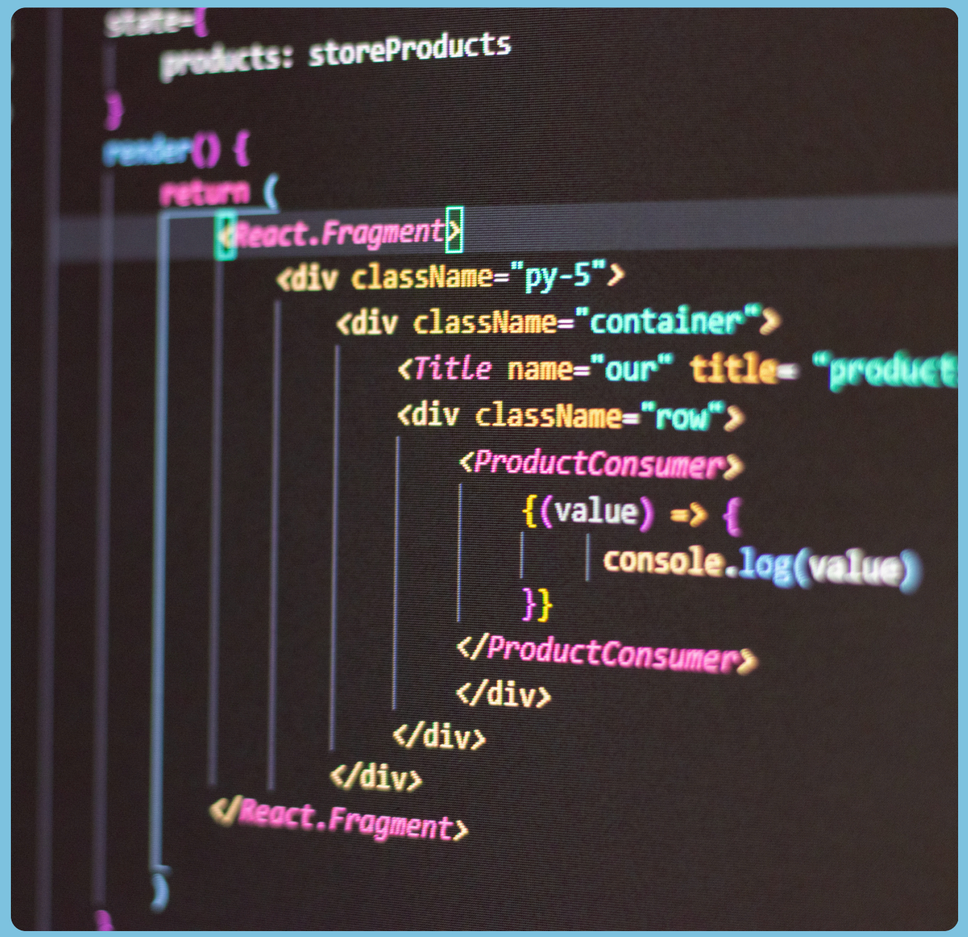
Web Scraping with Beautiful Soup

DIGITAL SKILL FAIR 44.0 - Faculty of Data : Data Engineering

Participant : Endah Rakhmawati

Project Overview



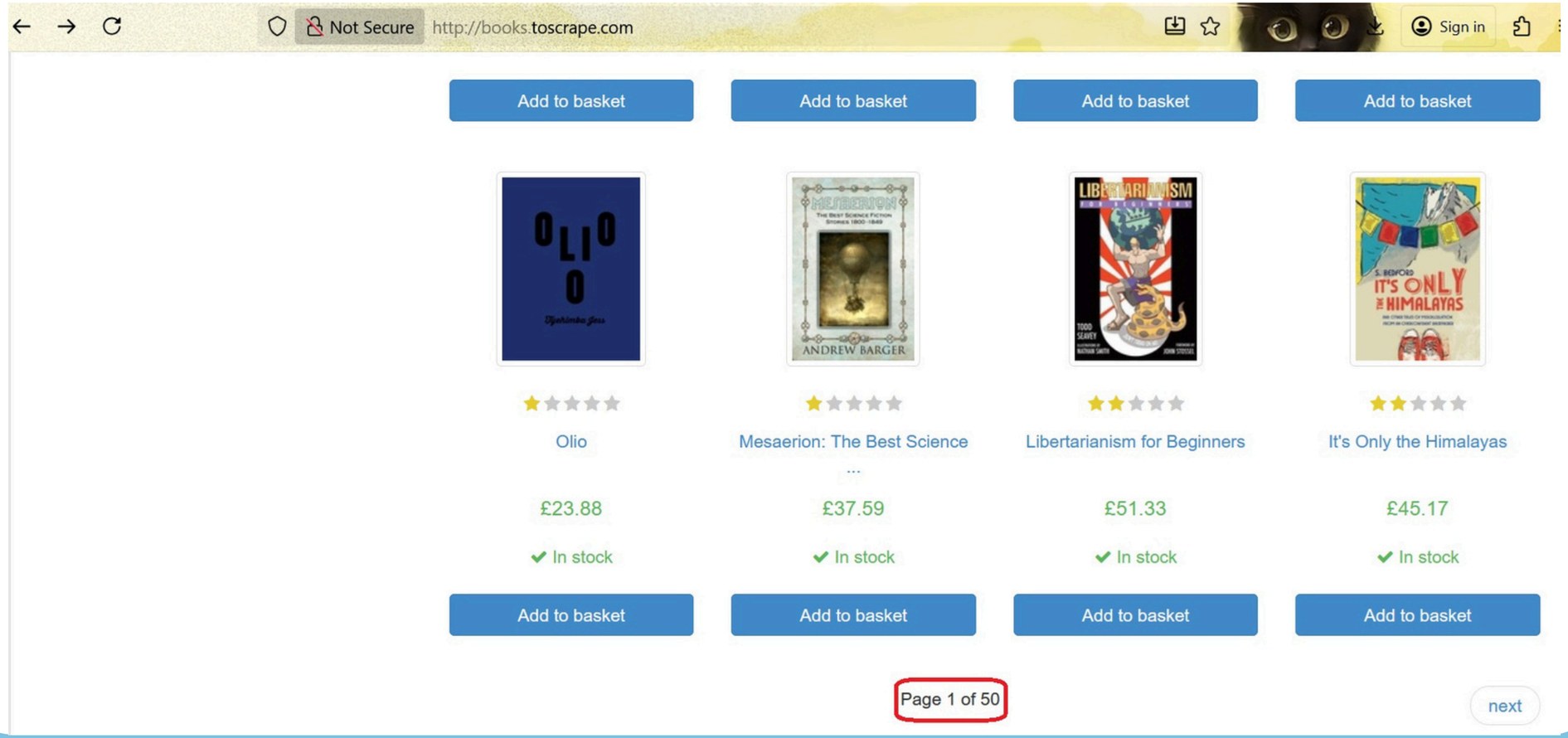
Web scraping is the process of automatically extracting data from websites.
This project parsing the data from bookstore website using Beautiful Soup tool. The scraper reads the HTML structure and extracts specific elements (like titles, prices, star ratings, etc.). It allows you to collect large amounts 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).

