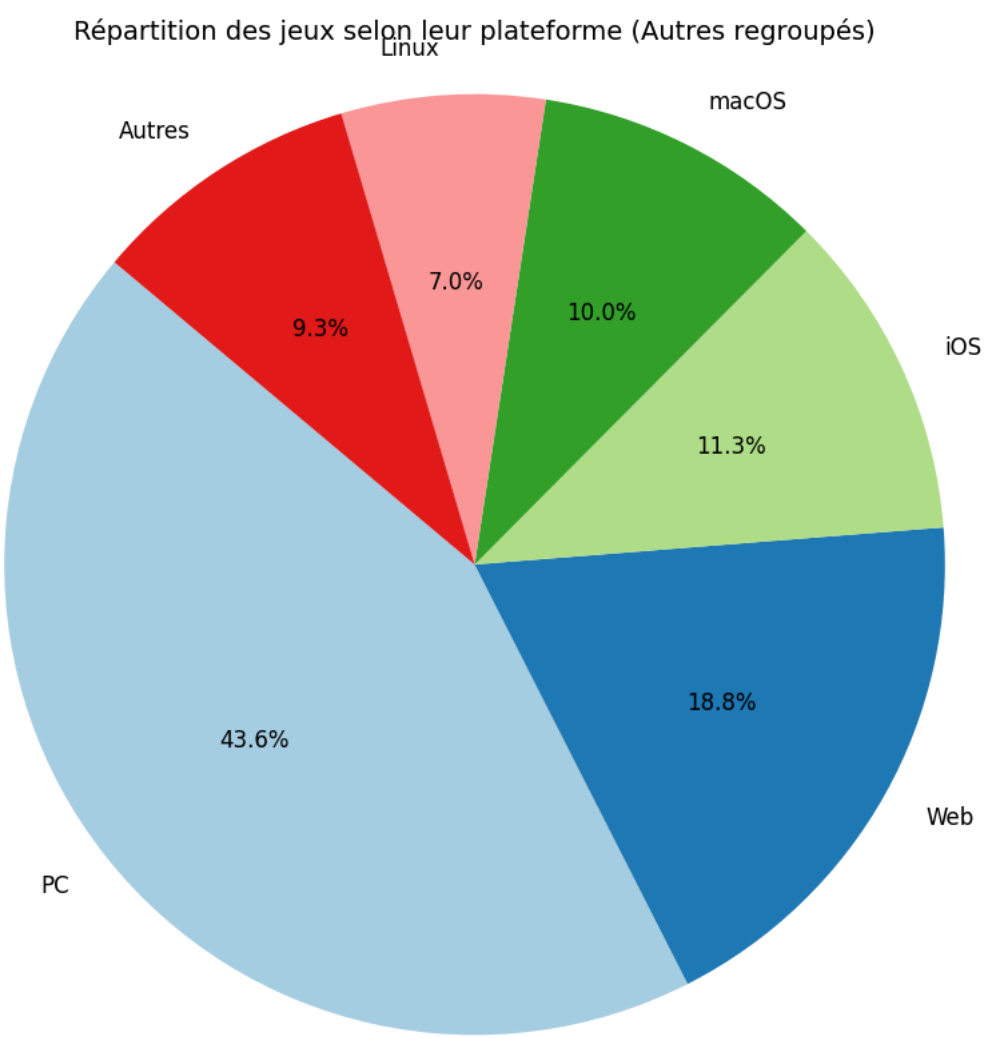


Répartition des jeux par plateforme

```
# Séparer les plateformes et les mettre dans une liste
platforms_list = df_listejeux['platforms'].dropna().str.split('\|\|', expand=False)

# Compter la fréquence de chaque plateforme
platforms_flat = platforms_list.explode()
platform_counts = platforms_flat.value_counts()

# Création du graphique
plt.figure(figsize=(12,8))
platform_counts.plot(kind='bar', color='teal')

# Ajouter un titre et des étiquettes
plt.title("Répartition des jeux selon leur plateforme")
plt.xlabel("Plateforme")
plt.ylabel("Nombre de jeux")

# Incliner les labels des plateformes pour une meilleure lisibilité
plt.xticks(rotation=45, ha="right")

# Ajuster la mise en page
plt.tight_layout()

# Afficher le graphique
plt.show()
```

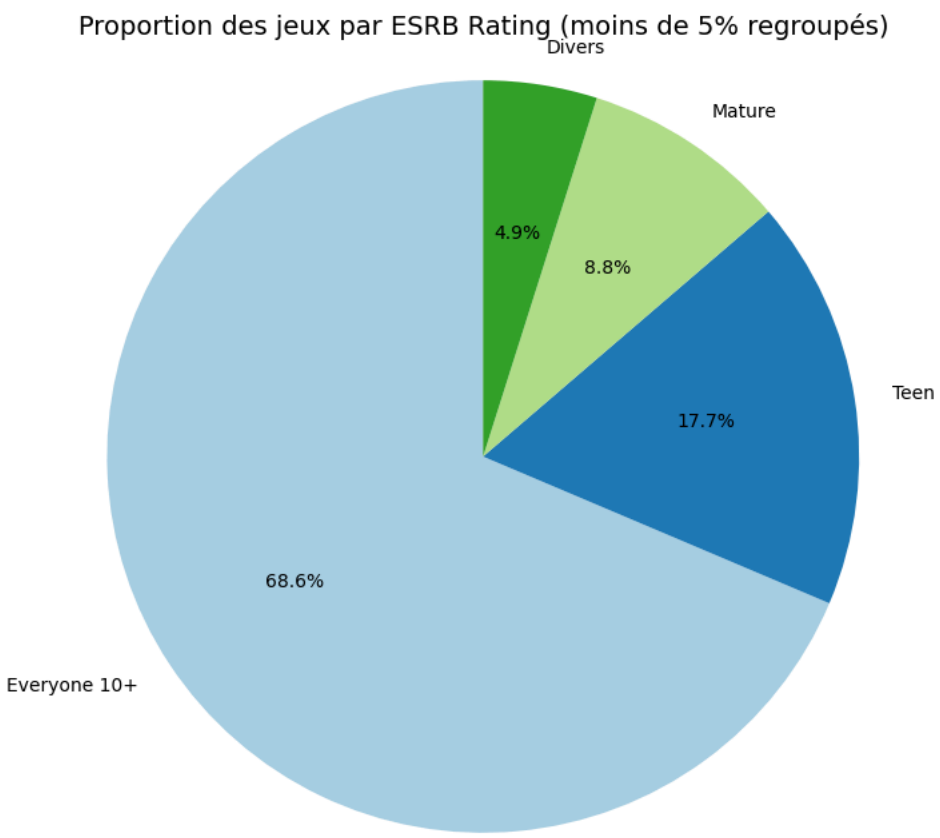



```
# Séparer les plateformes et les mettre dans une liste
platforms_list = df_listejeux['platforms'].dropna().str.split('\|\|', expand=False)

# Compter la fréquence de chaque plateforme
platforms_flat = platforms_list.explode()
platform_counts = platforms_flat.value_counts()

# Regrouper les plateformes représentant moins de 5%
threshold = 0.05  # Seuil de 5%
total_games = platform_counts.sum()

# Calculer la proportion et regrouper les petites catégories
platform_counts['Autres'] = platform_counts[platform_counts / total_games < threshold].sum()
platform_counts = platform_counts[platform_counts / total_games >= threshold]

# Création du graphique en camembert
plt.figure(figsize=(10, 10))
plt.pie(platform_counts, labels=platform_counts.index, autopct='%1.1f%%', startangle=140, colors=plt.cm.Paired.colors, textp

# Ajouter un titre
plt.title("Répartition des jeux selon leur plateforme (Autres regroupés)", fontsize=14)

# Afficher le graphique
plt.axis('equal')  # Assure que le graphique est un cercle
plt.show()
```

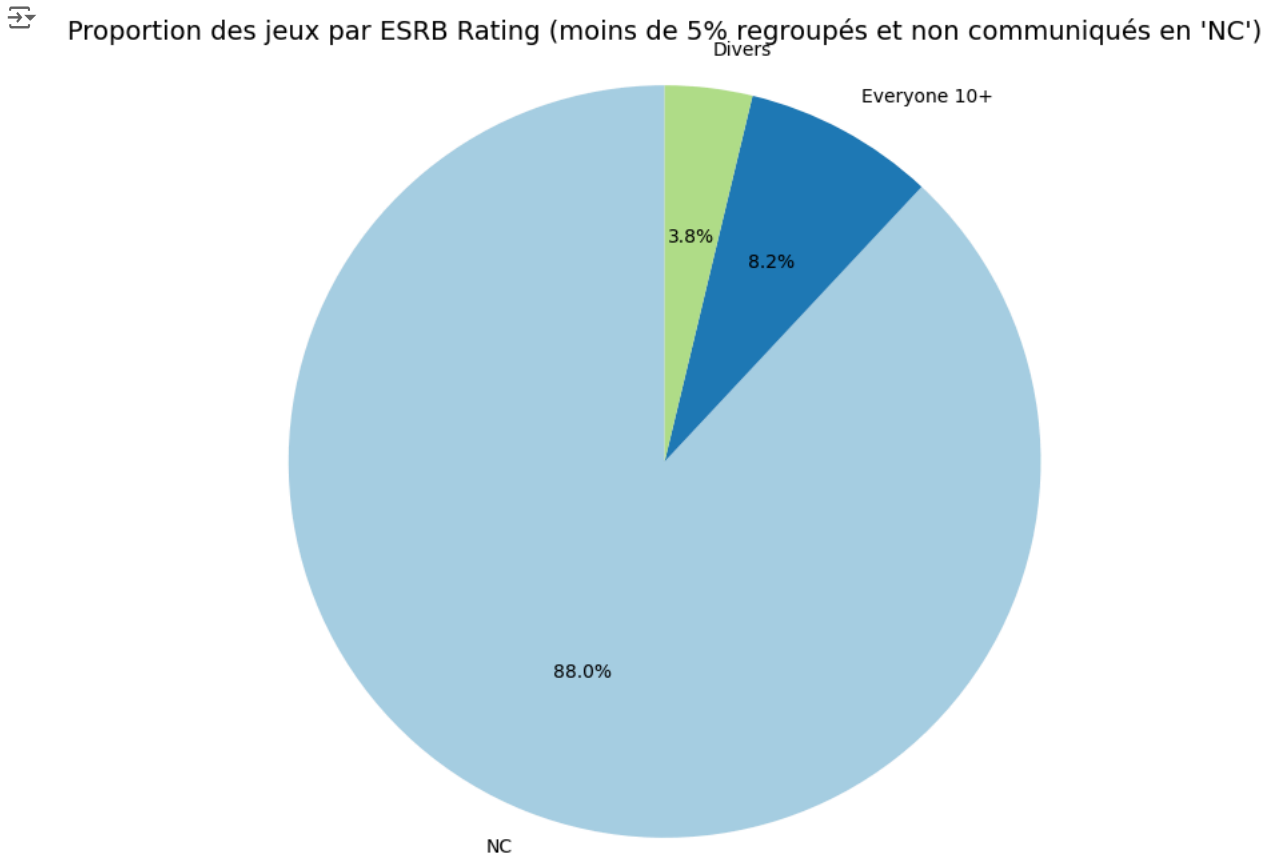


Répartition des jeux par classification d'âge

```
# Compter le nombre de jeux par ESRB Rating
proportion_esrb = df_listejeux['esrb_rating'].value_counts()

# Calculer le pourcentage de chaque catégorie
pourcentages = proportion_esrb / proportion_esrb.sum() * 100

# Regrouper les catégories avec moins de 5% dans une catégorie 'Divers'
proportion_esrb_regroupe = pourcentages[pourcentages >= 5].copy()
proportion_esrb_regroupe['Divers'] = pourcentages[pourcentages < 5].sum()

# Création du graphique camembert
plt.figure(figsize=(8, 8))
plt.pie(proportion_esrb_regroupe, labels=proportion_esrb_regroupe.index, autopct='%1.1f%%', startangle=90, colors=plt.cm.Pai

# Titre
plt.title("Proportion des jeux par ESRB Rating (moins de 5% regroupés)", fontsize=14)

# Affichage du graphique
plt.axis('equal')  # Pour un camembert circulaire
plt.show()
```

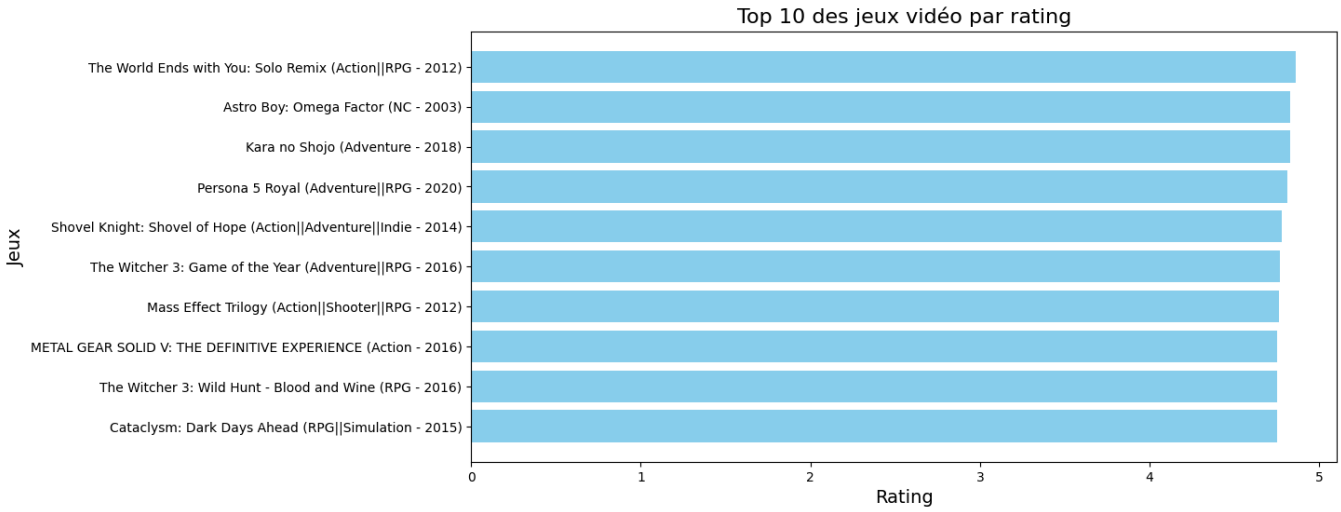



```
# Compter le nombre de jeux par ESRB Rating, en incluant les NaN comme 'NC'
proportion_esrb = df_listejeux['esrb_rating'].value_counts(dropna=False)

# Remplacer les valeurs NaN par 'NC'
proportion_esrb.index = proportion_esrb.index.fillna('NC')

# Calculer le pourcentage de chaque catégorie
pourcentages = proportion_esrb / proportion_esrb.sum() * 100

# Regrouper les catégories avec moins de 5% dans une catégorie 'Divers'
proportion_esrb_regroupe = pourcentages[pourcentages >= 5].copy()
proportion_esrb_regroupe['Divers'] = pourcentages[pourcentages < 5].sum()

# Création du graphique camembert
plt.figure(figsize=(8, 8))
plt.pie(proportion_esrb_regroupe, labels=proportion_esrb_regroupe.index, autopct='%1.1f%%', startangle=90, colors=plt.cm.Pai

# Titre
plt.title("Proportion des jeux par ESRB Rating (moins de 5% regroupés et non communiqués en 'NC')", fontsize=14)

# Affichage du graphique
plt.axis('equal')  # Pour un camembert circulaire
plt.show()
```

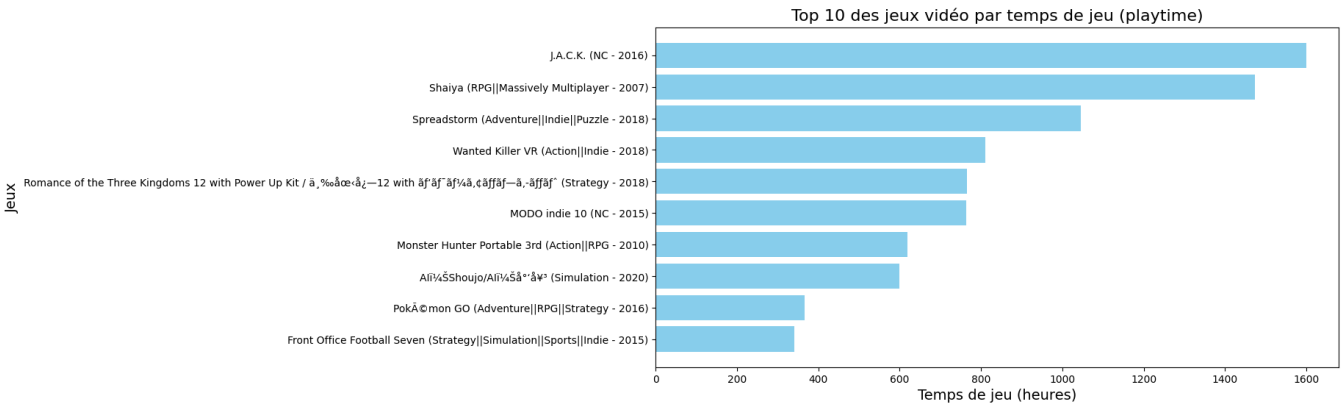


TOP 10

```
# Trier le DataFrame par 'rating' en ordre décroissant et sélectionner les 10 premiers
top_10_games = df_listejeux.sort_values(by='rating', ascending=False).head(10)

# Créer une nouvelle colonne pour afficher le nom du jeu, le genre et l'année sous le format : "nom (genre - année)"
top_10_games['label'] = top_10_games['name'] + ' (' + top_10_games['genres'] + ' - ' + top_10_games['Annee'].astype(str) + '

# Création du graphique à barres
plt.figure(figsize=(12, 6))
plt.barh(top_10_games['label'], top_10_games['rating'], color='skyblue')

# Titre et labels
plt.title('Top 10 des jeux vidéo par rating', fontsize=16)
plt.xlabel('Rating', fontsize=14)
plt.ylabel('Jeux', fontsize=14)

# Affichage du graphique
plt.gca().invert_yaxis()  # Inverser l'axe y pour avoir le jeu avec le rating le plus élevé en haut
plt.show()
```

