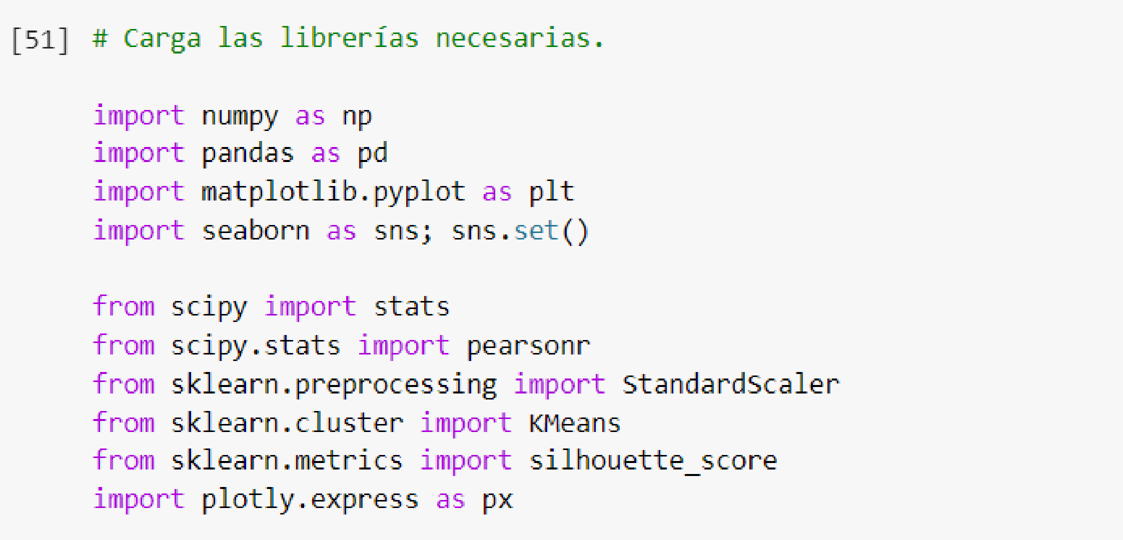
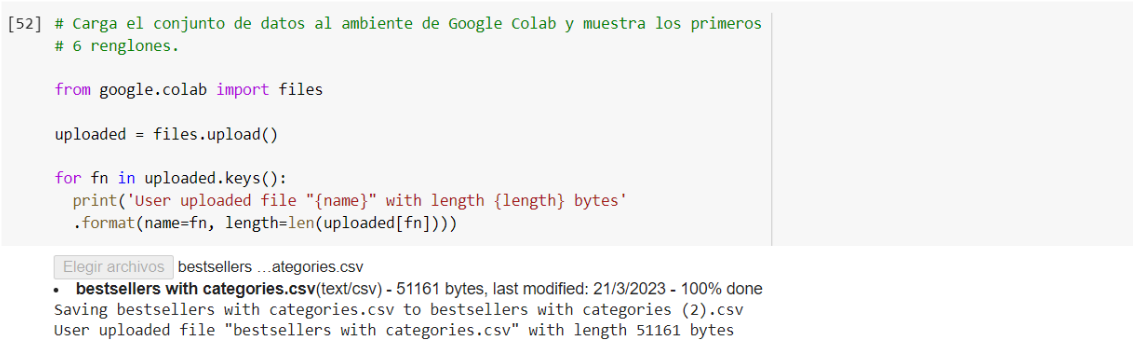
Yuna Chung A01709043

ACTIVIDAD 09 - K-MEANS



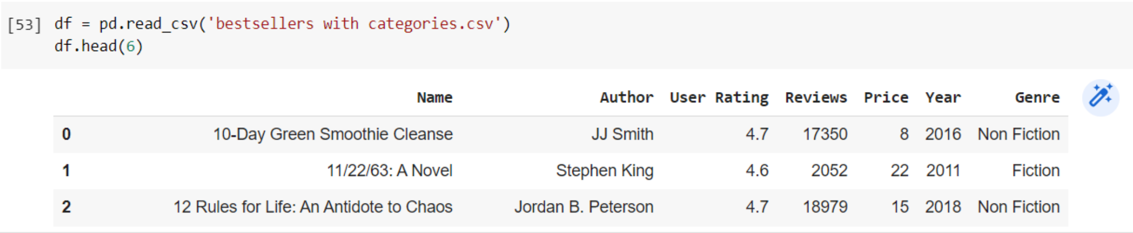
```
[51] # Carga las librerías necesarias.

     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns; sns.set()

     from scipy import stats
     from scipy.stats import pearsonr
     from sklearn.preprocessing import StandardScaler
     from sklearn.cluster import KMeans
     from sklearn.metrics import silhouette_score
     import plotly.express as px
```



```
[52] # Carga el conjunto de datos al ambiente de Google Colab y muestra los primeros
     # 6 renglones.

     from google.colab import files

     uploaded = files.upload()

     for fn in uploaded.keys():
       print('User uploaded file "{name}" with length {length} bytes'
       .format(name=fn, length=len(uploaded[fn])))
```

