WEB SCRAPPING

PROJET:

AMAZON SCRAPING

AMAZON'S GIFTS CARD

REALISER PAR:

AMZIL Hafsa n °4
ELGOUAR Ahmed n°14
LABYADY Ibtissame n°23
MANSSOURI Othmane n°27
MOUMNASSI Mohammed n°31

TABLE DE MATIERE

INTRODUCTION

1.1 Web scraping :

Le **scraping Web** consiste à extraire des données d'un site Web et à les enregistrer afin de les analyser ou de les utiliser de toute autre manière. Le scraping permet de récolter des informations de nature très différente. Il peut, par ex. Il peut s'agir de coordonnées, telles que des adresses e-mail ou des numéros de téléphone, ou de mots clés ou d'URL individuels. Ces informations sont ensuite collectées dans une base de données ou une table locale.

1.2 Gift card :

Une carte-cadeau peut sembler être **un cadeau idéal**, surtout quand on sait que "tant ceux qui achètent des cadeaux de Noël que ceux qui les reçoivent subissent une pression énorme. Le premier est la peur de la déception, le second est la peur de la déception", PriceMinister observe le PDG Olivier Mathiot. En plus de cela, faire des cadeaux est devenu de plus en plus difficile car beaucoup de gens ont déjà tout ce dont ils ont besoin. Un autre objet s'avère maintenant difficile à fournir. Alors forcément, la grande tendance de Noël est aux expériences et aux cadeaux immatériels (abonnements streaming, voyages, tickets restaurant…). de nombreux *achats* de *cartes-cadeaux* sont effectués chez Amazon .Pour cela on va effectuer une extraction et une analyse des donnés lié par les cartes cadeaux

1.3 Scraping :

L'objectif de ce projet est de créer un webscraper qui extrait les résultats de recherche de produits de **www.amazon.com.** Ce projet est conçu pour donner à un programmeur débutant ou intermédiaire une expérience du webscraping dans une application pratique.

CODE ET COMMENTAIRES

2.1 DEMARRAGE DE WEBDRIVER :

Pour commencer, Import des bibliothèques : Tout d'abord, les bibliothèques nécessaires sont importées, notamment time, csv, BeautifulSoup et web driver de Selenium..

```
import time
import csv
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
import csv
from bs4 import BeautifulSoup
```

Installation de Chromedriver : La commande "!apt-get update" est utilisée pour mettre à jour les packages de système d'exploitation de la machine virtuelle. Ensuite, l'installation de Chromedriver est effectuée avec "!apt install chromium-chromedriver" et les fichiers nécessaires sont copiés avec "!cp /usr/lib/chromium-browser/chromedriver /usr/bin".

Entrée [2]:
```
!apt-get update
!apt install chromium-chromedriver
!cp /usr/lib/chromium-browser/chromedriver /usr/bin
!pip install selenium
```

Définition des options Chrome : La variable "options" est définie pour utiliser le navigateur Chrome en mode headless (sans interface graphique), sans sandbox et avec une utilisation de la mémoire partagée désactivée. Cette configuration est couramment utilisée pour les scripts de scraping, car elle permet de récupérer les données sans l'affichage graphique du navigateur, ce qui rend le script plus rapide et moins gourmand en ressources.

Entrée [3]:
```
from selenium import webdriver
options = webdriver.ChromeOptions()
options.add_argument('--headless')
options.add_argument('--no-sandbox')
options.add_argument('--disable-dev-shm-usage')
driver = webdriver.Chrome('chromedriver',options=options)
driver.implicitly_wait(10)
```

Initialisez une instance du webdriver :

```
driver.get("https://google.com")
from selenium.webdriver.common.by import By
q = driver.find_element(By.NAME, 'q')
q.send_keys('cyublog')
q.submit
from selenium.webdriver.common.action_chains import ActionChains
action = ActionChains(driver)
driver.get("https://www.amazon.com")
```

Maintenant que le pilote Web a démarré, accédez au site Web d'Amazon.
Navigation sur la page Amazon : La fonction "get_url" est définie pour générer une URL de recherche en fonction d'un texte de recherche. Ensuite, l'URL pour rechercher des cartes-cadeaux sur Amazon est récupérée et la page correspondante est ouverte avec "driver.get(url)". La fonction "get_url" remplace simplement les espaces dans le texte de recherche par des signes "+" pour construire l'URL de recherche.