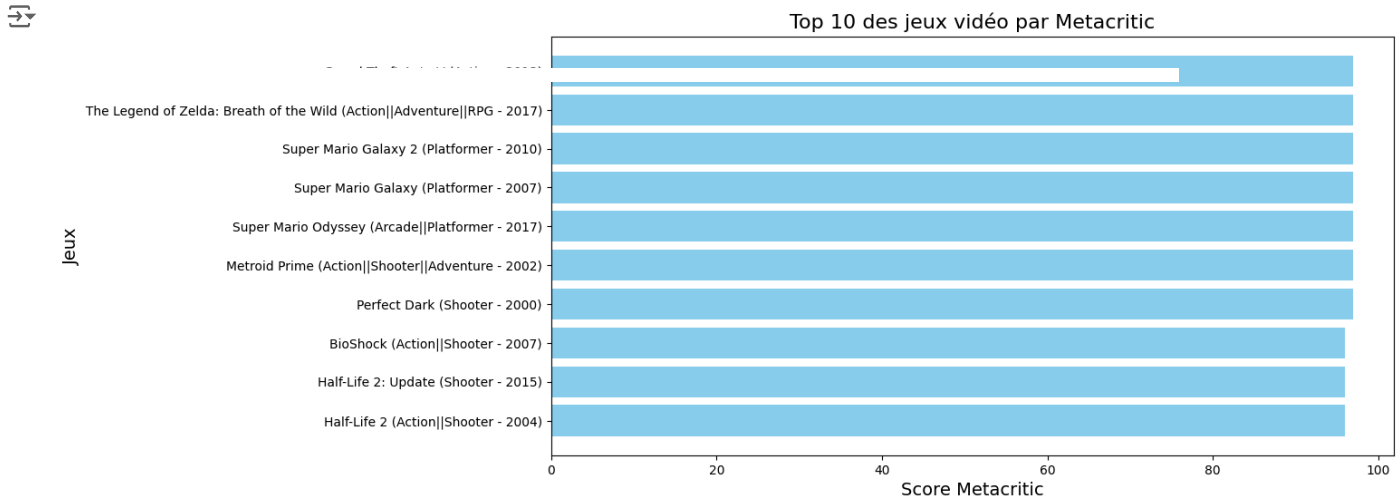



```
# Trier le DataFrame par 'playtime' en ordre décroissant et sélectionner les 10 premiers
top_10_playtime = df_listejeux.sort_values(by='playtime', ascending=False).head(10)

# Créer une nouvelle colonne pour afficher le nom du jeu, le genre et l'année sous le format : "nom (genre - année)"
top_10_playtime['label'] = top_10_playtime['name'] + ' (' + top_10_playtime['genres'] + ' - ' + top_10_playtime['Annee'].ast

# Création du graphique à barres
plt.figure(figsize=(12, 6))
plt.barh(top_10_playtime['label'], top_10_playtime['playtime'], color='skyblue')

# Titre et labels
plt.title('Top 10 des jeux vidéo par temps de jeu (playtime)', fontsize=16)
plt.xlabel('Temps de jeu (heures)', fontsize=14)
plt.ylabel('Jeux', fontsize=14)

# Inverser l'axe y pour avoir le jeu avec le playtime le plus élevé en haut
plt.gca().invert_yaxis()

# Affichage du graphique
plt.show()
```

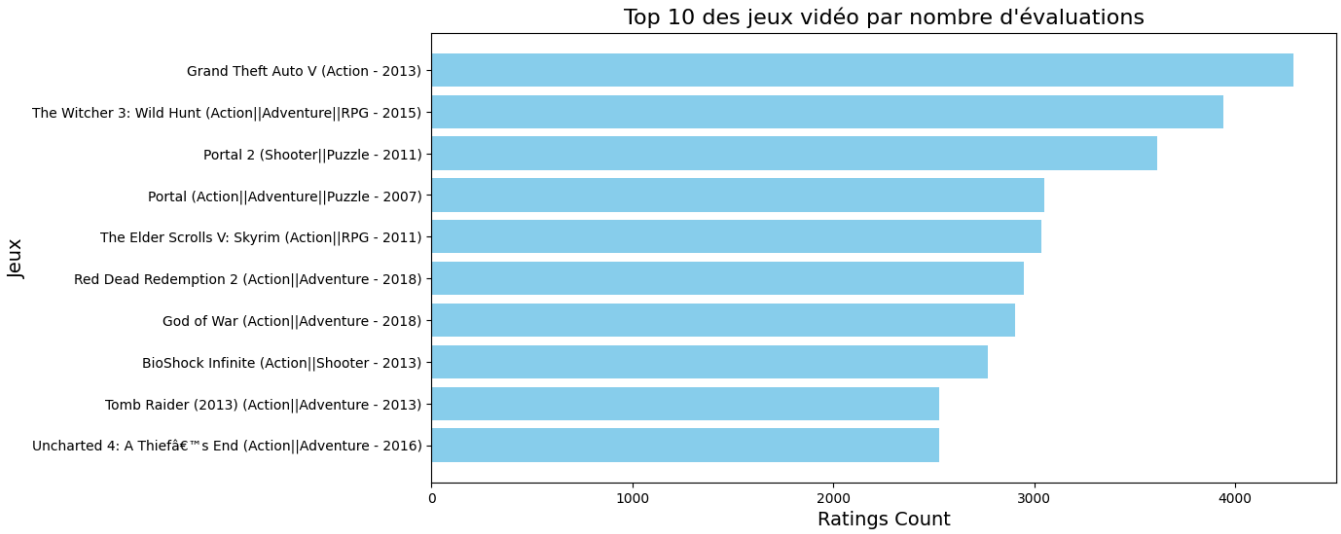



```
# Trier le DataFrame par 'metacritic' en ordre décroissant et sélectionner les 10 premiers
top_10_metacritic = df_listejeux.sort_values(by='metacritic', ascending=False).head(10)
```

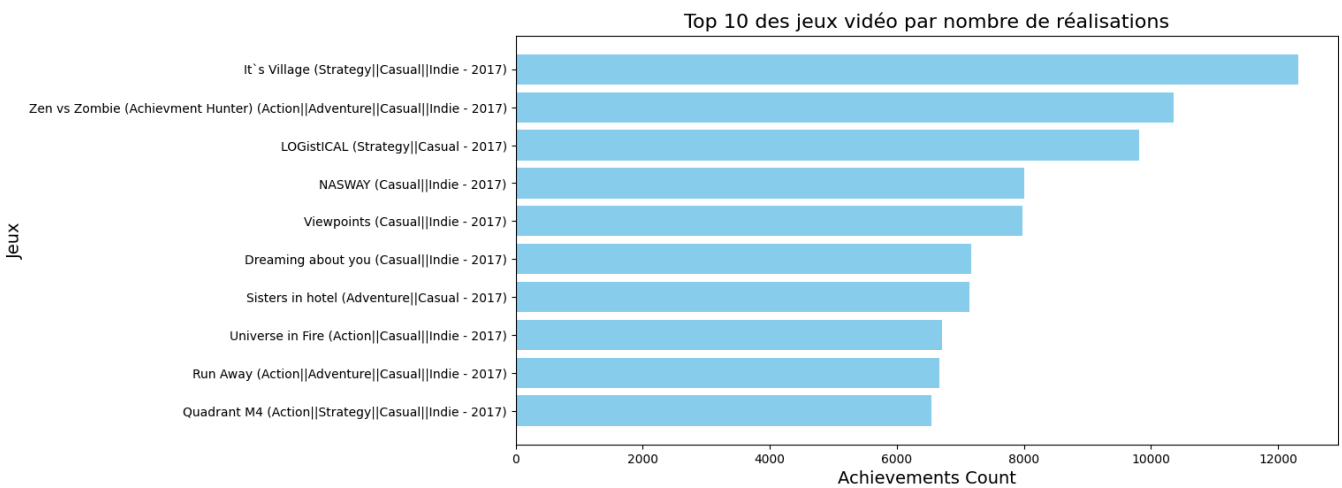
```
# Créer une nouvelle colonne pour afficher le nom du jeu, le genre et l'année sous le format : "nom (genre - année)"
top_10_metacritic['label'] = top_10_metacritic['name'] + ' (' + top_10_metacritic['genres'] + ' - ' + top_10_metacritic['Anr


# Créer le graphique à barres horizontales
plt.figure(figsize=(12, 6))
plt.barh(top_10_metacritic['label'], top_10_metacritic['metacritic'], color='skyblue')

# Titre et labels
plt.title('Top 10 des jeux vidéo par Metacritic', fontsize=16)
plt.xlabel('Score Metacritic', fontsize=14)
plt.ylabel('Jeux', fontsize=14)

# Inverser l'axe y pour avoir le jeu avec le playtime le plus élevé en haut
plt.gca().invert_yaxis()

# Affichage du graphique
plt.show()
```




```
# Trier le DataFrame par 'ratings_count' en ordre décroissant et sélectionner les 10 premiers
top_10_ratingscount = df_listejeux.sort_values(by='ratings_count', ascending=False).head(10)

# Créer une nouvelle colonne pour afficher le nom du jeu, le genre et l'année sous le format : "nom (genre - année)"
top_10_ratingscount['label'] = top_10_ratingscount['name'] + ' (' + top_10_ratingscount['genres'] + ' - ' + top_10_ratingscc

# Création du graphique à barres
plt.figure(figsize=(12, 6))
plt.barh(top_10_ratingscount['label'], top_10_ratingscount['ratings_count'], color='skyblue')

# Titre et labels
plt.title("Top 10 des jeux vidéo par nombre d'évaluations", fontsize=16)
plt.xlabel('Ratings Count', fontsize=14)
plt.ylabel('Jeux', fontsize=14)

# Affichage du graphique
plt.gca().invert_yaxis()  # Inverser l'axe y pour avoir le jeu avec le rating le plus élevé en haut
plt.show()
```




```
# Trier le DataFrame par 'achievements_count' en ordre décroissant et sélectionner les 10 premiers
top_10_achievementscount = df_listejeux.sort_values(by='achievements_count', ascending=False).head(10)

# Créer une nouvelle colonne pour afficher le nom du jeu, le genre et l'année sous le format : "nom (genre - année)"
top_10_achievementscount['label'] = top_10_achievementscount['name'] + ' (' + top_10_achievementscount['genres'] + ' - ' + t

# Création du graphique à barres
plt.figure(figsize=(12, 6))
plt.barh(top_10_achievementscount['label'], top_10_achievementscount['achievements_count'], color='skyblue')

# Titre et labels
plt.title("Top 10 des jeux vidéo par nombre de réalisations", fontsize=16)
plt.xlabel('Achievements Count', fontsize=14)
plt.ylabel('Jeux', fontsize=14)

# Affichage du graphique
plt.gca().invert_yaxis()  # Inverser l'axe y pour avoir le jeu avec le rating le plus élevé en haut
plt.show()
```



Fichier VideoGame Sales 1980-2016

```
# Import du fichier "Liste des ventes de jeux vidéos"
df_ventesjeux = pd.read_excel('/content/drive/MyDrive/Datasets/UOI Games/VideoGame_Sales_1980-2016.xlsb', sheet_name='vgsale
```

```
df_ventesjeux.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16545 entries, 0 to 16544
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Rank          16545 non-null  int64
 1   Name          16545 non-null  object
 2   Platform      16545 non-null  object
 3   Year          16274 non-null  float64
 4   Genre         16545 non-null  object
 5   Publisher     16487 non-null  object
 6   Constructeur  16545 non-null  object
 7   Portable      16545 non-null  object
 8   NA_Sales      16545 non-null  float64
 9   EU_Sales      16545 non-null  float64
 10  JP_Sales      16545 non-null  float64
 11  Other_Sales   16545 non-null  float64
 12  Global_Sales  16545 non-null  float64
dtypes: float64(6), int64(1), object(6)
memory usage: 1.6+ MB
```

```
# Vérifier le nombre de doublons complets dans le DataFrame
doublons_complets = df_ventesjeux.duplicated().sum()
print(f"Nombre de doublons complets : {doublons_complets}")
```

```
Nombre de doublons complets : 0
```

```
# Supprimer les espaces dans les noms de colonnes
df_ventesjeux.columns = df_ventesjeux.columns.str.strip() #présent dans la colonne "Portable"
```

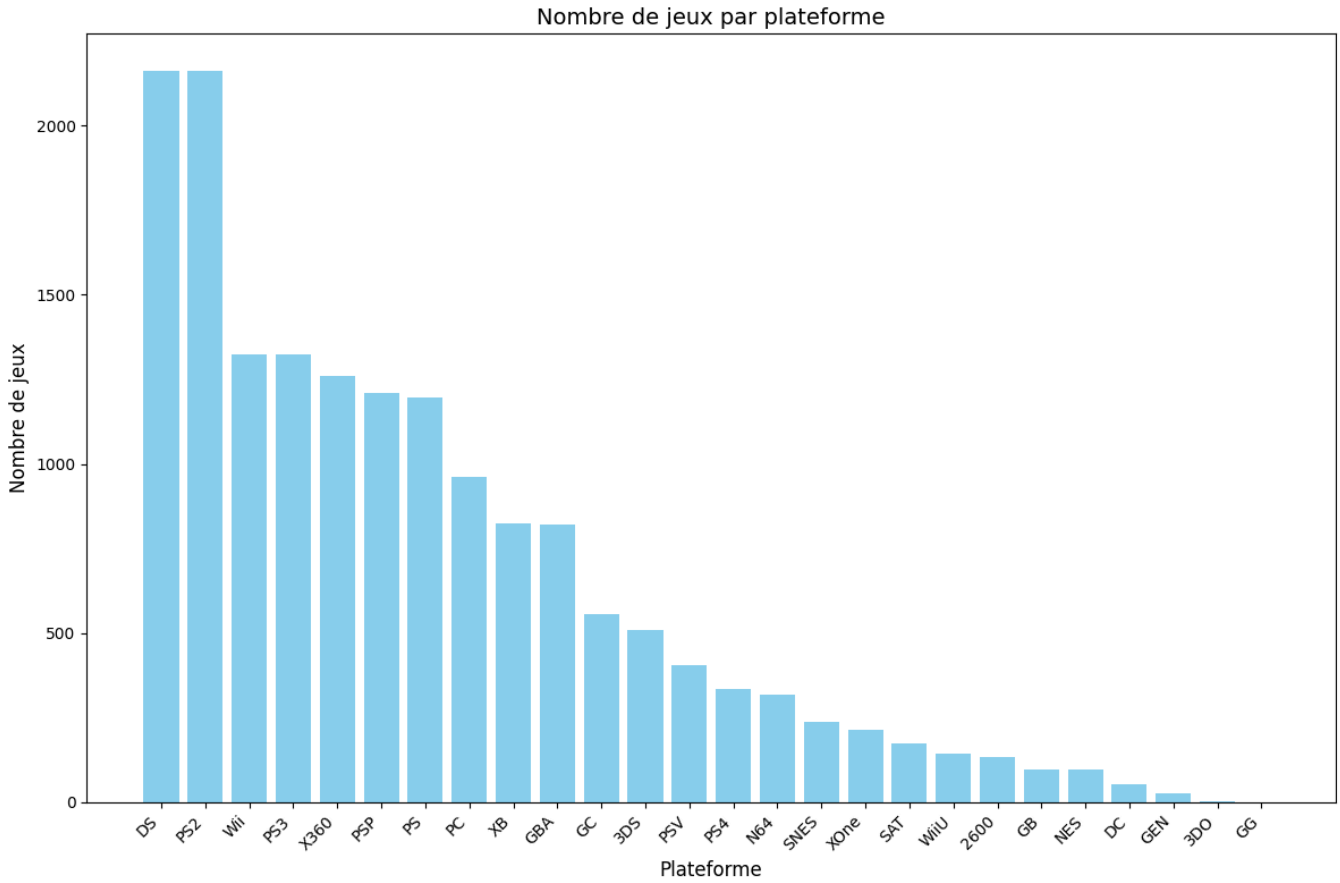
Nombre de jeux total

```
# Nombre total de jeux
nombre_de_jeux = len(df_ventesjeux)
print(f"Nombre total de jeux : {nombre_de_jeux}")
```

```
Nombre total de jeux : 16545
```

