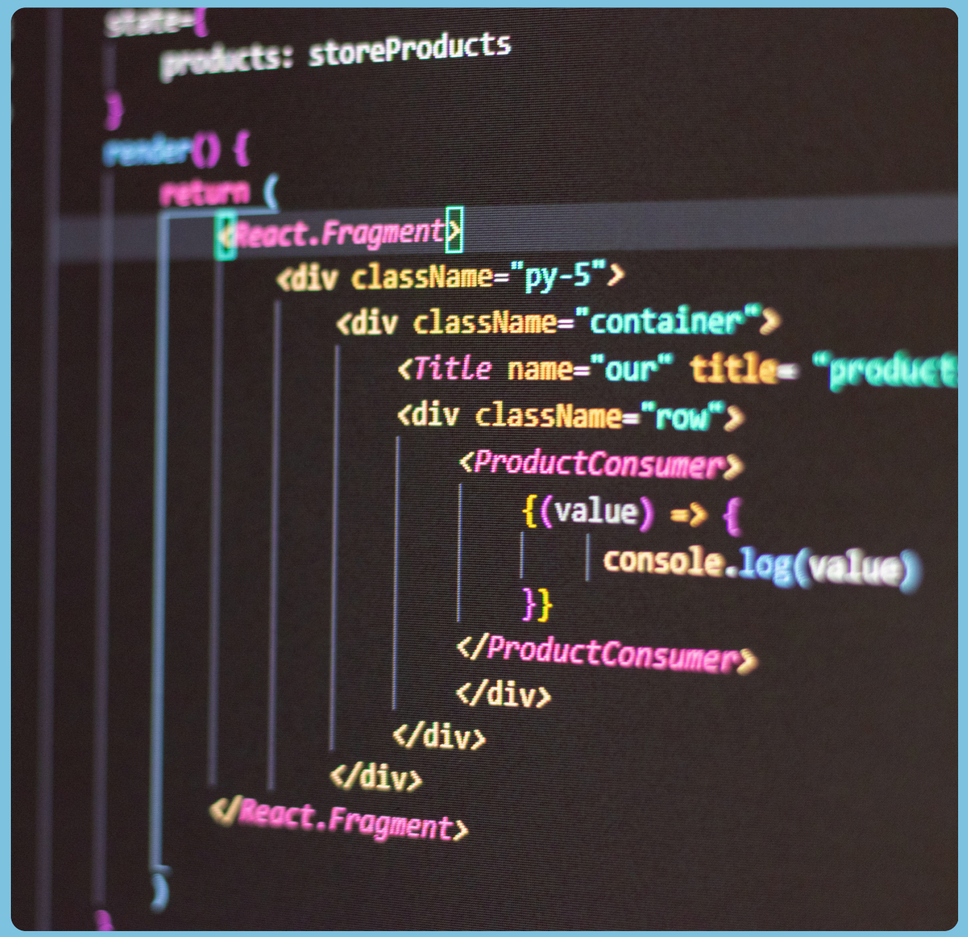
ibimbing

Web Scraping using Beautiful Soup (remastered)

DIGITAL SKILL FAIR 44.0 - Faculty of Data : Data Engineering

Participant : Endah Rakhmawati

Project Overview



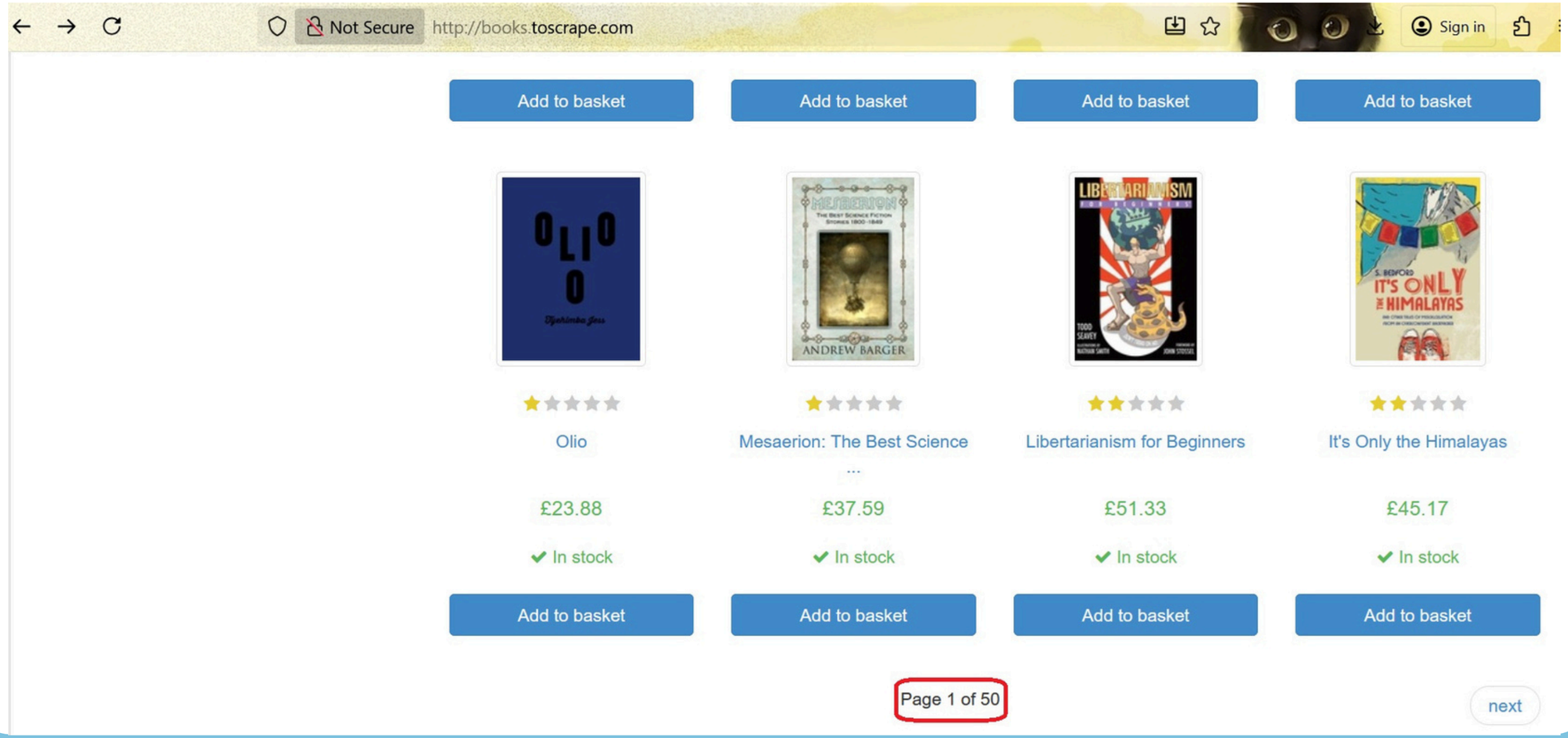
Web scraping is the process of automatically extracting data from websites.
This project parsing the data from bookstore website using Beautiful Soup tool. The scraper reads the HTML structure and extracts specific elements (like titles, prices, star ratings, etc.). It allows you to collect large amounts of information quickly, which can then be analyzed, stored in postgreSQL database, or used for various applications (like market research, data analysis, or machine learning).

