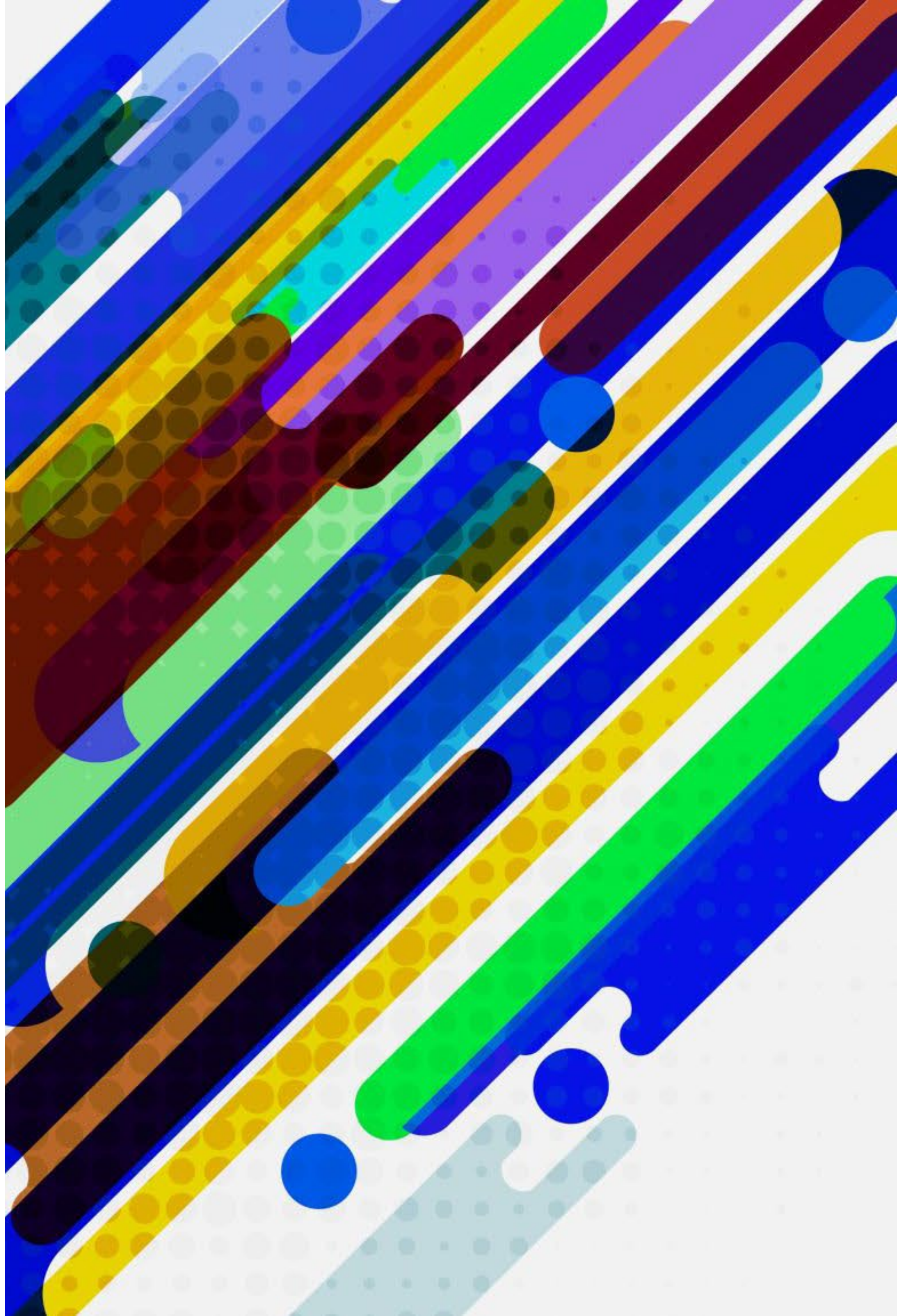
SQLALCHEMY & WEB SCRIPTING (I)

Johnny Zhang

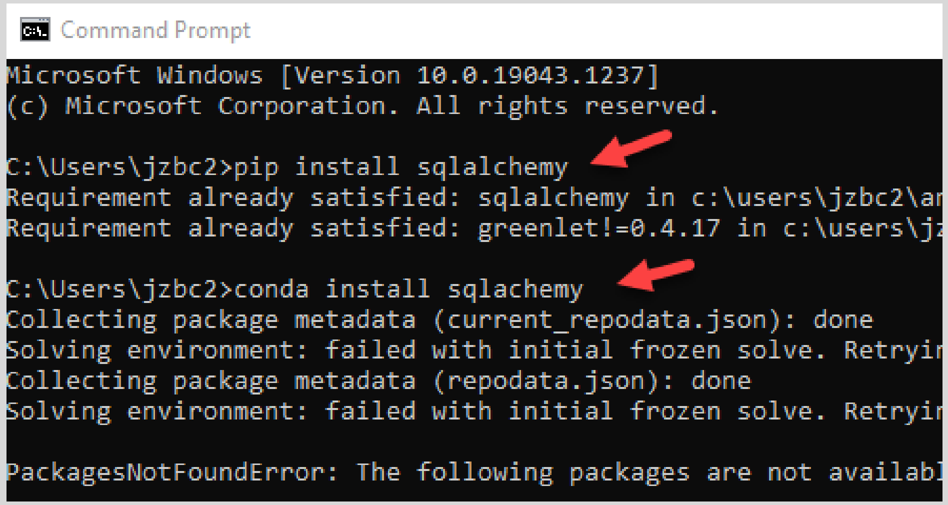


SQLAlchemy

❖SQLAlchemy provides an interface that allows you to query DataFrame with SQL as if the DataFrame is a database table.

❖How to install SQLAlchemy

- Pycharm: Python Packages->Search SQLAlchemy->Click Install
- Python: Command Prompt>pip install sqlalchemy
- Anaconda: Command Prompt>conda install sqlalchemy

