**✅ Project Title:**

**E-Book Scraper & Organizer**

**Goal:**

Scrape book data (title, price, availability, rating) from [http://books.toscrape.com](http://books.toscrape.com/), store it in JSON or SQLite, and organize books by price or rating.

**Prerequisites:**

* Python installed
* VS Code installed
* Python extensions in VS Code
* Libraries: requests, beautifulsoup4, json, sqlite3 (standard), pandas (optional)

**📁 Folder Structure:**

ebook\_scraper\_project/

├── scraper.py # Main script for scraping

├── database.py # (Optional) DB-related functions

├── utils.py # (Optional) Helper functions

├── data/

│ ├── books.json # JSON output

│ └── books.sqlite # SQLite DB (optional)

├── analysis.py # Script to organize/filter data

├── README.md # Project documentation

└── requirements.txt # Python libraries used

**🪜 Step-by-Step Plan:**

**🟢 STEP 1: Setup the Project**

1. Create the project folder ebook\_scraper\_project/
2. Open it in VS Code
3. Create all the files and folders shown in the structure
4. Set up a virtual environment (optional but recommended):
5. python -m venv venv
6. source venv/bin/activate # For Linux/Mac
7. venv\Scripts\activate # For Windows

**🟡 STEP 2: Install Required Libraries**

In terminal:

pip install requests beautifulsoup4

pip install pandas # Optional, for analysis

Add these to requirements.txt:

requests

beautifulsoup4

pandas

**🔵 STEP 3: Explore the Website**

Visit: [http://books.toscrape.com](http://books.toscrape.com/)

**Observe:**

* Each book is in a div with class product\_pod
* Title is inside the <a> tag in <h3>
* Price is inside p with class price\_color
* Availability is in p with class instock availability
* Rating is a class in the p tag (e.g., star-rating Three)
* There are **50 pages** to paginate

**🟣 STEP 4: Write the Scraper Logic**

In scraper.py:

* Use requests to get HTML of a page
* Use BeautifulSoup to parse it
* Extract: title, price, availability, and rating
* Loop through all pages (1 to 50)
* Store each book as a Python dictionary

**🟤 STEP 5: Store the Data**

Choose any **one or both** formats:

**✅ JSON:**

* Store book dictionaries in a list
* Use json.dump() to save into books.json

**✅ SQLite:**

* Create a table with columns: title, price, availability, rating
* Insert data using sqlite3 or keep functions in database.py

**⚫ STEP 6: Data Cleanup (Optional)**

* Convert price to float (strip £)
* Normalize rating (e.g., "Three" → 3)
* Clean whitespace in availability

**🔵 STEP 7: Organize/Analyze the Data**

In analysis.py:

You can:

* Sort by price
* Sort by rating
* Filter books under a certain price
* Count books available vs. out of stock
* (If using pandas) Load JSON/DB and analyze easily

**🟢 STEP 8: Display or Export Results**

* Print top 10 cheapest books
* Export filtered results to new JSON/CSV files
* Optionally, display results in a simple CLI menu

**🔴 STEP 9: Test Everything**

* Run the scraper and check JSON / DB
* Try out the analysis features
* Handle exceptions like network issues or missing tags

**🔵 STEP 10: Document the Project**

In README.md, write:

* Project overview
* How to run the scraper
* What data it collects
* Sample output
* Future improvement ideas

**PHASE 1: Polish & Document the Core**

**✅ 1. Add a README.md**

Include:

* What the project does (in plain English)
* Technologies used
* How to run it
* Sample output (e.g., book\_titles.json or SQLite screenshot)

✅ Bonus: Show before/after data  
✅ Use GitHub-flavored markdown (with headers, code, tables, emojis)

**✅ 2. Structure Your Project Folder**

book-scraper/

│

├── main.py # Your scraping + SQLite code

├── book\_scraping.json # Saved JSON

├── books.db # SQLite DB

├── requirements.txt # pip freeze > requirements.txt

├── README.md # Project overview

└── screenshots/ # Optional visuals

**🌟 PHASE 2: Add Smart Extensions**

**🔹 1. Price Normalization**

Convert £10.99 → 10.99 (as REAL in SQLite)  
So you can run:

SELECT \* FROM books WHERE price < 20 ORDER BY rating DESC;

**🔹 2. Data Analysis with Pandas + Matplotlib**

Create a Jupyter Notebook or script to:

* Plot number of books per rating
* Average price per rating
* Price distribution histogram

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_sql("SELECT \* FROM books", conn)

df['price'] = df['price'].str.replace("£", "").astype(float)

df.groupby('rating')['price'].mean().plot(kind='bar')

plt.title("Average Book Price by Rating")

plt.show()

**🔹 3. Visual Dashboard (Optional but Wow)**

Create a small **Streamlit** or **Tkinter** app to:

* Let users search books by title, rating, price
* Display results from SQLite in a table
* Allow filtering or sorting dynamically

Streamlit example:

pip install streamlit

streamlit run app.py

**💼 PHASE 3: Present It Professionally**

**✅ Host on GitHub:**

* Push your project repo
* Use a clean commit history
* Add tags like #webscraping #python #sqlite #portfolio

**✅ Add Screenshots**

* JSON preview
* SQLite browser view
* Jupyter plots
* Your terminal running the scraper

**✅ Mention in Your Resume / Portfolio**

Include:

“Built a Python-based web scraper to extract and store 1000+ book entries from a public website using BeautifulSoup, Requests, JSON, and SQLite. Extended project to include data visualization and queryable database.”

**✨ Optional Expert Add-ons (Only if you want to go big)**

| **Idea** | **Description** |
| --- | --- |
| 🔄 Auto-Scraping Mode | Add a button to refresh latest data |
| 🕵️ Scrape Book Descriptions | Go into each book’s detail page and extract more fields |
| 🌐 Export as CSV | Let users download the scraped data |
| 📊 Power BI Dashboard | Connect SQLite to Power BI or Tableau |
| 🧪 Test Cases | Add unit tests to show code reliability |

**🎯 Summary: Your Next Steps**

| **Priority** | **Action** |
| --- | --- |
| ✅ Must-Have | Add README, JSON + SQLite output, clean structure |
| 🚀 Impressive | Visualizations, price normalization, Streamlit |
| 💼 Professional | Host on GitHub, explain in resume/LinkedIn, show screenshots |

Would you like a **custom README template** or help turning it into a **Streamlit portfolio project**? I’d be happy to help you finalize it!