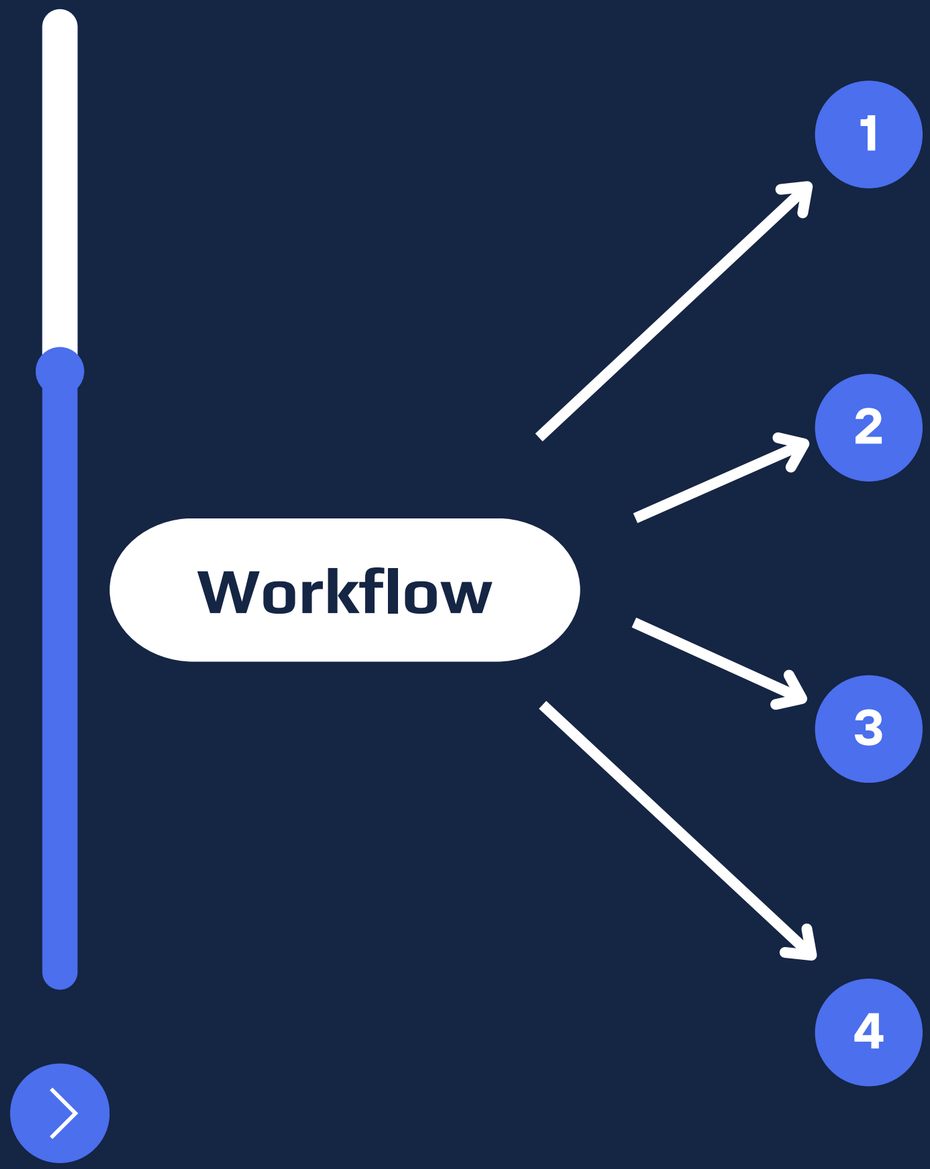
Tools

jupyter

PostgreSQL

http://books.toscrape.com/





Web Scraping with Beautiful Soup

Using tools like BeautifulSoup, the scraper reads the HTML structure and extracts specific elements (like titles, prices, star ratings, etc.)

Data Cleaning

Detecting and correcting (or removing) errors, inconsistencies, and inaccuracies in the extracted data to ensure it’s accurate, consistent, and ready for analysis.

Database Initialization and Data Storage

Creating db and table in postgreSQL for cleaned data storage. The cleaned data is saved into formats like CSV as well.

Data Visualization

The cleaned data can then be analyzed and applied into visualizations.

Web Scraping with BeautifulSoup

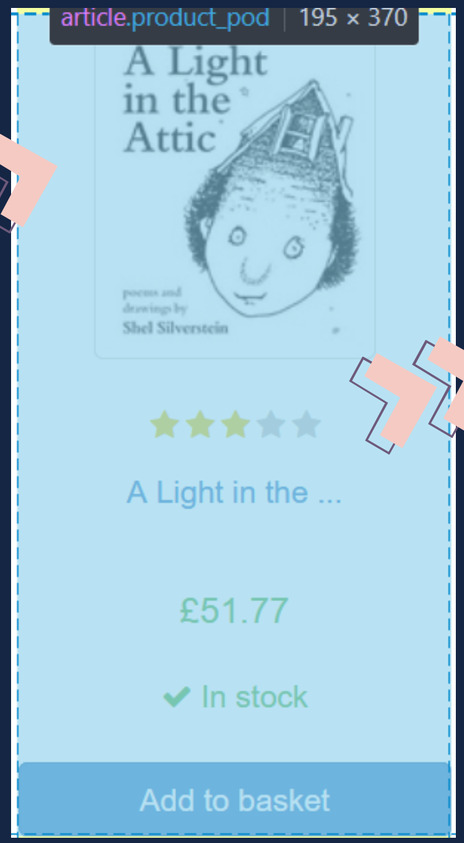
— Jupyter Notebook

Inspect home page

tag ‘ol’ : element that defines an ordered list of books in one page

```
<ol class="row">
  ::before
<li class="col-xs-6 col-sm-4 col-md-3 col-lg-3">
  <article class="product_pod">
    <div class="image_container">…</div>
    <p class="star-rating Three">…</p>
    <h3>
      <a href="catalogue/a-light-in-the-attic_1000/index.html" title="A Light in the Attic">
      A Light in the ...</a>
    </h3>
    <div class="product_price">…</div>
  </article>
</li>
```

tag ‘article’
class ‘product_pod’ :
element that defines for each book details



Data Extraction:

1. Book Title
2. Star Rating
3. Link product information

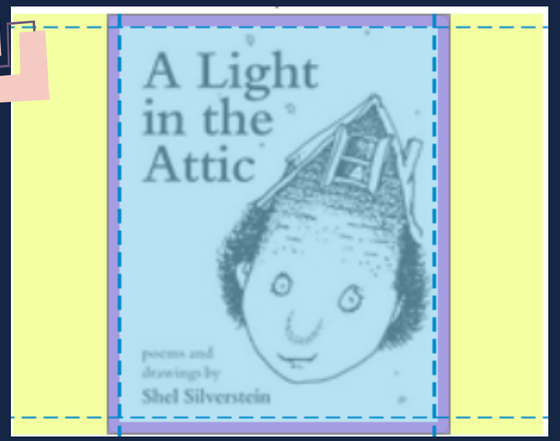
‘article’ element

Extracting Book Title

Inspect the ‘img’ tag and accessing value in attributes ‘alt’ using the .attrs attribute

Title

```
<div class="image_container">
  <a href="catalogue/a-light-in-the-attic_1000/index.html">
    <img class="thumbnail" src="media/cache/2c/da/2cdad67c44b002e7ead0cc35693c0e8b.jpg"
    alt="A Light in the Attic">
  </a>
</div>
```



‘article’ element

Star Rating

```
<p class="star-rating Three">
  <i class="icon-star">…</i>
  whitespace
  <i class="icon-star">…</i>
  whitespace
  <i class="icon-star">…</i>
  whitespace
  <i class="icon-star">…</i>
  whitespace
  <i class="icon-star">…</i>
</p>
```

Extracting Star Rating

Inspect the ‘p’ tag and get the star value with accessing name class in index 1.

‘article’ element

Extracting Link Prod Information

Inspect the ‘a’ tag and get the URL part with accessing 'href' value using the .attrs attribute

Extracting the data

How web scraping works?

1. Sending an HTTP Request
2. Receiving the Response
3. Parsing the Data
4. Storing the Data

Extracting Home Page Data

For extracting data in home page, the 'ol' tag is retrieved repeatedly from page 1 to page 50. On each page, the 'article' tag is parsed again to collect the book title and star ratings into books list and the links of product information links list.