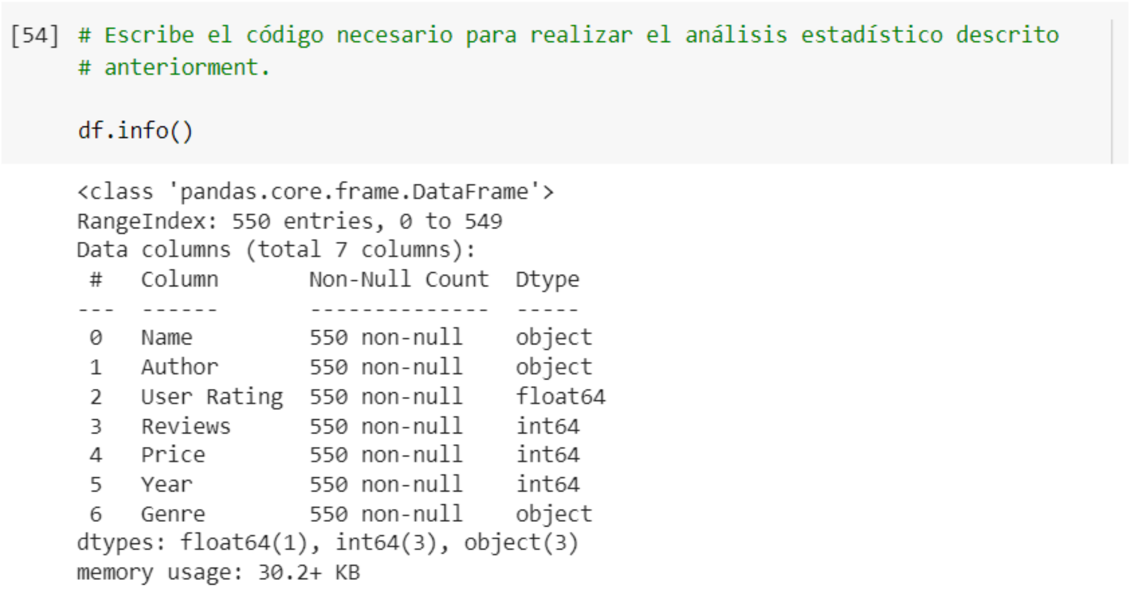
- **bestsellers with categories.csv**(text/csv) - 51161 bytes, last modified: 21/3/2023 - 100% done

```
Saving bestsellers with categories.csv to bestsellers with categories (2).csv
User uploaded file "bestsellers with categories.csv" with length 51161 bytes
```



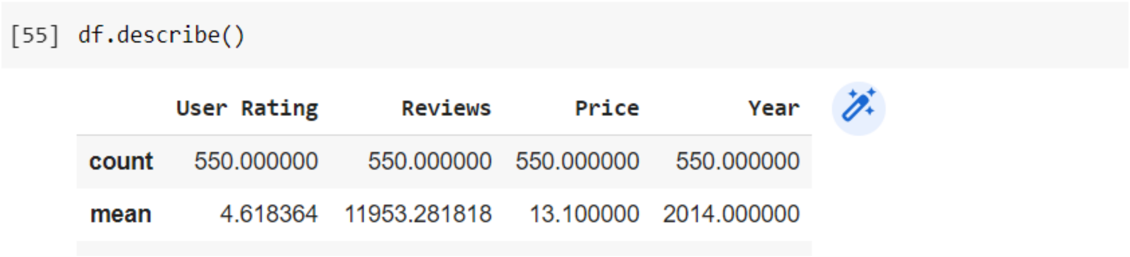
```
[53] df = pd.read_csv('bestsellers with categories.csv')
     df.head(6)
```

	Name	Author	User Rating	Reviews	Price	Year	Genre
0	10-Day Green Smoothie Cleanse	JJ Smith	4.7	17350	8	2016	Non Fiction
1	11/22/63: A Novel	Stephen King	4.6	2052	22	2011	Fiction
2	12 Rules for Life: An Antidote to Chaos	Jordan B. Peterson	4.7	18979	15	2018	Non Fiction