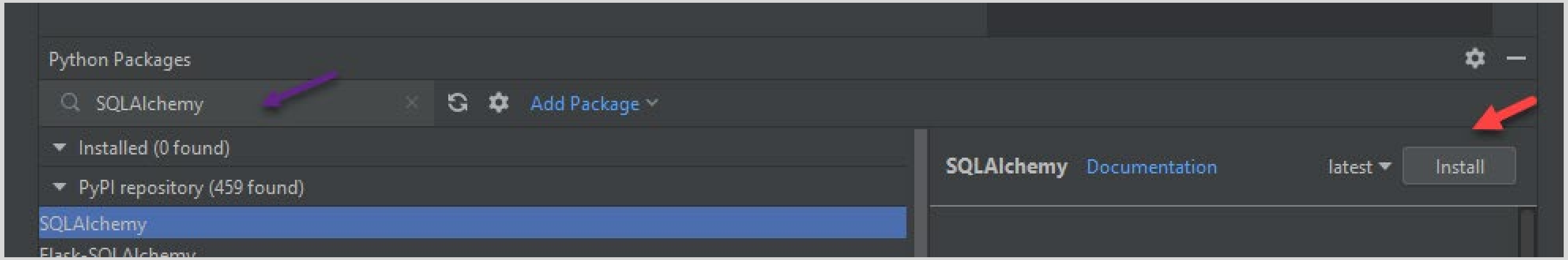


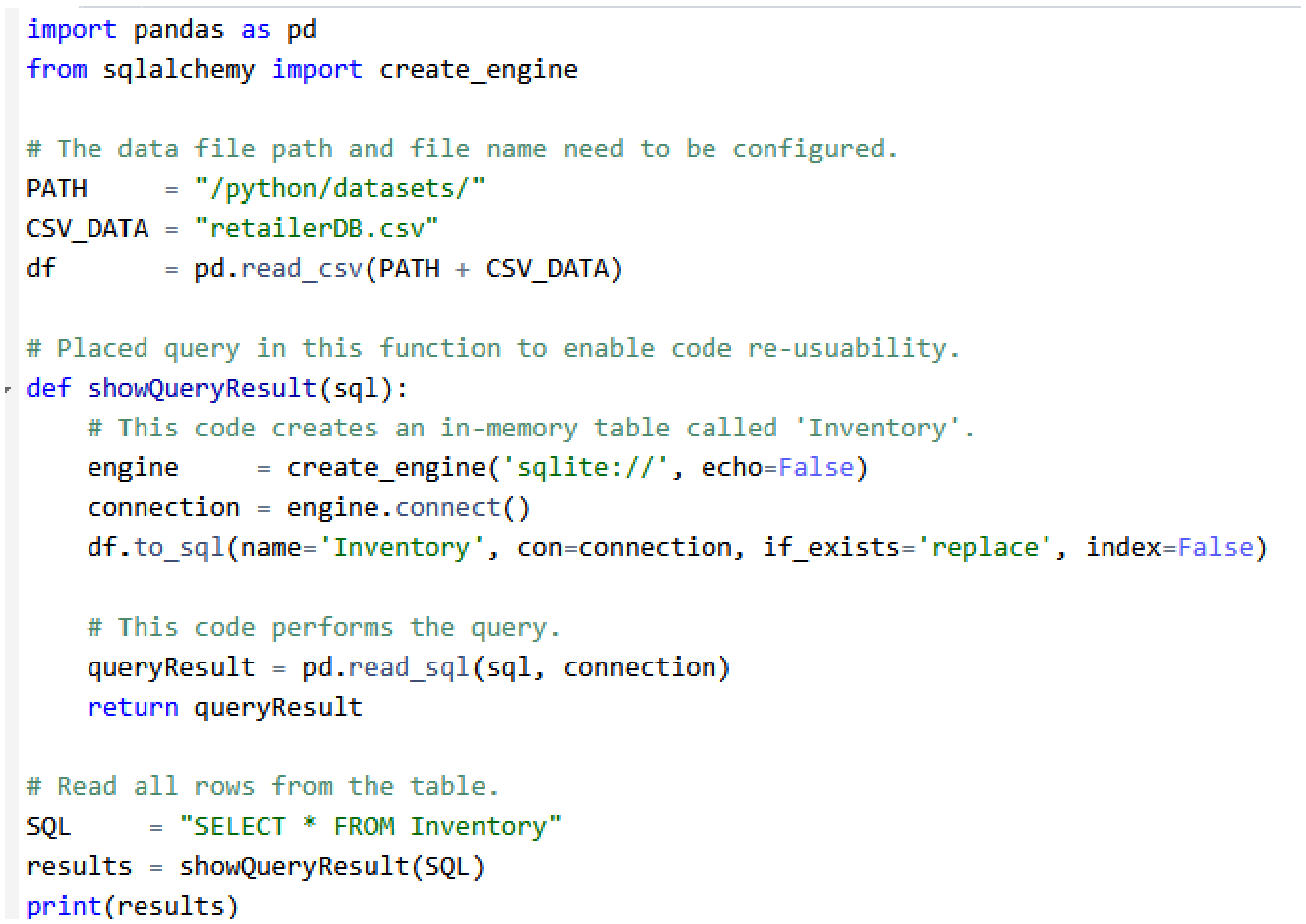
```
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Users\jzbc2>pip install sqlalchemy
Requirement already satisfied: sqlalchemy in c:\users\jzbc2\ar
Requirement already satisfied: greenlet!=0.4.17 in c:\users\jz

C:\Users\jzbc2>conda install sqlachemy
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retryir
Collecting package metadata (repodata.json): done
Solving environment: failed with initial frozen solve. Retryir

PackagesNotFoundError: The following packages are not availabl
```





```python
import pandas as pd
from sqlalchemy import create_engine

# The data file path and file name need to be configured.
PATH     = "/python/datasets/"
CSV_DATA = "retailerDB.csv"
df       = pd.read_csv(PATH + CSV_DATA)

# Placed query in this function to enable code re-usuability.
def showQueryResult(sql):
    # This code creates an in-memory table called 'Inventory'.
    engine     = create_engine('sqlite://', echo=False)
    connection = engine.connect()
    df.to_sql(name='Inventory', con=connection, if_exists='replace', index=False)

    # This code performs the query.
    queryResult = pd.read_sql(sql, connection)
    return queryResult

# Read all rows from the table.
SQL     = "SELECT * FROM Inventory"
results = showQueryResult(SQL)
print(results)
```

https://docs.sqlalchemy.org/en/20/core/engines.html

```python
import pandas as pd
from sqlalchemy import create_engine

engine = create_engine('mssql+pyodbc://sa:your_password@localhost/YourDB?driver=ODBC+Driver+17+for+SQL+Server')

sql = "SELECT * FROM Inventory"
df = pd.read_sql(sql, con=engine)

print(df.head())
```

SQL SELECT Statement

- The SELECT statement is used to select data from a DB.
- The result is stored in a result table (result-set)
- Syntax for SQL SELECT
  - Single Column: SELECT column_name FROM table_name
  - Multiple Columns: SELECT column_name1,column_name2 FROM table_name
  - Everything in a table: SELECT * FROM table_name

Exercise 1

Breakpoint

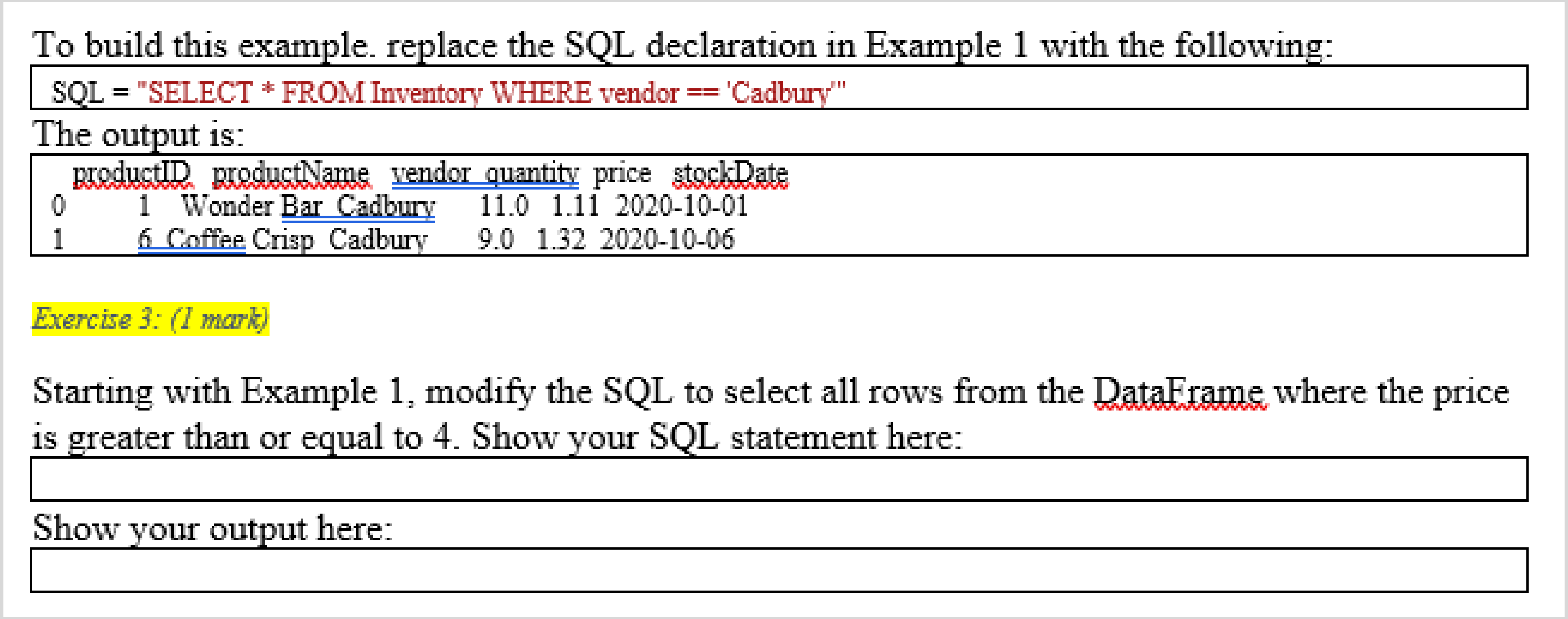
- Breakpoints are special markers that suspend program execution at a specific point. This lets you examine the program state and behavior.
- Breakpoints can be simple (for example, suspending the program on reaching some line of code) or involve more complex logic (checking against additional conditions, writing log messages).