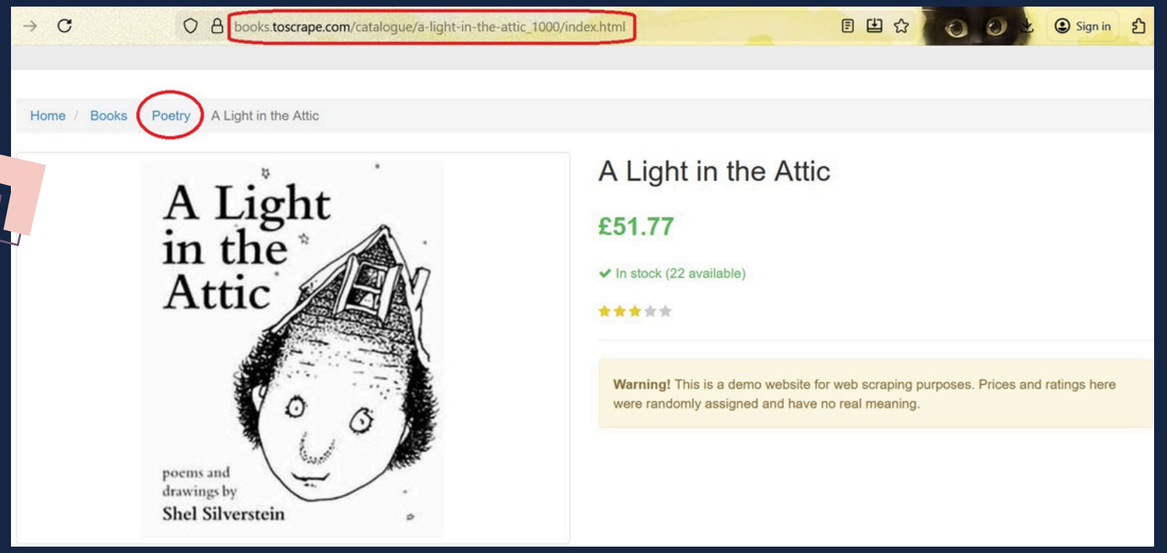
```
books = []
links = []

for i in range(1,51):   # Iteration page 1 to page 50
  url= f"http://books.toscrape.com/catalogue/page-{i}.html"
  response = requests.get(url)  # Use requests.get() to retrieve the HTML content of the target URL
  response = response.content

  # Create a BeautifulSoup object by passing the html_content and the parser type
  soup = BeautifulSoup(response,'html.parser')
  ol = soup.find('ol')  # Find the ol tag to inspect

  # Under the ol tag find all the article tag with class 'product_pod' into articles list
  articles = ol.findAll('article', class_='product_pod')

  for article in articles:  # Get the value of articles list one by one as article variable
    image = article.find('img')
    title = image.attrs['alt'] # Accessing attribute 'alt' to get title of the book
    #print(title)
    star = article.find('p')
    star = star['class'][1] # Get the value of name class in index 1 as star rating
    #print(star)
     detail = article.find('a') # Find a element in article element

     # In 'a' tag we can access attribute 'href' using .attrs to get the URL for each article
     link = 'https://books.toscrape.com/catalogue/' + detail.attrs['href']

     links.append(link)          # Collecting URL of detail book into links list
     books.append([title, star]) # Collecting title and star rating values into books list
```

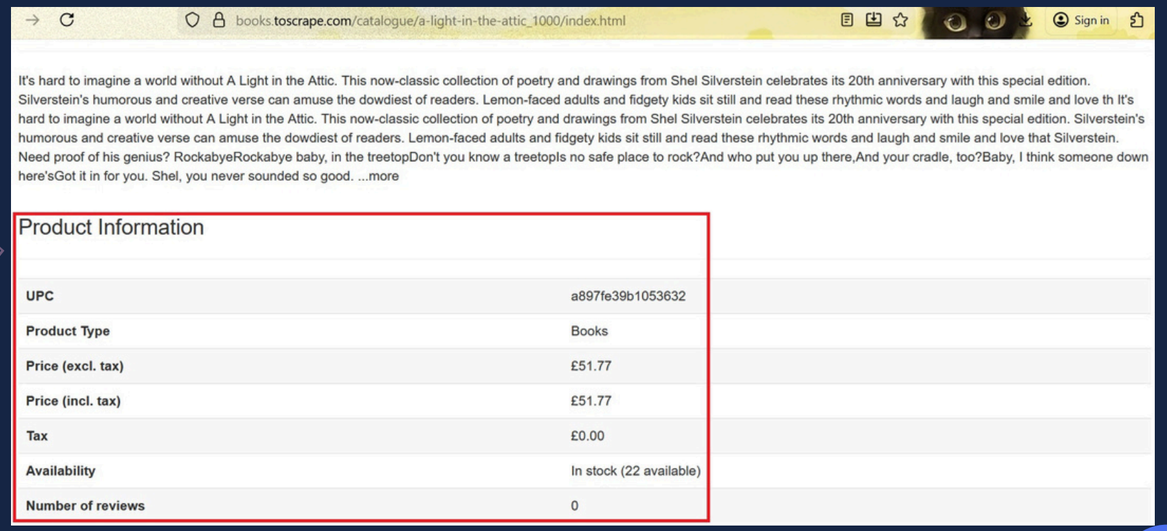
```
# Showing 5 links of details for each book
links[0:5]
```

```
['https://books.toscrape.com/catalogue/a-light-in-the-attic_1000/index.html',
 'https://books.toscrape.com/catalogue/tipping-the-velvet_999/index.html',
 'https://books.toscrape.com/catalogue/soumission_998/index.html',
 'https://books.toscrape.com/catalogue/sharp-objects_997/index.html',
 'https://books.toscrape.com/catalogue/sapiens-a-brief-history-of-humankind_996/index.html']
```

Inspect product information page

```
<ol class="row">
  ::before
  <li class="col-xs-6 col-sm-4 col-md-3 col-lg-3">
    <article class="product_pod">
      <div class="image_container">
        <a href="catalogue/a-light-in-the-attic_1000/index.html">...</a>
      </div>
      <p class="star-rating Three">...</p>
      <h3>...</h3>
      <div class="product_price">...</div>
    </article>
  </li>
```

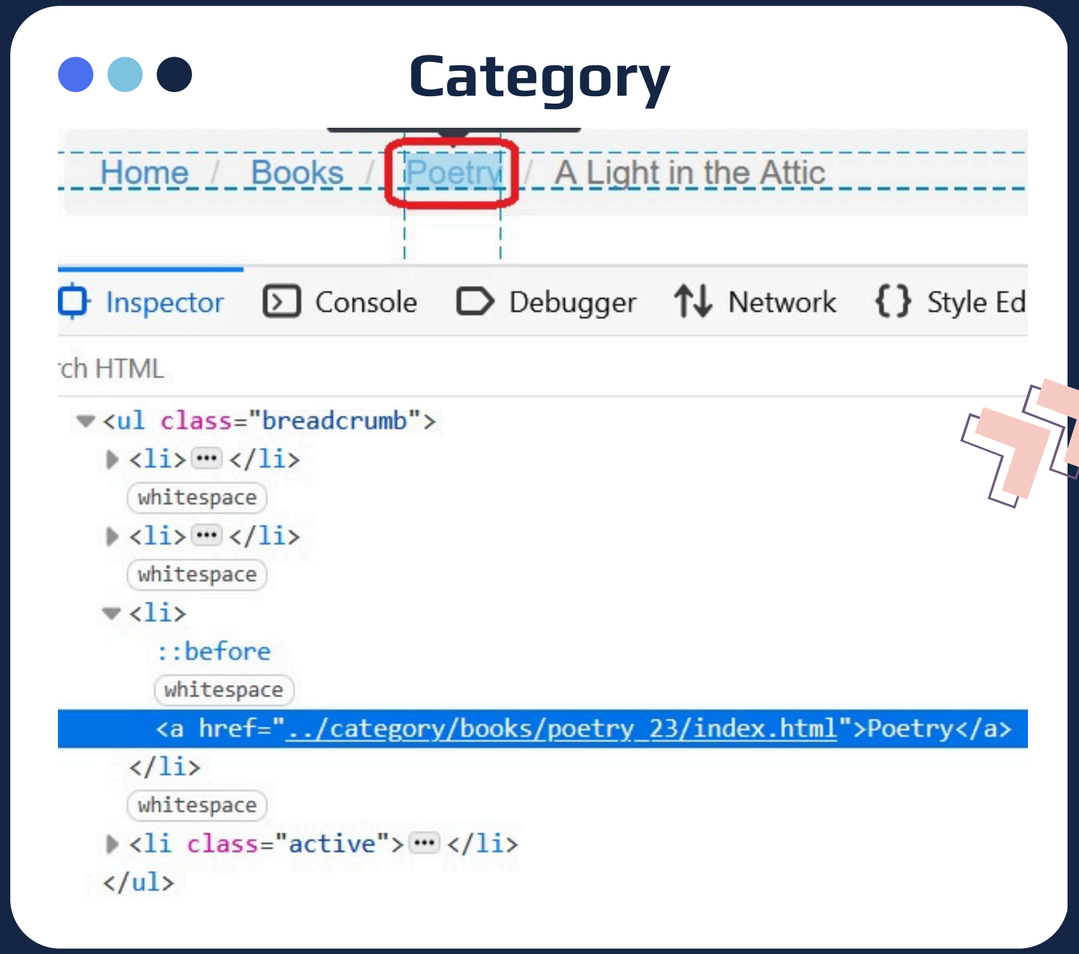




Data Extraction:

1. Book category
2. Product Information includes UPC, ProdType, Price_ExcTax, Price_IncTax, Tax, Availability, NumReviews

‘ul’ element



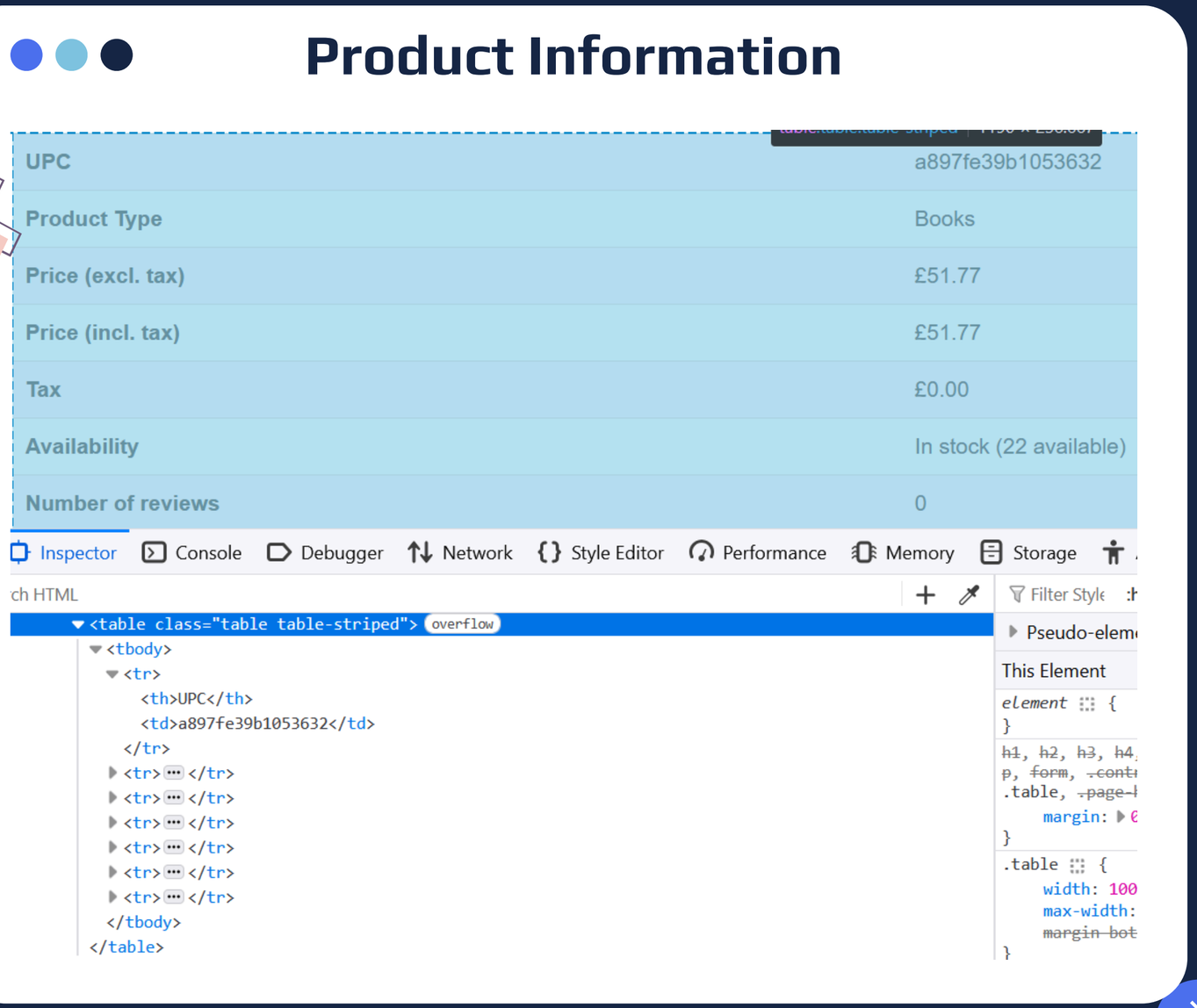
Extracting Book Category

Inspect the ul tag to find the set of li tags within it. Then, find the value of the second index, which represents the book category, for each extracted link from pages 1 to 50 on the home page.

‘table’ element

Extracting Product Information

Inspect the table tag named 'table-stripped' to get product information in a collection of tr tags. Each tr row contains the header cell name in the th tag and the product data in the td tag. To extract the td value, split each row into its index and td value.



Extracting the data

```
category = []
genre = []

val_upc = []
val_prod_type = []
val_price_exctax = []
val_price_inctax = []
val_tax = []
val_availability = []
val_num_reviews = []

for detail in links:    # Open the links one by one as detail variable
  res = requests.get(detail)
  res = res.content
  sp = BeautifulSoup(res,'html.parser')

  #Book Category
  ul = sp.find('ul', class_='breadcrumb') # Find ul tag with class 'breadcrumb'
  li = ul.find_all('a') # Under the ul tag find all the 'a' tag into li list

  for i, a in enumerate(li):    # Splitting into index and a value of each li
    if i==2:                    # Index 2 is a category of the book
      category.append(a.text)   # Collecting all values of index 2 into category list
  val_category = category[-1]   # Get the last result of all book categories
  genre.append(val_category)    # Insert the result into genre list
```

```
#Book Details
table = sp.find('table', {'class' : 'table-striped'}) # Inspect table tag with class 'table-stripped'
rows = table.find_all('tr')   # Find all tr tag under table tag as rows list

for index, tr in enumerate(rows): # Splitting into index and tr value of each rows
  if index==0:    # If index is 0 then this tr is called upc
    td = tr.find('td')              # Find td tag under upc tr as td variable
    val_upc.append(td.text.strip()) # Collect all td values for each link into val_upc list
  elif index==1:
    td1 = tr.find('td')
    val_prod_type.append(td1.text.strip())
  elif index==2:
    td2 = tr.find('td')
    val_price_exctax.append(td2.text.strip())
  elif index==3:
    td3 = tr.find('td')
    val_price_inctax.append(td3.text.strip())
  elif index==4:
    td4 = tr.find('td')
    val_tax.append(td4.text.strip())
  elif index==5:
    td5 = tr.find('td')
    val_availability.append(td5.text.strip())
  elif index==6:
    td6 = tr.find('td')
    val_num_reviews.append(td6.text.strip())
```

The Extracted Data

```
# Concatenate Books and Details dataframe
merged_df = pd.concat([df1, df2], axis=1)
merged_df.head()
```

	Title	StarRating	UPC	Category	ProdType	Price_ExcTax	Price_IncTax	Tax	Availability	NumReviews
0	A Light in the Attic	Three	a897fe39b1053632	Poetry	Books	£51.77	£51.77	£0.00	In stock (22 available)	0
1	Tipping the Velvet	One	90fa61229261140a	Historical Fiction	Books	£53.74	£53.74	£0.00	In stock (20 available)	0
2	Soumission	One	6957f44c3847a760	Fiction	Books	£50.10	£50.10	£0.00	In stock (20 available)	0
3	Sharp Objects	Four	e00eb4fd7b871a48	Mystery	Books	£47.82	£47.82	£0.00	In stock (20 available)	0
4	Sapiens: A Brief History of Humankind	Five	4165285e1663650f	History	Books	£54.23	£54.23	£0.00	In stock (20 available)	0

```
merged_df.shape
```

```
(1000, 10)
```

Data Cleaning

— Jupyter Notebook

Cleaning the extracted data

Setting UPC Column as an Index

Removing Currency Symbol in Price and Tax Columns

Extract a Substring from Availability column

Removing Special Characters and Punctuation

Identify Duplicates Based on a Subset of Columns

Checking Blank or Null Values

Custom or equal-width bins of Price and Stock

Label Encoding of Star Rating Values

Dropping Unnecessary Columns

Setting UPC Column as an Index

```
# Column to move
column_to_move = 'UPC'

# Pop the column (removes it and returns its Series)
col_series = merged_df.pop(column_to_move)

# Insert the column at index 0
merged_df.insert(0, column_to_move, col_series)
merged_df.head()
```