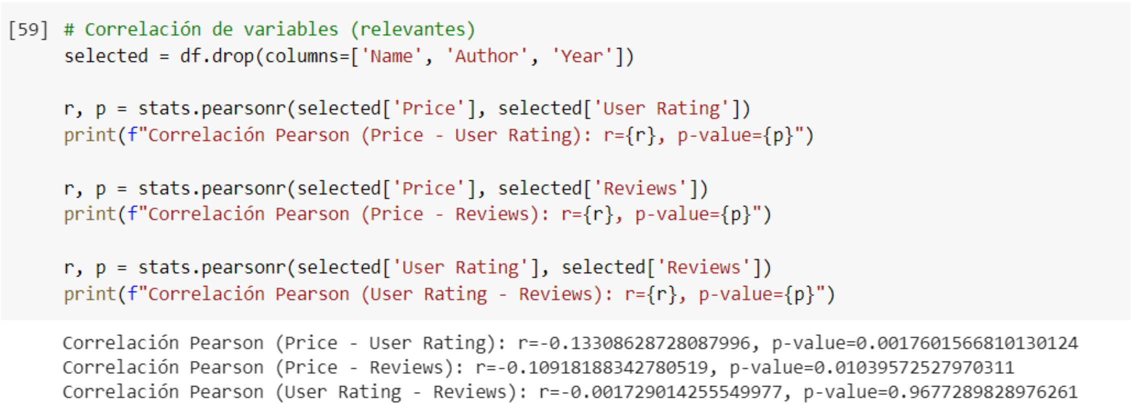
```
[54] # Escribe el código necesario para realizar el análisis estadístico descrito
     # anteriorment.

     df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550 entries, 0 to 549
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Name         550 non-null    object 
 1   Author       550 non-null    object 
 2   User Rating  550 non-null    float64
 3   Reviews      550 non-null    int64  
 4   Price        550 non-null    int64  
 5   Year         550 non-null    int64  
 6   Genre        550 non-null    object 
dtypes: float64(1), int64(3), object(3)
memory usage: 30.2+ KB
```



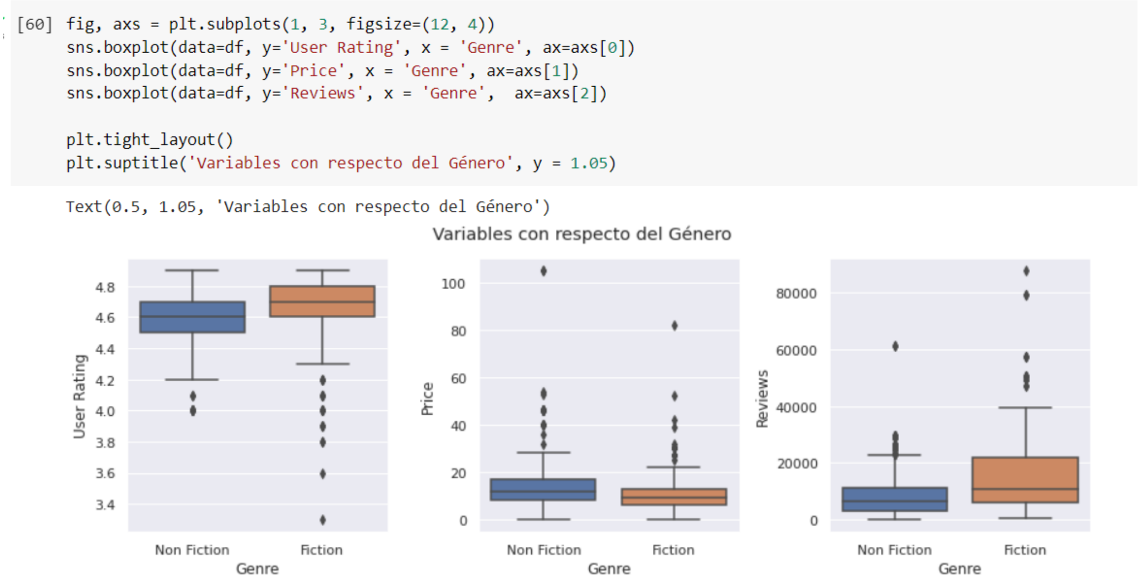
```
[55] df.describe()
```

	User Rating	Reviews	Price	Year
count	550.000000	550.000000	550.000000	550.000000
mean	4.618364	11953.281818	13.100000	2014.000000



```
[59] # Correlación de variables (relevantes)
     selected = df.drop(columns=['Name', 'Author', 'Year'])

     r, p = stats.pearsonr(selected['Price'], selected['User Rating'])
     print(f"Correlación Pearson (Price - User Rating): r={r}, p-value={p}")

     r, p = stats.pearsonr(selected['Price'], selected['Reviews'])
     print(f"Correlación Pearson (Price - Reviews): r={r}, p-value={p}")

     r, p = stats.pearsonr(selected['User Rating'], selected['Reviews'])
     print(f"Correlación Pearson (User Rating - Reviews): r={r}, p-value={p}")
```

