Connected to Python 3.12.5

In [ ]:

```python
import os
import pandas as pd
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from bs4 import BeautifulSoup
```

In [ ]:

```python
# Configuración del driver
driver = webdriver.Chrome()

# Navegar al sitio web
driver.get("https://books.toscrape.com/")

# Esperar a que los elementos estén presentes
wait = WebDriverWait(driver, 10)
elementos_libros = wait.until(EC.presence_of_all_elements_located((By.CSS_SELECTOR, ".product_pod")))
# Extraer datos
books = []
while True:
    soup = BeautifulSoup(driver.page_source, 'html.parser')
    book_elements = soup.find_all('article', class_='product_pod')

    for book in book_elements:
        title = book.h3.a['title']
        price = book.find('p', class_='price_color').text
        rating = book.p['class'][1]
        books.append({'title': title, 'price': price, 'rating': rating})

    # Intentar ir a la siguiente página
    next_button = soup.find('li', class_='next')
    if next_button:
        next_page_url = next_button.a['href']
        driver.get('https://books.toscrape.com/catalogue/' + next_page_url)
    else:
        break

driver.quit()
# Guardar los datos en un archivo CSV
df = pd.DataFrame(books)
file_path = os.path.join(os.getcwd(), "libros.csv")
df.to_csv(file_path, index=False)
df.head()
```

Out[ ]:

| | title | price | rating |
|---|---|---|---|
| **0** | A Light in the Attic | £51.77 | Three |
| **1** | Tipping the Velvet | £53.74 | One |
| **2** | Soumission | £50.10 | One |
| **3** | Sharp Objects | £47.82 | Four |
| **4** | Sapiens: A Brief History of Humankind | £54.23 | Five |

In [ ]:

```python
from flask import Flask, jsonify, request
import pandas as pd

app = Flask(__name__)
df_libros = pd.read_csv("libros.csv")

# Crear un endpoint que devuelve los datos con un filtro
@app.route('/books', methods=['GET'])
def get_books():
    rating_filter = request.args.get('rating')
    if rating_filter:
        filtered_books = df_libros[df_libros['rating'] == rating_filter]
    else:
        filtered_books = df_libros

    return jsonify(filtered_books.to_dict(orient='records'))

if __name__ == '__main__':
    app.run(debug=True)

app.run(debug=True)  # Ejecutar directamente sin el bloque `if __name__ == '__main__':`
```

```
 * Serving Flask app 'proyectoestebanfinal'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
An exception has occurred, use %tb to see the full traceback.

SystemExit: 1
C:\Users\f\AppData\Roaming\Python\Python312\site-packages\IPython\core\interactiveshell.py:3585: UserWarning: To exit: use 'exit', 'quit', or Ctrl-D.
  warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)
```

In [ ]:

```python
from flask import Flask, jsonify, request
import pandas as pd

app = Flask(__name__)
df_libros = pd.read_csv("libros.csv")

# Crear un endpoint que devuelve los datos con un filtro
@app.route('/books', methods=['GET'])
def get_books():
    rating_filter = request.args.get('rating')
    if rating_filter:
        filtered_books = df_libros[df_libros['rating'] == rating_filter]
    else:
        filtered_books = df_libros

    return jsonify(filtered_books.to_dict(orient='records'))

if __name__ == '__main__':
    app.run(debug=True)

app.run(debug=True)  # Ejecutar directamente sin el bloque `if __name__ == '__main__':`
```

```
 * Serving Flask app 'proyectoestebanfinal'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
An exception has occurred, use %tb to see the full traceback.

SystemExit: 1
C:\Users\f\AppData\Roaming\Python\Python312\site-packages\IPython\core\interactiveshell.py:3585: UserWarning: To exit: use 'exit', 'quit', or Ctrl-D.
  warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)
```

In [ ]:

```python
from flask import Flask, jsonify, request
import pandas as pd

app = Flask(__name__)
df_libros = pd.read_csv("libros.csv")

# Crear un endpoint que devuelve los datos con un filtro
@app.route('/books', methods=['GET'])
def get_books():
    rating_filter = request.args.get('rating')
    if rating_filter:
        filtered_books = df_libros[df_libros['rating'] == rating_filter]
    else:
        filtered_books = df_libros

    return jsonify(filtered_books.to_dict(orient='records'))


if __name__ == '__main__':
    app.run(debug=True)

app.run(debug=True)  # Ejecutar directamente sin el bloque `if __name__ == '__main__':`
```

```
 * Serving Flask app 'proyectoestebanfinal'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
An exception has occurred, use %tb to see the full traceback.

SystemExit: 1
C:\Users\f\AppData\Roaming\Python\Python312\site-packages\IPython\core\interactiveshell.py:3585: UserWarning: To exit: use 'exit', 'quit', or Ctrl-D.
  warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)
```