2.2 Prototype :

Extraction de données : Les données sont extraites en utilisant BeautifulSoup pour analyser le code HTML de la page et en utilisant des fonctions pour extraire les informations de chaque élément trouvé. Les résultats de la recherche sont stockés dans un objet "soup" qui contient tout le code HTML de la page. Ensuite, les données pertinentes sont extraites à l'aide de la fonction "extract_record", qui prend un élément HTML de la page et extrait la description, le prix, le rating, le nombre d'avis et l'URL du produit. Cette fonction renvoie un tuple contenant ces informations. Les résultats sont stockés dans une liste de tuples nommée "records".

```
def extract_record(item):
    """Extract and return data from a single record"""

    # description and url
    atag = item.h2.a
    description = atag.text.strip()
    url = 'https://www.amazon.com' + atag.get('href')
    try:
        # product price
        price_parent = item.find('span', 'a-price')
        price = price_parent.find('span', 'a-offscreen').text
    except AttributeError:
        return

    try:
        # rating and review count
        rating = item.i.text
        review_count = item.find('span', {'class': 'a-size-base s-underline-text'}).text
    except AttributeError:
        rating = ''
        review_count = ''

    result = (description, price, rating, review_count, url)

    return result
```

```
records = []

results = soup.find_all('div', {'data-component-type': 's-search-result'})

for item in results:
    record = extract_record(item)
    if record:
        records.append(record)
records
```

2.3 Généraliser :

Après avoir prototyper un article, il est maintenant temps de généraliser ce modèle dans une fonction afin que nous puissions l'appliquer à tous les enregistrements de la page.
Il existe des enregistrements sans prix, sans classements, ni notes, etc. Donc, on a ajouté une gestion des erreurs à notre code pour ces situations. S'il n'y a pas de prix, cela signifie que l'article n'est pas disponible, ce qui ne nous soucie pas, donc le code renvoie juste un enregistrement vide. Cependant, si l'évaluation ou la note est manquante, on les définira simplement comme vides.

```
 results = soup.find_all('div', {'data-component-type': 's-search-result'})
for item in results:
    record = extract_record(item)
    if record:
        records.append(record)
```

```
while True:
    soup = BeautifulSoup(driver.page_source, 'html.parser')
    results = soup.find_all('div', {'data-component-type': 's-search-result'})
    for item in results:
        record = extract_record(item)
        if record:
            records.append(record)
    try:
        next_button = driver.find_element(By.XPATH, "//a[contains(text(),'Next')]")
        ActionChains(driver).move_to_element(next_button).click().perform()
        time.sleep(3)
    except:
        break
```

2.4 Création du csv :

Les résultats sont stockés dans une liste de tuples nommée "records" et sont finalement écrits dans un fichier CSV nommé "gift_cards.csv" en utilisant la bibliothèque csv. La première ligne du fichier CSV est un en-tête contenant les noms des colonnes ("Description", "Price", "Rating", "Review Count", "URL") et chaque ligne suivante contient les données d'un produit.

```
# Write results to CSV file
with open('gft_cards.csv', mode='w', newline='', encoding='utf-8') as file:
    writer = csv.writer(file)
    writer.writerow(['Description', 'Price', 'Rating', 'Review Count', 'URL'])
    writer.writerows(records)
```

Finalement, le driver est fermé avec "driver.quit()". Cela permet de libérer les ressources utilisées par le navigateur et de terminer le script.

```
# Close the driver
driver.quit()
```

ANALYSE

3.1 DATA INFOS

Apres l'importation des bibliotheques : pandas, numpy, matplotlib.pyplot et seaborn :

```
## importation des packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

On cherche les informations suffisantes de notre data :

```
In [4]: data.head()
```

Out[4]:

	Description	Price	Rating	Review Count	URL
0	Amazon.com eGift Card	$1.00	4.8 out of 5 stars	(902,600)	https://www.amazon.com/Amazon-eGift-Card-Logo/...
1	Amazon.com Gift Card Balance Reload	$0.50	4.6 out of 5 stars	(372,292)	https://www.amazon.com/Amazon-com-Gift-Card-Ba...
2	Amazon.com Print at Home Gift Card	$1.00	4.8 out of 5 stars	(84,199)	https://www.amazon.com/Amazon-Gift-Card-Print-...
3	Amazon.com Gift Card in a Reveal (Various Desi...	$20.00	4.9 out of 5 stars	(78,768)	https://www.amazon.com/Amazon-com-Reveal-Class...
4	Amazon.com Gift Card in a Greeting Card (Vario...	$10.00	4.8 out of 5 stars	(70,425)	https://www.amazon.com/Amazon-com-Greeting-Bir...