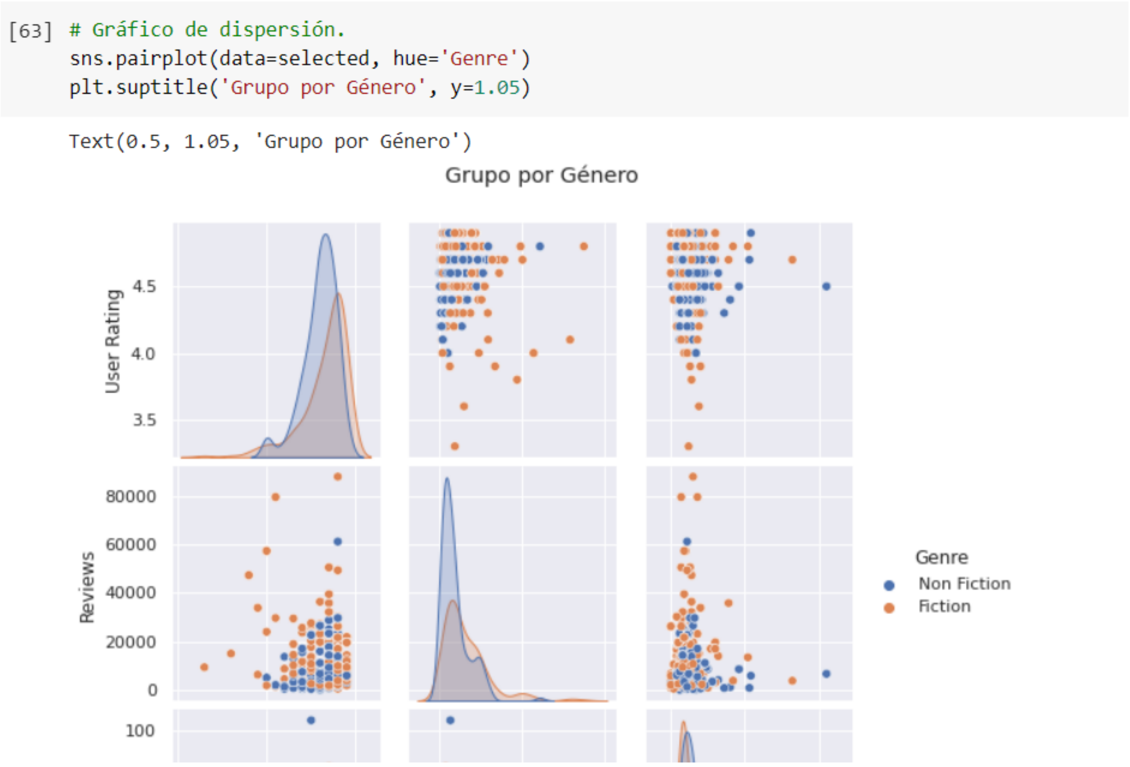
```
Correlación Pearson (Price - User Rating): r=-0.13308628728087996, p-value=0.0017601566810130124
Correlación Pearson (Price - Reviews): r=-0.10918188342780519, p-value=0.010395725279703̶11
Correlación Pearson (User Rating - Reviews): r=-0.001729014255549977, p-value=0.9677289828976261
```



```
[60] fig, axs = plt.subplots(1, 3, figsize=(12, 4))
     sns.boxplot(data=df, y='User Rating', x = 'Genre', ax=axs[0])
     sns.boxplot(data=df, y='Price', x = 'Genre', ax=axs[1])
     sns.boxplot(data=df, y='Reviews', x = 'Genre',  ax=axs[2])

     plt.tight_layout()
     plt.suptitle('Variables con respecto del Género', y = 1.05)
```

```
Text(0.5, 1.05, 'Variables con respecto del Género')
```

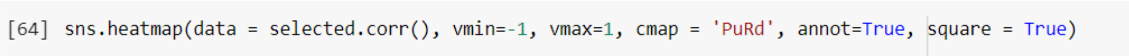
Variables con respecto del Género



```
[63] # Gráfico de dispersión.
     sns.pairplot(data=selected, hue='Genre')
     plt.suptitle('Grupo por Género', y=1.05)
```

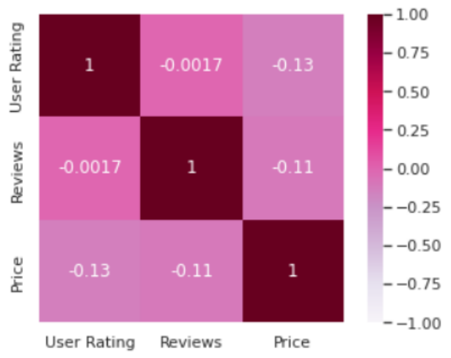
```
Text(0.5, 1.05, 'Grupo por Género')
```