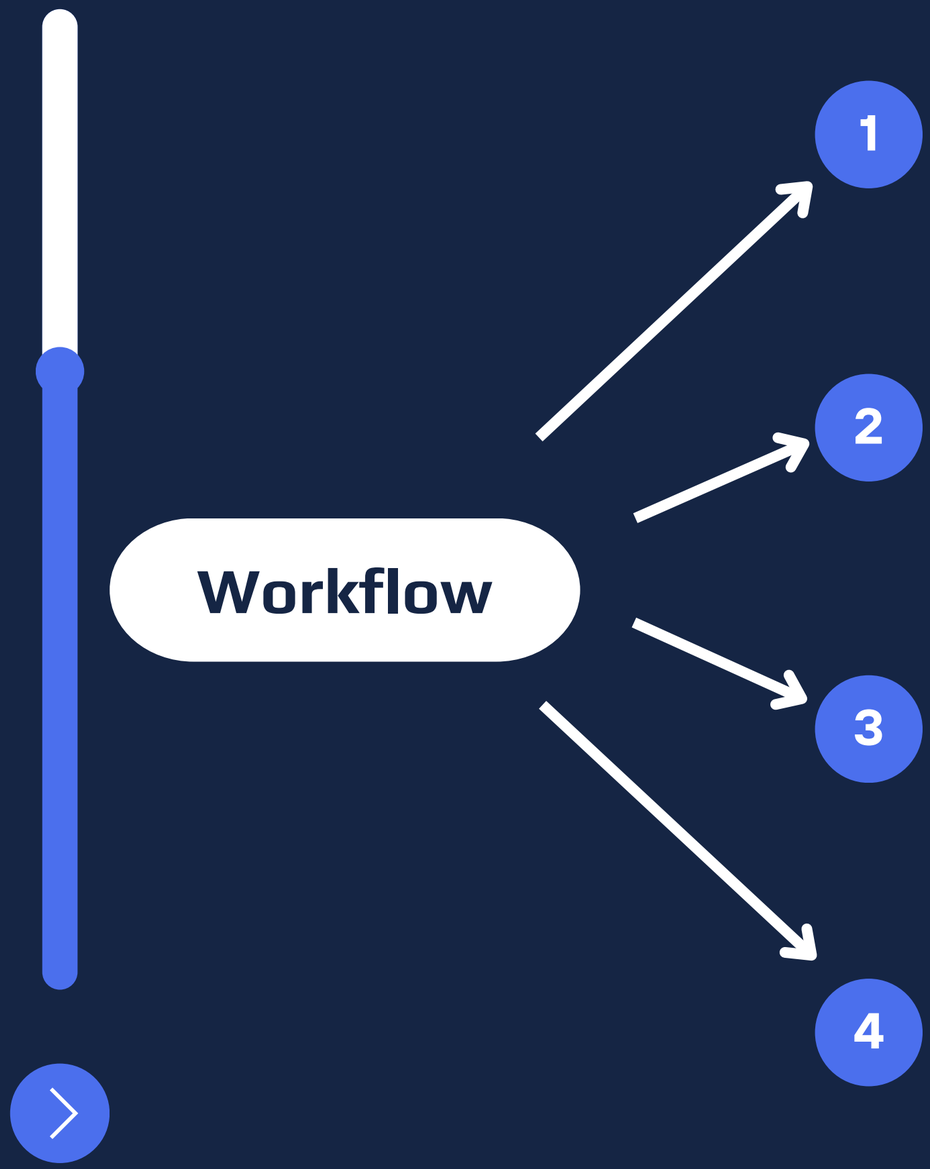
Tools

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

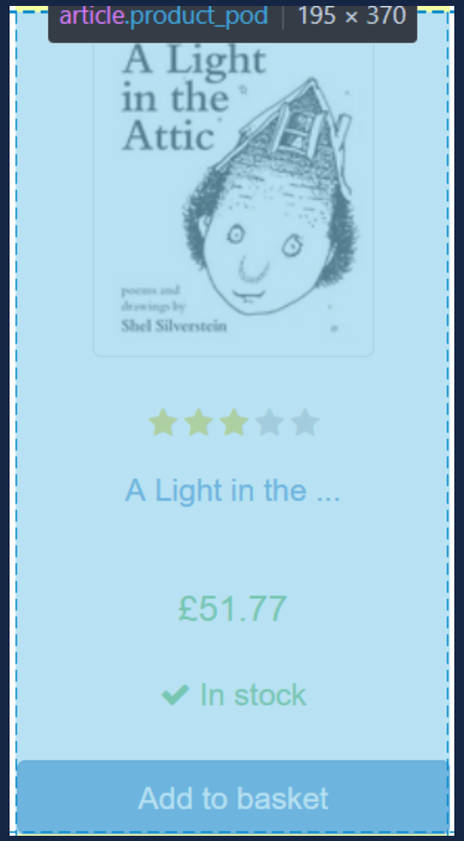
— Jupyter Notebook

Inspect the page

element ‘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>
```

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



Extracting the data

How web scraping works?

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

Explanations:

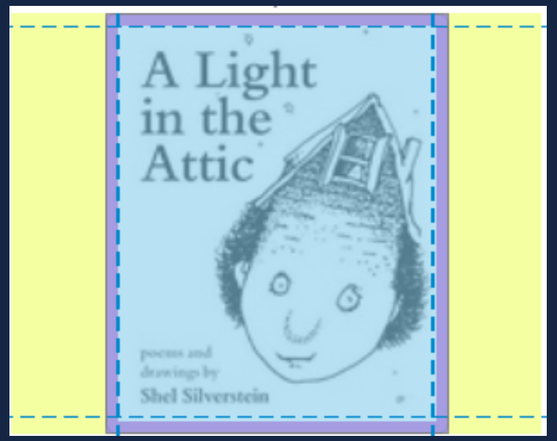
To get data on the 'article' element, the 'ol' element is retrieved repeatedly from page 1 to page 50. On each page, the 'article' element is parsed again to get the required book details into books array.