```
In [5]: print(data.shape)
(366, 5)

In [6]: data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 366 entries, 0 to 365
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Description   366 non-null    object
 1   Price         366 non-null    object
 2   Rating        343 non-null    object
 3   Review Count  343 non-null    object
 4   URL           366 non-null    object
dtypes: object(5)
memory usage: 14.4+ KB
```

On obtient que notre data compose de 366 lignes et 5 colonnes de type 'Object'.

3.2 CLEANING DATA

On cherche si y a :

Des lignes dupliquées :

```
In [12]: data.duplicated().sum()
```

```
Out[12]: 0
```

Des valeurs nulles :

```
In [13]: #checking for null values
         data.isnull().sum()
```

```
Out[13]: Description      0
         Price            0
         Rating          23
         Review Count    23
         URL              0
         dtype: int64
```

Or notre dataset est de 366 lignes on a décidé de supprimer les valeurs nulles :

```
In [90]: data.dropna(inplace=True)
```

```
In [15]: data.isnull().sum()
```

```
Out[15]: Description     0
         Price           0
         Rating          0
         Review Count    0
         URL             0
         dtype: int64
```

Maintenant on cherche de changer le type des colonnes : **Price, Rating, ReviewCount** :
Pour la colonne **Price** on a supprimé '$' et changer le type :

```
In [16]: data['Price'] = data['Price'].str.replace('$', '')
```

```
<ipython-input-16-ad71304e4ed8>:1: FutureWarning: The default value of regex will c
n. In addition, single character regular expressions will *not* be treated as liter
  data['Price'] = data['Price'].str.replace('$', '')
```

```
In [17]: data['Price'] = data['Price'].astype(float)
```

Pour la colonne **Rating** on a extrait que les nombres et changer le type :

```
In [18]: data['Rating'] = data['Rating'].str.extract('(\d+\.\d+)')
```

```
In [19]: data['Rating'] = data['Rating'].astype(float)
```

Pour la colonne **ReviewCount** on a modifier les ‘,’ par des ‘.’ et changer le type :

In [20]:
```
# Replace commas with periods in the 'ReviewCount' column
data['Review Count'] = data['Review Count'].str.replace(',', '.')
data['Review Count'] = data['Review Count'].str.replace('(', '')
data['Review Count'] = data['Review Count'].str.replace(')', '')

# Convert the 'ReviewCount' column to a float
data['Review Count'] = data['Review Count'].astype(float)
```

Finalement :

In [21]:
```
data.head()
```

Out[21]:

	Description	Price	Rating	Review Count	URL
0	Amazon.com eGift Card	1.0	4.8	902.600	https://www.amazon.com/Amazon-eGift-Card-Logo/...
1	Amazon.com Gift Card Balance Reload	0.5	4.6	372.292	https://www.amazon.com/Amazon-com-Gift-Card-Ba...
2	Amazon.com Print at Home Gift Card	1.0	4.8	84.199	https://www.amazon.com/Amazon-Gift-Card-Print-...
3	Amazon.com Gift Card in a Reveal (Various Desi...	20.0	4.9	78.768	https://www.amazon.com/Amazon-com-Reveal-Class...
4	Amazon.com Gift Card in a Greeting Card (Vario...	10.0	4.8	70.425	https://www.amazon.com/Amazon-com-Greeting-Bir...

In [22]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 343 entries, 0 to 365
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Description   343 non-null    object
 1   Price         343 non-null    float64
 2   Rating        343 non-null    float64
 3   Review Count  343 non-null    float64
 4   URL           343 non-null    object
dtypes: float64(3), object(2)
memory usage: 16.1+ KB
```

3.3 DATA VALUES

DATA VALUES

##Analyse descriptive

```
In [23]: data.describe()
```

Out[23]:

	Price	Rating	Review Count
count	343.000000	343.000000	343.000000
mean	29.090117	4.601166	147.483125
std	20.523186	0.569445	229.640458
min	0.500000	1.000000	1.000000
25%	25.000000	4.600000	4.628500
50%	25.000000	4.800000	39.000000
75%	25.000000	4.900000	174.000000
max	250.000000	5.000000	998.000000

La méthode **data.describe()** renvoie un résumé statistique comprenant le nombre d'observations (count), la moyenne (mean), l'écart-type (standard deviation), la valeur minimum (min), le quartile inférieur (25%), la médiane (50%), le quartile supérieur (75%) et la valeur maximum (max) pour chaque colonne numérique du base de données.

3.3.1 ETUDE DU PRIX

ETUDE DU PRIX

```
[91] data["Price"].min()
```

```
0.0
```

```
[92] data["Price"].max()
```

```
250.0
```

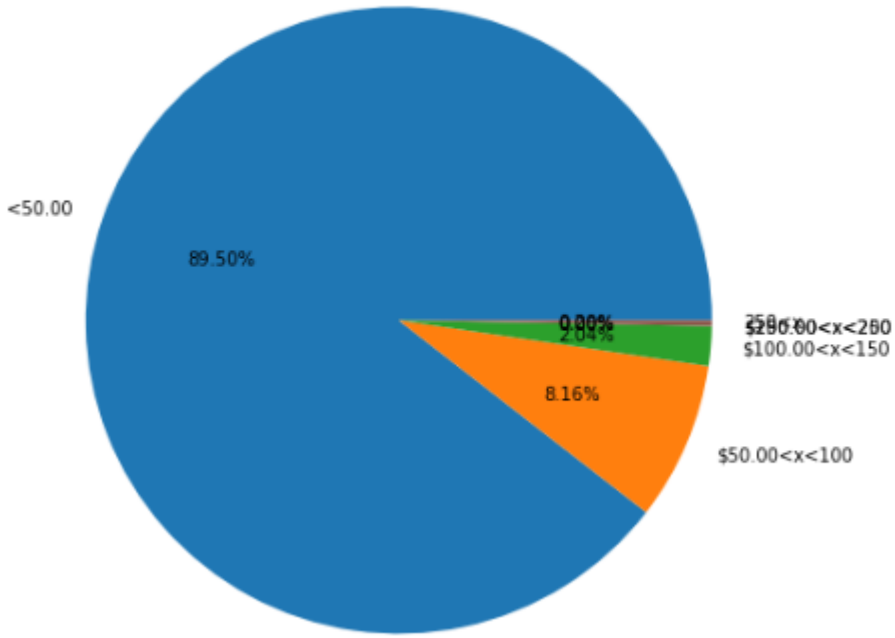
```
data["Price"].mean()
```

```
26.461836734693875
```

```
[94] data["Price"].std()
```

```
22.026541296135047
```

```
[100] plt.figure(figsize = (12,8))
      labels = ["<50.00","$50.00<x<100","$100.00<x<150","$150.00<x<200","$200.00<x<250","250<x"]
      plt.pie([307,28,7,0,0,1],labels = labels, autopct = '%2.2f%%')
```



D’après la figure obtenue on déduit que la majorité des gifts cards ont un prix inferieure a 50$ et de pourcentage 89.50%. Pour les gifts card de 50$<prix<100$ ont un pourcentage de 8.16% de fréquenté, ...

3.3.2 ETUDE DE RATING

ETUDE DU RATING (MIN ET MAX)

```
[95] data["Rating"].max()

     5.0

[96] data["Rating"].min()

     1.0

[97] data["Rating"].mean()

     4.601166180758017
```