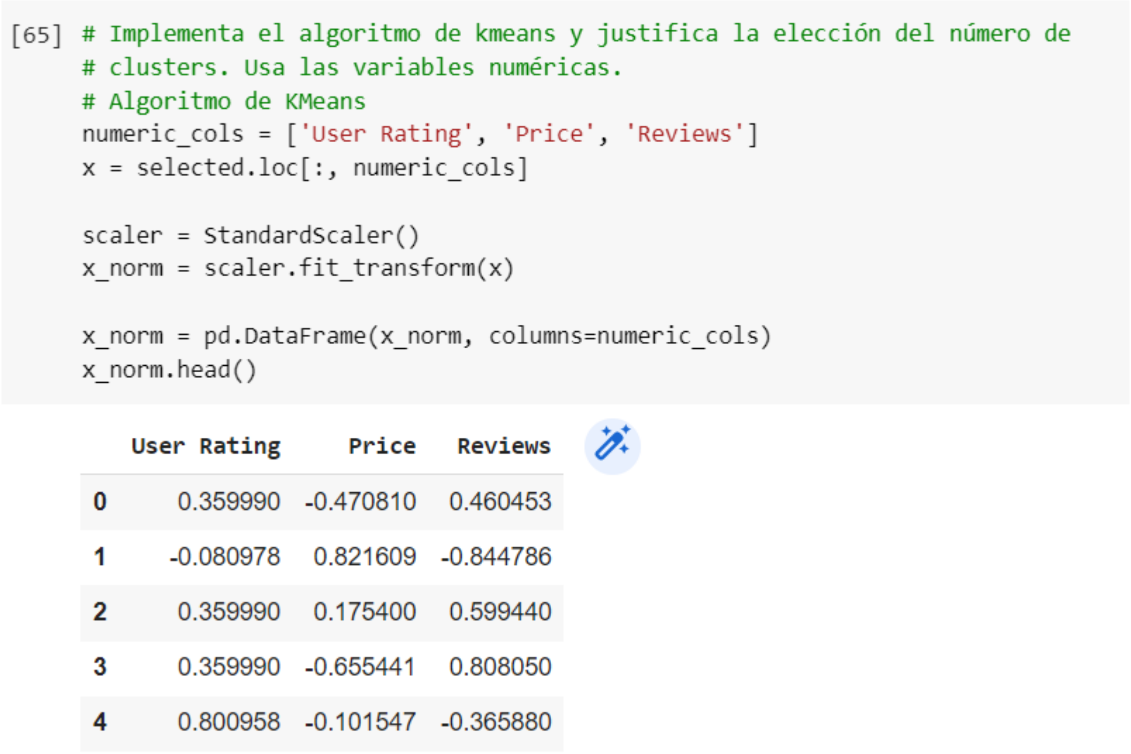
Grupo por Género



```
[64] sns.heatmap(data = selected.corr(), vmin=-1, vmax=1, cmap = 'PuRd', annot=True, square = True)
```