In [ ]:

```python
# Consumir la API
import requests
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Consumir la API
try:
    respuesta = requests.get('http://127.0.0.1:5000/libros')
    respuesta.raise_for_status()  # Lanza una excepción para errores HTTP
    datos_libros = respuesta.json()

    if not datos_libros:  # Verifica si la respuesta está vacía
        raise ValueError("La respuesta de la API está vacía.")

    # Convertir a DataFrame
    df_libros = pd.DataFrame(datos_libros)

    # Crear gráficos
    # Gráfico categórico: conteo de libros por calificación
    sns.countplot(x='rating', data=df_libros)
    plt.title('Conteo de Libros por Calificación')
    plt.show()

    # Gráfico relacional: precio vs calificación
    df_libros['price'] = df_libros['price'].str.replace('£', '').astype(float)
    sns.scatterplot(x='price', y='rating', data=df_libros)
    plt.title('Precio vs Calificación')
    plt.show()

    # Gráfico de distribución: distribución de precios
    sns.histplot(df_libros['price'], kde=True)
    plt.title('Distribución de Precios de los Libros')
    plt.show()

except requests.RequestException as e:
    print(f"Error al realizar la solicitud a la API: {e}")
except ValueError as e:
    print(f"Error en los datos de la API: {e}")
except Exception as e:
    print(f"Ocurrió un error inesperado: {e}")


## He aprendi bastante al realizar este proyecto, en especial sobre como se conforman nuestras pags web y como es necesario conocer su estructuracion para el minado de datos en ellas, en lo pe
# repositorio = https://github.com/estebanrzk/Proyecto_final.git
##
```

```
Error al realizar la solicitud a la API: HTTPConnectionPool(host='127.0.0.1', port=5000): Max retries exceeded with url: /libros (Caused by NewConnectionError('<urllib3.connection.HTTPConnect
ion object at 0x000001D8C7683920>: Failed to establish a new connection: [WinError 10061] No se puede establecer una conexión ya que el equipo de destino denegó expresamente dicha conexión'))
```

In [ ]:

```python
from flask import Flask, jsonify, request
import pandas as pd

app = Flask(__name__)
df_libros = pd.read_csv("libros.csv")

# Crear un endpoint que devuelve los datos con un filtro
@app.route('/libros', methods=['GET'])
def get_books():
    rating_filter = request.args.get('rating')
    if rating_filter:
        filtered_books = df_libros[df_libros['rating'] == rating_filter]
    else:
        filtered_books = df_libros

    return jsonify(filtered_books.to_dict(orient='records'))


if __name__ == '__main__':
    app.run(debug=True)

app.run(debug=True)  # Ejecutar directamente sin el bloque `if __name__ == '__main__':`
```

```
 * Serving Flask app 'proyectoestebanfinal'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
An exception has occurred, use %tb to see the full traceback.

SystemExit: 1
C:\Users\f\AppData\Roaming\Python\Python312\site-packages\IPython\core\interactiveshell.py:3585: UserWarning: To exit: use 'exit', 'quit', or Ctrl-D.
  warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)
```

In [ ]:

```python
# Consumir la API
import requests
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Consumir la API
try:
    respuesta = requests.get('http://127.0.0.1:5000/libros')
    respuesta.raise_for_status()  # Lanza una excepción para errores HTTP
    datos_libros = respuesta.json()

    if not datos_libros:  # Verifica si la respuesta está vacía
        raise ValueError("La respuesta de la API está vacía.")

    # Convertir a DataFrame
    df_libros = pd.DataFrame(datos_libros)

    # Crear gráficos
    # Gráfico categórico: conteo de libros por calificación
    sns.countplot(x='rating', data=df_libros)
    plt.title('Conteo de Libros por Calificación')
    plt.show()

    # Gráfico relacional: precio vs calificación
    df_libros['price'] = df_libros['price'].str.replace('£', '').astype(float)
    sns.scatterplot(x='price', y='rating', data=df_libros)
    plt.title('Precio vs Calificación')
    plt.show()

    # Gráfico de distribución: distribución de precios
    sns.histplot(df_libros['price'], kde=True)
    plt.title('Distribución de Precios de los Libros')
    plt.show()

except requests.RequestException as e:
    print(f"Error al realizar la solicitud a la API: {e}")
except ValueError as e:
    print(f"Error en los datos de la API: {e}")
except Exception as e:
    print(f"Ocurrió un error inesperado: {e}")


## He aprendi bastante al realizar este proyecto, en especial sobre como se conforman nuestras pags web y como es necesario conocer su estructuracion para el minado de datos en ellas, en lo pe
# repositorio = https://github.com/estebanrzk/Proyecto_final.git
##
```

```
Error al realizar la solicitud a la API: HTTPConnectionPool(host='127.0.0.1', port=5000): Max retries exceeded with url: /libros (Caused by NewConnectionError('<urllib3.connection.HTTPConnect
ion object at 0x000001D8C77985F0>: Failed to establish a new connection: [WinError 10061] No se puede establecer una conexión ya que el equipo de destino denegó expresamente dicha conexión'))
```