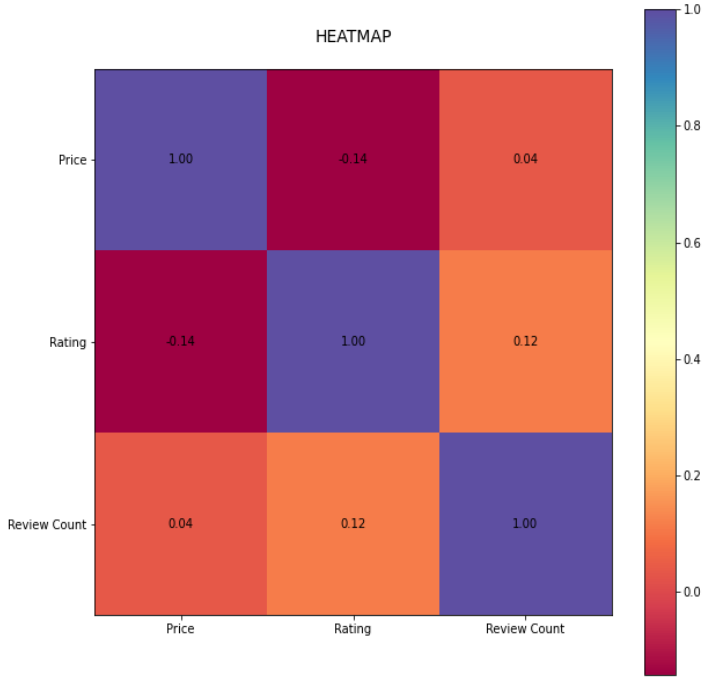
Les valeurs du rating sont :

```
[1.0, 2.1, 2.5, 2.8, 2.9, 3.0, 3.1, 3.2, 3.3, 3.5, 3.6, 3.7, 3.8, 3.9, 4.0, 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 5.0]
```

3.4 DATA MANIPULATING

Premièrement on cherche si y a une corrélation entre Price, Rating et ReviewCount :

```
In [34]: plt.imshow(data.corr(), cmap="Spectral")
         plt.colorbar()
         plt.gcf().set_size_inches(10, 10)
         plt.xticks(range(len(data.corr().columns)), data.corr().columns)
         plt.yticks(range(len(data.corr().columns)), data.corr().columns)
         labels = data.corr().values
         for a in range(labels.shape[0]):
             for b in range(labels.shape[1]):
                 plt.text(a, b, '{:.2f}'.format(labels[b, a]), ha='center', va='center', color='black')
         plt.title('HEATMAP\n', fontsize=14)
         plt.show()
         plt.show()
```

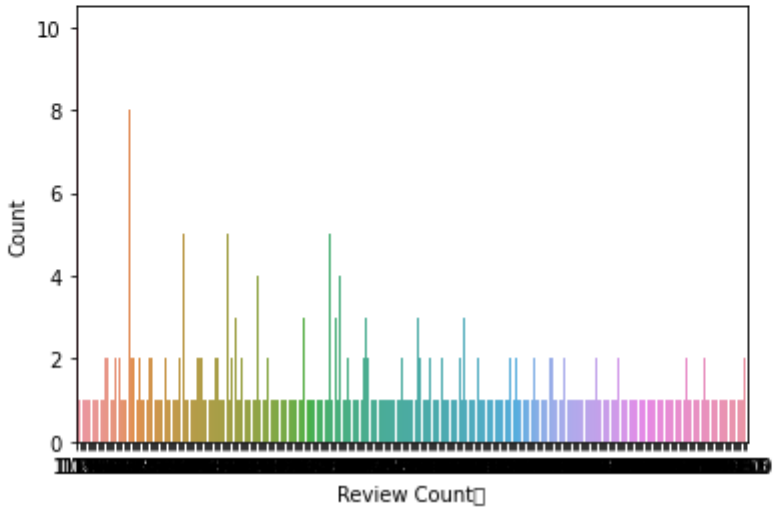


On remarque qu'on n' a pas un corrélation modéree entre le prix , rating et Review count.

On a essayé d'avoir une visualisation sur les types des gifts cards :

```
In [25]: # Create a bar chart of gift card counts by Review Count
sns.countplot(x='Review Count', data=data)
plt.xlabel('Review Count')
plt.ylabel('Count')
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/matplotlib/backends/backend_ag
t.
  font.set_text(s, 0.0, flags=flags)
/usr/local/lib/python3.8/dist-packages/matplotlib/backends/backend_ag
t.
  font.set_text(s, 0, flags=flags)
```



on a pensé à **catégorer les types** des gifts cards **et créer un autre colonne nommée 'Category'** par la fonction :

```
def assign_category(description):
    for category, keywords in categories.items():
        for keyword in keywords:
            if keyword in description.lower():
                return category
    return 'Other'

# Apply the function to the 'Description' column and create a new column 'Category'
data['Category'] = data['Description'].apply(assign_category)
```