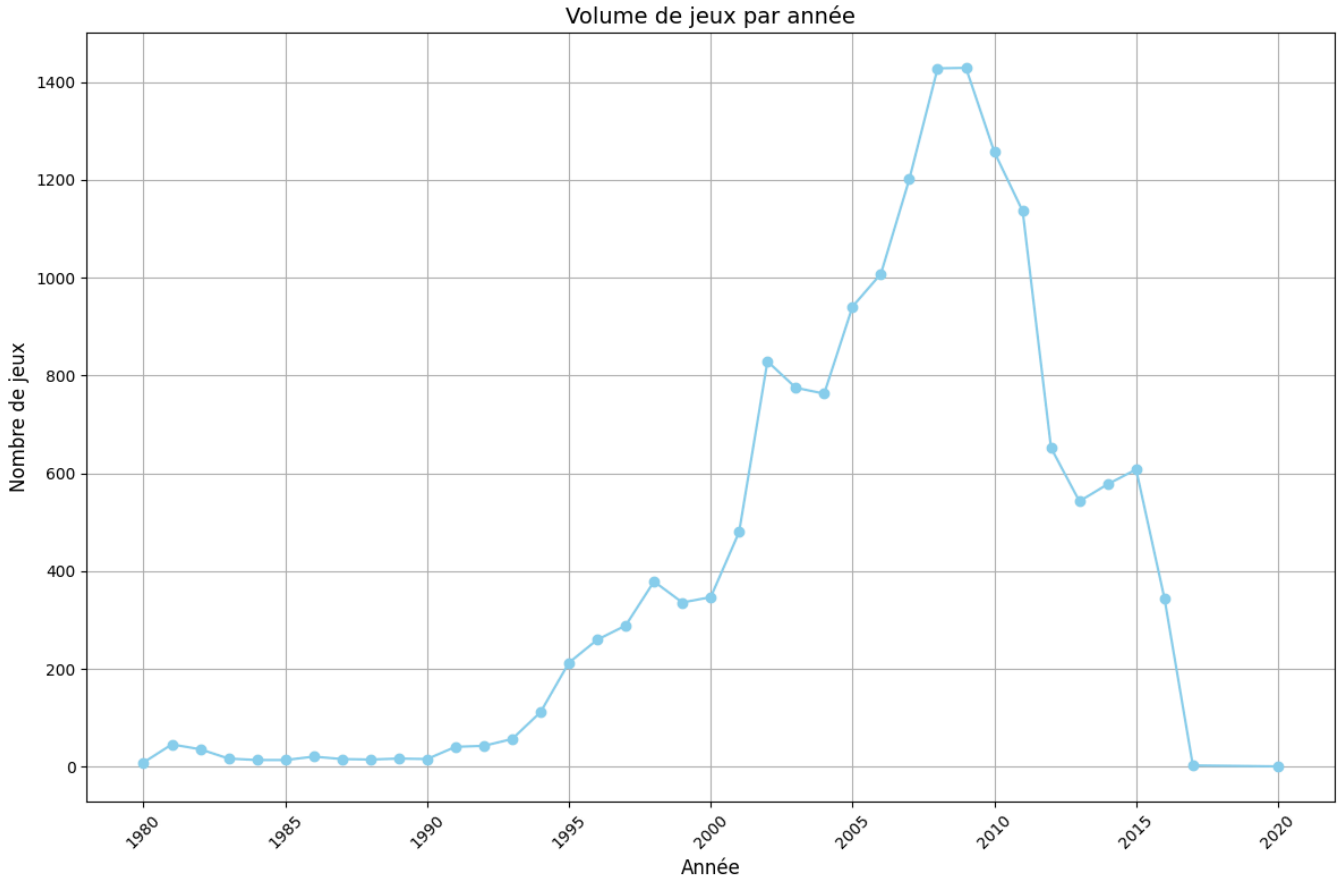
Nombre de jeux par plateforme

```
# Grouper par plateforme et compter le nombre de jeux
jeux_par_plateforme = df_ventesjeux['Platform'].value_counts()

# Convertir les index en chaînes de caractères pour éviter l'erreur
jeux_par_plateforme_index_str = jeux_par_plateforme.index.astype(str)

# Création du graphique avec Matplotlib
plt.figure(figsize=(12, 8))
plt.bar(jeux_par_plateforme_index_str, jeux_par_plateforme.values, color='skyblue')

# Titre et labels
plt.title("Nombre de jeux par plateforme", fontsize=14)
plt.xlabel("Plateforme", fontsize=12)
plt.ylabel("Nombre de jeux", fontsize=12)

# Rotation des labels de l'axe des x pour meilleure lisibilité
plt.xticks(rotation=45, ha='right')

# Affichage du graphique
plt.tight_layout()
plt.show()
```



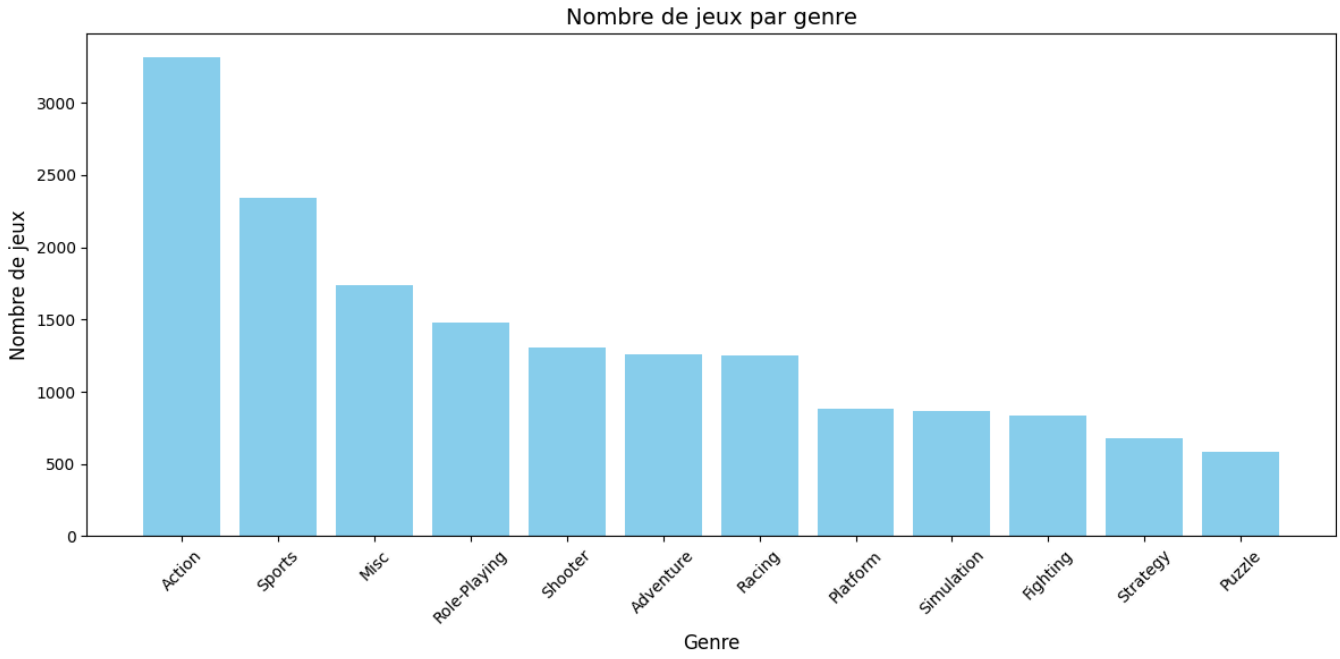
Volume de jeux par année

```
# Suppression des valeurs manquantes dans la colonne Year
df_ventesjeux_clean = df_ventesjeux.dropna(subset=['Year'])

# Groupement par année et comptage du nombre de jeux par année
jeux_par_annee = df_ventesjeux_clean.groupby('Year')['Name'].count()

# Création du graphique
plt.figure(figsize=(12, 8))
plt.plot(jeux_par_annee.index.astype(int), jeux_par_annee.values, marker='o', linestyle='-', color='skyblue')

# Titre et labels
plt.title("Volume de jeux par année", fontsize=14)
plt.xlabel("Année", fontsize=12)
plt.ylabel("Nombre de jeux", fontsize=12)

# Ajustement des labels de l'axe des x pour meilleure lisibilité
plt.xticks(rotation=45)

# Affichage du graphique
plt.tight_layout()
plt.grid(True)
plt.show()
```



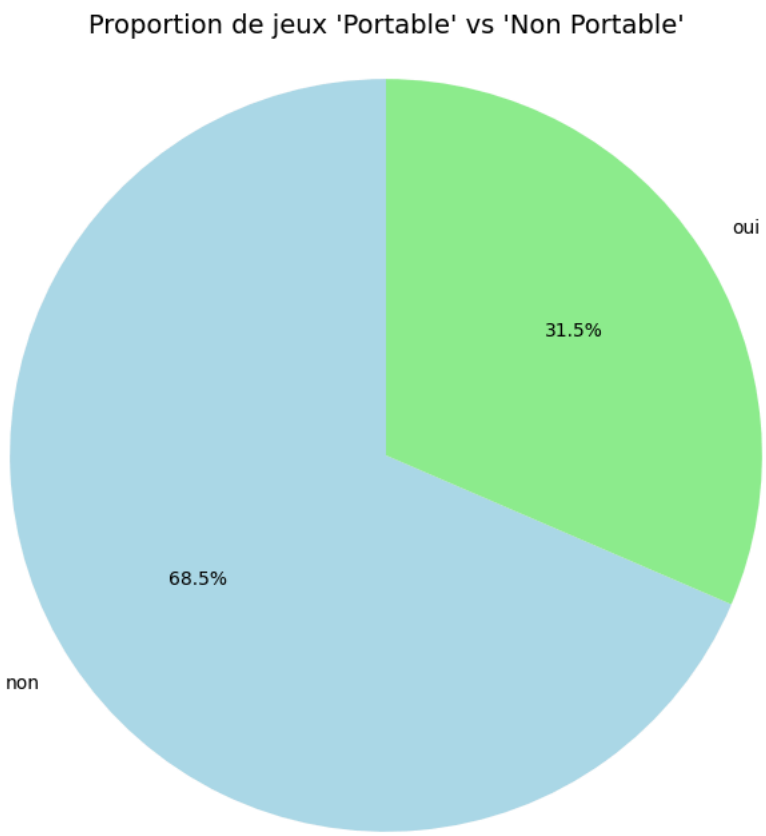
Nombre de jeux par genre

```
# Groupement par genre et comptage du nombre de jeux par genre
jeux_par_genre = df_ventesjeux.groupby('Genre')['Name'].count()

# Tri des genres par nombre de jeux
jeux_par_genre = jeux_par_genre.sort_values(ascending=False)

# Création du graphique
plt.figure(figsize=(12, 6))
plt.bar(jeux_par_genre.index, jeux_par_genre.values, color='skyblue')

# Titre et labels
plt.title("Nombre de jeux par genre", fontsize=14)
plt.xlabel("Genre", fontsize=12)
plt.ylabel("Nombre de jeux", fontsize=12)

# Rotation des labels de l'axe des x pour meilleure lisibilité
plt.xticks(rotation=45)

# Affichage du graphique
plt.tight_layout()
plt.show()
```



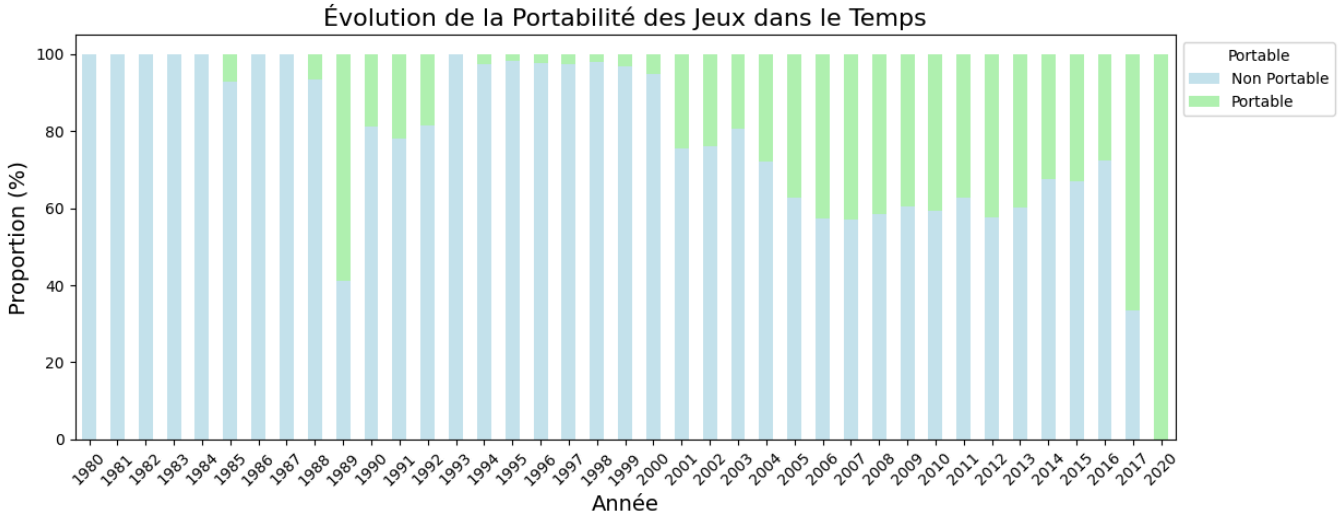
Jeux portables

```
# Convertir les valeurs de la colonne 'Portable' en minuscules
df_ventesjeux['Portable'] = df_ventesjeux['Portable'].str.lower() # pour corriger la valeur "oUI"

# Comptage des jeux "Oui" et "Non" dans la colonne "Portable"
proportion_portable = df_ventesjeux['Portable'].value_counts()

# Création du graphique camembert
plt.figure(figsize=(8, 8))
plt.pie(proportion_portable, labels=proportion_portable.index, autopct='%1.1f%%', startangle=90, colors=['lightblue', 'light

# Titre
plt.title("Proportion de jeux 'Portable' vs 'Non Portable'", fontsize=14)

# Affichage du graphique
plt.axis('equal')  # Pour un camembert circulaire
plt.show()
```

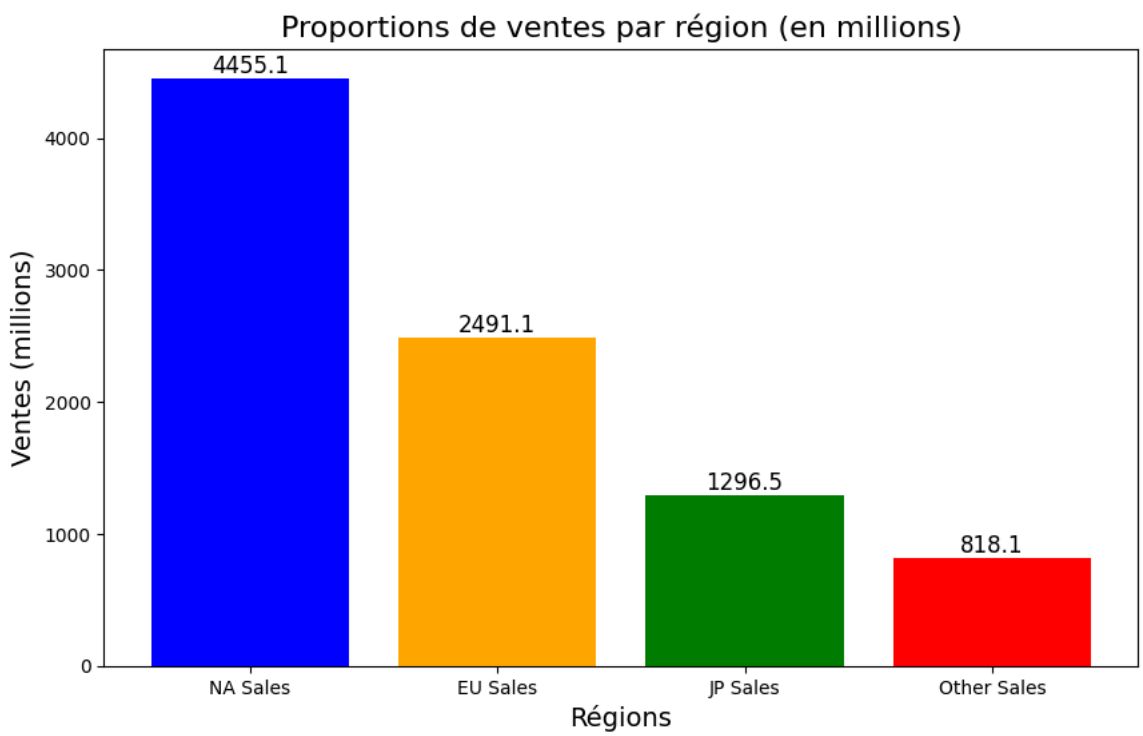



```
# Compter le nombre de jeux par année et par portabilité
evolution_portable = df_ventesjeux.groupby(['Year', 'Portable']).size().unstack(fill_value=0)

# Normaliser les valeurs pour obtenir des proportions
evolution_portable_percent = evolution_portable.div(evolution_portable.sum(axis=1), axis=0) * 100

# Création de l'histogramme empilé
plt.figure(figsize=(12, 6))
evolution_portable_percent.plot(kind='bar', stacked=True, color=['lightblue', 'lightgreen'], alpha=0.7)

# Titre et labels
plt.title("Évolution de la Portabilité des Jeux dans le Temps", fontsize=16)
plt.xlabel("Année", fontsize=14)
plt.ylabel("Proportion (%)", fontsize=14)

# Suppression des virgules des étiquettes d'années
plt.xticks(ticks=range(len(evolution_portable_percent.index)), labels=evolution_portable_percent.index.astype(int), rotatior

# Affichage de la légende à l'extérieur
plt.legend(title="Portable", loc='upper left', bbox_to_anchor=(1, 1), labels=["Non Portable", "Portable"])

# Ajuster l'espacement entre les barres
plt.subplots_adjust(right=1.7)

# Affichage du graphique
plt.xticks(rotation=45)
plt.show()
```

