Web Data Extraction/ Web Scraping Report

Rishabh Kaushick
College of Engineering
Northeastern University
Toronto, ON
kaushick.r@northeastern.edu

Introduction

In this project, I aim to gather data about people selling and placing bids on iconic luxury or sports cars. I would like to understand and gain data about different car brands, models, engine, and how much money people are willing to spend on such exotic vehicles.

Methodology

The methods used in this project:

1. Language: Python 3 on Jupyter Notebook
2. IDE: VS Code
3. Browser: Google Chrome
4. Python Libraries:
    a. bs4 (BeautifulSoup)
    b. requests
    c. selenium

Idea

I have decided to scrape the website – 'https://carsandbids.com/'. The website homepage looks as follows. The homepage shows the auctions of different car models, the highest bid, and the day when the car has been set for auctioning.

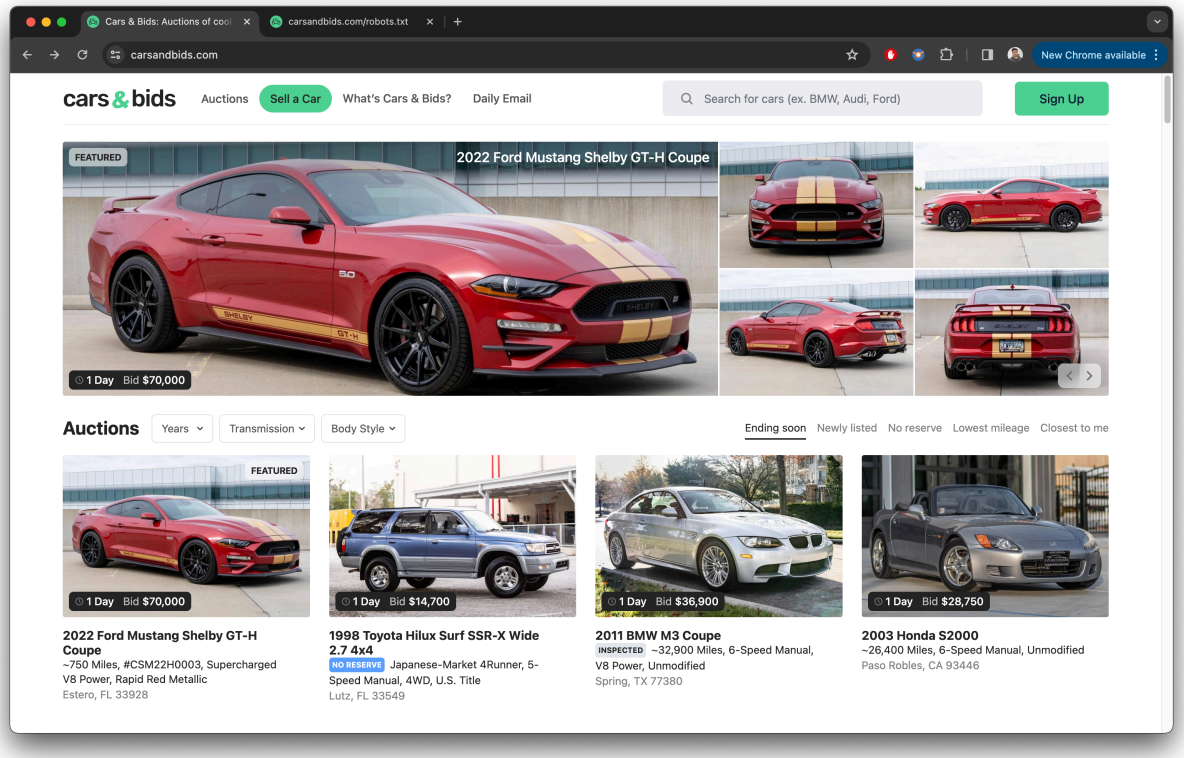


Image 1: Home Page Image (1/2)