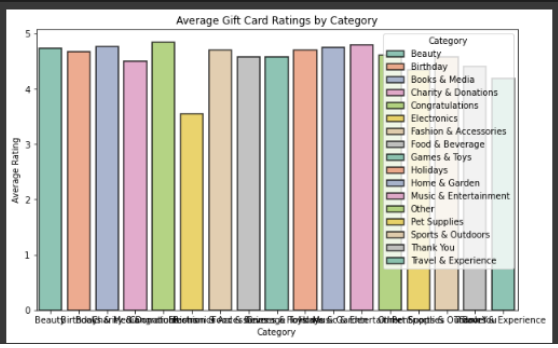
```
In [29]: data.head()
```

Out[29]:

	Description	Price	Rating	Review Count	URL	Category
0	Amazon.com eGift Card	1.0	4.8	902.600	https://www.amazon.com/Amazon-eGift-Card-Logo/...	Other
1	Amazon.com Gift Card Balance Reload	0.5	4.6	372.292	https://www.amazon.com/Amazon-com-Gift-Card-Ba...	Other
2	Amazon.com Print at Home Gift Card	1.0	4.8	84.199	https://www.amazon.com/Amazon-Gift-Card-Print-...	Home & Garden
3	Amazon.com Gift Card in a Reveal (Various Desi...	20.0	4.9	78.768	https://www.amazon.com/Amazon-com-Reveal-Class...	Other
4	Amazon.com Gift Card in a Greeting Card (Vario...	10.0	4.8	70.425	https://www.amazon.com/Amazon-com-Greeting-Bir...	Congratulations

On cherche pour chaque catégorie le **rate** le plus fréquent :

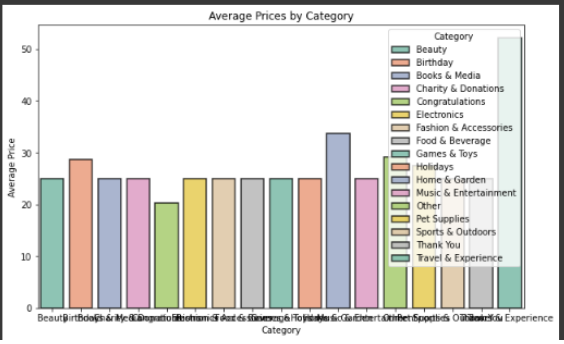
```
# Calculate the mean rating for each category
mean_ratings = data.groupby("Category")["Rating"].mean().reset_index()
# Set the figure size
plt.figure(figsize=(10, 6))
# Create the bar plot
sns.barplot(x="Category", y="Rating", data=mean_ratings, hue="Category", dodge=False, palette="Set2", alpha=0.8, ci=None, edgecolor='black', linewidth=1.5)
# Set the x-axis label
plt.xlabel("Category")
# Set the y-axis label
plt.ylabel("Average Rating")
# Set the title
plt.title("Average Gift Card Ratings by Category")
# Display the plot
plt.show()
```



On remarque que la majorité des ‘’gifts cards’’ a un bon ‘rating’. Cela tend vers que ces cartes d’Amazon sont fiables

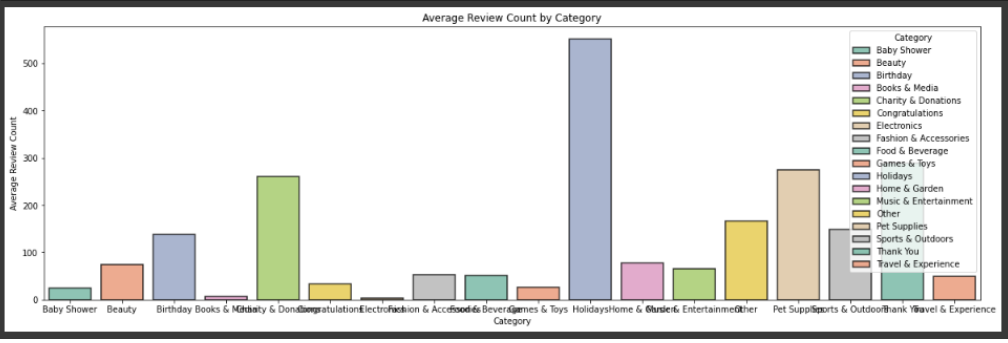
On cherche pour chaque catégorie le **prix** le plus fréquent :

```
# Group the data by 'Category' and calculate the mean price
mean_price = data.groupby('Category')['Price'].mean()

# Convert the resulting Series back to a DataFrame
mean_price = mean_price.to_frame().reset_index()

# Set the size of the figure
plt.figure(figsize=(10, 6))

# Create a bar plot
sns.barplot(x='Category', y='Price', data=mean_price ,  hue="Category", dodge=False, palette="Set2", alpha=0.8, ci=None, edgecolor='black', linewidth=1.5)

# Set the title and axis labels
plt.title('Average Prices by Category')
plt.xlabel('Category')
plt.ylabel('Average Price')

# Display the plot
plt.show()
```