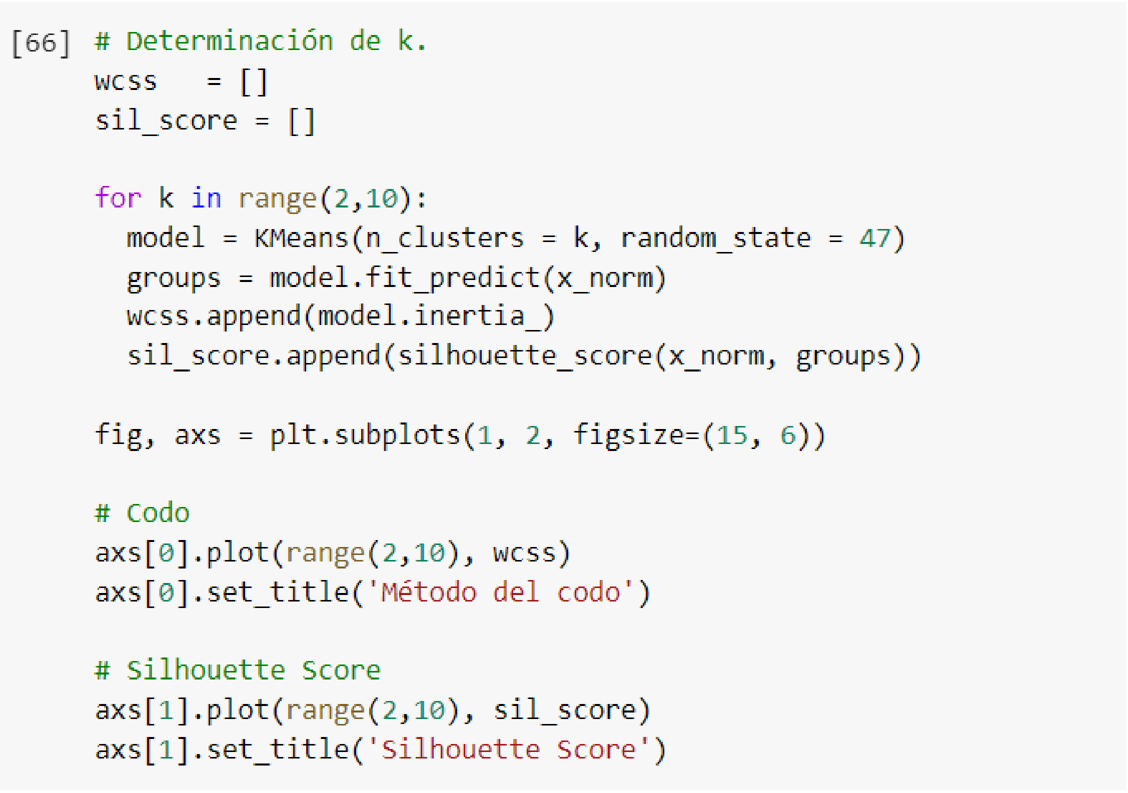
```
<Axes: >
```



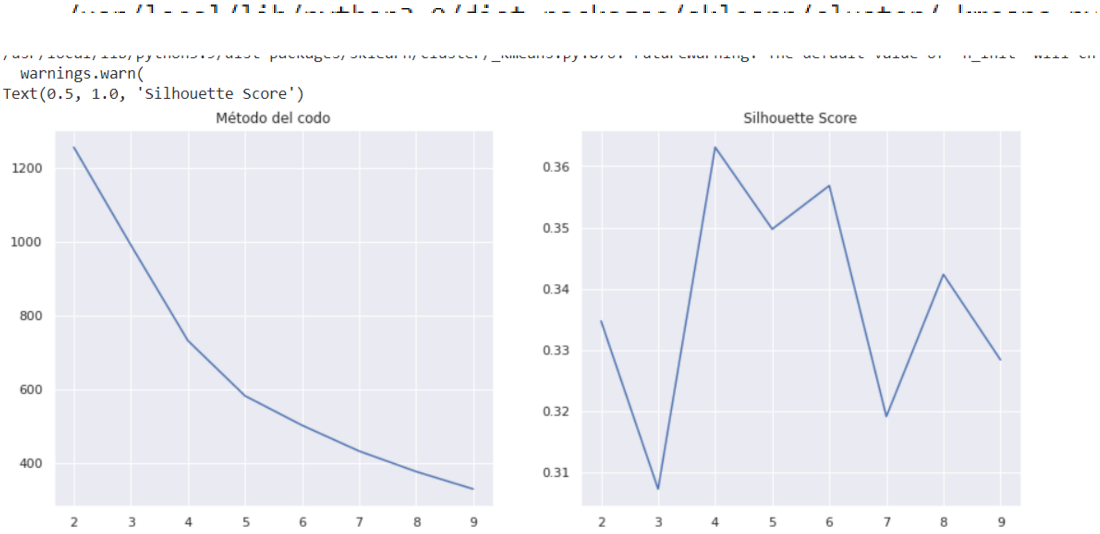


```
[65] # Implementa el algoritmo de kmeans y justifica la elección del número de
     # clusters. Usa las variables numéricas.
     # Algoritmo de KMeans
     numeric_cols = ['User Rating', 'Price', 'Reviews']
     x = selected.loc[:, numeric_cols]

     scaler = StandardScaler()
     x_norm = scaler.fit_transform(x)

     x_norm = pd.DataFrame(x_norm, columns=numeric_cols)
     x_norm.head()
```