	UPC	Title	StarRating	Category	ProdType	Price_ExcTax	Price_IncTax	Tax	Availability	NumReviews
0	a897fe39b1053632	A Light in the Attic	Three	Poetry	Books	£51.77	£51.77	£0.00	In stock (22 available)	0
1	90fa61229261140a	Tipping the Velvet	One	Historical Fiction	Books	£53.74	£53.74	£0.00	In stock (20 available)	0
2	6957f44c3847a760	Soumission	One	Fiction	Books	£50.10	£50.10	£0.00	In stock (20 available)	0
3	e00eb4fd7b871a48	Sharp Objects	Four	Mystery	Books	£47.82	£47.82	£0.00	In stock (20 available)	0
4	4165285e1663650f	Sapiens: A Brief History of Humankind	Five	History	Books	£54.23	£54.23	£0.00	In stock (20 available)	0

Removing Currency Symbol in Price and Tax Columns

```
# Extracts characters from index 1
merged_df['Price_ExcTax'] = merged_df['Price_ExcTax'].str[1:]
merged_df['Price_IncTax'] = merged_df['Price_IncTax'].str[1:]
merged_df['Tax'] = merged_df['Tax'].str[1:]
merged_df.head()
```

Price_ExcTax	Price_IncTax	Tax
51.77	51.77	0.00
53.74	53.74	0.00
50.10	50.10	0.00
47.82	47.82	0.00
54.23	54.23	0.00

Extract a Substring from Availability column

```
# Extracts characters from index 0 up to (but not including) 8 into Available Column
merged_df['Available'] = merged_df['Availability'].str.slice(0, 8)

# Extracts characters from index 10 up to (but not including) 12 into Stock Column
merged_df['Stock'] = merged_df['Availability'].str.slice(10, 12)

# Drop Availability column
merged_df.drop('Availability', axis=1, inplace=True)
merged_df.head()
```

Available	Stock
In stock	22
In stock	20
In stock	20
In stock	20
In stock	20

Removing Special Characters and Punctuation

```python
import re

def remove_special_chars(text):
    # Remove anything that is not a letter, number, or whitespace
    text = re.sub(r'[^A-Za-z0-9\s]', '', text)
    return text
```

```python
merged_df['Title'] = merged_df['Title'].apply(remove_special_chars)
```

```python
merged_df['Title'].head()
```

```
0                    A Light in the Attic
1                      Tipping the Velvet
2                              Soumission
3                           Sharp Objects
4    Sapiens A Brief History of Humankind
Name: Title, dtype: object
```

Identify Duplicates Based on a Subset of Columns

```
duplicate_title = merged_df.duplicated(subset=['Title'], keep=False)
duplicate_title_rows = merged_df[duplicate_title]
print("\nDuplicate rows based on 'Title':")
duplicate_title_rows
```

```
Duplicate rows based on 'Title':
```

	UPC	Title	StarRating	Category	ProdType	Price_ExcTax	Price_IncTax	Tax	NumReviews	Available	Stock
236	1528279aec1f3dce	The StarTouched Queen	Five	Fantasy	Books	46.02	46.02	0.00	0	In stock	14
358	4a7a25be293ad678	The StarTouched Queen	Five	Fantasy	Books	32.30	32.30	0.00	0	In stock	12

```
# Drop duplicate rows based on 'Title' and keep the last occurrence
df_cleaned = merged_df.drop_duplicates(subset=['Title'], keep='last')
```

```
# The total number of rows in a df after dropping duplicates
df_cleaned.shape[0]
```

```
999
```

Checking Blank or Null Values

```
# Change data type of columns
df_cleaned['Price_ExcTax'] = df_cleaned['Price_ExcTax'].astype(float)
df_cleaned['Price_IncTax'] = df_cleaned['Price_IncTax'].astype(float)
df_cleaned['Tax'] = df_cleaned['Tax'].astype(float)
df_cleaned['NumReviews'] = df_cleaned['NumReviews'].astype(int)
df_cleaned['Stock'] = df_cleaned['Stock'].astype(int)
```

```
blank = df_cleaned[(df_cleaned['UPC'] == '') | (df_cleaned['Title'] == '')\
                   | (df_cleaned['StarRating'] == '') | (df_cleaned['Category'] == '')\
                   | (df_cleaned['ProdType'] == '') | (df_cleaned['Price_ExcTax'] == 0.0)\
                    | (df_cleaned['Price_IncTax'] == 0.0) | (df_cleaned['Available'] == '')]
blank.shape[0]
```

```
0
```

```
df_cleaned.isnull().sum()
```

```
UPC             0
Title           0
StarRating      0
Category        0
ProdType        0
Price_ExcTax    0
Price_IncTax    0
Tax             0
NumReviews      0
Available       0
Stock           0
dtype: int64
```

Custom or equal-width bins of Price and Stock

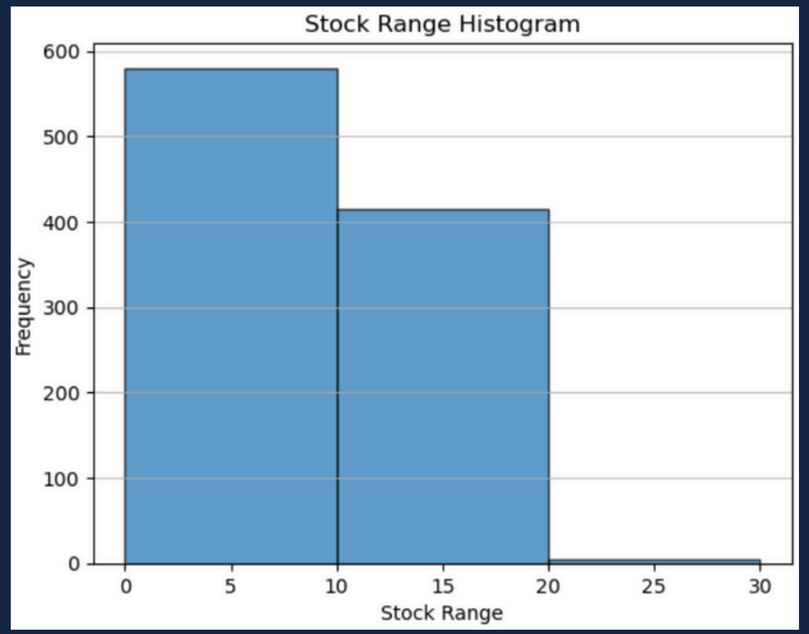
```
bins = [0, 25, 50, 75]
labels = ['Low', 'Medium', 'High']
df_cleaned['PriceCategory'] = pd.cut(df_cleaned['Price_IncTax'], bins=bins, labels=labels, right=True)
df_cleaned.sample(5)
```

PriceCategory
Medium
Low
Medium
High
High

Custom or equal-width bins of Price and Stock

```
bins = [0, 10, 20, 30]
labels = ['0-10', '11-20', '21-30']
df_cleaned['Stock_Rg'] = pd.cut(df_cleaned['Stock'], bins=bins, labels=labels, right=True)
df_cleaned.sample(5)
```

Stock_Rg
0-10
0-10
0-10
0-10
0-10



Label Encoding of Star Rating Values

```
star_map = {'One': '1', 'Two': '2', 'Three': '3', 'Four': '4', 'Five': '5'}
df_cleaned['StarRating'].replace(star_map, inplace=True)
df_cleaned['StarRating'].value_counts().sort_index()
```

```
StarRating
1    226
2    196
3    203
4    179
5    195
Name: count, dtype: int64
```

Dropping Unnecessary Columns

```
# All books are not sold out yet
soldout = df_cleaned[~df_cleaned['Available'].str.contains('in stock', case=False)]
soldout.shape[0]
```

```
0
```

```
# The values of price_exctax and price_inctax are same
diff_price = df_cleaned[df_cleaned['Price_ExcTax'] != df_cleaned['Price_IncTax']]
diff_price.shape[0]
```

```
0
```

```
# All filled only 1 value
df_cleaned['Tax'].value_counts()
```

```
Tax
0.0    999
Name: count, dtype: int64
```

```
# All filled only 1 value
df_cleaned['NumReviews'].value_counts()
```

```
NumReviews
0    999
Name: count, dtype: int64
```

The Cleaned Data

```
# Removing unnecessary columns
df_cleaned.drop(columns=['Price_ExcTax','Tax','NumReviews'], axis=1, inplace=True)
#df_cleaned.to_string(justify='left')
df_cleaned.head()
```