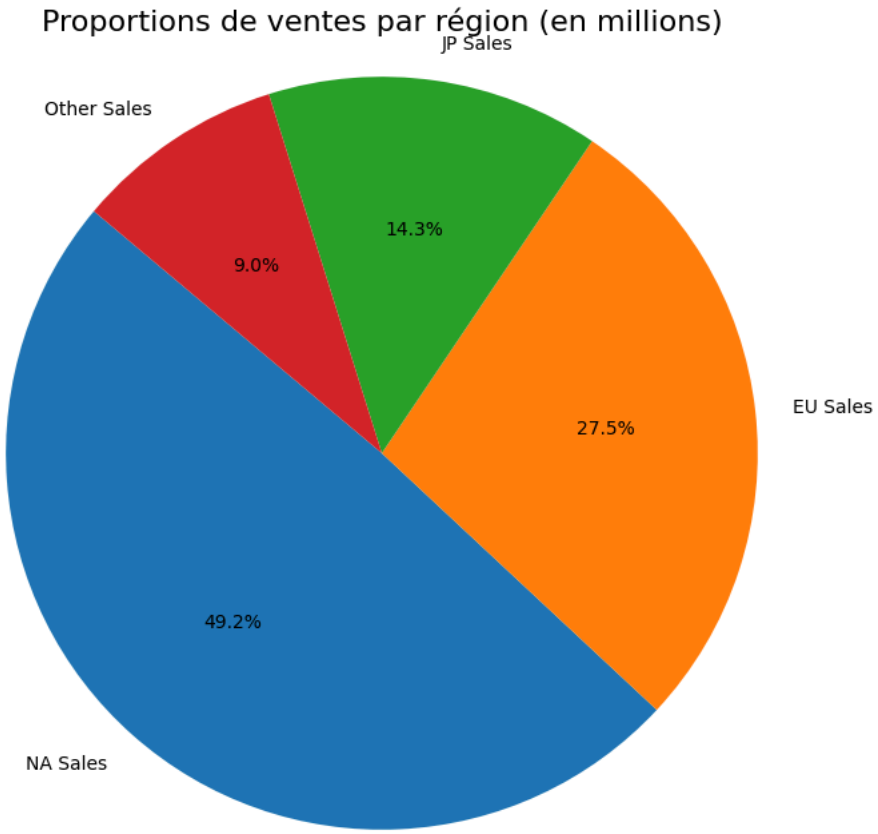
```
<Figure size 1200x600 with 0 Axes>
```




```
# Calcul des ventes totales par région
ventes_par_region = {
    'NA Sales': df_ventesjeux['NA_Sales'].sum(),
    'EU Sales': df_ventesjeux['EU_Sales'].sum(),
    'JP Sales': df_ventesjeux['JP_Sales'].sum(),
    'Other Sales': df_ventesjeux['Other_Sales'].sum()
}

# Création du graphique à barres
plt.figure(figsize=(10, 6))
plt.bar(ventes_par_region.keys(), ventes_par_region.values(), color=['blue', 'orange', 'green', 'red'])

# Titre et labels
plt.title('Proportions de ventes par région (en millions)', fontsize=16)
plt.xlabel('Régions', fontsize=14)
plt.ylabel('Ventes (millions)', fontsize=14)

# Affichage des valeurs sur les barres
for i, value in enumerate(ventes_par_region.values()):
    plt.text(i, value, f'{value:.1f}', ha='center', va='bottom', fontsize=12)

# Affichage du graphique
plt.show()
```

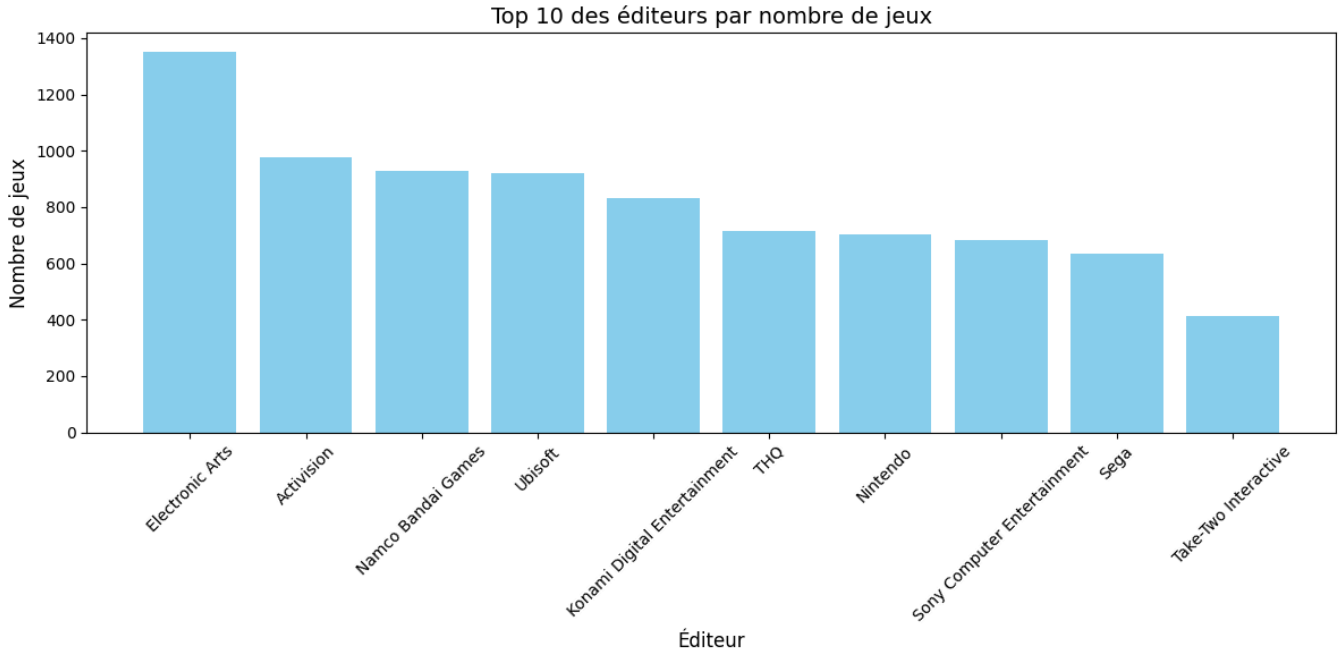


```
# Création du graphique en camembert
plt.figure(figsize=(8, 8))
plt.pie(ventes_par_region.values(), labels=ventes_par_region.keys(), autopct='%1.1f%%', startangle=140)

# Titre du graphique
plt.title('Proportions de ventes par région (en millions)', fontsize=16)

# Affichage du graphique
plt.axis('equal')  # Pour assurer que le camembert soit bien circulaire
plt.show()
```

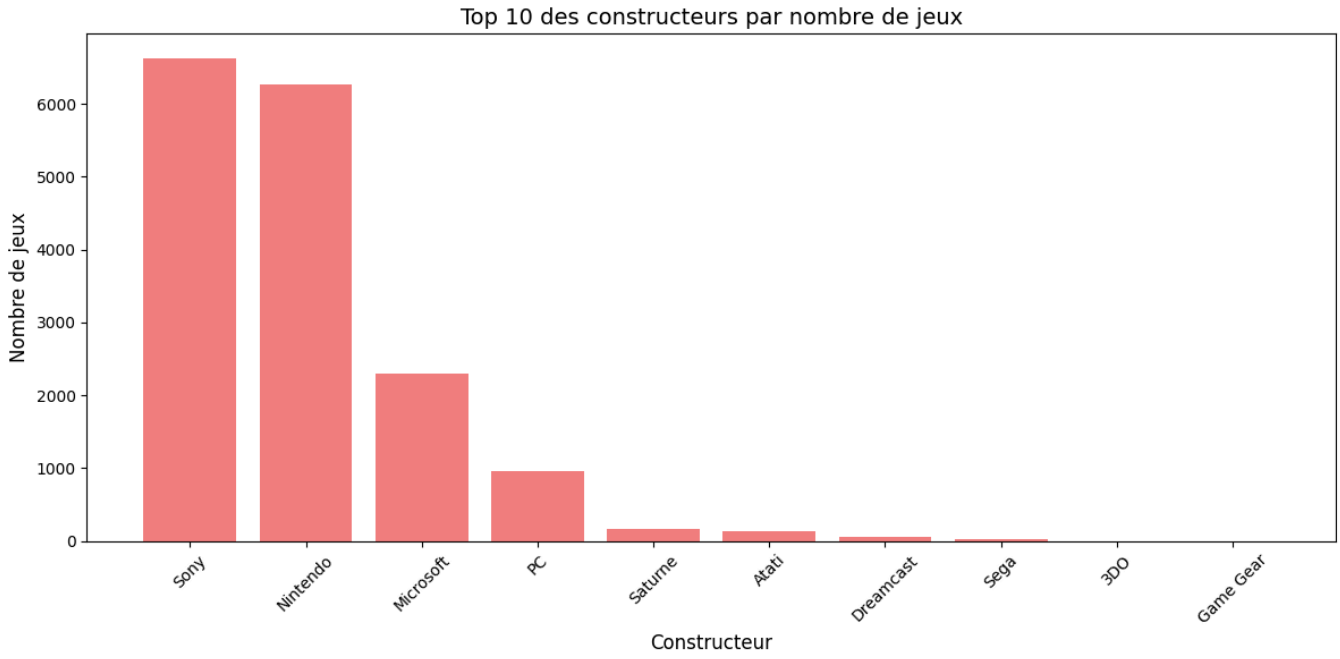


Top 10

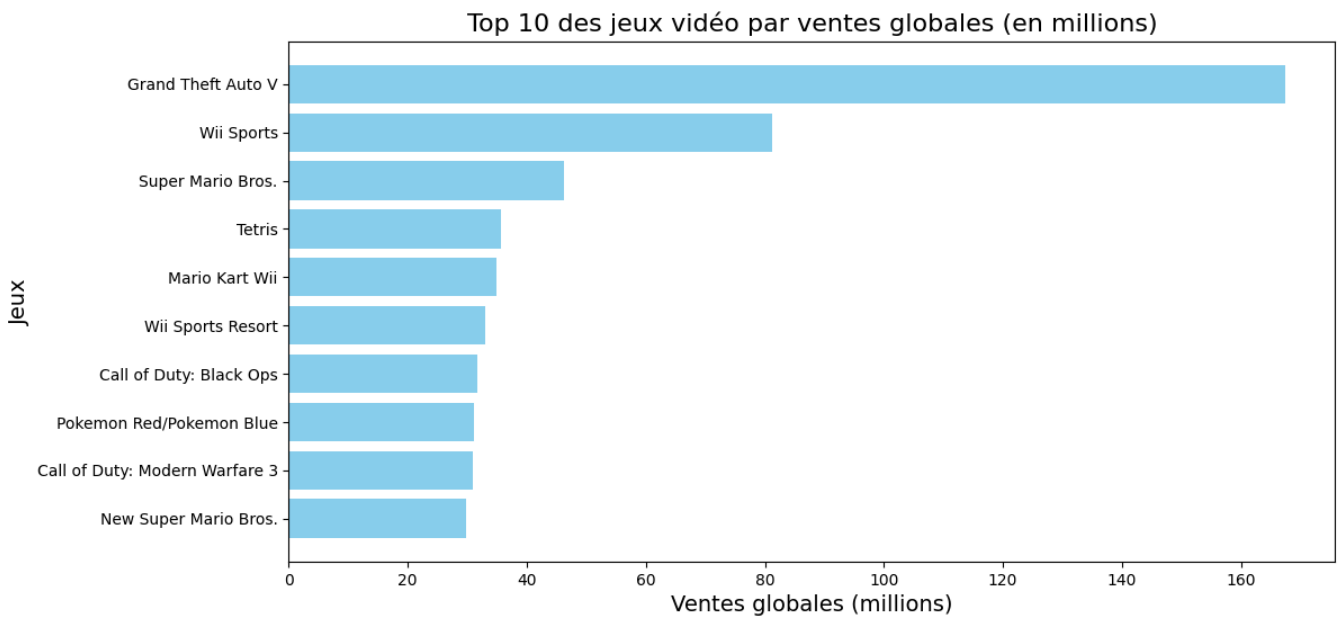
```
# Groupement par éditeur et comptage du nombre de jeux par éditeur
jeux_par_editeur = df_ventesjeux.groupby('Publisher')['Name'].count()

# Tri des éditeurs par nombre de jeux et sélection des 10 premiers
jeux_par_editeur = jeux_par_editeur.sort_values(ascending=False).head(10)

# Création du graphique
plt.figure(figsize=(12, 6))
plt.bar(jeux_par_editeur.index, jeux_par_editeur.values, color='skyblue')

# Titre et labels
plt.title("Top 10 des éditeurs par nombre de jeux", fontsize=14)
plt.xlabel("Éditeur", fontsize=12)
plt.ylabel("Nombre de jeux", fontsize=12)

# Rotation des labels de l'axe des x pour meilleure lisibilité
plt.xticks(rotation=45)

# Affichage du graphique
plt.tight_layout()
plt.show()
```



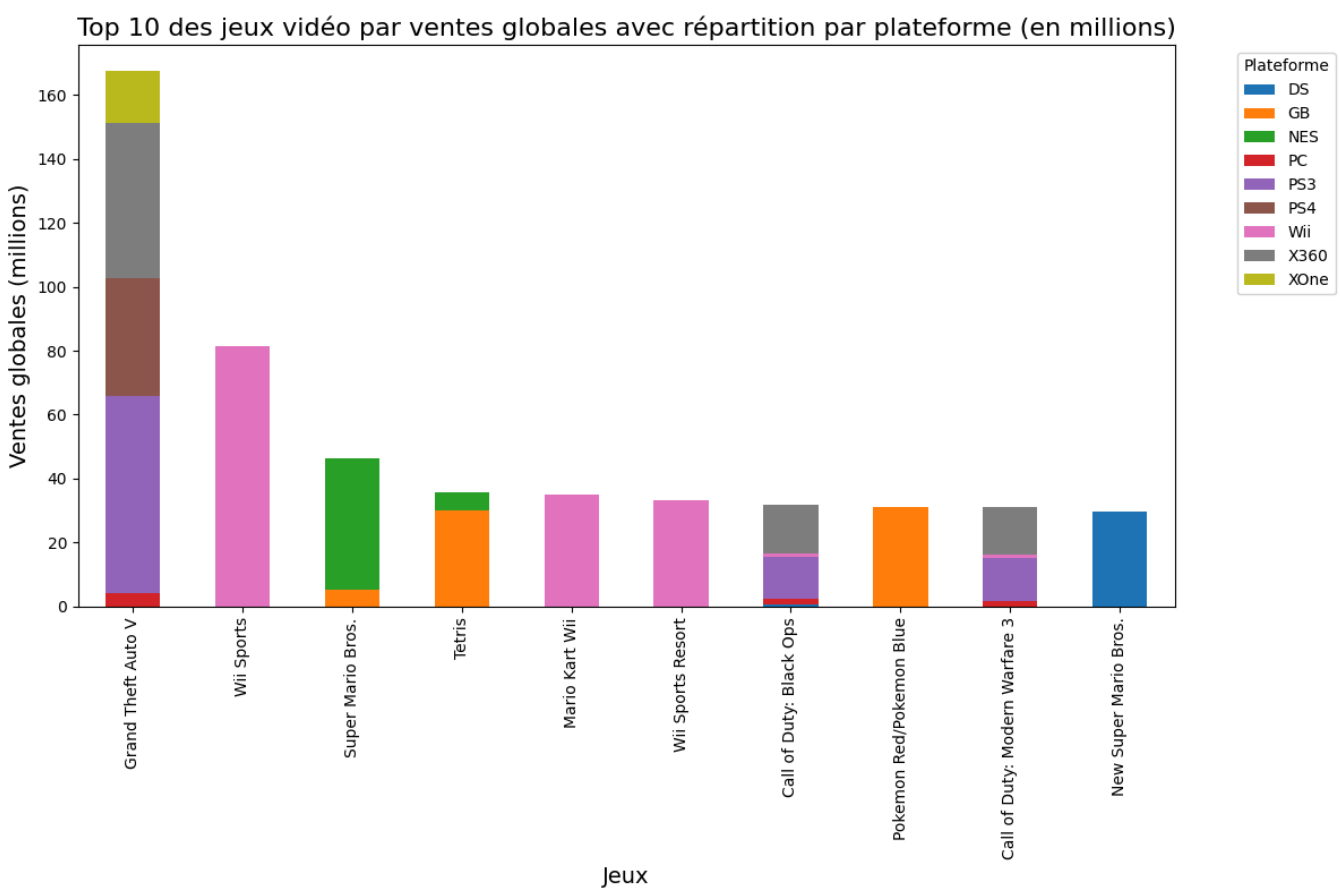

```
# Groupement par constructeur et comptage du nombre de jeux par constructeur
jeux_par_constructeur = df_ventesjeux.groupby('Constructeur')['Name'].count()

# Tri des constructeurs par nombre de jeux et sélection des 10 premiers
jeux_par_constructeur = jeux_par_constructeur.sort_values(ascending=False).head(10)

# Création du graphique
plt.figure(figsize=(12, 6))
plt.bar(jeux_par_constructeur.index, jeux_par_constructeur.values, color='lightcoral')

# Titre et labels
plt.title("Top 10 des constructeurs par nombre de jeux", fontsize=14)
plt.xlabel("Constructeur", fontsize=12)
plt.ylabel("Nombre de jeux", fontsize=12)

# Rotation des labels de l'axe des x pour meilleure lisibilité
plt.xticks(rotation=45)

# Affichage du graphique
plt.tight_layout()
plt.show()
```




```
# Regrouper par 'Name' et faire la somme des 'Global_Sales'
ventes_par_jeu = df_ventesjeux.groupby('Name')['Global_Sales'].sum().reset_index()

# Trier le DataFrame par Global_Sales en ordre décroissant et garder les 10 premiers
top_10_global_sales = ventes_par_jeu.sort_values(by='Global_Sales', ascending=False).head(10)

# Création du graphique à barres
plt.figure(figsize=(12, 6))
plt.barh(top_10_global_sales['Name'], top_10_global_sales['Global_Sales'], color='skyblue')

# Titre et labels
plt.title('Top 10 des jeux vidéo par ventes globales (en millions)', fontsize=16)
plt.xlabel('Ventes globales (millions)', fontsize=14)
plt.ylabel('Jeux', fontsize=14)

# Affichage du graphique
plt.gca().invert_yaxis()  # Inverser l'axe y pour avoir le jeu avec les ventes les plus élevées en haut
plt.show()
```