```
books=[]
for i in range(1,51):
  url= f"http://books.toscrape.com/catalogue/page-{i}.html"
  response = requests.get(url)
  response = response.content
  soup = BeautifulSoup(response,'html.parser')
  ol = soup.find('ol')
  articles= ol.findAll('article',class_='product_pod')

  for article in articles:
    image = article.find('img')
    title = image.attrs['alt']
    #print(title)
    star = article.find('p')
    star = star['class'][1]
    #print(star)
    price = article.find('p',class_='price_color').text
    price = float(price[1:])
    #print(price)
    stock = article.find('p',class_='instock availability').text.strip()
    #stock = stock.replace('\n', '')
    #print(stock)
    books.append([title,price,star,stock])
```

‘article’ element

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>
```



Find the ‘img’ element and accessing attributes ‘alt’ with .attrs

‘article’ element

Star Rating

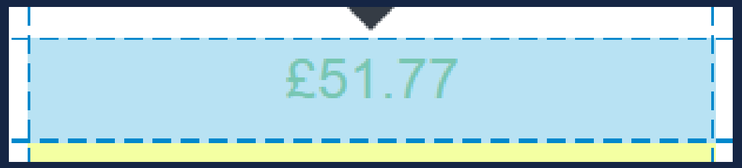
```
<p class="star-rating Three">
  <i class="icon-star">…</i>
  whitespace
  <i class="icon-star">…</i>
  whitespace
  <i class="icon-star">…</i>
  whitespace
  <i class="icon-star">…</i>
  whitespace
  <i class="icon-star">…</i>
</p>
```

Find the ‘p’ element and get the star value with accessing class index 1

‘article’ element

Price

```
<div class="product_price">
    <p class="price_color">£51.77</p>
  ▶ <p class="instock availability"> ••• </p>
  ▶ <form> ••• </form>
  </div>
```



Find the ‘p’ element class ‘price_color’ and set the price with float type without currency symbol

‘article’ element

Stock Availability

```
<p class="instock availability">
  <i class="icon-ok"> ••• </i>
  In stock
</p>
```

Find the ‘p’ element class ‘instock_availability’ and get the value with method strip() to remove leading and trailing characters from a string

The Extracted Data

```
df = pd.DataFrame(books, columns=['Title', 'Price', 'StarRating', 'Available'])
```

```
df.head()
```

	Title	Price	StarRating	Available
0	A Light in the Attic	51.77	Three	In stock
1	Tipping the Velvet	53.74	One	In stock
2	Soumission	50.10	One	In stock
3	Sharp Objects	47.82	Four	In stock
4	Sapiens: A Brief History of Humankind	54.23	Five	In stock

Data Cleaning

— Jupyter Notebook

Cleaning the extracted data

Add ID Column and Setting as an Index

Removing Special Characters and Punctuation

Identify Duplicates Based on a Subset of Columns

Custom or equal-width bins of Price

Checking Blank or Null Values

Add ID Column and Setting as an Index

```python
# Add a book ID column with a range of numbers
df['Book_ID'] = range(len(df))
df.set_index('Book_ID', inplace=True)
df.head()
```

Book_ID	Title	Price	StarRating	Available
0	A Light in the Attic	51.77	Three	In stock
1	Tipping the Velvet	53.74	One	In stock
2	Soumission	50.10	One	In stock
3	Sharp Objects	47.82	Four	In stock
4	Sapiens: A Brief History of Humankind	54.23	Five	In stock

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
df['Title'] = df['Title'].apply(remove_special_chars)
```

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

```
Book_ID
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

```
# The total number of rows in a df
df.shape[0]
```

```
1000
```

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

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

Book_ID	Title	Price	StarRating	Available
236	The StarTouched Queen	46.02	Five	In stock
358	The StarTouched Queen	32.30	Five	In stock

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

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

```
999
```

Custom or equal-width bins of Price

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

df_cleaned.head()
```

Book_ID	Title	Price	StarRating	Available	PriceCategory
0	A Light in the Attic	51.77	Three	In stock	High
1	Tipping the Velvet	53.74	One	In stock	High
2	Soumission	50.10	One	In stock	High
3	Sharp Objects	47.82	Four	In stock	Medium
4	Sapiens A Brief History of Humankind	54.23	Five	In stock	High

Checking Blank or Null Values