In [ ]:

```python
# Configuración del driver
driver = webdriver.Chrome()

# Navegar al sitio web
driver.get("https://books.toscrape.com/")

# Esperar a que los elementos estén presentes
wait = WebDriverWait(driver, 10)
elementos_libros = wait.until(EC.presence_of_all_elements_located((By.CSS_SELECTOR, ".product_pod")))
# Extraer datos
books = []
while True:
    soup = BeautifulSoup(driver.page_source, 'html.parser')
    book_elements = soup.find_all('article', class_='product_pod')

    for book in book_elements:
        title = book.h3.a['title']
        price = book.find('p', class_='price_color').text
        rating = book.p['class'][1]
        books.append({'title': title, 'price': price, 'rating': rating})

    # Intentar ir a la siguiente página
    next_button = soup.find('li', class_='next')
    if next_button:
        next_page_url = next_button.a['href']
        driver.get('https://books.toscrape.com/catalogue/' + next_page_url)
    else:
        break

driver.quit()
# Guardar los datos en un archivo CSV
df = pd.DataFrame(books)
file_path = os.path.join(os.getcwd(), "libros.csv")
df.to_csv(file_path, index=False)
df.head()
```

Out[ ]:

| | title | price | rating |
|---|---|---|---|
| **0** | A Light in the Attic | £51.77 | Three |
| **1** | Tipping the Velvet | £53.74 | One |
| **2** | Soumission | £50.10 | One |
| **3** | Sharp Objects | £47.82 | Four |
| **4** | Sapiens: A Brief History of Humankind | £54.23 | Five |

In [ ]:

```python
from flask import Flask, jsonify, request
import pandas as pd

app = Flask(__name__)
df_libros = pd.read_csv("libros.csv")

# Crear un endpoint que devuelve los datos con un filtro
@app.route('/libros', methods=['GET'])
def get_books():
    rating_filter = request.args.get('rating')
    if rating_filter:
        filtered_books = df_libros[df_libros['rating'] == rating_filter]
    else:
        filtered_books = df_libros

    return jsonify(filtered_books.to_dict(orient='records'))


if __name__ == '__main__':
    app.run(debug=True)

app.run(debug=True)  # Ejecutar directamente sin el bloque `if __name__ == '__main__':`
```

```
 * Serving Flask app 'proyectoestebanfinal'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
An exception has occurred, use %tb to see the full traceback.

SystemExit: 1
C:\Users\f\AppData\Roaming\Python\Python312\site-packages\IPython\core\interactiveshell.py:3585: UserWarning: To exit: use 'exit', 'quit', or Ctrl-D.
  warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)
```

In [ ]:

```python
from flask import Flask, jsonify, request
import pandas as pd

app = Flask(__name__)
df_libros = pd.read_csv("libros.csv")

# Crear un endpoint que devuelve los datos con un filtro
@app.route('/libros', methods=['GET'])
def get_books():
    rating_filter = request.args.get('rating')
    if rating_filter:
        filtered_books = df_libros[df_libros['rating'] == rating_filter]
    else:
        filtered_books = df_libros

return jsonify(filtered_books.to_dict(orient='records'))


if __name__ == '__main__':
    app.run(debug=True)

app.run(debug=True)  # Ejecutar directamente sin el bloque `if __name__ == '__main__':`
```

```
  Cell In[11], line 16
    return jsonify(filtered_books.to_dict(orient='records'))
    ^
SyntaxError: 'return' outside function
```

In [ ]:

```python
from flask import Flask, jsonify, request
import pandas as pd

app = Flask(__name__)
df_libros = pd.read_csv("libros.csv")

# Crear un endpoint que devuelve los datos con un filtro
@app.route('/libros', methods=['GET'])
def get_books():
    rating_filter = request.args.get('rating')
    if rating_filter:
        filtered_books = df_libros[df_libros['rating'] == rating_filter]
    else:
        filtered_books = df_libros

    return jsonify(filtered_books.to_dict(orient='records'))


if __name__ == '__main__':
    app.run(debug=True)

app.run(debug=True)  # Ejecutar directamente sin el bloque `if __name__ == '__main__':`
```

```
 * Serving Flask app 'proyectoestebanfinal'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
An exception has occurred, use %tb to see the full traceback.

SystemExit: 1
C:\Users\f\AppData\Roaming\Python\Python312\site-packages\IPython\core\interactiveshell.py:3585: UserWarning: To exit: use 'exit', 'quit', or Ctrl-D.
  warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)
```