Exercise 2: Set a breakpoint inside the showQueryResult() function. With the debugger window open, while the program is halted, show a screenshot of the program from Example 1 while it is halted at the breakpoint. The value that is stored in the sql parameter must be visible in the debugger window.

Syntax for the SQL WHERE

- ❖The WHERE clause is used to filter records.
- ❖The WHERE clause is used to extract only the records that fulfill the specific parameter
- ❖Syntax for SQL WHERE:
  - **SELECT** column_name
  - **FROM** table_name
  - **WHERE** column_name (math operator) desired_value;

Example 2: Filtering with a WHERE clause



To build this example. replace the SQL declaration in Example 1 with the following:

```
SQL = "SELECT * FROM Inventory WHERE vendor == 'Cadbury'"
```

The output is:

```
   productID  productName  vendor  quantity  price   stockDate
0          1   Wonder Bar  Cadbury      11.0   1.11  2020-10-01
1          6  Coffee Crisp  Cadbury      9.0   1.32  2020-10-06
```

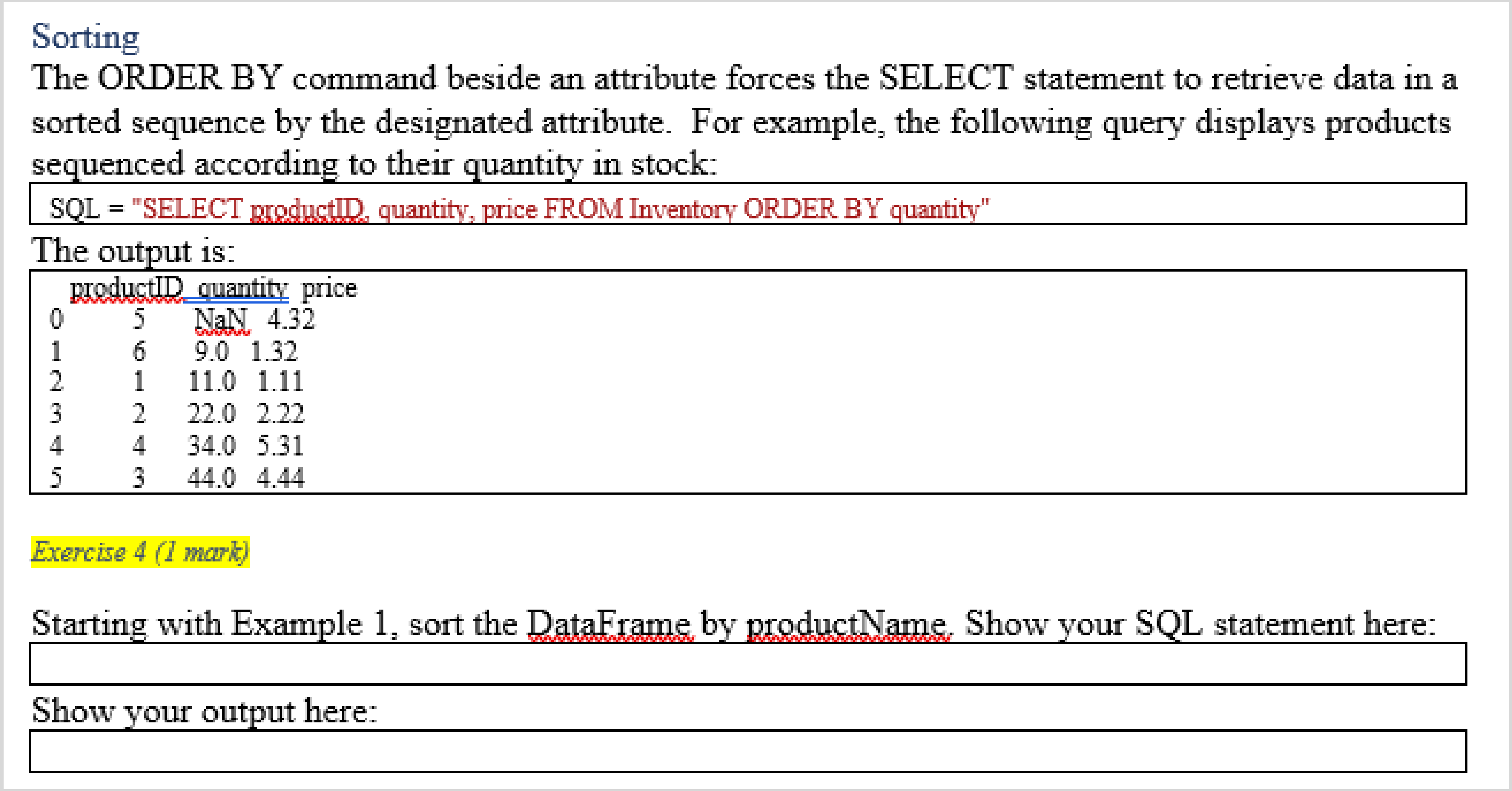
Exercise 3: (1 mark)

Starting with Example 1, modify the SQL to select all rows from the DataFrame where the price is greater than or equal to 4. Show your SQL statement here:

Show your output here:

SQL ORDER BY

- ❖ The ORDER BY keyword is used to sort the result-set by one or more columns.
- ❖ The ORDER BY keyword sorts the records in ascending order by default.
- ❖ To sort the records in a descending order, you can use the keyword, DESC.
- ❖ Syntax:
  - **SELECT** column_name(s)
  - **FROM** table_name
  - **ORDER BY** column_name(s) ASC or DESC



Sorting

The ORDER BY command beside an attribute forces the SELECT statement to retrieve data in a sorted sequence by the designated attribute. For example, the following query displays products sequenced according to their quantity in stock:

```
SQL = "SELECT productID, quantity, price FROM Inventory ORDER BY quantity"
```

The output is:

```
   productID  quantity  price
0          5       NaN   4.32
1          6       9.0   1.32
2          1      11.0   1.11
3          2      22.0   2.22
4          4      34.0   5.31
5          3      44.0   4.44
```

Exercise 4 (1 mark)

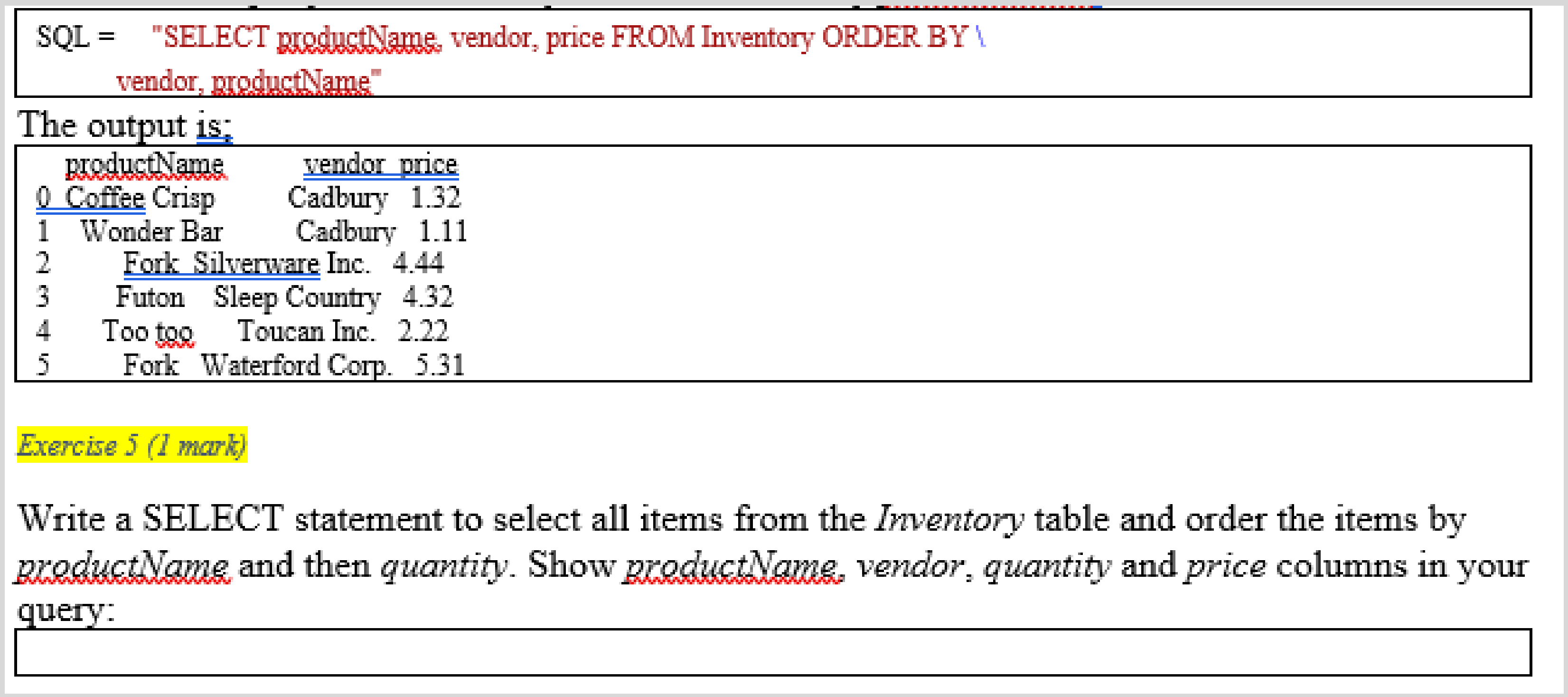
Starting with Example 1, sort the DataFrame by productName. Show your SQL statement here:

Show your output here:

Sorting on Multiple Columns

To group items, you can sort on one column and then another with a comma separated list of columns.

The query below sorts by *vendor* and then by *productName*;



```
SQL =    "SELECT productName, vendor, price FROM Inventory ORDER BY \
        vendor, productName"
```

The output is;

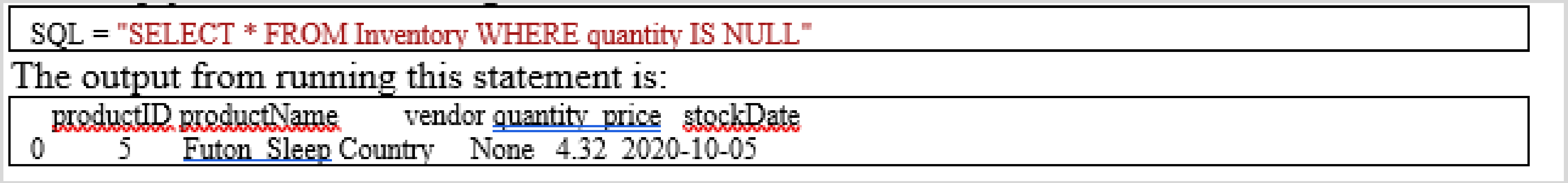
```
   productName        vendor  price
0  Coffee Crisp       Cadbury   1.32
1    Wonder Bar        Cadbury   1.11
2         Fork  Silverware Inc.   4.44
3        Futon    Sleep Country   4.32
4      Too too      Toucan Inc.   2.22
5         Fork   Waterford Corp.   5.31
```

Exercise 5 (1 mark)

Write a SELECT statement to select all items from the *Inventory* table and order the items by *productName* and then *quantity*. Show *productName*, *vendor*, *quantity* and *price* columns in your query:

IS NULL/ IS NOT NULL

It is possible to check for NULL values in the WHERE clause. This statement only pulls records with empty values for the designated attribute.



```
SQL = "SELECT * FROM Inventory WHERE quantity IS NULL"
```

The output from running this statement is:

```
   productID productName        vendor quantity  price   stockDate
0          5       Futon  Sleep Country     None   4.32  2020-10-05
```

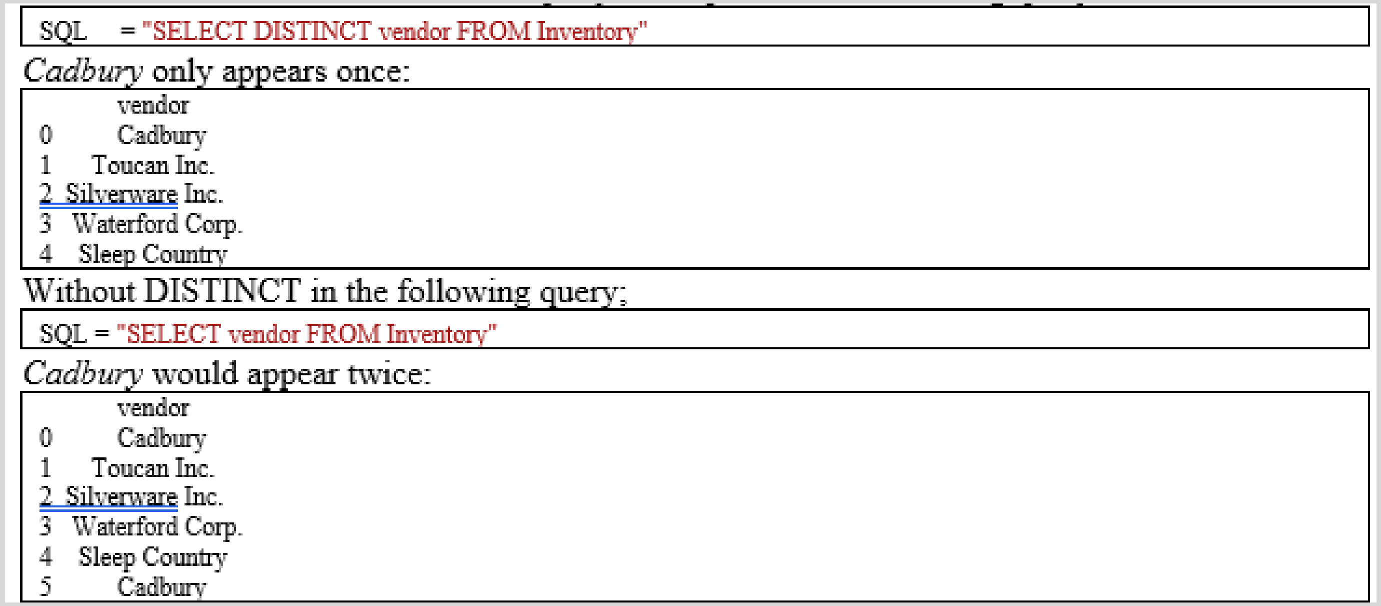
Syntax for the SQL Distinct Statement

❖In a table, a column may contain duplicate values. You may only want to list the unique values.

❖You can use the DISKTINCT keyword to return only unique value.

❖Syntax:

- **SELECT DISTINCT** column_name
- **FROM** table_name



```
SQL    = "SELECT DISTINCT vendor FROM Inventory"
```

Cadbury only appears once:

```
        vendor
0       Cadbury
1     Toucan Inc.
2  Silverware Inc.
3   Waterford Corp.
4    Sleep Country
```

Without DISTINCT in the following query;

```
SQL = "SELECT vendor FROM Inventory"
```

Cadbury would appear twice:

```
        vendor
0       Cadbury
1     Toucan Inc.
2  Silverware Inc.
3   Waterford Corp.
4    Sleep Country
5       Cadbury
```

SQL Wildcards

- A wildcards character can be used to substitute for other characters in a string.
- In SQL, wildcards characters are used with SQL LIKE operator.
- The LIKE operator is used in a WHERE clause to search a specified pattern in a column.
- The LIKE command is used in combination with regular expression symbols.
- Syntax for LIKE in SQL