On remarque que la majorité a tendence d'un prix entre 25 et 35 en moyenne . Cependant les cartes cadeaux de type 'Travel & Experience ' ont le plus large prix qui dépasse 50 en moyenne.
On cherche pour chaque catégorie le **average ReviewRate** le plus fréquent :

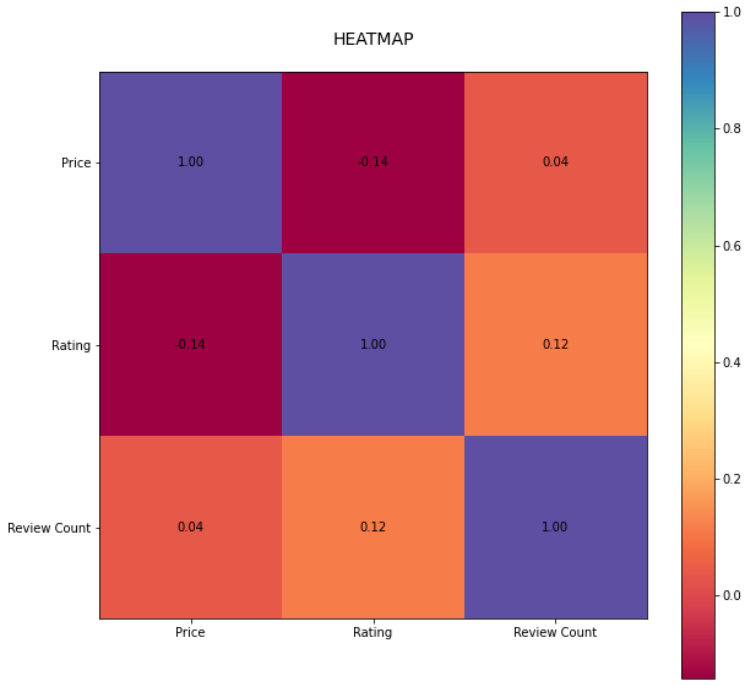
```
# Group the data by 'Category' and calculate the mean Review Count
mean_Review = data.groupby('Category')['Review Count'].mean()

# Convert the resulting Series back to a DataFrame
mean_Review= mean_Review.to_frame().reset_index()

# Set the size of the figure
plt.figure(figsize=(20, 6))

# Create a bar plot
sns.barplot(x='Category', y='Review Count', data=mean_Review  ,  hue="Category", dodge=False, palette="Set2", alpha=0.8, ci=None, edgecolor='black', linewidth=1.5)

# Set the title and axis labels
plt.title('Average Review Count by Category')
plt.xlabel('Category')
plt.ylabel('Average Review Count')

# Display the plot
plt.show()
```



On remarque que les cartes de types Holidays ont un grand important nombre de commentaires qui dépasse 500 en moyenne . Cela signifie que les client réagissent plus avec ce type la .
Ainsi les cartes de type "charity and donation" , "pet supplies" et "thank you" ont un important nombre de commentare de 250 en moyennes .