```
blank = df_cleaned[(df_cleaned['Title'] == '') | (df_cleaned['Price'] == 0.0)\
                   | (df_cleaned['StarRating'] == '') | (df_cleaned['Available'] == '')\
                   | (df_cleaned['PriceCategory'] == '')]

blank
```

Book_ID	Title	Price	StarRating	Available	PriceCategory

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

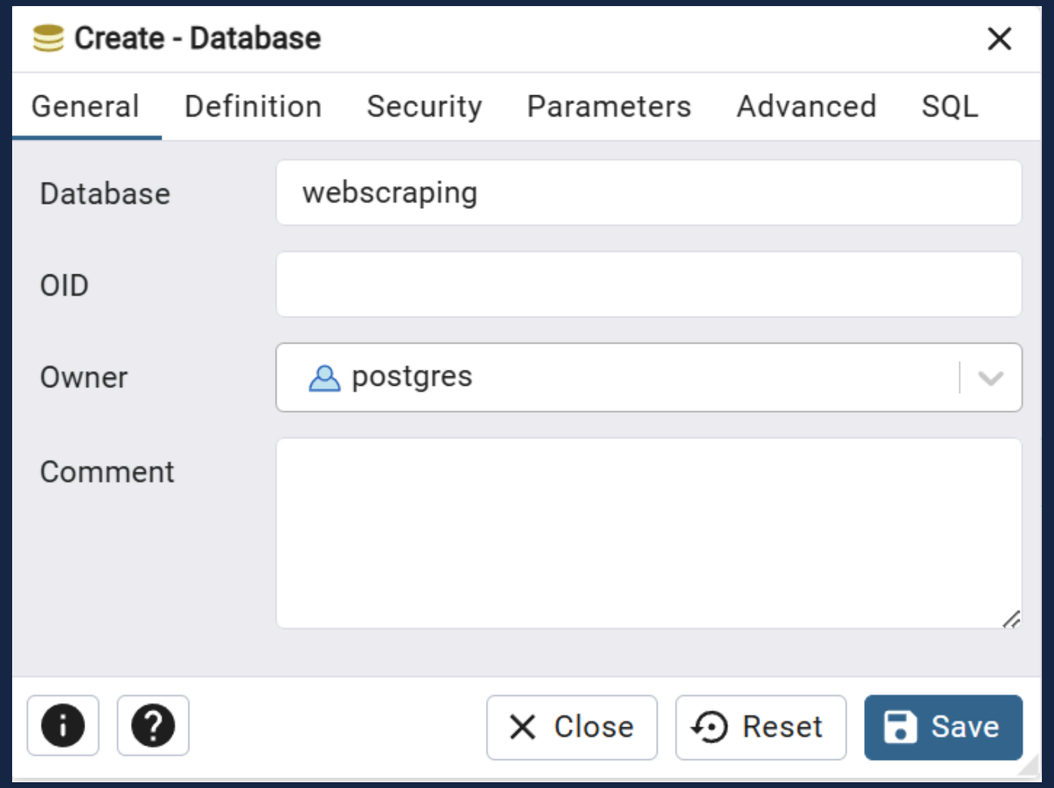
```
Title           0
Price           0
StarRating      0
Available       0
PriceCategory   0
dtype: int64
```

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.



```sql
CREATE TABLE books(
    Book_ID INT PRIMARY KEY,
    Title VARCHAR(50) NOT NULL,
    Price NUMERIC(10,2) NOT NULL,
    StarRating VARCHAR(10),
    Available VARCHAR(10) NOT NULL,
    PriceCategory VARCHAR(10) NOT NULL
);
```

```sql
SELECT * FROM books
```

Output | Messages | Notifications

SQL

book_id [PK] integer	title character varying (50)	price numeric (10,2)	starrating character varying (10)	available character varying (10)	pricecategory character varying (10)

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()
```

```
df_cleaned.to_sql(
    table_name,
    engine,
    if_exists='replace',  # Options: 'fail', 'replace', 'append'
    index=True           # 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 'books' in PostgreSQL.
```