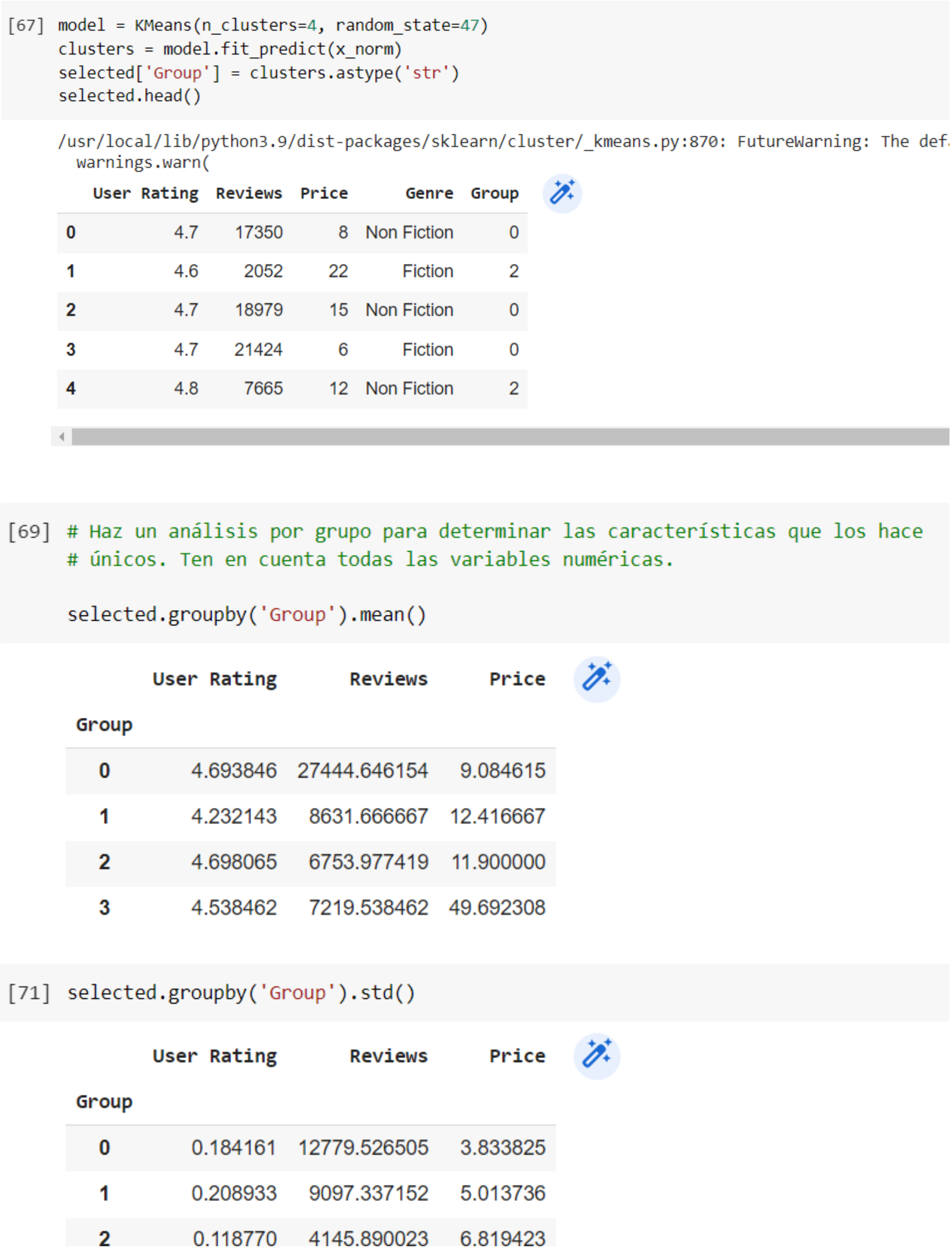
	User Rating	Price	Reviews
0	0.359990	-0.470810	0.460453
1	-0.080978	0.821609	-0.844786
2	0.359990	0.175400	0.599440
3	0.359990	-0.655441	0.808050
4	0.800958	-0.101547	-0.365880



```
[66] # Determinación de k.
     wcss    = []
     sil_score = []

     for k in range(2,10):
       model = KMeans(n_clusters = k, random_state = 47)
       groups = model.fit_predict(x_norm)
       wcss.append(model.inertia_)
       sil_score.append(silhouette_score(x_norm, groups))

     fig, axs = plt.subplots(1, 2, figsize=(15, 6))

     # Codo
     axs[0].plot(range(2,10), wcss)
     axs[0].set_title('Método del codo')

     # Silhouette Score
     axs[1].plot(range(2,10), sil_score)
     axs[1].set_title('Silhouette Score')
```



```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will ch
  warnings.warn(
Text(0.5, 1.0, 'Silhouette Score')
```



```
[67] model = KMeans(n_clusters=4, random_state=47)
     clusters = model.fit_predict(x_norm)
     selected['Group'] = clusters.astype('str')
     selected.head()
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The def
  warnings.warn(
```

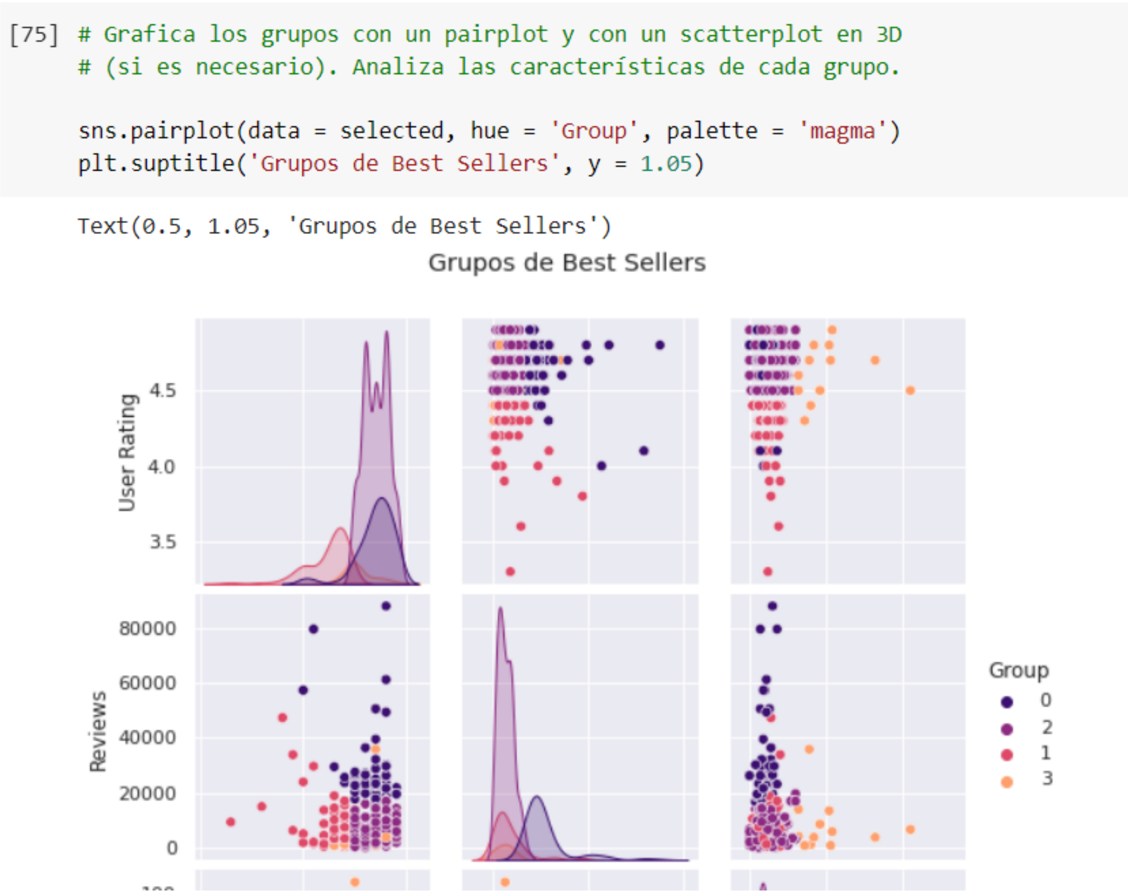
	User Rating	Reviews	Price	Genre	Group
0	4.7	17350	8	Non Fiction	0
1	4.6	2052	22	Fiction	2
2	4.7	18979	15	Non Fiction	0
3	4.7	21424	6	Fiction	0
4	4.8	7665	12	Non Fiction	2

```
[69] # Haz un análisis por grupo para determinar las características que los hace
     # únicos. Ten en cuenta todas las variables numéricas.

     selected.groupby('Group').mean()
```