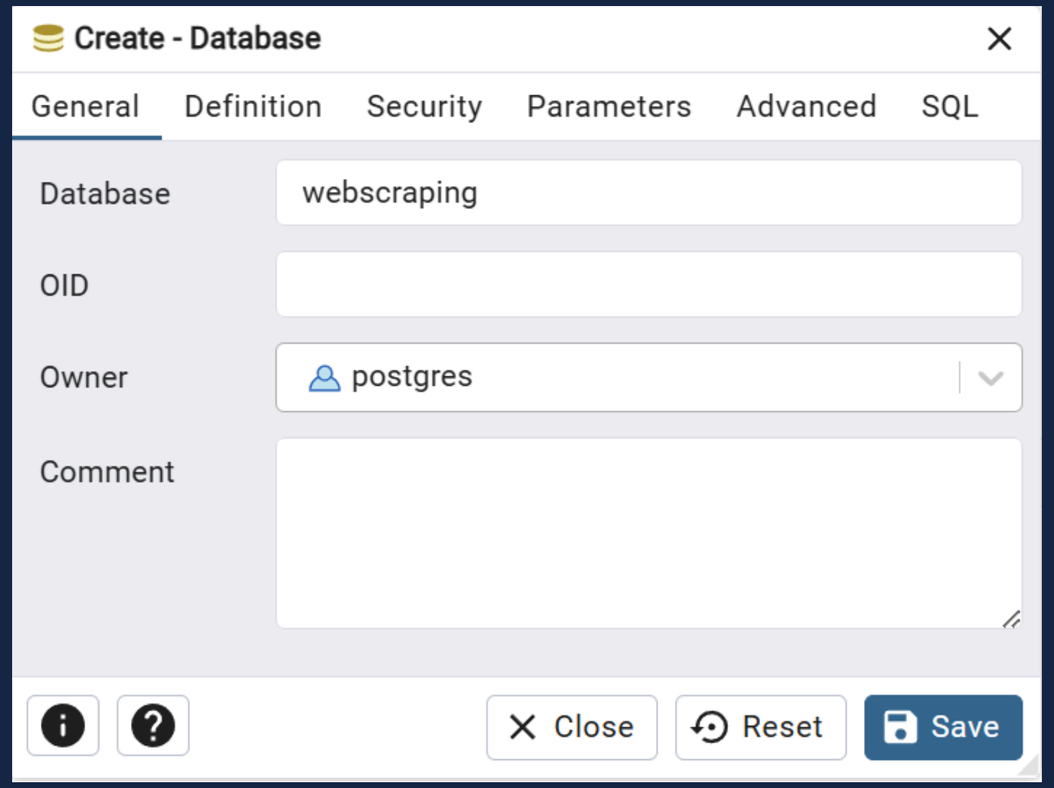
	UPC	Title	StarRating	Category	ProdType	Price_IncTax	Available	Stock	PriceCategory	Stock_Rg
0	a897fe39b1053632	A Light in the Attic	3	Poetry	Books	51.77	In stock	22	High	21-30
1	90fa61229261140a	Tipping the Velvet	1	Historical Fiction	Books	53.74	In stock	20	High	11-20
2	6957f44c3847a760	Soumission	1	Fiction	Books	50.10	In stock	20	High	11-20
3	e00eb4fd7b871a48	Sharp Objects	4	Mystery	Books	47.82	In stock	20	Medium	11-20
4	4165285e1663650f	Sapiens A Brief History of Humankind	5	History	Books	54.23	In stock	20	High	11-20

Database Initialization and Data Storage

— PostgreSQL

To create a new database in pgAdmin 4, follow these steps:

- Open pgAdmin 4 and connect to your PostgreSQL server. You may need to enter your server password if prompted.
- Navigate to Databases: In the pgAdmin 4 browser tree (left panel), expand your server connection and locate the "Databases" node.
- Initiate Database Creation: Right-click on the "Databases" node, then select "Create" and subsequently choose "Database...".
- Configure Database Details: A "Create - Database" dialog will appear.



Exporting Cleaned Data into postgreSQL

```
from sqlalchemy import create_engine
import pandas as pd

# Replace with your PostgreSQL connection details
db_user = "postgres"
db_password = [redacted]
db_host = "localhost"  # or your host IP/name
db_port = "5432"
db_name = "webscraping"

# Create the connection string
engine_str = f"postgresql+psycopg2://{db_user}:{db_password}@{db_host}:{db_port}/{db_name}"
engine = create_engine(engine_str)
#conn = engine.connect()
```

```
table_name = "books2"  # Name of the table in PostgreSQL

df_cleaned.to_sql(
  table_name,
  engine,
  if_exists = 'replace',  # Options: 'fail', 'replace', 'append'
  index = False           # Set to True if you want to export the DataFrame index as a column
)

print(f"DataFrame successfully exported to '{table_name}' in PostgreSQL.")
```

```
DataFrame successfully exported to 'books2' in PostgreSQL.
```

BOOKS2 TABLE

```
2   select * from books2 LIMIT 10
```

Data Output | Messages | Notifications

Showing rows: 1 to 10 | Page No: 1 of 1

	UPC text	Title text	StarRating text	Category text	ProdType text	Price_IncTax double precision	Available text	Stock integer	PriceCategory text	Stock_Rg text
1	a897fe39b1053632	A Light in the At...	3	Poetry	Books	51.77	In stock	22	High	21-30
2	90fa61229261140a	Tipping the Velv...	1	Historical Fiction	Books	53.74	In stock	20	High	11-20
3	6957f44c3847a760	Soumission	1	Fiction	Books	50.1	In stock	20	High	11-20
4	e00eb4fd7b871a48	Sharp Objects	4	Mystery	Books	47.82	In stock	20	Medium	11-20
5	4165285e1663650f	Sapiens A Brief ...	5	History	Books	54.23	In stock	20	High	11-20
6	f77dbf2323deb740	The Requiem Red	1	Young Adult	Books	22.65	In stock	19	Low	11-20
7	2597b5a345f45e1b	The Dirty Little ...	4	Business	Books	33.34	In stock	19	Medium	11-20
8	e72a5dfc7e9267b2	The Coming Wo...	3	Default	Books	17.93	In stock	19	Low	11-20
9	e10e1e165dc8be4a	The Boys in the ...	4	Default	Books	22.6	In stock	19	Low	11-20
10	1dfe412b8ac00530	The Black Maria	1	Poetry	Books	52.15	In stock	19	High	11-20

Books2 Schema

```
select
column_name, data_type, character_maximum_length
from INFORMATION_SCHEMA.COLUMNS
where table_name ='books2';
```

Output | Messages | Notifications

column_name name	data_type character varying	character_maximum_length integer
UPC	text	[null]
Title	text	[null]
StarRating	text	[null]
Category	text	[null]
ProdType	text	[null]
Price_IncTax	double precision	[null]
Available	text	[null]
Stock	integer	[null]
PriceCategory	text	[null]
Stock_Rg	text	[null]

Querying from Postgres DB

```
import psycopg2
conn = psycopg2.connect(host="localhost",dbname="webscraping",user="postgres",password="ndahadmin",port=5432)
```

```
cur = conn.cursor()
```

```
query = """select * from books2 LIMIT 10"""

cur.execute(query)

data = cur.fetchall()

select_data = pd.DataFrame(data, columns=['UPC','Title','StarRating','Category','ProdType','Price_IncTax','Available','Stock','PriceCategory','Stock_Rg'])
select_data
```