BOOKS TABLE

```
SELECT * FROM books
```

Output | Messages | Notifications

Showing rows: 1 to 999 | Page No: 1 of 1

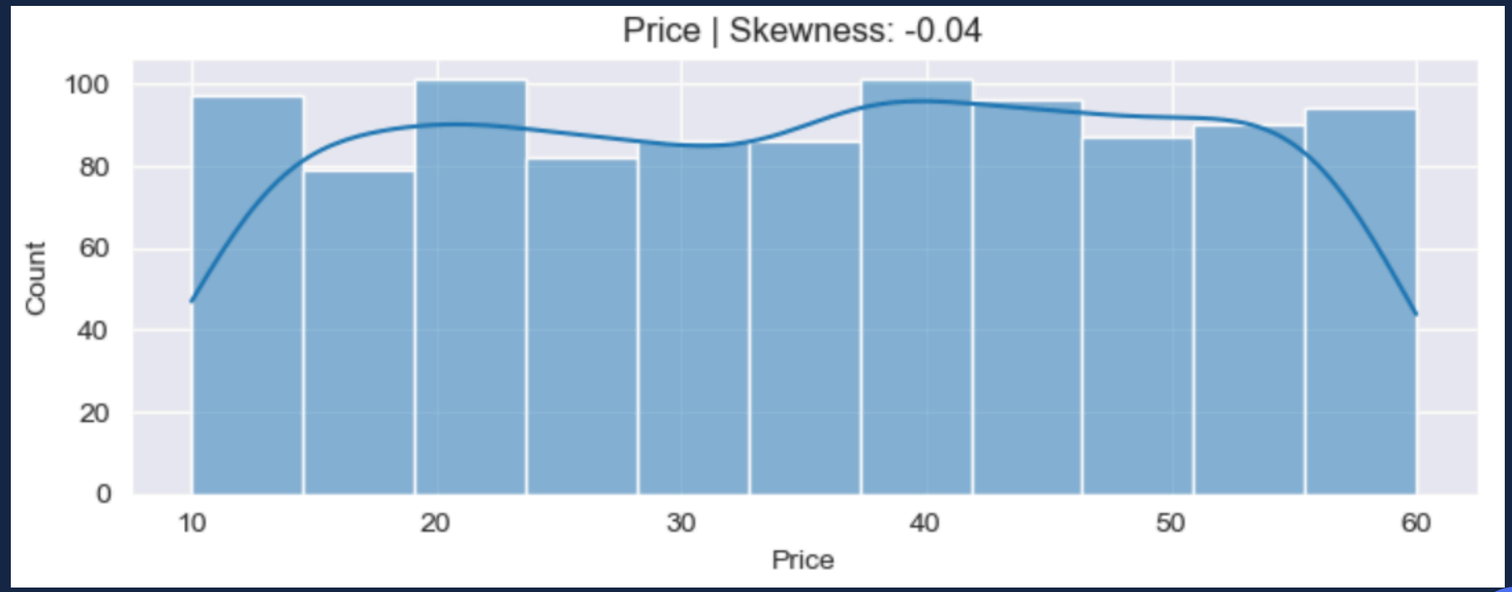
Book_ID bigint	Title text	Price double precision	StarRating text	Available text	PriceCategory text
0	A Light in the Attic	51.77	Three	In stock	High
1	Tipping the Velvet	53.74	One	In stock	High
2	Soumission	50.1	One	In stock	High
3	Sharp Objects	47.82	Four	In stock	Medium
4	Sapiens A Brief History of Humankind	54.23	Five	In stock	High
5	The Requiem Red	22.65	One	In stock	Low
6	The Dirty Little Secrets of Getting Your Dream Job	33.34	Four	In stock	Medium
7	The Coming Woman A Novel Based on the Life of the Infamous Feminist Victoria ...	17.93	Three	In stock	Low
8	The Boys in the Boat Nine Americans and Their Epic Quest for Gold at the 1936 Ber...	22.6	Four	In stock	Low
9	The Black Maria	52.15	One	In stock	High
10	Starving Hearts Triangular Trade Trilogy 1	13.99	Two	In stock	Low