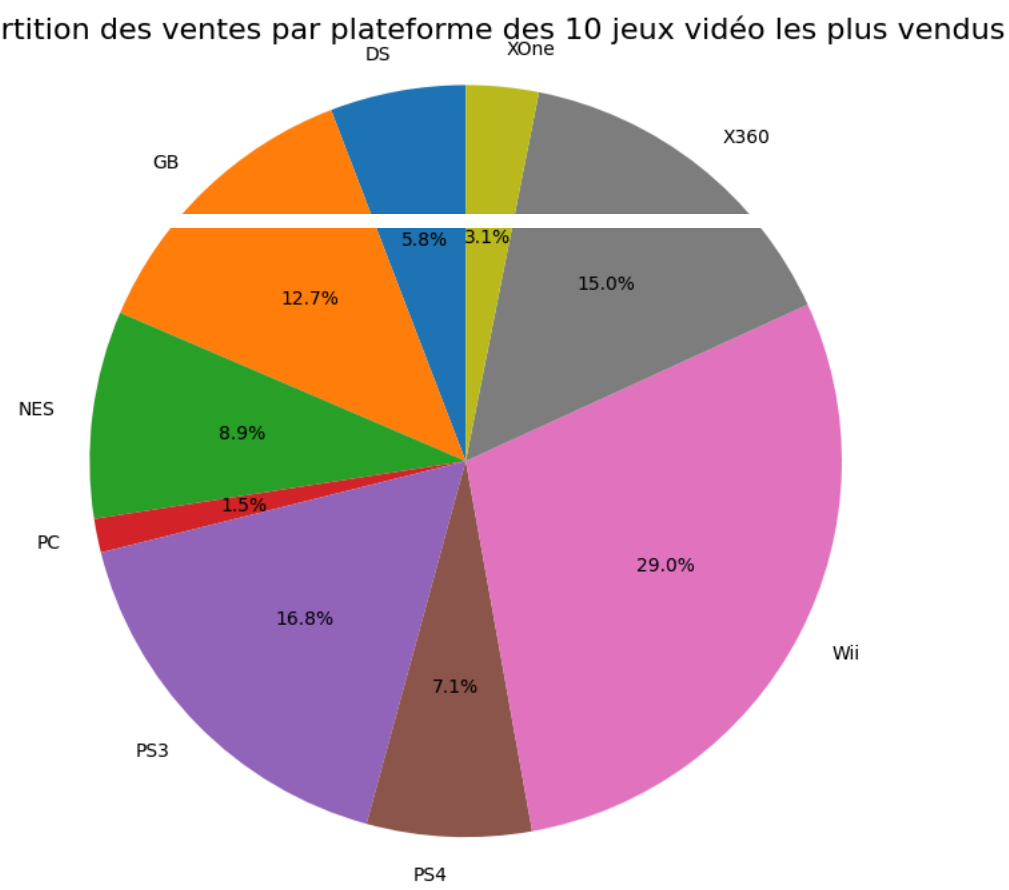


```
# Sélectionner le Top 10 des jeux par ventes globales
top_10_global_sales = df_ventesjeux.groupby(['Name', 'Platform'])['Global_Sales'].sum().reset_index()
top_10_sales_data = top_10_global_sales.groupby('Name')['Global_Sales'].sum().nlargest(10).index
top_10_sales_data = df_ventesjeux[df_ventesjeux['Name'].isin(top_10_sales_data)]

# Regrouper les ventes par jeu et par plateforme
ventes_par_plateforme_top10 = top_10_sales_data.groupby(['Name', 'Platform'])['Global_Sales'].sum().unstack(fill_value=0)

# Trier les jeux par ventes globales
ventes_par_plateforme_top10 = ventes_par_plateforme_top10.loc[ventes_par_plateforme_top10.sum(axis=1).sort_values(ascending=

# Définir des couleurs pour chaque plateforme en utilisant la nouvelle méthode
colors = plt.colormaps['tab10'].colors

# Création de l'histogramme empilé
plt.figure(figsize=(12, 8))
ventes_par_plateforme_top10.plot(kind='bar', stacked=True, ax=plt.gca(), color=colors)

# Titre et labels
plt.title('Top 10 des jeux vidéo par ventes globales avec répartition par plateforme (en millions)', fontsize=16)
plt.xlabel('Jeux', fontsize=14)
plt.ylabel('Ventes globales (millions)', fontsize=14)

# Affichage de la légende
plt.legend(title='Plateforme', bbox_to_anchor=(1.05, 1), loc='upper left')  # Ajuster la légende
plt.tight_layout()  # Ajuster l'espacement

# Afficher le graphique
plt.show()
```




```
# Sélectionner le Top 10 des jeux par ventes globales
top_10_global_sales = df_ventesjeux.groupby(['Name', 'Platform'])['Global_Sales'].sum().reset_index()
top_10_sales_data = top_10_global_sales.groupby('Name')['Global_Sales'].sum().nlargest(10).index
top_10_sales_data = df_ventesjeux[df_ventesjeux['Name'].isin(top_10_sales_data)]

# Regrouper les ventes par plateforme pour les 10 jeux
ventes_par_plateforme_top10 = top_10_sales_data.groupby('Platform')['Global_Sales'].sum()

# Création du camembert
```

```
plt.figure(figsize=(8, 8))
plt.pie(ventes_par_plateforme_top10, labels=ventes_par_plateforme_top10.index, autopct='%1.1f%%', startangle=90, colors=plt.

# Titre
plt.title('Répartition des ventes par plateforme des 10 jeux vidéo les plus vendus', fontsize=16)

# Afficher le graphique
plt.axis('equal')  # Égaliser les axes pour un camembert
plt.show()
```



Fichier Vente de consoles 2022

```
# Import du fichier "Liste de tous les jeux avec note des joueurs"
df_venteconsoles = pd.read_excel('/content/drive/MyDrive/Datasets/UOI Games/Vente-de-console_2022.xlsx')
```

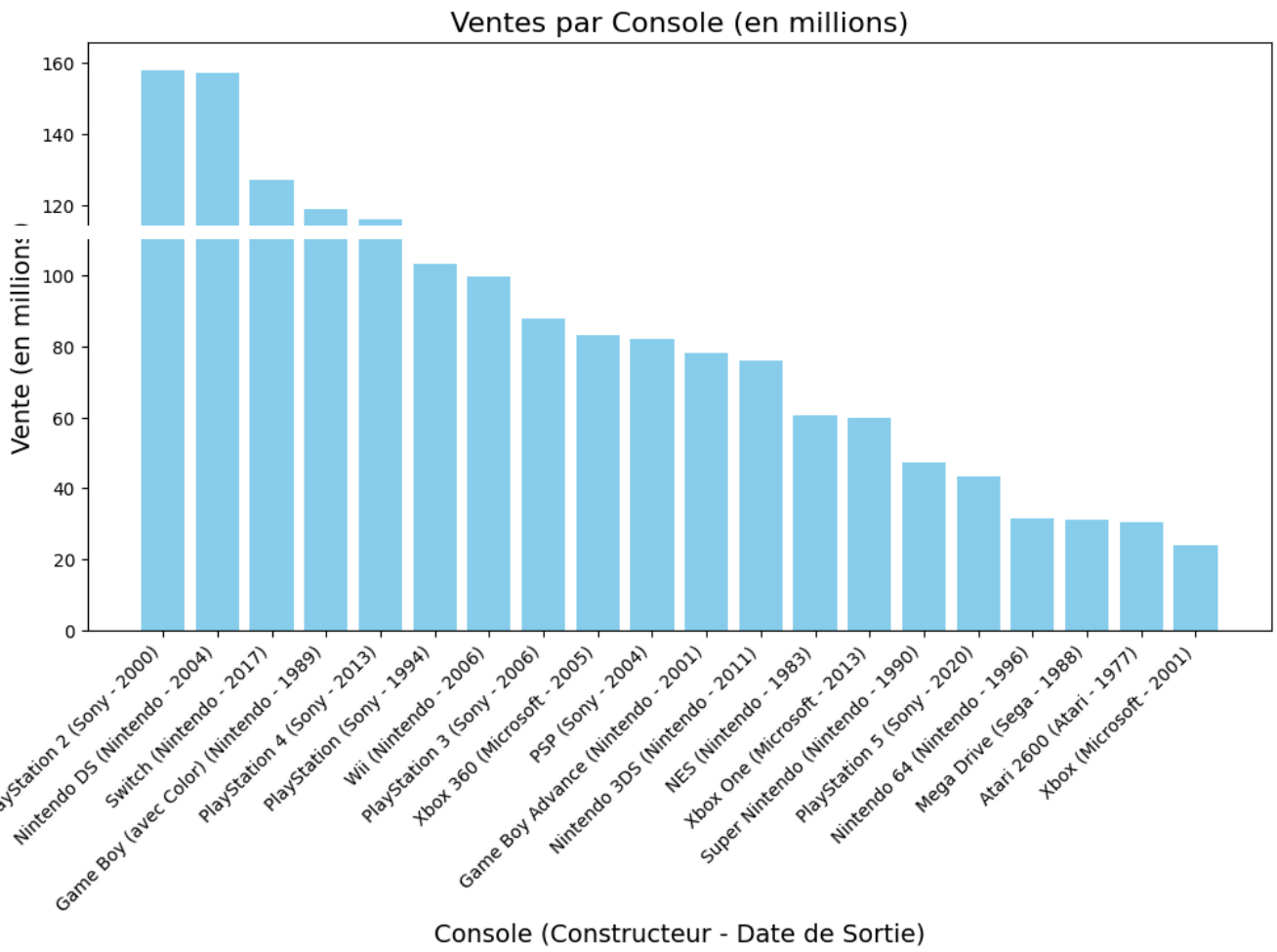
```
df_venteconsoles.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 4 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Console             20 non-null     object
 1   Constructeur        20 non-null     object
 2   Date de Sortie      20 non-null     int64
 3   Vente (en million)  20 non-null     float64
dtypes: float64(1), int64(1), object(2)
memory usage: 768.0+ bytes
```

Ventes par console

```
# Créer les étiquettes pour les x en combinant Console, Constructeur et Date de Sortie
df_venteconsoles['Label'] = df_venteconsoles['Console'] + ' (' + df_venteconsoles['Constructeur'] + ' - ' + df_venteconsoles

# Création du graphique à barres
plt.figure(figsize=(12, 6))
plt.bar(df_venteconsoles['Label'], df_venteconsoles['Vente (en million)'], color='skyblue')

# Titre et labels
plt.title('Ventes par Console (en millions)', fontsize=16)
plt.xlabel('Console (Constructeur - Date de Sortie)', fontsize=14)
```

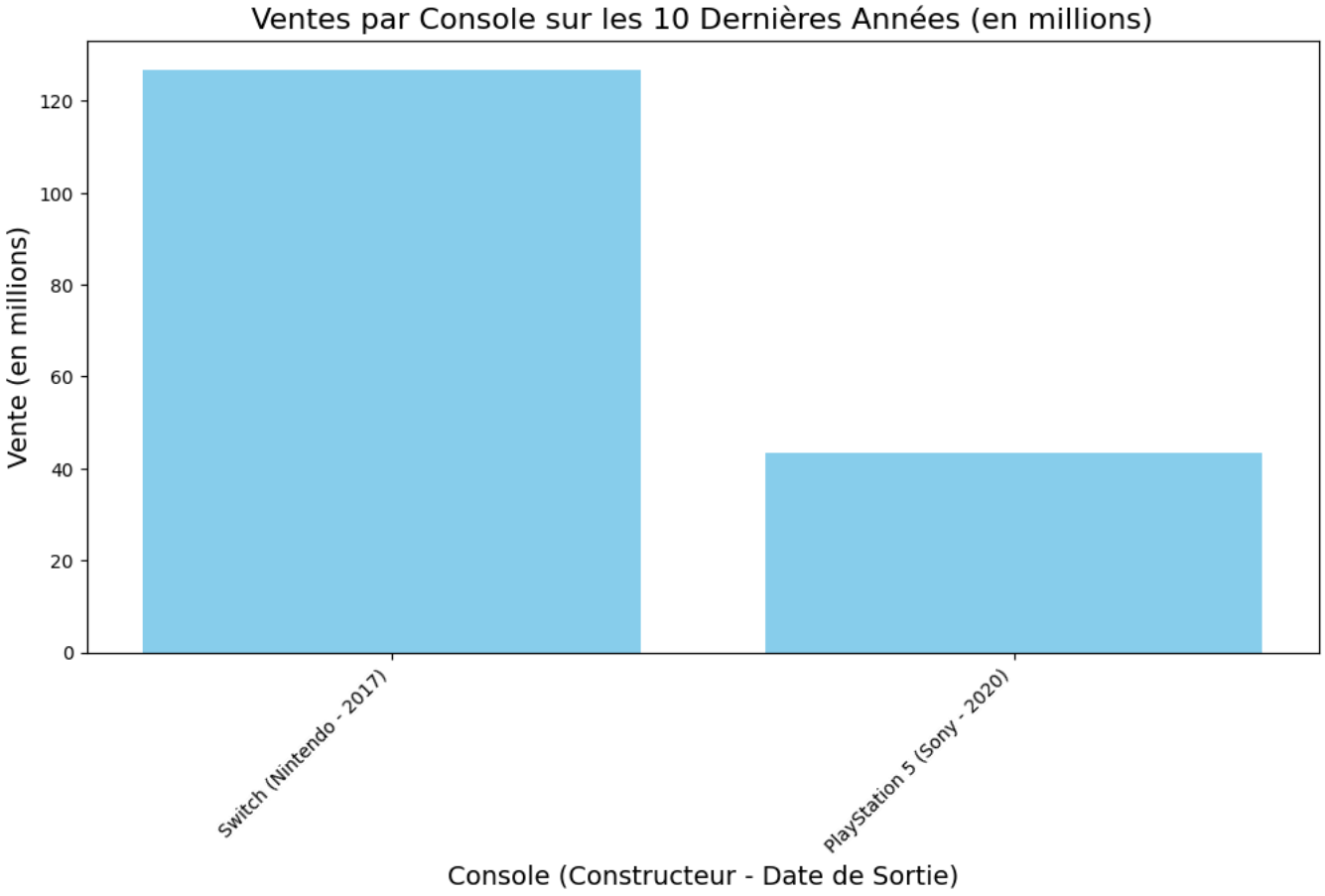
```
plt.ylabel('Vente (en millions)', fontsize=14)

# Rotation des labels pour une meilleure lisibilité
plt.xticks(rotation=45, ha='right')

# Affichage du graphique
plt.show()
```



Ventes par console sur les 10 denières années

```
from datetime import datetime
# Obtenir l'année actuelle
current_year = datetime.now().year

# Filtrer pour les 10 dernières années
df_recent_consoles = df_venteconsoles[df_venteconsoles['Date de Sortie'] >= current_year - 10]

# Créer les étiquettes pour les x en combinant Console, Constructeur et Date de Sortie
df_recent_consoles['Label'] = (
    df_recent_consoles['Console'] +
    ' (' + df_recent_consoles['Constructeur'] +
    ' - ' + df_recent_consoles['Date de Sortie'].astype(str) + ')'
)

# Création du graphique à barres
plt.figure(figsize=(12, 6))
plt.bar(df_recent_consoles['Label'], df_recent_consoles['Vente (en million)'], color='skyblue')

# Titre et labels
plt.title('Ventes par Console sur les 10 Dernières Années (en millions)', fontsize=16)
plt.xlabel('Console (Constructeur - Date de Sortie)', fontsize=14)
plt.ylabel('Vente (en millions)', fontsize=14)

# Rotation des labels pour une meilleure lisibilité
plt.xticks(rotation=45, ha='right')

# Affichage du graphique
plt.show()
```