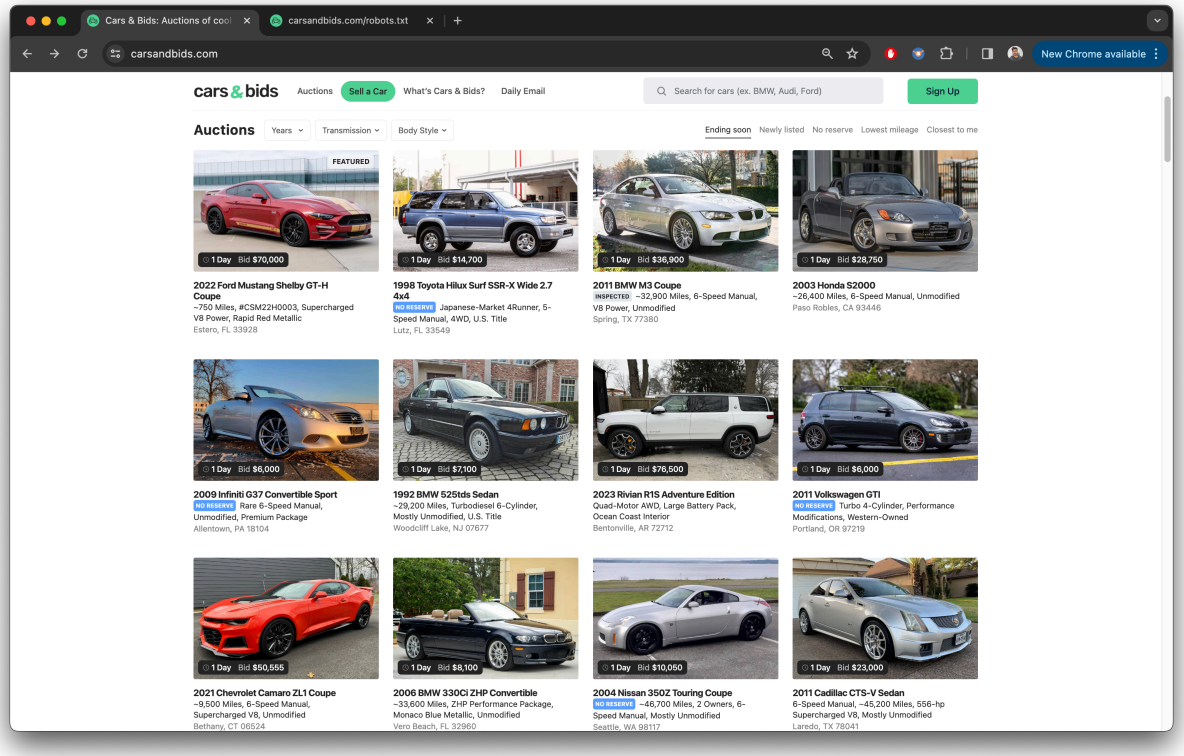


Image 2: Home Page Image (2/2)

From the homepage, we have a lot of information about the car, but we cannot see separate distinctions for car brand, model, engine, etc. When we click on a car in the auction, we can see all these details stored in a tabular form.

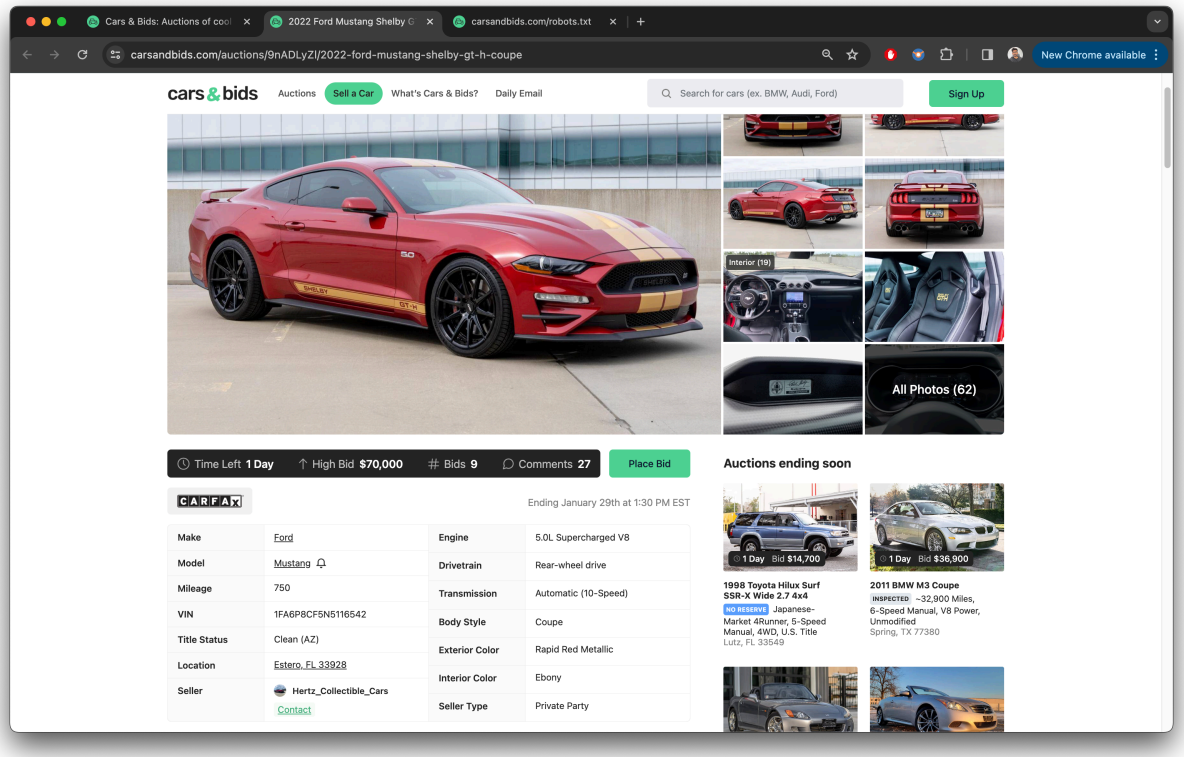


Image 3: Car Page

Therefore, in this project, my idea is to get all the links of the different cars from the homepage. After I have all these links in a list, I will loop through this list and I will visit each page individually to scrape all the data specific to each car.