Data Visualization

— Jupyter Notebook

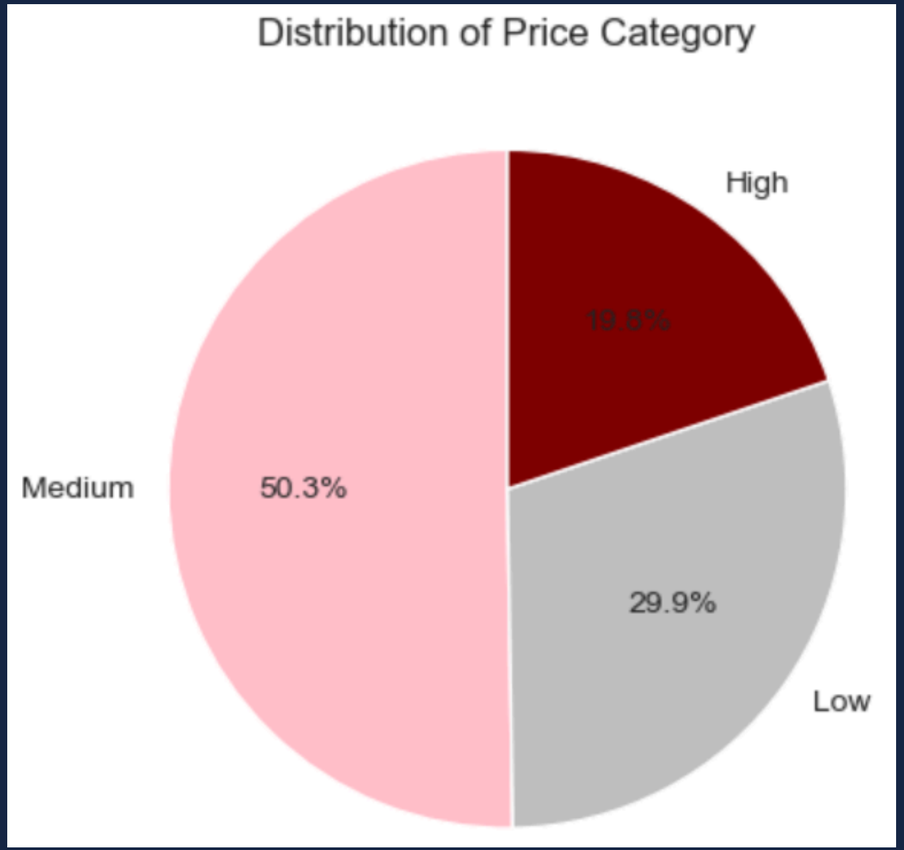
Distribution of Numeric Column

```python
# Distribution of Numerical columns
sns.set_style("darkgrid")

numerical_columns = df_cleaned.select_dtypes(include=["int64","float64"]).columns

plt.figure(figsize=(14, len(numerical_columns) * 3))
for idx, feature in enumerate(numerical_columns, 1):
    plt.subplot(len(numerical_columns), 2, idx)
    sns.histplot(df_cleaned[feature], kde=True)
    plt.title(f"{feature} | Skewness: {round(df_cleaned[feature].skew(), 2)}")

plt.tight_layout()
plt.show()
```



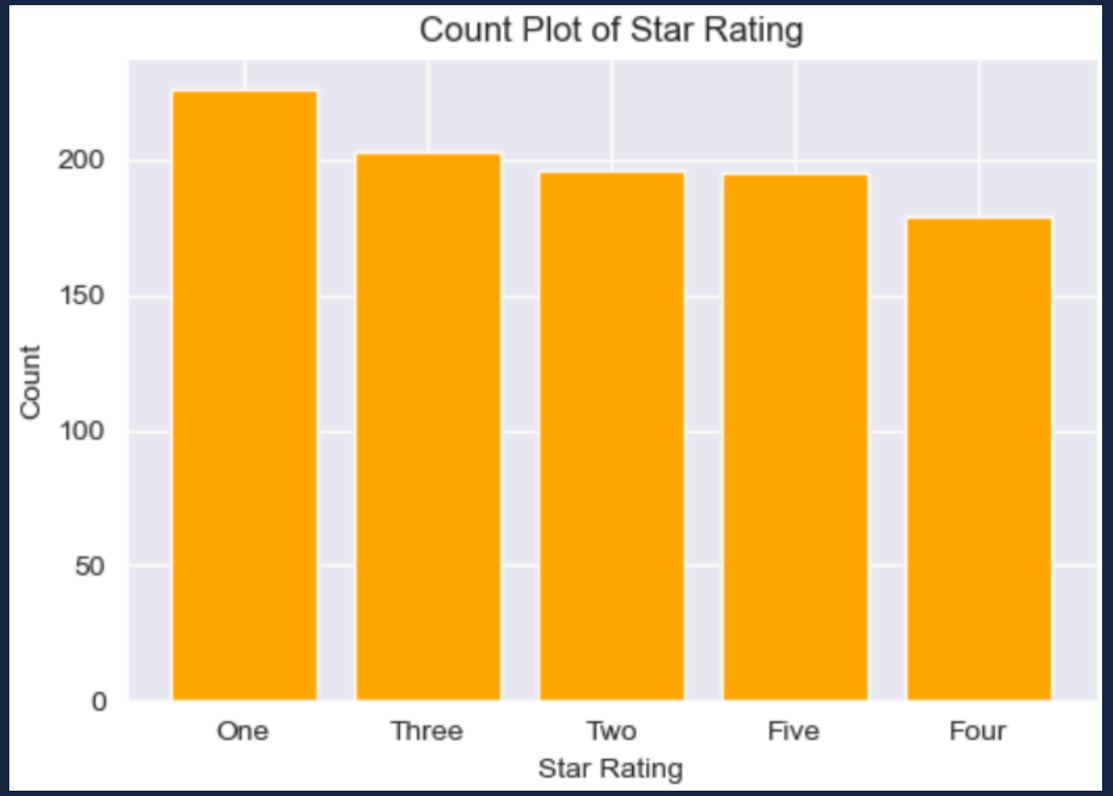
Pie Chart of Price Category

```
# Pie chart of Price Category
# Most of the books here have medium price, between 25-50 dollars
a = ['pink', 'silver', 'maroon'] # colors

counts = df_cleaned['PriceCategory'].value_counts()
counts.plot.pie(autopct='%1.1f%%', startangle=90, colors=a)
plt.title('Distribution of Price Category')
plt.ylabel('')
plt.show()
```