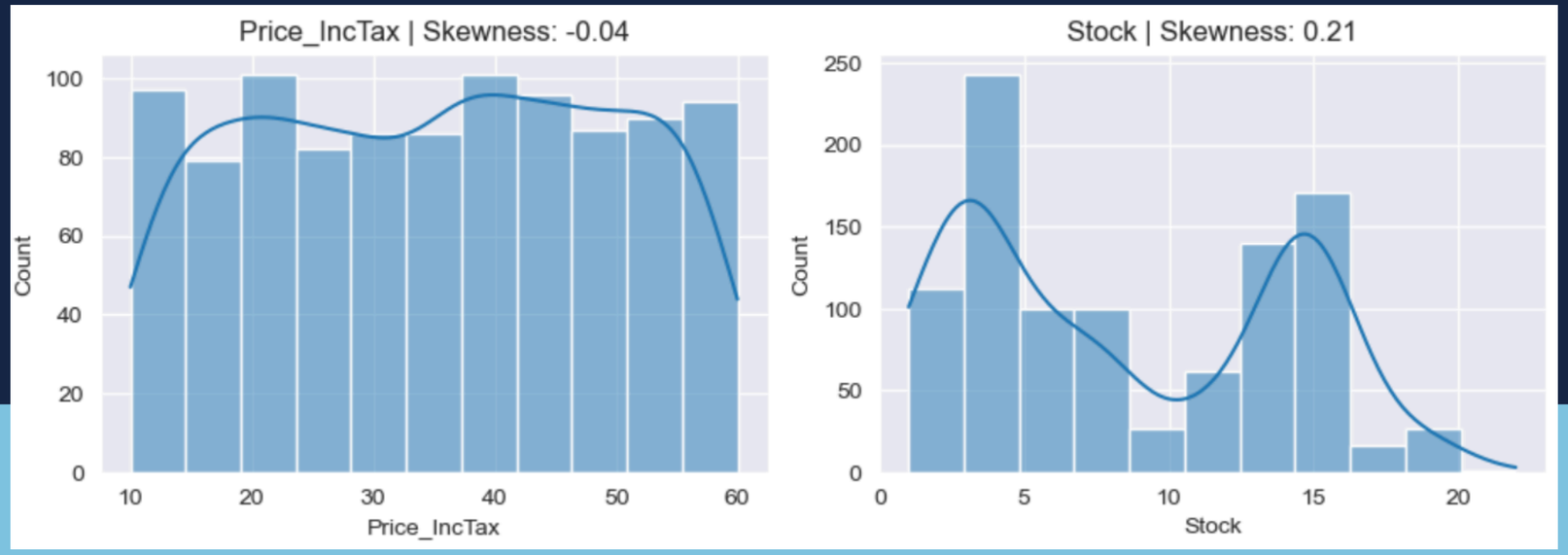
Querying from Postgres DB

	UPC	Title	StarRating	Category	ProdType	Price_IncTax	Available	Stock	PriceCategory	Stock_Rg
0	a897fe39b1053632	A Light in the Attic	3	Poetry	Books	51.77	In stock	22	High	21-30
1	90fa61229261140a	Tipping the Velvet	1	Historical Fiction	Books	53.74	In stock	20	High	11-20
2	6957f44c3847a760	Soumission	1	Fiction	Books	50.10	In stock	20	High	11-20
3	e00eb4fd7b871a48	Sharp Objects	4	Mystery	Books	47.82	In stock	20	Medium	11-20
4	4165285e1663650f	Sapiens A Brief History of Humankind	5	History	Books	54.23	In stock	20	High	11-20
5	f77dbf2323deb740	The Requiem Red	1	Young Adult	Books	22.65	In stock	19	Low	11-20
6	2597b5a345f45e1b	The Dirty Little Secrets of Getting Your Dream...	4	Business	Books	33.34	In stock	19	Medium	11-20
7	e72a5dfc7e9267b2	The Coming Woman A Novel Based on the Life of ...	3	Default	Books	17.93	In stock	19	Low	11-20
8	e10e1e165dc8be4a	The Boys in the Boat Nine Americans and Their ...	4	Default	Books	22.60	In stock	19	Low	11-20
9	1dfe412b8ac00530	The Black Maria	1	Poetry	Books	52.15	In stock	19	High	11-20

Data Visualization

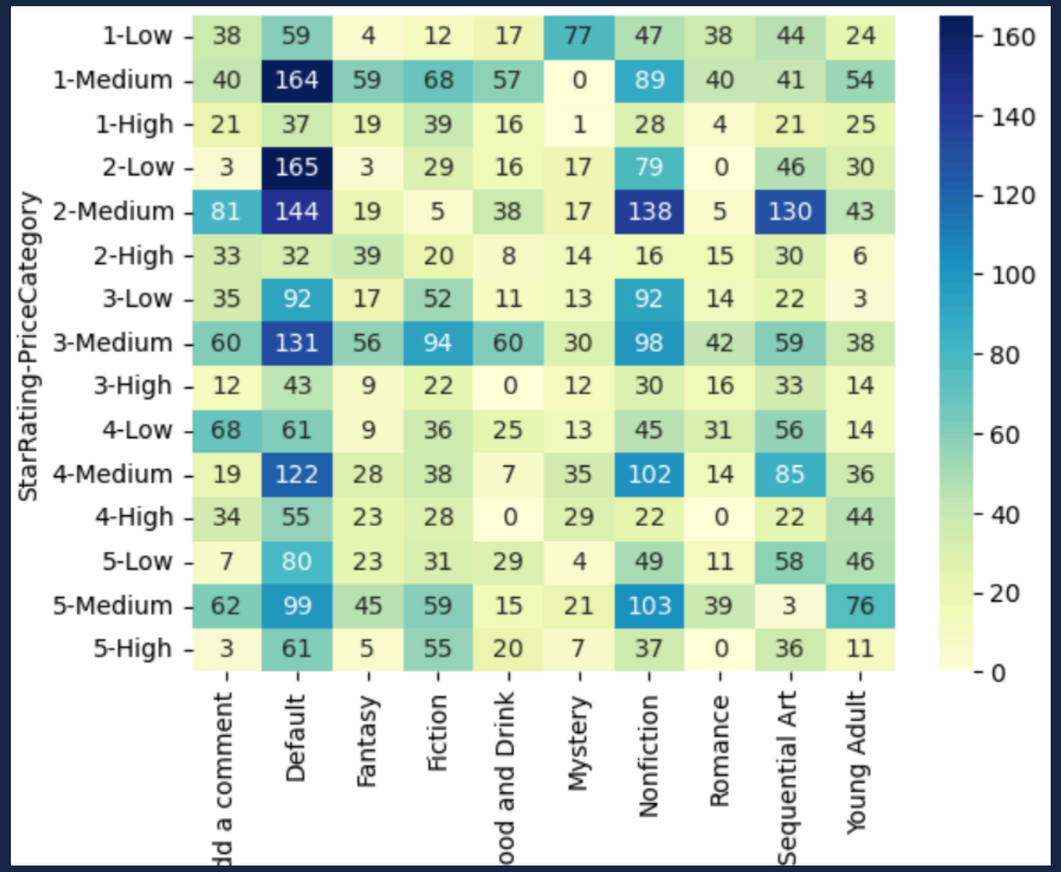
— Jupyter Notebook

Distribution of Numeric Column



- Negative skewness, also known as left-skewed, means the distribution's tail is longer on the left side, indicating that the majority of the data points are concentrated on the right side of the mean.
- Positive skewness means the data has a long right tail and most data points are concentrated on the left side. This indicates that there are more frequent, lower values in the dataset, with a few extreme high values that pull the mean to the right of the median.

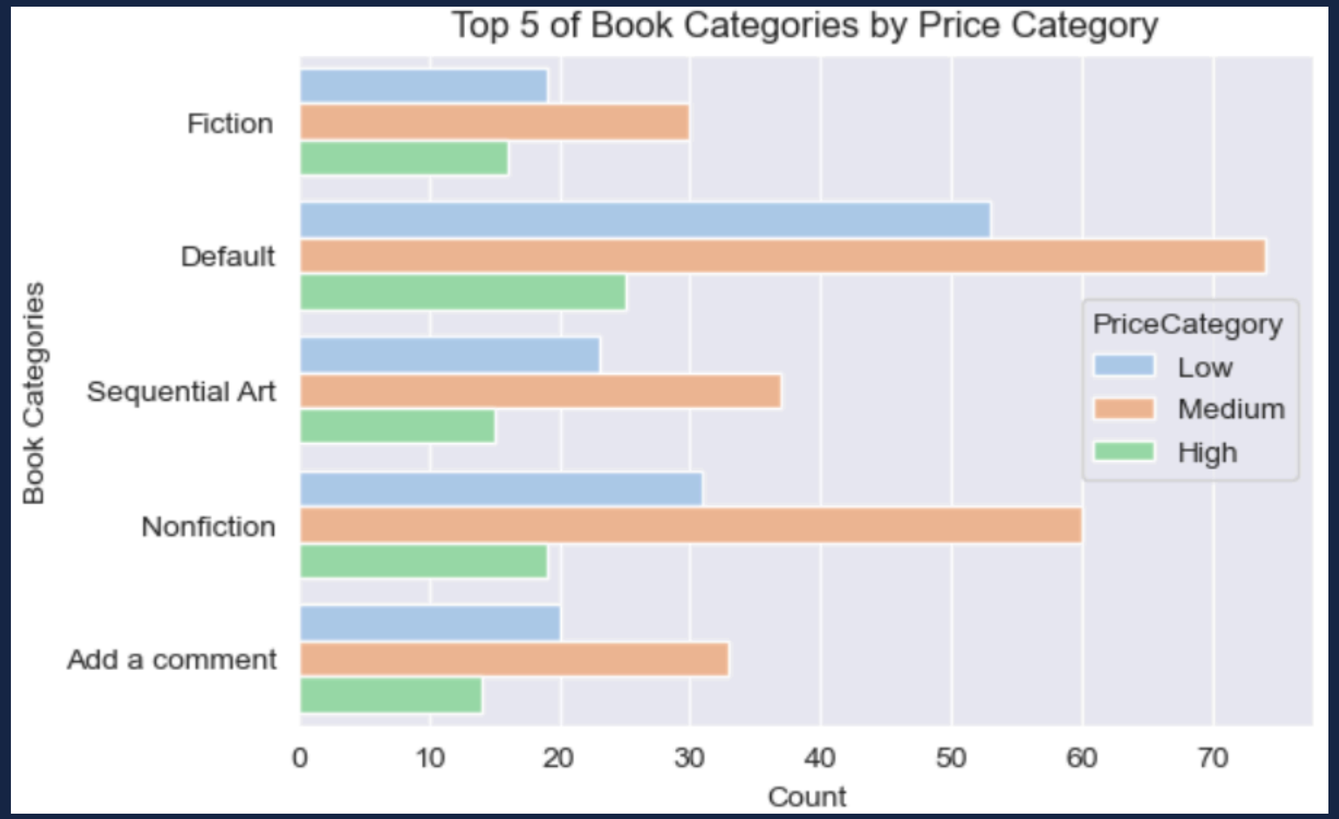
View Top 10 Book Categories that Have Large Stocks