On cherche la correlation entre chaque categorie et Price,Rating, et ReviewCount à l'aide de HeatMap

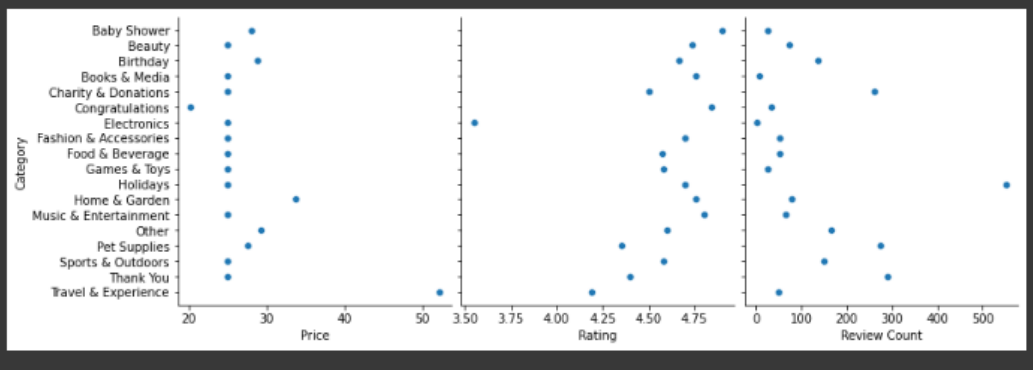
```
In [34]: plt.imshow(data.corr(), cmap="Spectral")
         plt.colorbar()
         plt.gcf().set_size_inches(10, 10)
         plt.xticks(range(len(data.corr().columns)), data.corr().columns)
         plt.yticks(range(len(data.corr().columns)), data.corr().columns)
         labels = data.corr().values
         for a in range(labels.shape[0]):
             for b in range(labels.shape[1]):
                 plt.text(a, b, '{:.2f}'.format(labels[b, a]), ha='center', va='center', color='black')
         plt.title('HEATMAP\n', fontsize=14)
         plt.show()
         plt.show()
```



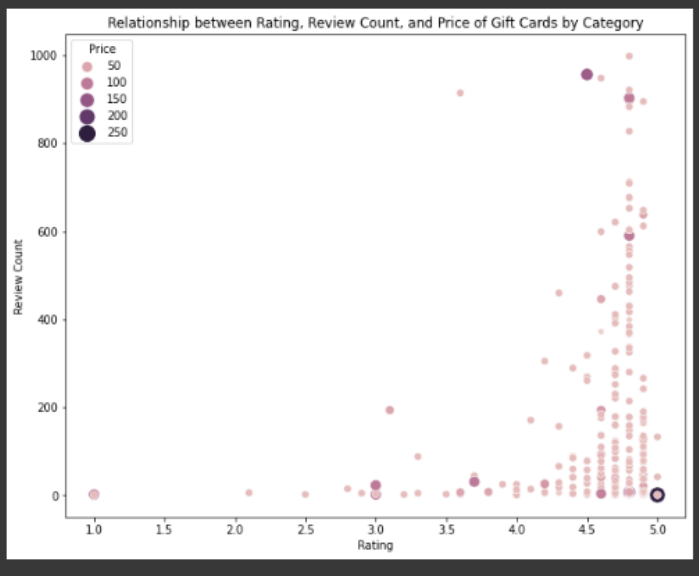
A partir HeatMap on déduit que les variables sont indépendants

On cherche maintenant par un nuage de point la relation entre chaque categorie et Price,Rating, et ReviewCount

```
[ ] sns.pairplot(grouped_df, x_vars=['Price','Rating','Review Count'], y_vars=['Category'], height=4)
    plt.show()
```



```
# Create a scatter plot of rating vs. review count, colored by price
plt.figure(figsize=(10, 8))
sns.scatterplot(data=giftcards_df, x='Rating', y='Review Count', hue='Price', size='Price', sizes=(30, 200))
plt.title('Relationship between Rating, Review Count, and Price of Gift Cards by Category')
plt.xlabel('Rating')
plt.ylabel('Review Count')
plt.show()
```



Dans certaines catégories, les cartes-cadeaux Amazon avec des prix moyens ont des évaluations plus élevées et un grand nombre de commentaires. Cela peut être dû au fait que les clients ont des attentes plus élevées pour les produits et sont donc plus courir à laisser un commentaire s'ils sont satisfaits ou insatisfaits.

Ainsi, il y a également beaucoup de dispersion dans les données, ce qui indique que ce n'est pas une règle absolue et que d'autres facteurs peuvent influencer les évaluations.