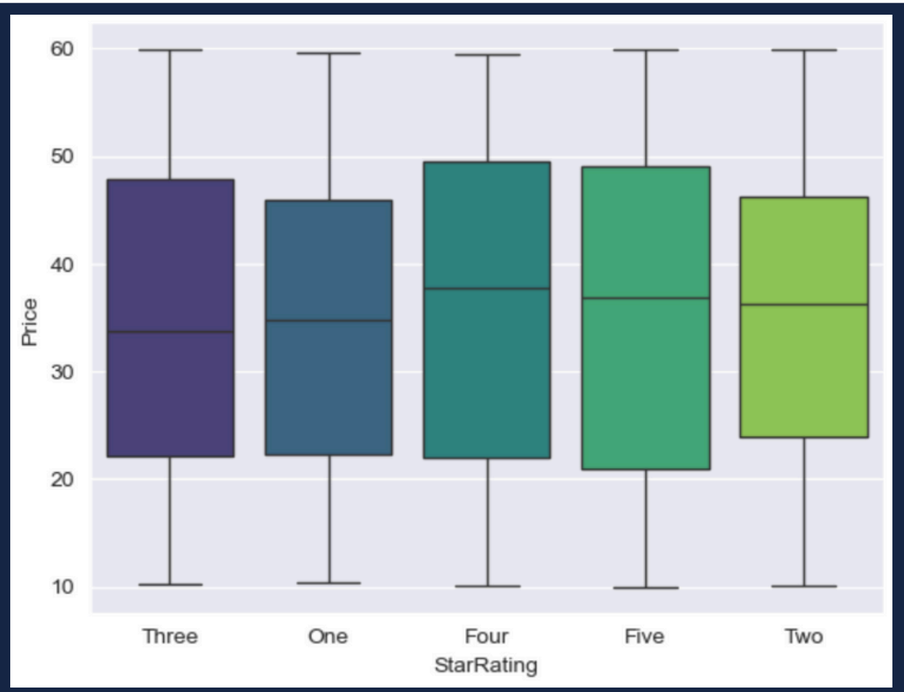
Bar plot of Star Rating

```
# Bar plot of Star Rating
# Most of the books here have a 1 star rating
reviewstar = df_cleaned['StarRating'].value_counts()

plt.figure(figsize=(6, 4))
plt.bar(reviewstar.index, reviewstar, color='orange')
plt.title('Count Plot of Star Rating')
plt.xlabel('Star Rating')
plt.ylabel('Count')
plt.show()
```



Correlation Between Price and Star Rating

```
# Distribution of Star Rating and Price
# 5 star books is worth to buy, its have a wider price range and the most dominant price is around 37 dollars
sns.boxplot(x='StarRating', y='Price', data=df_cleaned, palette='viridis')
```



THANK YOU !!

endahen12@gmail.com

www.linkedin.com/in/endahrakhmawati

ibimbing