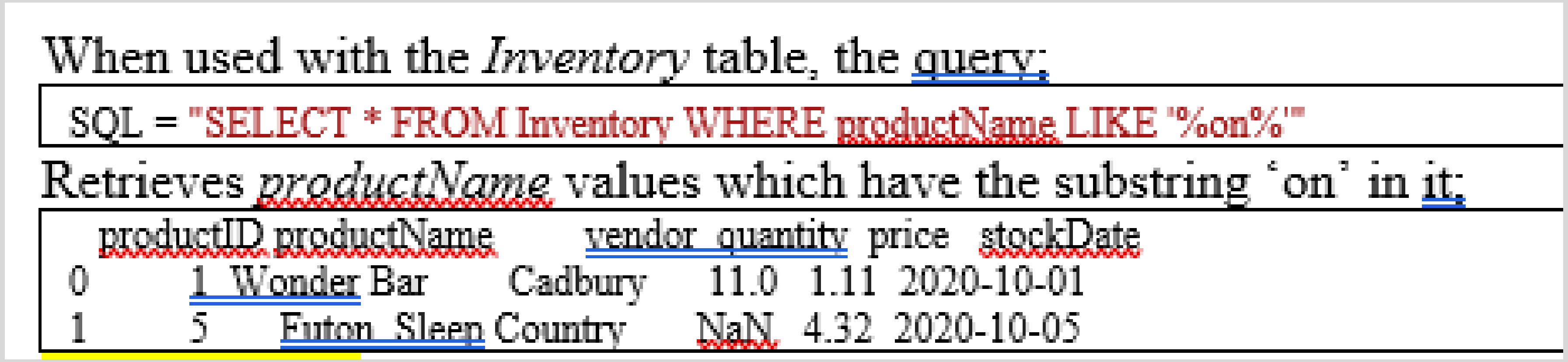
```
SELECT column_name
FROM table_name
WHERE column_name LIKE regular express;
```

%	**Any string of zero or more characters**
_	Any single character
[ ]	Any single character within the specified range (for example, [a-f]) or set (for example, [abcdef])
[^]	Any single character not within the specified range (for example, [^a - f]) or set (for example, [^abcdef])

Examples:

- LIKE 'Mc%' searches for all names that begin with the letters "Mc" (eg. McBadden).
- LIKE '%inger' searches for all names that end with the letters "inger" (eg. Ringer, Stringer).
- LIKE '%en%' searches for all names that have the letters "en" (eg. Bennet, Green, McBadden).
- LIKE '_heryl' searches for all six-letter names ending with the letters "heryl" (eg. Cheryl, Sheryl).
- LIKE '[CK]ars[eo]n' searches for Carsen, Karsen, Carson, and Karson (eg. Carson).
- LIKE '[M-Z]inger' searches for all names ending with the letters "inger" that begin with any single letter from M to Z (eg. Ringer).
- LIKE 'M[^c]%' searches for all names beginning with the letter M that don't have the letter c as the second letter (eg. MacFeather).

Example


When used with the *Inventory* table, the query:

```
SQL = "SELECT * FROM Inventory WHERE productName LIKE '%on%'"
```

Retrieves *productName* values which have the substring 'on' in it:

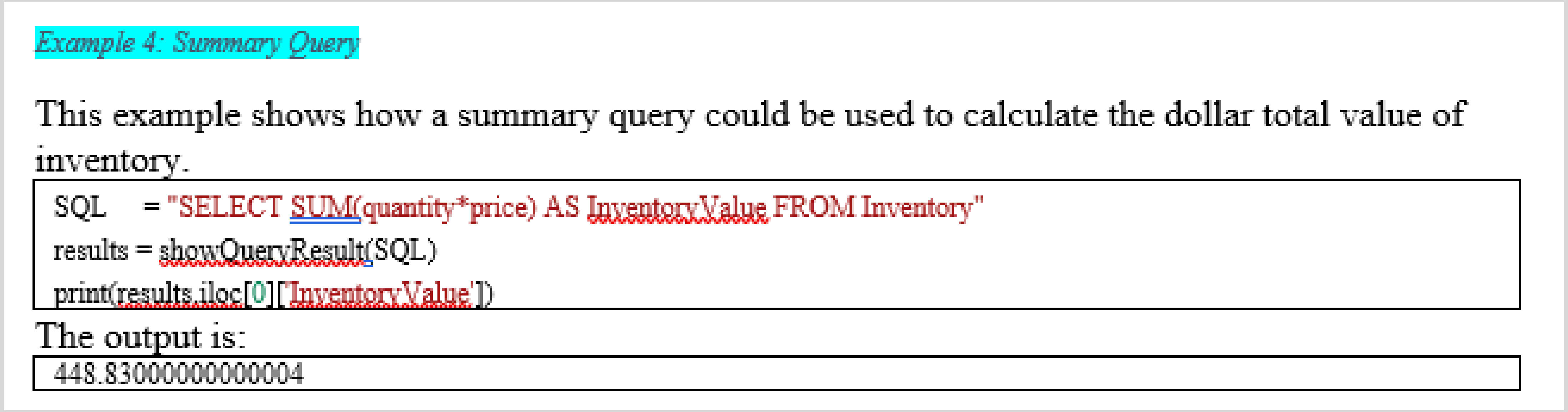
```
   productID productName        vendor  quantity  price   stockDate
0          1  Wonder Bar       Cadbury      11.0   1.11  2020-10-01
1          5       Futon  Sleep Country      NaN   4.32  2020-10-05
```

Exercise 8

Show the query needed to display vendor and productName values where the vendor name ends with 'ry':

Aggregate (Summary) Functions

- **Aggregate Functions**
  - AVG() - Returns the average value.
  - COUNT() - Returns the number of rows.
  - MAX() - Returns the largest value.
  - MIN() - Returns the smallest value.
  - SUM() - Returns the sum.
- **Syntax for Aggregate function in SQL**
  - **SELECT** column_name, aggregate_functrion(column_name)
  - **FROM** table_name
  - **WHERE** column_name



Example 4: Summary Query

This example shows how a summary query could be used to calculate the dollar total value of inventory.

```
SQL    = "SELECT SUM(quantity*price) AS InventoryValue FROM Inventory"
results = showQueryResult(SQL)
print(results.iloc[0]['InventoryValue'])
```

The output is:

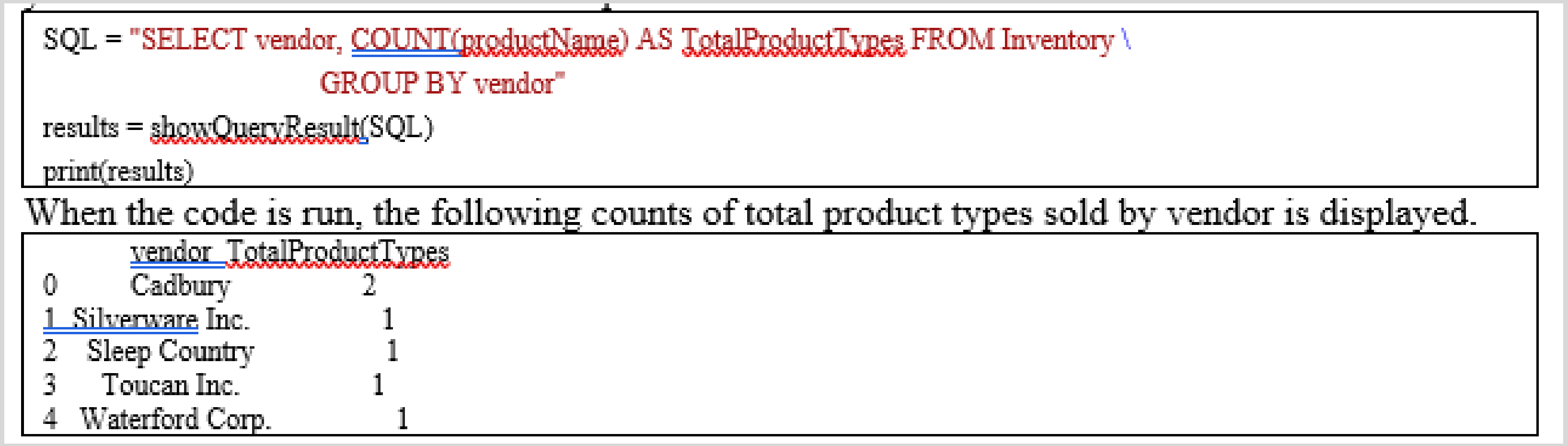
```
448.8300000000004
```

GROUP BY

❖The GROUP BY statement is used with the aggregate functions to group the results by one or more columns.