	User Rating	Reviews	Price
Group			
0	4.693846	27444.646154	9.084615
1	4.232143	8631.666667	12.416667
2	4.698065	6753.977419	11.900000
3	4.538462	7219.538462	49.692308

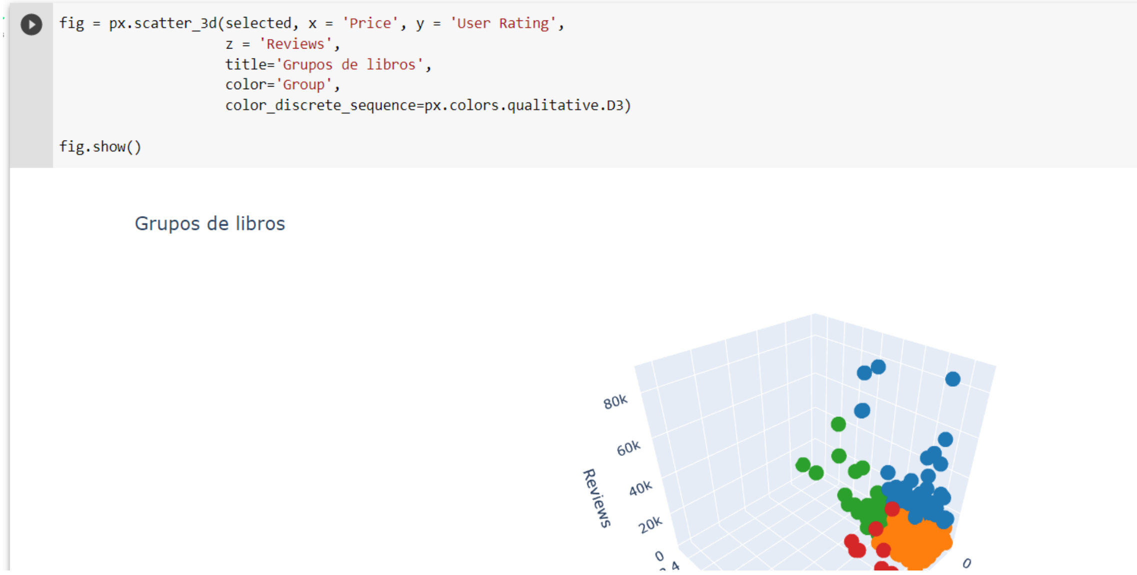
```
[71] selected.groupby('Group').std()
```

	User Rating	Reviews	Price
Group			
0	0.184161	12779.526505	3.833825
1	0.208933	9097.337152	5.013736
2	0.118770	4145.890023	6.819423



```
[75] # Grafica los grupos con un pairplot y con un scatterplot en 3D
     # (si es necesario). Analiza las características de cada grupo.

     sns.pairplot(data = selected, hue = 'Group', palette = 'magma')
     plt.suptitle('Grupos de Best Sellers', y = 1.05)
```

```
Text(0.5, 1.05, 'Grupos de Best Sellers')
```



```
fig = px.scatter_3d(selected, x = 'Price', y = 'User Rating',
                    z = 'Reviews',
                    title='Grupos de libros',
                    color='Group',
                    color_discrete_sequence=px.colors.qualitative.D3)

fig.show()
```