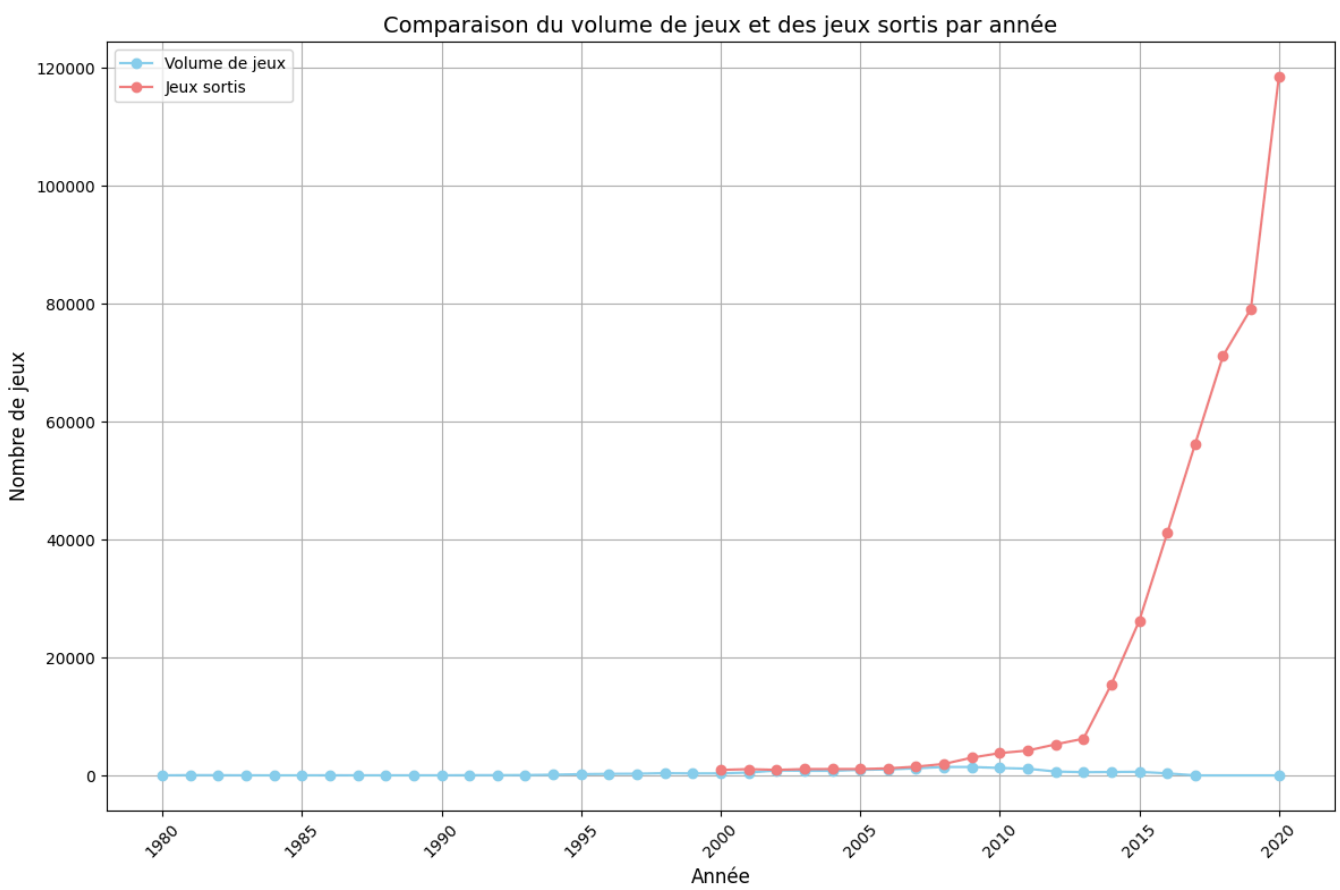
```
<ipython-input-50-ceaa3c4f9f73>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
  df_recent_consoles['Label'] = (
```



Analyses croisées

Jeux sortis vs jeux vendus

```
# Suppression des valeurs manquantes dans la colonne Year pour le premier DataFrame
df_ventesjeux_clean = df_ventesjeux.dropna(subset=['Year'])
jeux_par_annee = df_ventesjeux_clean.groupby('Year')['Name'].count()

# Nombre de jeux par année pour le second DataFrame
sorties_par_annee = df_listejeux['Annee'].value_counts().sort_index()

# Création du graphique
plt.figure(figsize=(12, 8))

# Tracer le premier graphique (volume de jeux par année)
plt.plot(jeux_par_annee.index.astype(int), jeux_par_annee.values, marker='o', linestyle='-', color='skyblue', label='Volume

# Tracer le second graphique (nombre de jeux sortis par année)
plt.plot(sorties_par_annee.index, sorties_par_annee.values, marker='o', linestyle='-', color='lightcoral', label='Jeux sorti

# Titre et labels
plt.title("Comparaison du volume de jeux et des jeux sortis par année", fontsize=14)
plt.xlabel("Année", fontsize=12)
plt.ylabel("Nombre de jeux", fontsize=12)

# Ajustement des ticks de l'axe x pour meilleure lisibilité
plt.xticks(rotation=45)
plt.grid(True)
plt.legend()  # Ajouter une légende pour identifier chaque série
plt.tight_layout()  # Pour un espacement optimal
plt.show()
```



Données complémentaires

Fichier all_video_games.csv (from Kaggle)

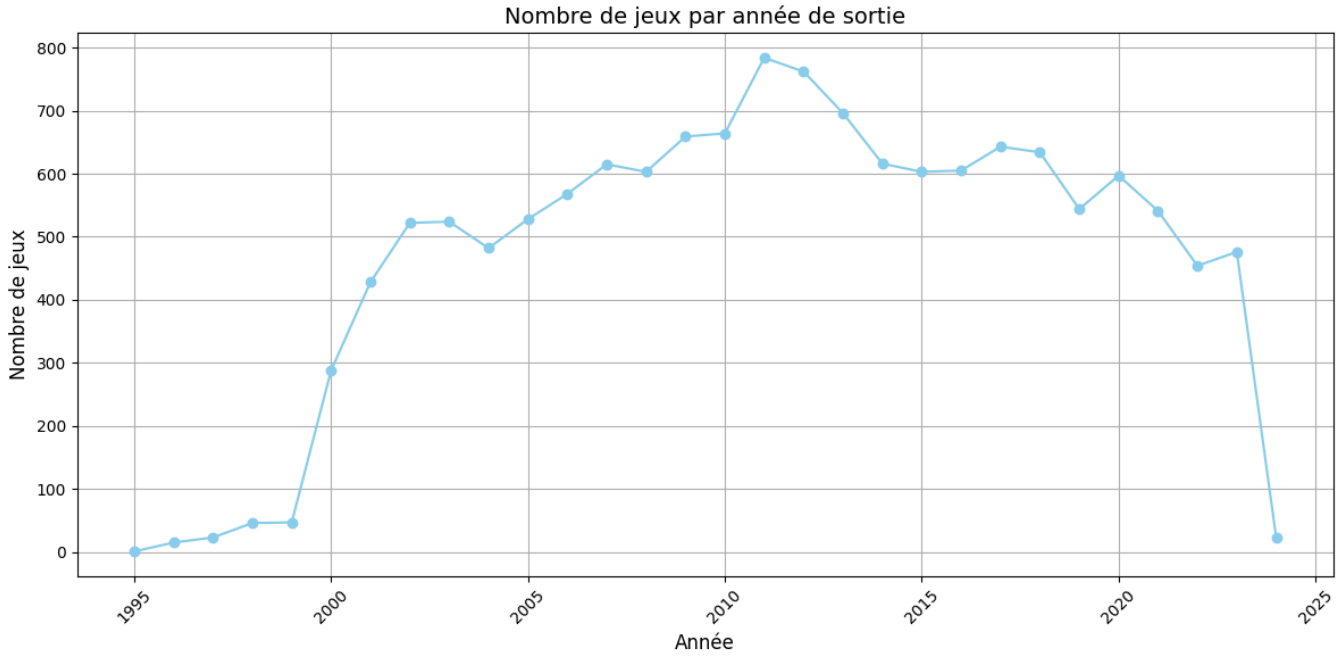
```
# Charger le fichier CSV dans un DataFrame
df_kagglevideogames = pd.read_csv('/content/drive/MyDrive/Datasets/UOI Games/all_video_games(kaggle).csv')
```

```
df_kagglevideogames.info()
```

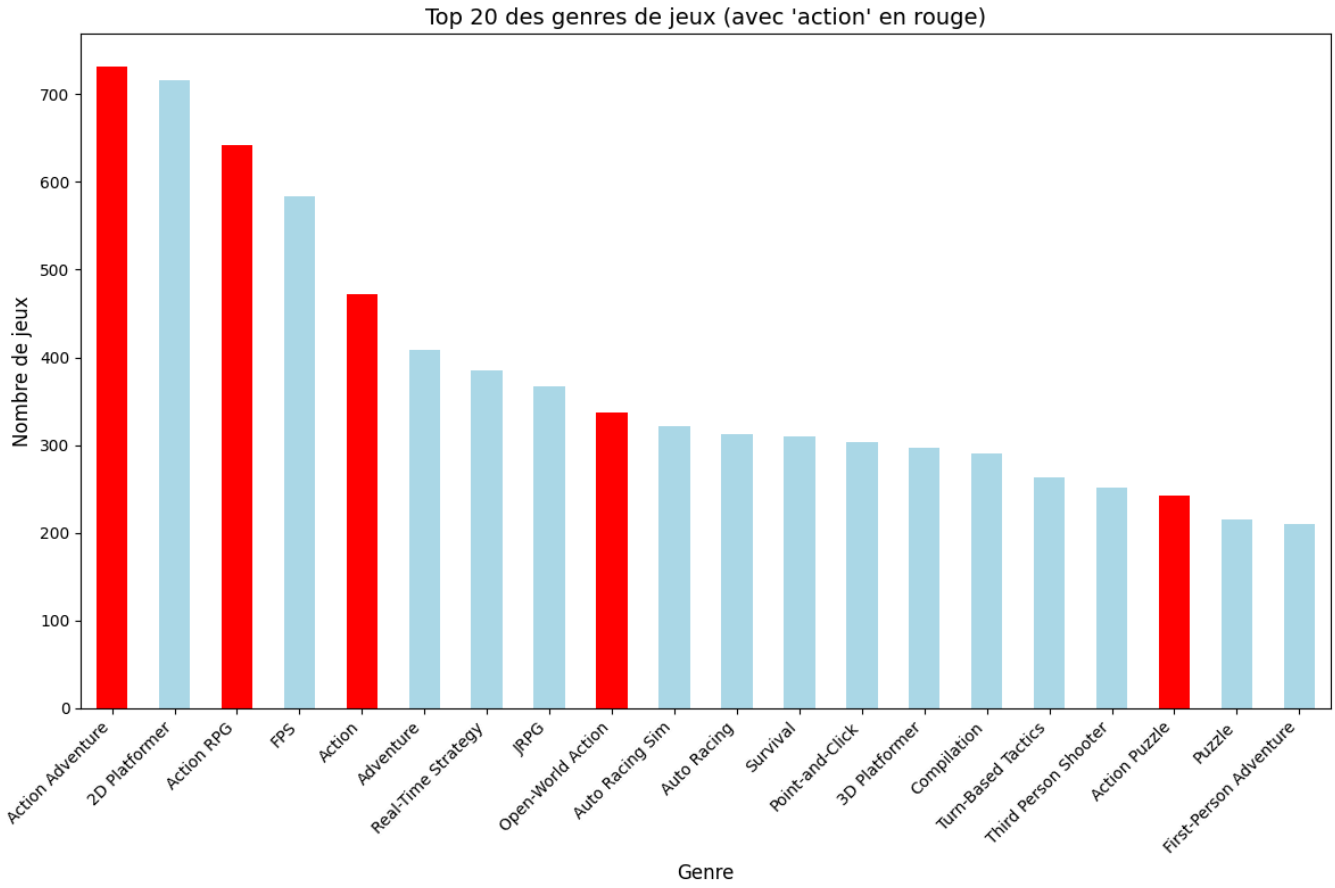
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14055 entries, 0 to 14054
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Title               14034 non-null  object
 1   Release Date        13991 non-null  object
 2   Developer           13917 non-null  object
 3   Publisher           13917 non-null  object
 4   Genres              14034 non-null  object
 5   Product Rating      11005 non-null  object
 6   User Score          11714 non-null  float64
 7   User Ratings Count  11299 non-null  float64
 8   Platforms Info      14055 non-null  object
dtypes: float64(2), object(7)
memory usage: 988.4+ KB
```

```
# Convertir la colonne 'Release Date' en format datetime
df_kagglevideogames['Release Date'] = pd.to_datetime(df_kagglevideogames['Release Date'], errors='coerce')

# Grouper les jeux par année de sortie
jeux_par_annee = df_kagglevideogames.groupby(df_kagglevideogames['Release Date'].dt.year)['Title'].count()

# Créer un graphique
plt.figure(figsize=(12, 6))
```

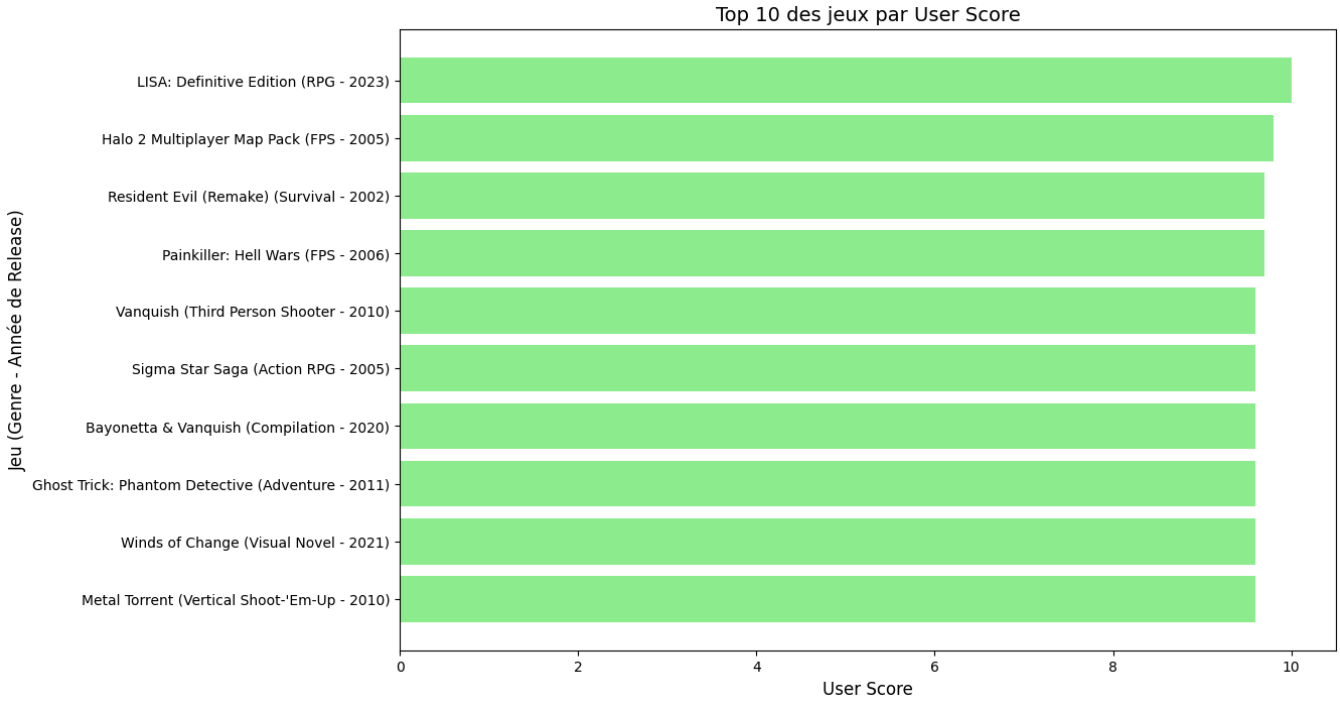
```
plt.plot(jeux_par_annee.index, jeux_par_annee.values, marker='o', linestyle='-', color='skyblue')

# Titre et labels
plt.title("Nombre de jeux par année de sortie", fontsize=14)
plt.xlabel("Année", fontsize=12)
plt.ylabel("Nombre de jeux", fontsize=12)

# Afficher les années sur l'axe des x avec une meilleure lisibilité
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()

# Afficher le graphique
plt.show()
```




```
# Remplacer les valeurs vides dans la colonne 'Genres' par 'NC'
df_kagglevideogames['Genres'] = df_kagglevideogames['Genres'].fillna('NC')

# Compter le nombre de jeux pour chaque genre
genres_count = df_kagglevideogames['Genres'].value_counts()

# Sélectionner uniquement les 20 premiers genres
top_20_genres = genres_count.head(20)

# Créer une liste de couleurs : rouge pour les genres contenant 'action', sinon bleu
colors = ['red' if 'action' in genre.lower() else 'lightblue' for genre in top_20_genres.index]

# Création du graphique
plt.figure(figsize=(12, 8))
top_20_genres.plot(kind='bar', color=colors)

# Titre et labels
plt.title("Top 20 des genres de jeux (avec 'action' en rouge)", fontsize=14)
plt.xlabel("Genre", fontsize=12)
plt.ylabel("Nombre de jeux", fontsize=12)

# Rotation des labels de l'axe x pour une meilleure lisibilité
plt.xticks(rotation=45, ha='right')

# Affichage du graphique
plt.tight_layout()
plt.show()
```



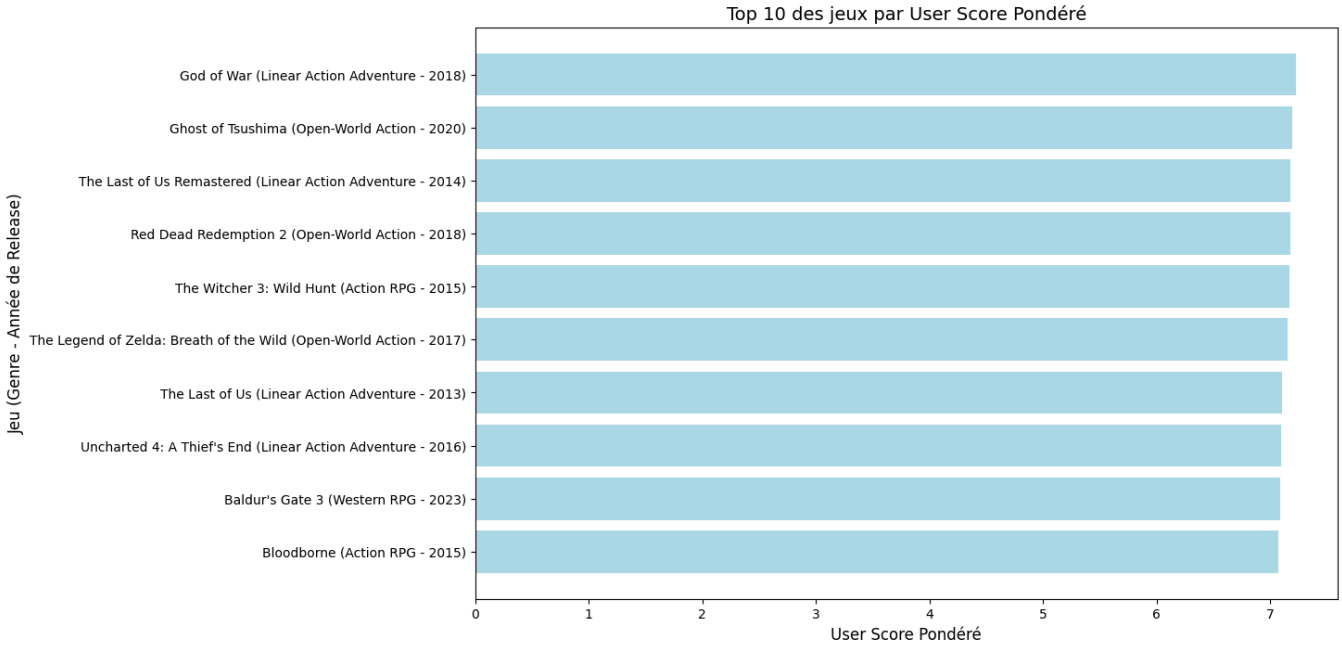

```
# Nettoyage et conversion des valeurs de 'Release Date' pour extraire l'année
df_kagglevideogames['Release Date'] = pd.to_datetime(df_kagglevideogames['Release Date'], errors='coerce')
df_kagglevideogames['Release Year'] = df_kagglevideogames['Release Date'].dt.year