❖Syntax:

- **SELECT** column_name, aggregate_function(column_name)
- **FROM** table_name
- **WHERE** column_name operator value
- **GROUP BY** column_name

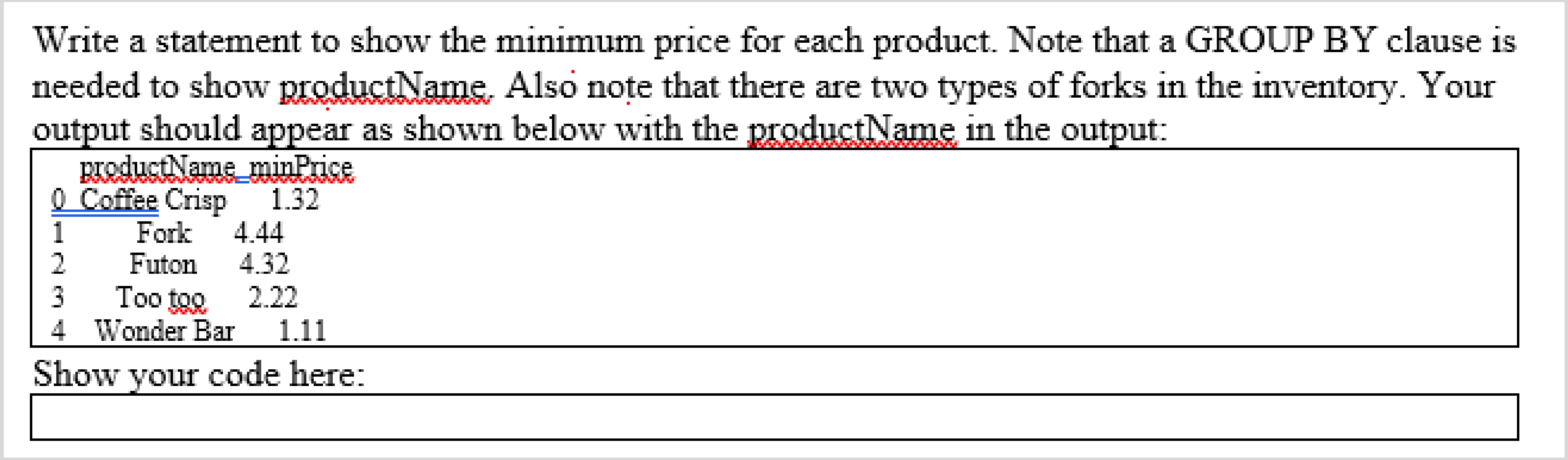


```
SQL = "SELECT vendor, COUNT(productName) AS TotalProductTypes FROM Inventory \
                              GROUP BY vendor"
results = showQueryResult(SQL)
print(results)
```

When the code is run, the following counts of total product types sold by vendor is displayed.

```
          vendor  TotalProductTypes
0        Cadbury                  2
1  Silverware Inc.                1
2    Sleep Country                1
3      Toucan Inc.                1
4   Waterford Corp.               1
```

Exercise 10



Write a statement to show the minimum price for each product. Note that a GROUP BY clause is needed to show productName. Also note that there are two types of forks in the inventory. Your output should appear as shown below with the productName in the output:

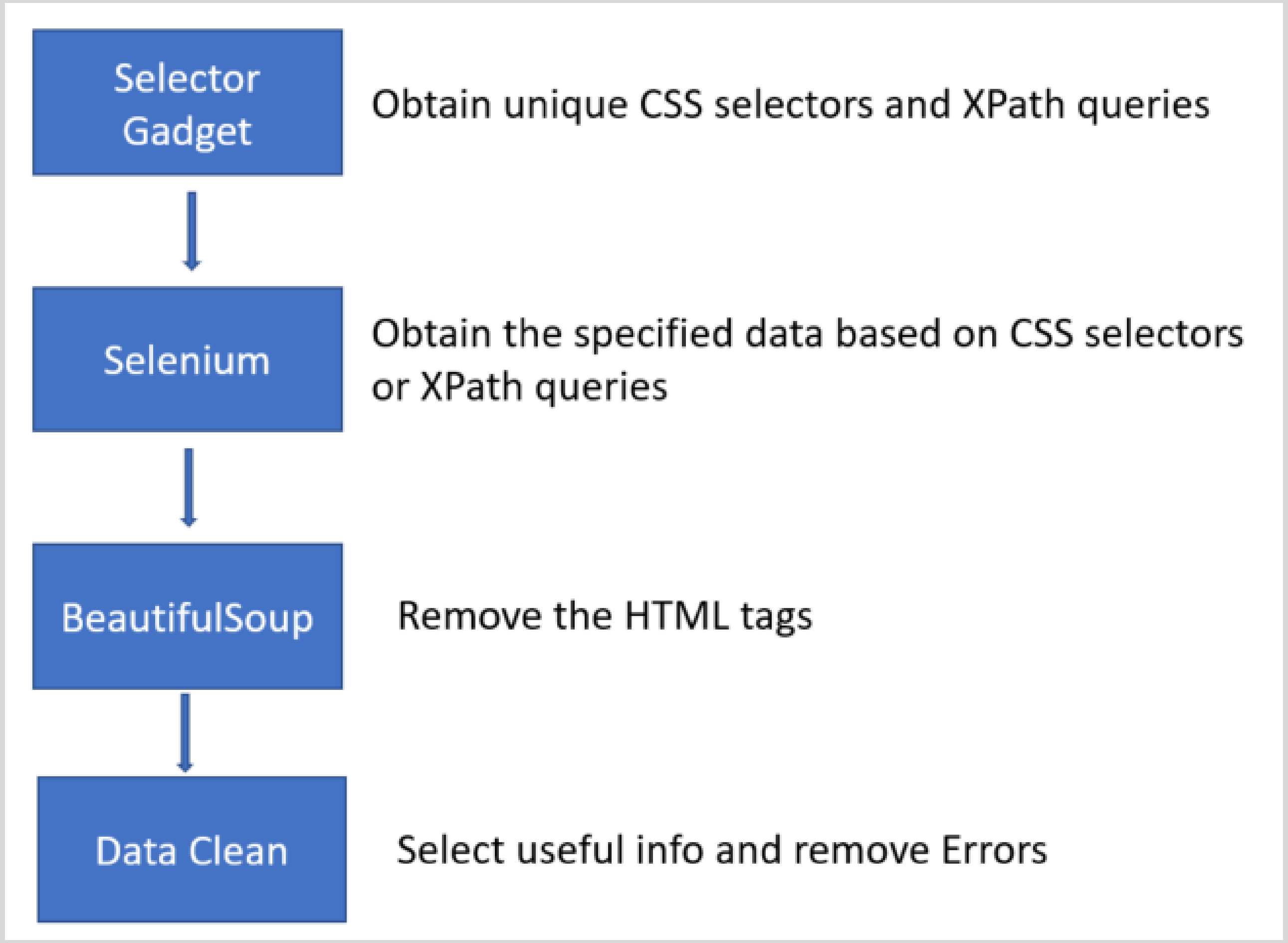
```
   productName  minPrice
0  Coffee Crisp     1.32
1         Fork     4.44
2        Futon     4.32
3      Too too     2.22
4   Wonder Bar     1.11
```

Show your code here:

Web Scraping

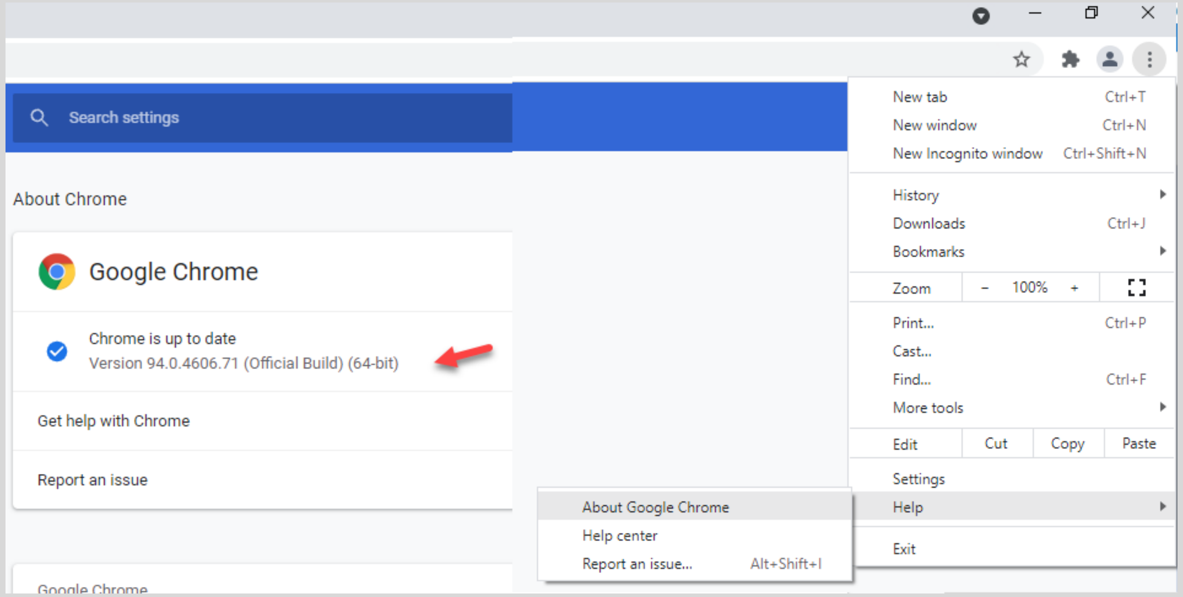
- Web Scraping is the process of extracting content from web sites. (Raw Data Collection)
- Before web scraping, consider the followings:
  - You should check a site's terms and conditions (Permission?). Note: Most sites will not permit public or commercial use fo scraped content without permission
  - Narrow down your scope. Do not overload the site's server
  - You may need to rewrite your code if web pages change their layout.
  - You need to clean up your raw date since web pages are usually inconsistent

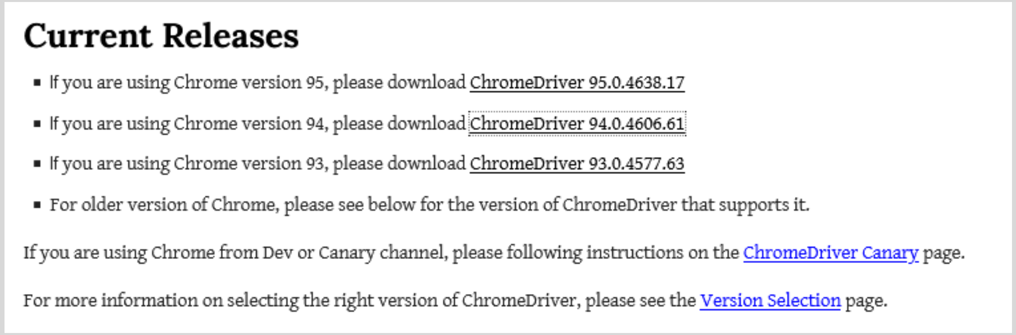
The Process of Web Scraping


Selector Gadget
Obtain unique CSS selectors and XPath queries
Selenium
Obtain the specified data based on CSS selectors or XPath queries
BeautifulSoup
Remove the HTML tags
Data Clean
Select useful info and remove Errors

Selenium and Selector Gadget Installation & Configuration

- Selenium
  - Install ChromeDriver
  - Download Chromedriver (https://chromedriver.chromium.org/downloads)



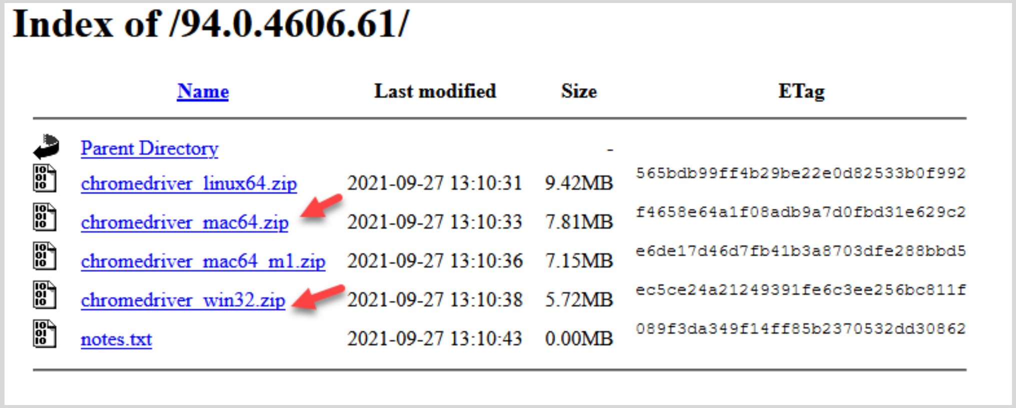


Current Releases

- If you are using Chrome version 95, please download ChromeDriver 95.0.4638.17
- If you are using Chrome version 94, please download ChromeDriver 94.0.4606.61
- If you are using Chrome version 93, please download ChromeDriver 93.0.4577.63
- For older version of Chrome, please see below for the version of ChromeDriver that supports it.

If you are using Chrome from Dev or Canary channel, please following instructions on the ChromeDriver Canary page.

For more information on selecting the right version of ChromeDriver, please see the Version Selection page.



