📘 Books Data Analytics Project

👨‍💻 Created by: Raj Bhirad

🤖 Guided by: ChatGPT AI Mentor

📅 Date: October 2025

📊 “Data Visualization Script (Matplotlib)”

(Code)

```
import pandas as pd import
matplotlib.pyplot as plt

# Load cleaned dataset df =
pd.read_csv("books_cleaned_v3.csv", encoding="latin1")

# Ensure Price is numeric (in case of weird characters) df["Price"]
= pd.to_numeric(df["Price"], errors="coerce")

# --- Chart 1: Average Price by Rating --- avg_price
= df.groupby("Rating")["Price"].mean()
plt.figure(figsize=(8, 5))
avg_price.plot(kind="bar", color="skyblue", edgecolor="black")
plt.title("Average Book Price by Rating", fontsize=14, weight='bold')
plt.xlabel("Rating (Stars)") plt.ylabel("Average Price (£)")
plt.grid(axis="y", linestyle="--", alpha=0.7) plt.tight_layout()
plt.show()
```

```
# --- Chart 2: Availability Breakdown ---

availability_counts = df["Availability"].value_counts()

plt.figure(figsize=(6, 6)) availability_counts.plot(

kind="pie",     autopct="%1.1f%%",

   startangle=90,
   colors=["lightgreen", "lightcoral"],

textprops={"fontsize": 12}

)

plt.title("Book Availability Status", fontsize=14, weight='bold')

plt.ylabel("") plt.tight_layout() plt.show()


# --- Chart 3: Top 10 Most Expensive Books --- top10 = df.nlargest(10, "Price")

plt.figure(figsize=(9, 6)) plt.barh(top10["Book Title"], top10["Price"],

color="orange", edgecolor="black") plt.gca().invert_yaxis()  # Highest at top

plt.title("Top 10 Most Expensive Books", fontsize=14, weight='bold')

plt.xlabel("Price (£)") plt.ylabel("Book Title") plt.tight_layout() plt.show()


print("✅ All visualizations created suc
```

“Data Scraping Script (BeautifulSoup)” (Code)

```
# scrape_books.py import

requests from bs4 import

BeautifulSoup import pandas

as pd import time


def parse_page(url):

   """Return list of (title, price) tuples from one page URL."""

resp = requests.get(url)     resp.raise_for_status()   # will raise
```

```
an error if request failed     soup = BeautifulSoup(resp.text,
"html.parser")     books = soup.find_all("article",
class_="product_pod")

   rows = []     for
book in books:
       title = book.h3.a.get("title", "").strip()        price_text =
book.find("p", class_="price_color").text.strip()
rows.append((title, price_text))     return rows

def clean_price(price_text):
   """Remove currency symbol and convert to float."""
# remove non-numeric characters except dot     cleaned
= price_text.replace("£", "").strip()
   try:        return
float(cleaned)     except:
       # if conversion fails, return None
return None

def main():
   base = "https://books.toscrape.com/catalogue/page-{}.html"
first_page = "https://books.toscrape.com/"

   all_rows = []

   # --------- OPTION A: Single page (homepage) ----------
print("Scraping single homepage...")     rows =
parse_page(first_page)     all_rows.extend(rows)

   # --------- OPTION B: Pagination (multiple pages) ----------
```

```
    # If you want to scrape pages 1..N, set N here.
    # Note: catalogue starts from page-1.html; homepage is a different url.
scrape_multiple_pages = True   # set to False if you only want homepage     if
scrape_multiple_pages:
       N = 5   # number of pages to scrape (adjust as needed, e.g., 50)
print(f"Scraping {N} catalogue pages (this may take a while)...")
for i in range(1, N+1):
          page_url = base.format(i)
          try:
             rows = parse_page(page_url)
all_rows.extend(rows)               print(f"  - page {i} scraped,
{len(rows)} books found")           except Exception as e:
             print(f"  ! error scraping page {i}: {e}")
          time.sleep(1)  # polite delay between requests

    # Build DataFrame and clean price     df = pd.DataFrame(all_rows,
columns=["Book Title", "PriceText"])     df["Price"] =
df["PriceText"].apply(clean_price)     df = df[["Book Title", "Price",
"PriceText"]]

    # Save to CSV     out_file = "books.csv"
df.to_csv(out_file, index=False, encoding="utf-8")
print(f"\n Saved {len(df)} rows to {out_file}")

if __name__ == "__main__":
    main()
```