# Remplacement des valeurs manquantes dans 'User Score' par NaN et filtrage
df_kagglevideogames = df_kagglevideogames.dropna(subset=['User Score'])

# Sélectionner les top 10 jeux par 'User Score'
top_10_user_score = df_kagglevideogames.nlargest(10, 'User Score')

# Créer un label avec 'Titre (Genre - Année de Release)'
top_10_user_score['Label'] = top_10_user_score['Title'] + ' (' + top_10_user_score['Genres'] + ' - ' + top_10_user_score['Re

# Création du graphique
plt.figure(figsize=(12, 8))
plt.barh(top_10_user_score['Label'], top_10_user_score['User Score'], color='lightgreen')

# Titre et labels
plt.title("Top 10 des jeux par User Score", fontsize=14)
plt.xlabel("User Score", fontsize=12)
plt.ylabel("Jeu (Genre - Année de Release)", fontsize=12)

# Affichage du graphique
plt.gca().invert_yaxis()
#plt.tight_layout()
plt.show()
```



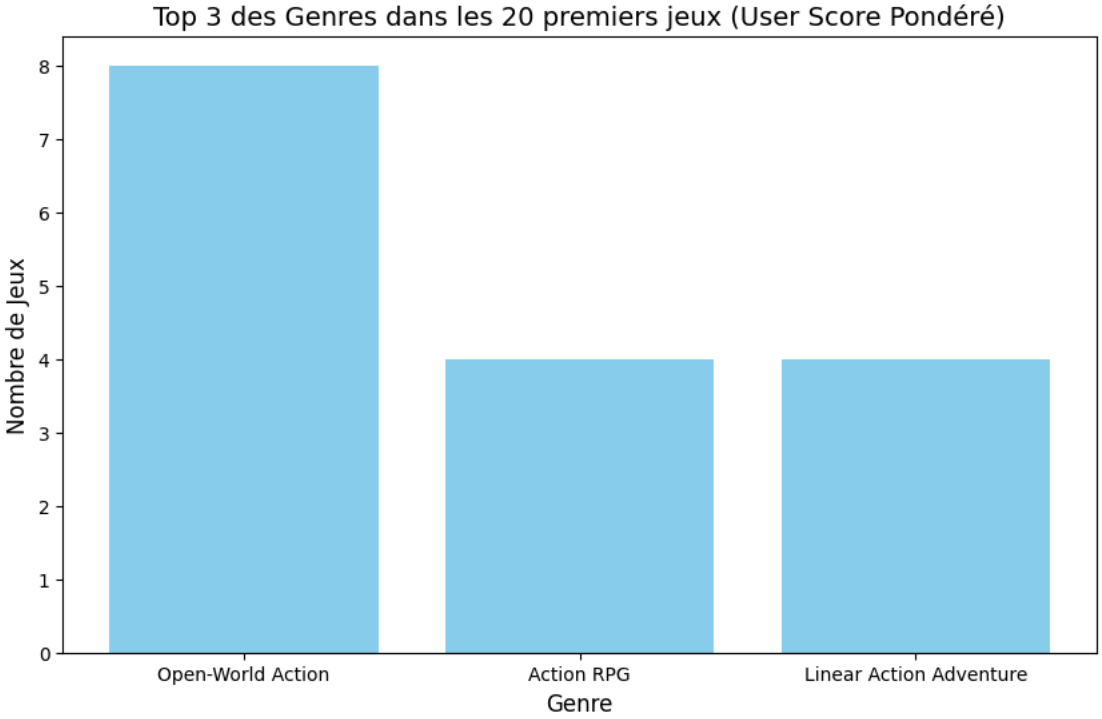
Pondération du score

- User Score est la note moyenne donnée par les utilisateurs
- User Ratings Count est le nombre total d'avis pour ce jeu
- Global Score Moyen est la moyenne des User Scores dans l'ensemble de données (c'est-à-dire, la note moyenne de tous les jeux)
- Total est la somme des votes, c'est-à-dire User Ratings Count + (Total - User Ratings Count)

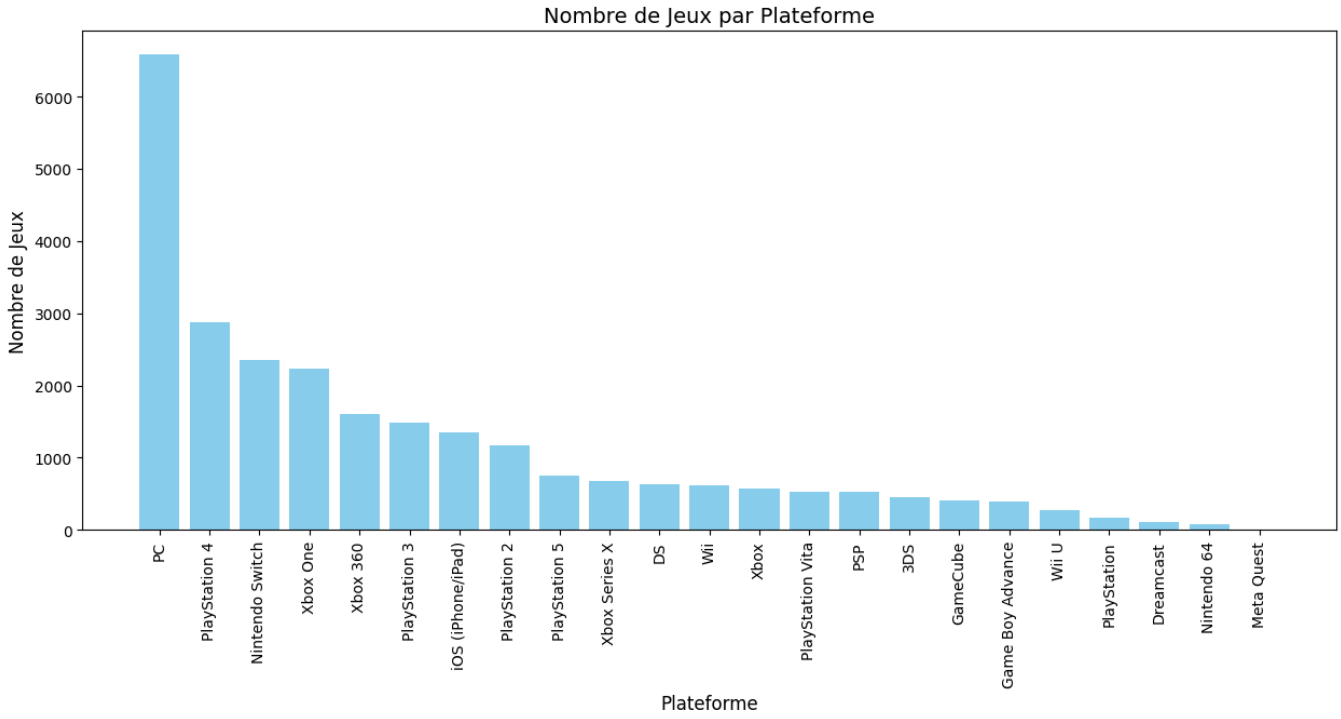
```
# Calculer la moyenne globale du User Score dans le dataset
global_mean_score = df_kagglevideogames['User Score'].mean()

# Pondérer le User Score en fonction du nombre d'avis
df_kagglevideogames['Weighted User Score'] = (
    (df_kagglevideogames['User Score'] * df_kagglevideogames['User Ratings Count']) +
    (global_mean_score * (df_kagglevideogames['User Ratings Count'].max() - df_kagglevideogames['User Ratings Count']))
) / df_kagglevideogames['User Ratings Count'].max()


# Sélectionner les top 10 jeux par Weighted User Score
top_10_weighted_user_score = df_kagglevideogames.nlargest(10, 'Weighted User Score')

# Création du label avec 'Titre (Genre - Année de Release)'
top_10_weighted_user_score['Label'] = top_10_weighted_user_score['Title'] + ' (' + top_10_weighted_user_score['Genres'] + '

# Inverser l'ordre des jeux
top_10_weighted_user_score = top_10_weighted_user_score[::-1]

# Création du graphique
plt.figure(figsize=(12, 8))
plt.barh(top_10_weighted_user_score['Label'], top_10_weighted_user_score['Weighted User Score'], color='lightblue')

# Titre et labels
plt.title("Top 10 des jeux par User Score Pondéré", fontsize=14)
plt.xlabel("User Score Pondéré", fontsize=12)
plt.ylabel("Jeu (Genre - Année de Release)", fontsize=12)

# Affichage du graphique
#plt.tight_layout()
plt.show()
```




```
# Calculer un score pondéré basé sur le User Score et le User Ratings Count
df_kagglevideogames['Weighted User Score'] = df_kagglevideogames['User Score'] * df_kagglevideogames['User Ratings Count']

# Trier les jeux par le score pondéré de manière décroissante
top_20_games_weighted = df_kagglevideogames.nlargest(20, 'Weighted User Score')

# Séparer les genres pour les jeux dans le top 20
# Supposons que les genres sont séparés par une virgule
top_20_games_weighted['Genres'] = top_20_games_weighted['Genres'].str.split(',')

# "Exploser" la colonne Genres pour avoir un genre par ligne
top_20_games_weighted_exploded = top_20_games_weighted.explode('Genres')

# Compter la fréquence des genres dans les 20 premiers jeux
genre_counts = top_20_games_weighted_exploded['Genres'].value_counts()

# Afficher le top 3 des genres
top_3_genres = genre_counts.head(3)

# Création du graphique
plt.figure(figsize=(10, 6))
plt.bar(top_3_genres.index, top_3_genres.values, color='skyblue')

# Ajout du titre et des labels
plt.title("Top 3 des Genres dans les 20 premiers jeux (User Score Pondéré)", fontsize=14)
plt.xlabel("Genre", fontsize=12)
plt.ylabel("Nombre de Jeux", fontsize=12)

# Affichage du graphique
plt.show()
```

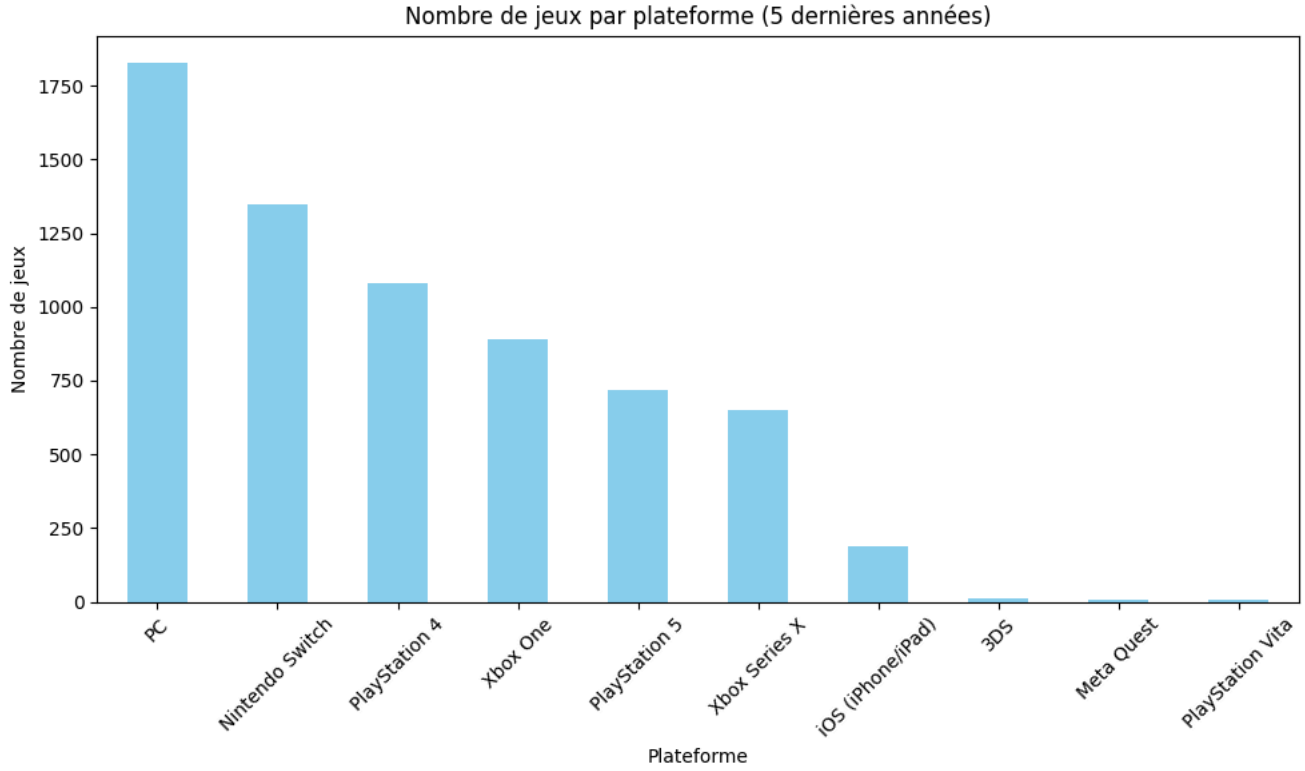



```
# Fonction pour extraire toutes les plateformes présentes dans la colonne 'Platforms Info'
def extract_platforms(platform_info, separator=', '):
    platforms = [platform['Platform'] for platform in platform_info]
    return separator.join(platforms) if platforms else None

# Appliquer la fonction à la colonne 'Platforms Info' et créer une nouvelle colonne 'Filtered Platforms'
df_kagglevideogames['Filtered Platforms'] = df_kagglevideogames['Platforms Info'].apply(eval).apply(extract_platforms)

# Supprimer les lignes où 'Filtered Platforms' est None (pas de plateforme filtrée trouvée)
df_platforms_filtered = df_kagglevideogames.dropna(subset=['Filtered Platforms'])

# Compter le nombre de jeux pour chaque plateforme
platform_counts = df_platforms_filtered['Filtered Platforms'].str.split(', ').explode().value_counts()

# Afficher le résultat
# print(platform_counts)

# Afficher graphique
plt.figure(figsize=(15, 6))
plt.bar(platform_counts.index, platform_counts.values, color='skyblue')
plt.title("Nombre de Jeux par Plateforme", fontsize=14)
plt.xlabel("Plateforme", fontsize=12)
plt.ylabel("Nombre de Jeux", fontsize=12)
plt.xticks(rotation=90, ha='center')
#plt.tight_layout()
plt.show()
```



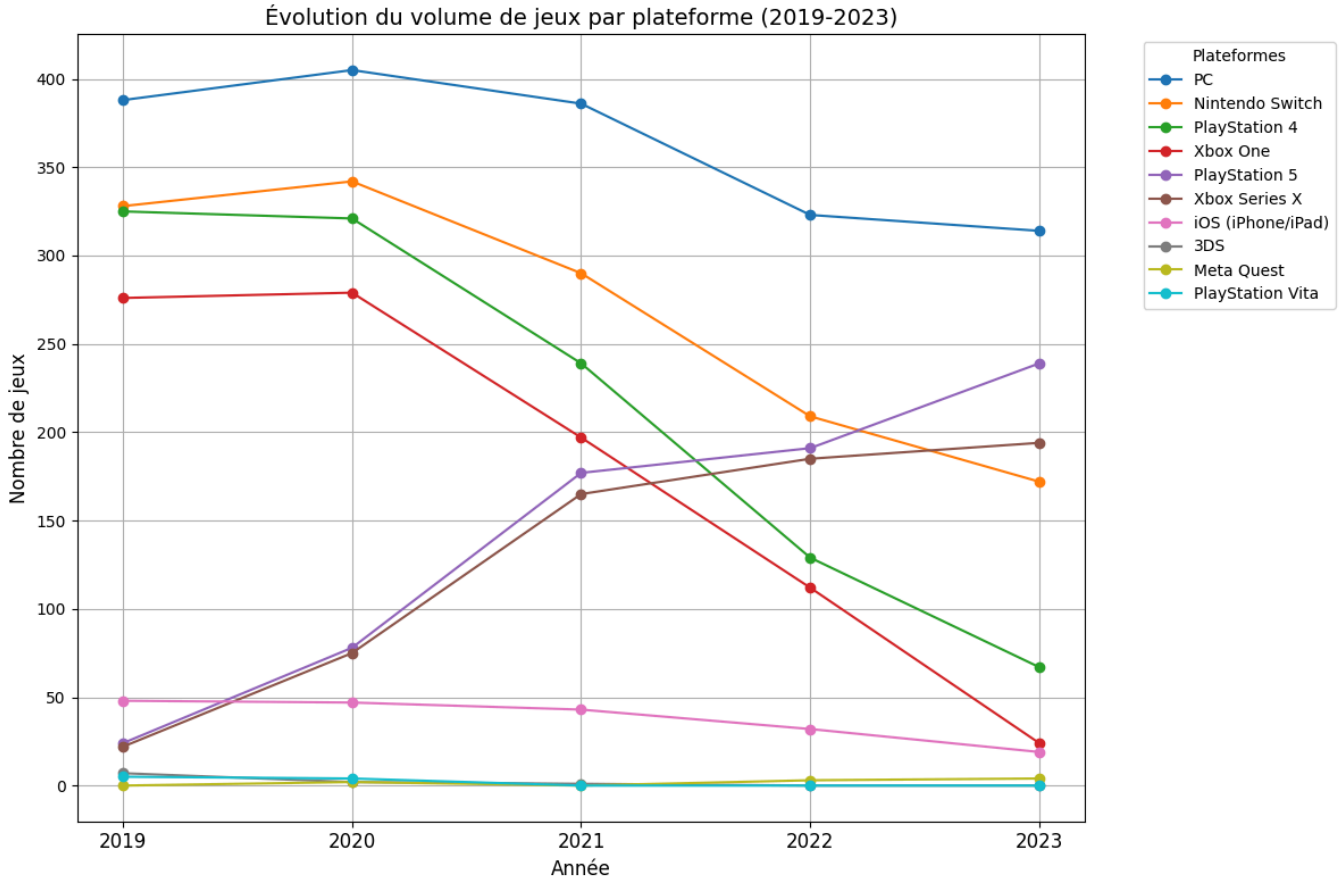