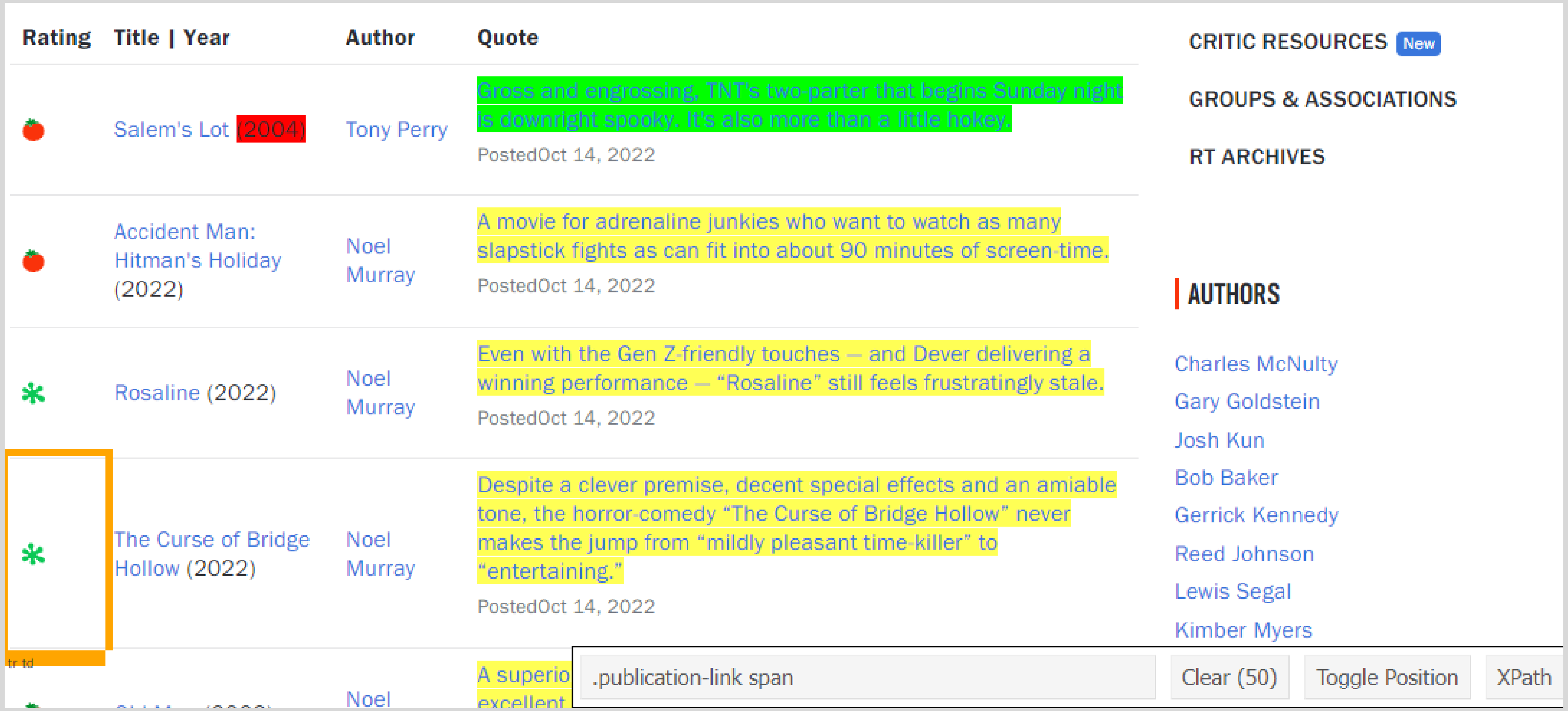
Index of /94.0.4606.61/

Name	Last modified	Size	ETag
Parent Directory		-	
chromedriver_linux64.zip	2021-09-27 13:10:31	9.42MB	565bdb99ff4b29be22e0d82533b0f992
chromedriver_mac64.zip	2021-09-27 13:10:33	7.81MB	f4658e64a1f08adb9a7d0fbd31e629c2
chromedriver_mac64_m1.zip	2021-09-27 13:10:36	7.15MB	e6de17d46d7fb41b3a8703dfe288bbd5
chromedriver_win32.zip	2021-09-27 13:10:38	5.72MB	ec5ce24a21249391fe6c3ee256bc811f
notes.txt	2021-09-27 13:10:43	0.00MB	089f3da349f14ff85b2370532dd30862

Tutorial: Week 4 Lecture: How to Set Up Web Scraping Environment

Selector Gadget

https://www.rottentomatoes.com/critics/source/268



Rating	Title | Year	Author	Quote
🍅	Salem's Lot (2004)	Tony Perry	Gross and engrossing, TNT's two-parter that begins Sunday night is downright spooky. It's also more than a little hokey. PostedOct 14, 2022
🍅	Accident Man: Hitman's Holiday (2022)	Noel Murray	A movie for adrenaline junkies who want to watch as many slapstick fights as can fit into about 90 minutes of screen-time. PostedOct 14, 2022
✱	Rosaline (2022)	Noel Murray	Even with the Gen Z-friendly touches — and Dever delivering a winning performance — “Rosaline” still feels frustratingly stale. PostedOct 14, 2022
✱	The Curse of Bridge Hollow (2022)	Noel Murray	Despite a clever premise, decent special effects and an amiable tone, the horror-comedy “The Curse of Bridge Hollow” never makes the jump from “mildly pleasant time-killer” to “entertaining.” PostedOct 14, 2022
		Noel	A superio excellent

tr td

CRITIC RESOURCES New

GROUPS & ASSOCIATIONS

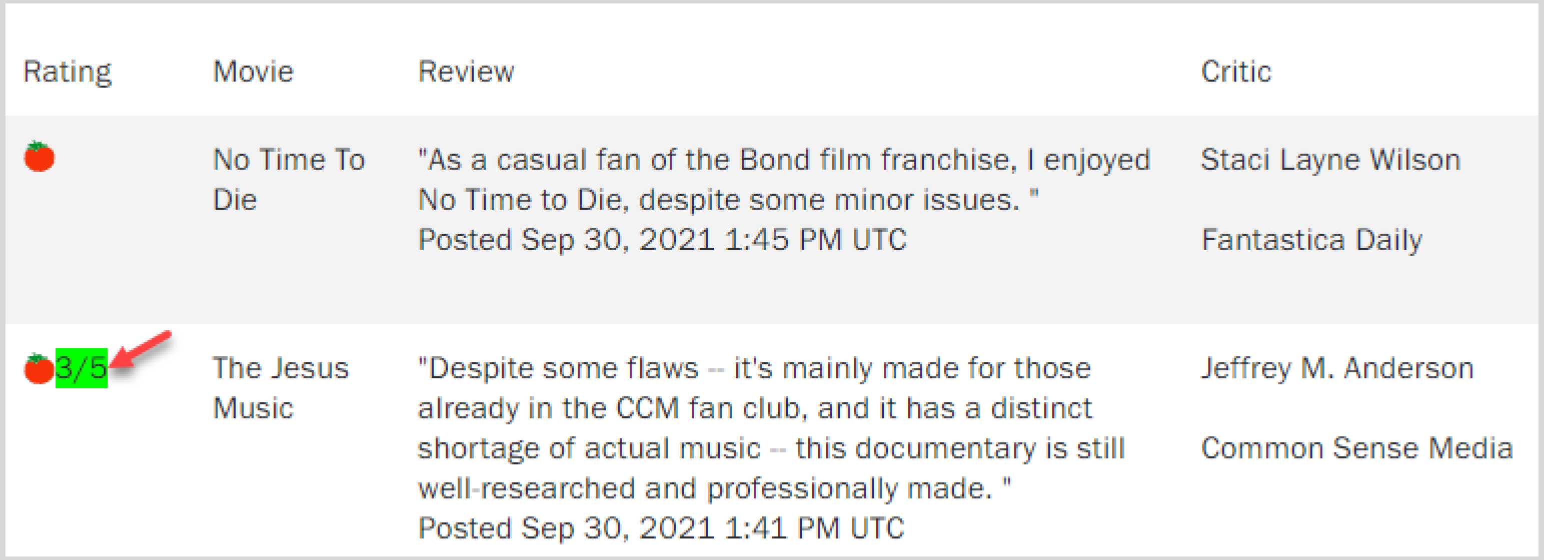
RT ARCHIVES

AUTHORS

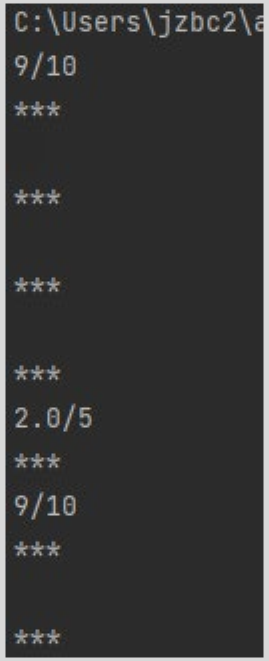
Charles McNulty
Gary Goldstein
Josh Kun
Bob Baker
Gerrick Kennedy
Reed Johnson
Lewis Segal
Kimber Myers

.publication-link span

Example: Selector Gadget



Rating	Movie	Review	Critic
🍅	No Time To Die	"As a casual fan of the Bond film franchise, I enjoyed No Time to Die, despite some minor issues. " Posted Sep 30, 2021 1:45 PM UTC	Staci Layne Wilson Fantastica Daily
🍅3/5	The Jesus Music	"Despite some flaws -- it's mainly made for those already in the CCM fan club, and it has a distinct shortage of actual music -- this documentary is still well-researched and professionally made. " Posted Sep 30, 2021 1:41 PM UTC	Jeffrey M. Anderson Common Sense Media



```
C:\Users\jzbc2\a
9/10
***

***

***

***
2.0/5
***
9/10
***

***
```

.critics-latest-reviews__score

Scraping Rotten Tomatoes

```
import time
from selenium import webdriver
from bs4 import BeautifulSoup

# driver = webdriver.Chrome(ChromeDriverManager().install())
DRIVER_PATH = "/Python/ChromeDriver/chromedriver"
URL = "https://www.rottentomatoes.com/critics/latest_reviews"

browser = webdriver.Chrome(DRIVER_PATH)
browser.get(URL)

# Give the browser time to load all content.
time.sleep(3)

content = browser.find_elements_by_css_selector(".critics-latest-reviews__data-
review")
for e in content:
  start = e.get_attribute('innerHTML')
  # Beautiful soup allows us to remove HTML tags from our content if it exists.
  soup = BeautifulSoup(start, features="lxml")
  print(soup.get_text())
  print("***")  # Go to new line.
```

find_element_by_id
find_element_by_name
find_element_by_xpath
find_element_by_link_text
find_element_by_partial_link_text
find_element_by_tag_name
find_element_by_class_name
find_element_by_css_selector

When we scrape with Selenium we are really extracting downloaded content that is delivered to our browser. To allow browser enough time to load content before we scrape it. We can use a sleep() function to pause the code.

find_elements_by_name

find_elements_by_xpath

find_elements_by_link_text

find_elements_by_partial_link_text

find_elements_by_tag_name

find_elements_by_class_name

find_elements_by_css_selector

Exercises