Analysis

All books in the Default and Nonfiction categories are less popular with buyers, as can be seen from the large available stock, including the Sequential Art category in 2-star rating. Even though the prices of books are Low and Medium levels, the available stock is still very large.

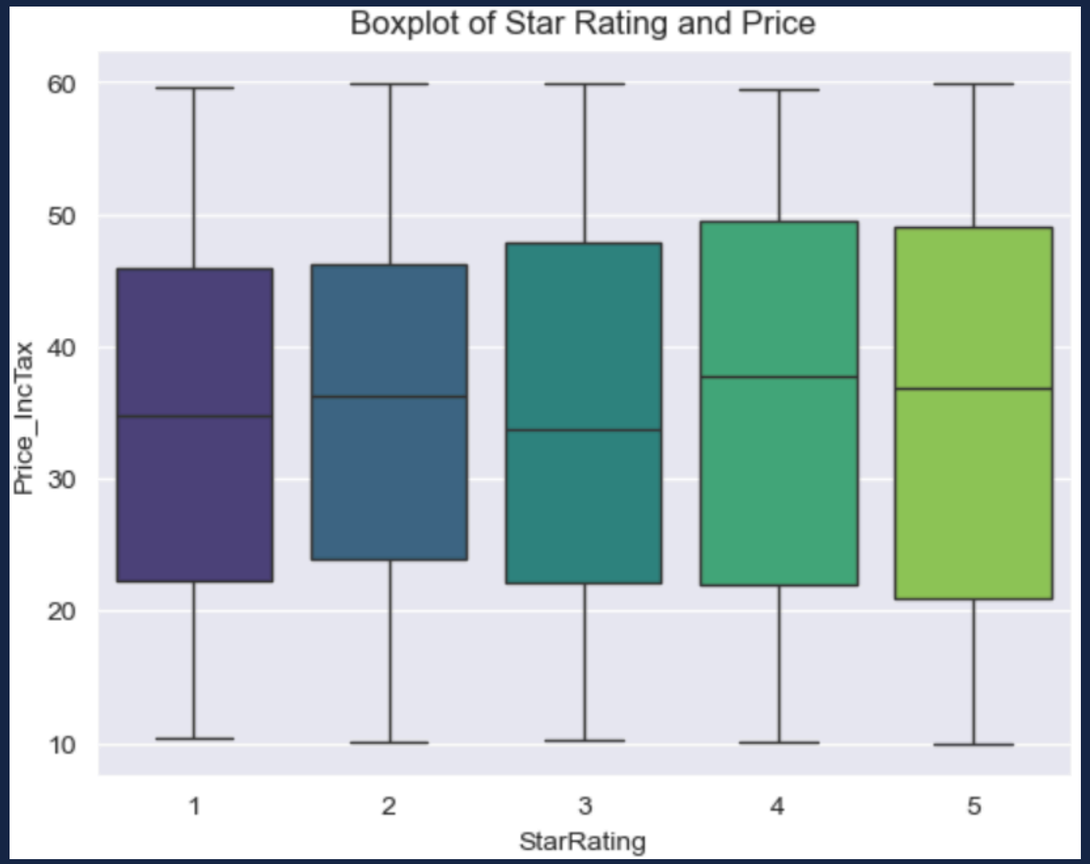
Top 5 of Book Categories and Its Price Category



Analysis

All books in the top 5 book categories are dominated in Medium level of price.

Correlation Between Price and Star Rating



Analysis

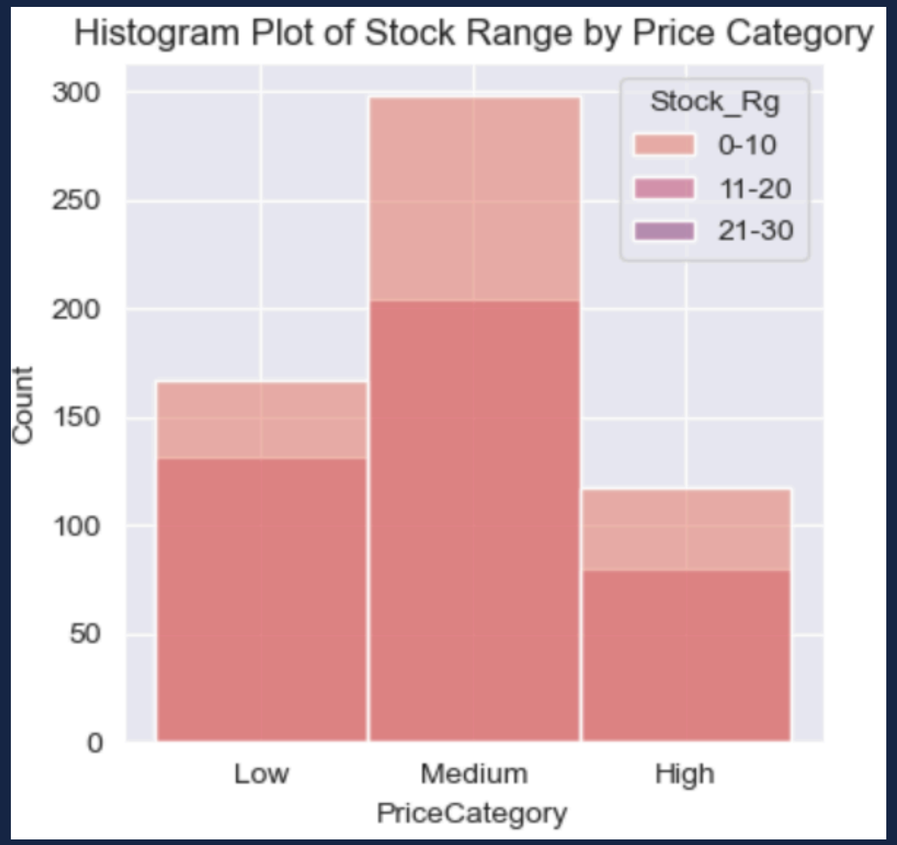
Quick visual comparisons of their distributions, central tendencies (medians), and variability (spread) of Star Rating and Price_IncTax.
5 star rating books are worth to buy, its have a wider price range and the mid price is around 37 dollars.

Count Plot of Price Category by Star Rating

Analysis

In general, the higher star rating, the more expensive books there are, the lower the star rating, the more cheap books there are. The number of 5 star ratings books priced at low and medium levels is greater than the number of books with 4 star ratings.

Histogram Plot of Stock Range by Price Category



Analysis

* The largest percentage of stock in range 11-20 occurs the books at low price. And the second highest percentage occurs the books at high price.
* Books with low-level prices make people hesitate to buy them because they assume the book is not good enough to read.
* Books with high-level prices make people hesitate to buy them because many people consider the book star ratings too, so there are still a lot of stocks of them.

THANK YOU !!

endahen12@gmail.com

www.linkedin.com/in/endahrakhmawati

ibimbing