```
# Convertir la colonne 'Release Date' en datetime
df_kagglevideogames['Release Date'] = pd.to_datetime(df_kagglevideogames['Release Date'], errors='coerce')

# Filtrer les jeux sortis au cours des 5 dernières années
current_year = pd.Timestamp.now().year
df_last5years = df_kagglevideogames[df_kagglevideogames['Release Date'].dt.year >= current_year - 5]

# Remplacer les valeurs manquantes de 'Platforms Info' par 'NC'
df_last5years['Platforms Info'] = df_last5years['Platforms Info'].fillna('NC')

# Extraire les informations de plateforme
import ast

def extract_platforms(platform_info):
    try:
        platforms = [item['Platform'] for item in ast.literal_eval(platform_info)]
        return ', '.join(platforms)
    except:
        return 'NC'

df_last5years['Platforms'] = df_last5years['Platforms Info'].apply(extract_platforms)

# Compter le nombre de jeux par plateforme
platform_counts = df_last5years['Platforms'].str.get_dummies(sep=', ').sum().sort_values(ascending=False)

# Afficher les 10 plateformes principales sur les 5 dernières années
top_platforms = platform_counts.head(10)

# Création du graphique
plt.figure(figsize=(10, 6))
top_platforms.plot(kind='bar', color='skyblue')
plt.title('Nombre de jeux par plateforme (5 dernières années)')
plt.xlabel('Plateforme')
plt.ylabel('Nombre de jeux')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
<ipython-input-61-06d497df6b5f>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
  df_last5years['Platforms Info'] = df_last5years['Platforms Info'].fillna('NC')
<ipython-input-61-06d497df6b5f>:21: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
  df_last5years['Platforms'] = df_last5years['Platforms Info'].apply(extract_platforms)
```




```
# Assurez-vous que 'Release Date' est bien au format datetime, en ignorant les erreurs
df_kagglevideogames['Release Date'] = pd.to_datetime(df_kagglevideogames['Release Date'], errors='coerce')

# Filtrer les jeux de 2019 à 2023 (exclure 2024)
df_last5years = df_kagglevideogames[(df_kagglevideogames['Release Date'].dt.year >= 2019) &
                                    (df_kagglevideogames['Release Date'].dt.year < 2024)].copy()

# Remplacer les valeurs manquantes dans 'Platforms Info' par 'NC'
df_last5years['Platforms Info'] = df_last5years['Platforms Info'].fillna('NC')

# Fonction pour extraire les plateformes
def extract_platforms(platform_info):
    try:
        # Conversion en liste d'objets Python via ast.literal_eval
        platforms = [item['Platform'] for item in ast.literal_eval(platform_info)]
        return ', '.join(platforms)  # Séparation par virgule et espace
    except (ValueError, SyntaxError):  # Gérer les erreurs de conversion
        return 'NC'

# Appliquer la fonction pour extraire les plateformes
df_last5years['Platforms'] = df_last5years['Platforms Info'].apply(extract_platforms)

# Extraire l'année de sortie des jeux
df_last5years['Year'] = df_last5years['Release Date'].dt.year

# Convertir la colonne 'Platforms' en colonnes binaires pour chaque plateforme
platforms_dummies = df_last5years['Platforms'].str.get_dummies(sep=', ')

# Supprimer les colonnes de plateformes avec uniquement des zéros (pas de jeux pour cette plateforme)
platforms_dummies = platforms_dummies.loc[:, (platforms_dummies != 0).any(axis=0)]

# Ajouter les colonnes de plateformes au DataFrame original
df_last5years = pd.concat([df_last5years, platforms_dummies], axis=1)

# Grouper les données par année et sommer les colonnes de plateformes disponibles
platforms_per_year = df_last5years.groupby('Year')[platforms_dummies.columns].sum()

# Limiter aux 10 plateformes principales
```

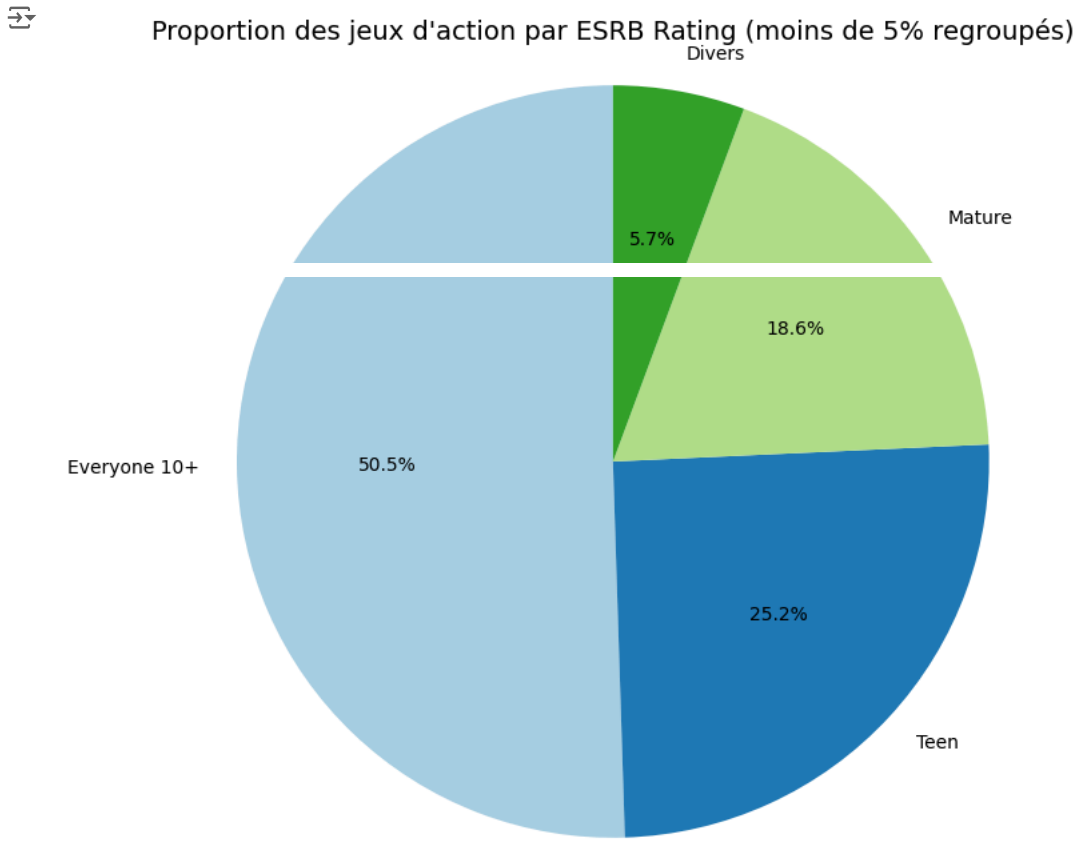
```
top_platforms = platforms_per_year.sum().sort_values(ascending=False).head(10).index
platforms_per_year_top = platforms_per_year[top_platforms]

# Création du graphique de l'évolution des jeux par plateforme
plt.figure(figsize=(12, 8))
for platform in platforms_per_year_top.columns:
    plt.plot(platforms_per_year_top.index, platforms_per_year_top[platform], label=platform, marker='o')

# Ajouter un titre et les labels des axes
plt.title("Évolution du volume de jeux par plateforme (2019-2023)", fontsize=14)
plt.xlabel("Année", fontsize=12)
plt.ylabel("Nombre de jeux", fontsize=12)

# Spécifier les ticks de l'axe x pour n'afficher que les années entières
plt.xticks(platforms_per_year_top.index, fontsize=12)

# Ajuster la légende pour qu'elle ne chevauche pas le graphique
plt.legend(title="Plateformes", bbox_to_anchor=(1.05, 1), loc='upper left')

# Afficher le graphique
plt.grid(True)
plt.tight_layout()
plt.show()
```



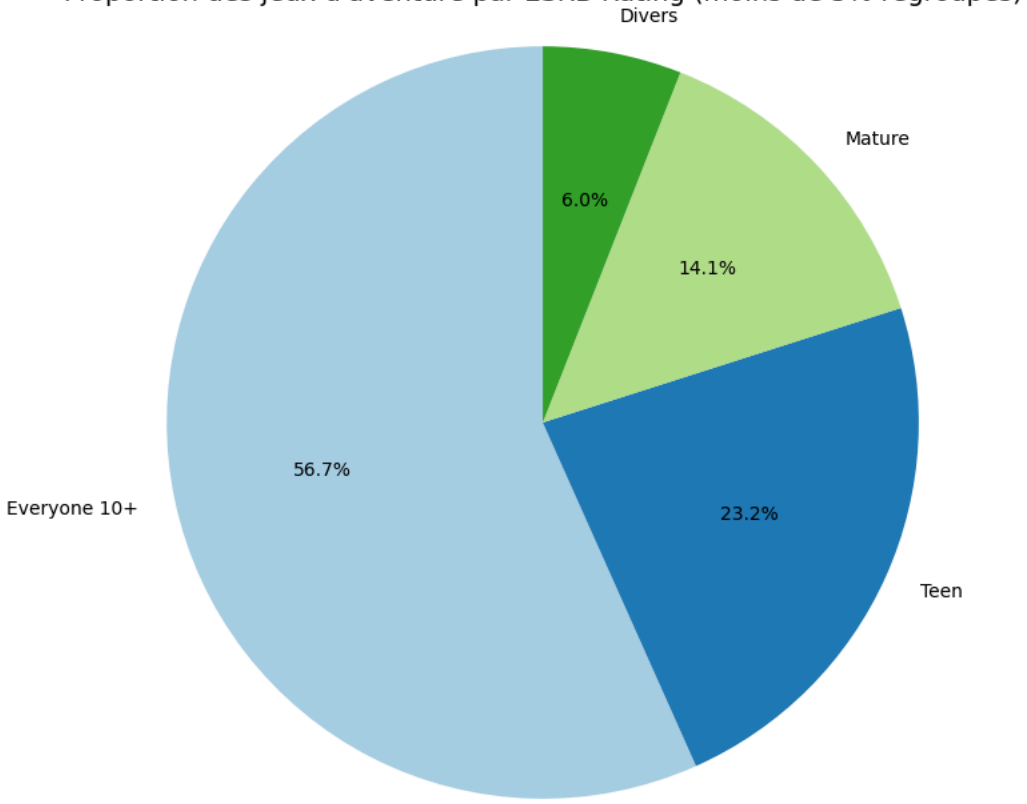
Age cible pour les jeux d'action / aventure

```
# Supposons que df_listejeux est déjà défini et contient une colonne 'genres'
# Filtrer pour obtenir uniquement les jeux de genre "Action"
df_action = df_listejeux[df_listejeux['genres'].str.contains('Action', case=False, na=False)]

# Compter le nombre de jeux par ESRB Rating
proportion_esrb = df_action['esrb_rating'].value_counts()

# Calculer le pourcentage de chaque catégorie
pourcentages = proportion_esrb / proportion_esrb.sum() * 100

# Regrouper les catégories avec moins de 5% dans une catégorie 'Divers'
```

```
proportion_esrb_regroupe = pourcentages[pourcentages >= 5].copy()
proportion_esrb_regroupe['Divers'] = pourcentages[pourcentages < 5].sum()

# Création du graphique camembert
plt.figure(figsize=(8, 8))
plt.pie(proportion_esrb_regroupe, labels=proportion_esrb_regroupe.index, autopct='%1.1f%%', startangle=90, colors=plt.cm.Pai

# Titre
plt.title("Proportion des jeux d'action par ESRB Rating (moins de 5% regroupés)", fontsize=14)

# Affichage du graphique
plt.axis('equal')  # Pour un camembert circulaire
plt.show()
```




```
# Supposons que df_listejeux est déjà défini et contient une colonne 'genres'
# Filtrer pour obtenir uniquement les jeux de genre "Aventure"
df_aventure = df_listejeux[df_listejeux['genres'].str.contains('Adventure', case=False, na=False)]

# Compter le nombre de jeux par ESRB Rating
proportion_esrb = df_aventure['esrb_rating'].value_counts()

# Calculer le pourcentage de chaque catégorie
pourcentages = proportion_esrb / proportion_esrb.sum() * 100

# Regrouper les catégories avec moins de 5% dans une catégorie 'Divers'
proportion_esrb_regroupe = pourcentages[pourcentages >= 5].copy()
proportion_esrb_regroupe['Divers'] = pourcentages[pourcentages < 5].sum()

# Création du graphique camembert
plt.figure(figsize=(8, 8))
plt.pie(proportion_esrb_regroupe, labels=proportion_esrb_regroupe.index, autopct='%1.1f%%', startangle=90, colors=plt.cm.Pai

# Titre
plt.title("Proportion des jeux d'aventure par ESRB Rating (moins de 5% regroupés)", fontsize=14)

# Affichage du graphique
plt.axis('equal')  # Pour un camembert circulaire
plt.show()
```




```
# Supposons que df_listejeux est déjà défini et contient une colonne 'genres'
# Filtrer pour obtenir uniquement les jeux de genre "Action"
df_action = df_listejeux[df_listejeux['genres'].str.contains('Action', case=False, na=False)]
# Filtrer pour obtenir uniquement les jeux de genre "Adventure"
df_adventure = df_listejeux[df_listejeux['genres'].str.contains('Adventure', case=False, na=False)]

# Compter le nombre de jeux par ESRB Rating pour les deux genres
proportion_esrb_action = df_action['esrb_rating'].value_counts()
proportion_esrb_adventure = df_adventure['esrb_rating'].value_counts()

# Calculer le pourcentage de chaque catégorie
pourcentages_action = proportion_esrb_action / proportion_esrb_action.sum() * 100
pourcentages_adventure = proportion_esrb_adventure / proportion_esrb_adventure.sum() * 100

# Regrouper les catégories avec moins de 5% dans une catégorie 'Divers'
proportion_esrb_action_regroupe = pourcentages_action[pourcentages_action >= 5].copy()
proportion_esrb_action_regroupe['Divers'] = pourcentages_action[pourcentages_action < 5].sum()

proportion_esrb_adventure_regroupe = pourcentages_adventure[pourcentages_adventure >= 5].copy()
proportion_esrb_adventure_regroupe['Divers'] = pourcentages_adventure[pourcentages_adventure < 5].sum()

# Préparer les données pour le donut
labels_action = proportion_esrb_action_regroupe.index
sizes_action = proportion_esrb_action_regroupe

labels_adventure = proportion_esrb_adventure_regroupe.index
sizes_adventure = proportion_esrb_adventure_regroupe

# Créer le graphique avec deux donuts
fig, ax = plt.subplots(figsize=(10, 10), subplot_kw=dict(aspect="equal"))

# Premier donut pour "Action"
wedges1, texts1, autotexts1 = ax.pie(sizes_action, labels=labels_action, autopct='%1.1f%%', startangle=90,
                                     colors=plt.cm.Paired.colors[:len(labels_action)])

# Dessiner un cercle central pour créer l'effet de donut
centre_circle1 = plt.Circle((0, 0), 0.70, fc='white')
fig.gca().add_artist(centre_circle1)

# Deuxième donut pour "Adventure"
wedges2, texts2, autotexts2 = ax.pie(sizes_adventure, labels=labels_adventure, autopct='%1.1f%%', startangle=90,
                                     radius=0.5, colors=plt.cm.Paired.colors[len(labels_action):len(labels_action)+len(lat

# Dessiner un cercle central pour créer l'effet de donut
centre_circle2 = plt.Circle((0, 0), 0.35, fc='white')
```

```
    fig.gca().add_artist(centre_circle2)

    # Titre
    plt.title("Proportion des jeux par ESRB Rating\n (Moins de 5% regroupés) pour Action et Adventure", fontsize=14)
```