Solution

Getting Data Elements & Links from the Home Page

Firstly, let us try to get the elements from the webpage using Python code. For this we can use the requests library to create a GET request as shown in Image 4. From the below image, we can see that the HTTP GET request has received a 200: OK response.

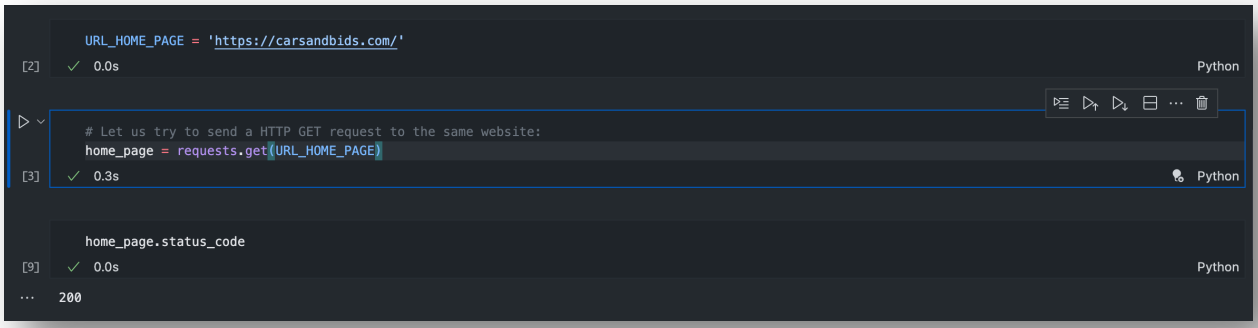


Image 4:

Let us try to parse this HTML and see the content:

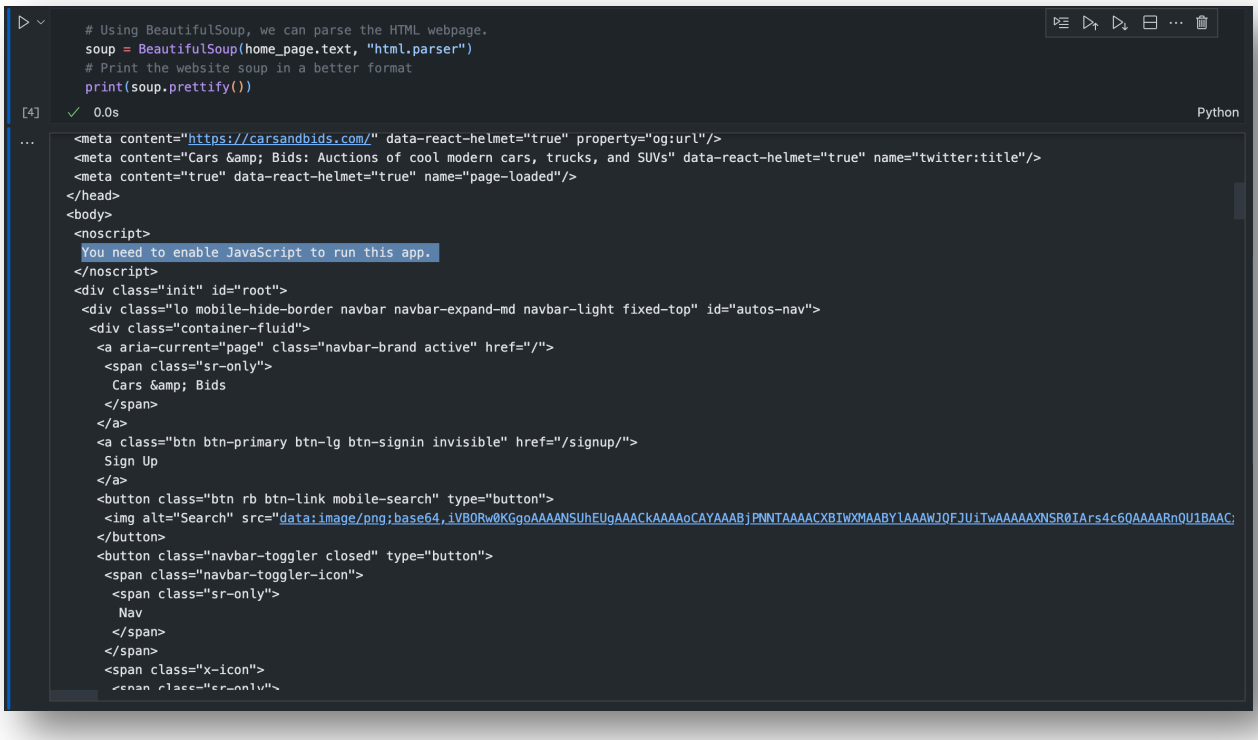
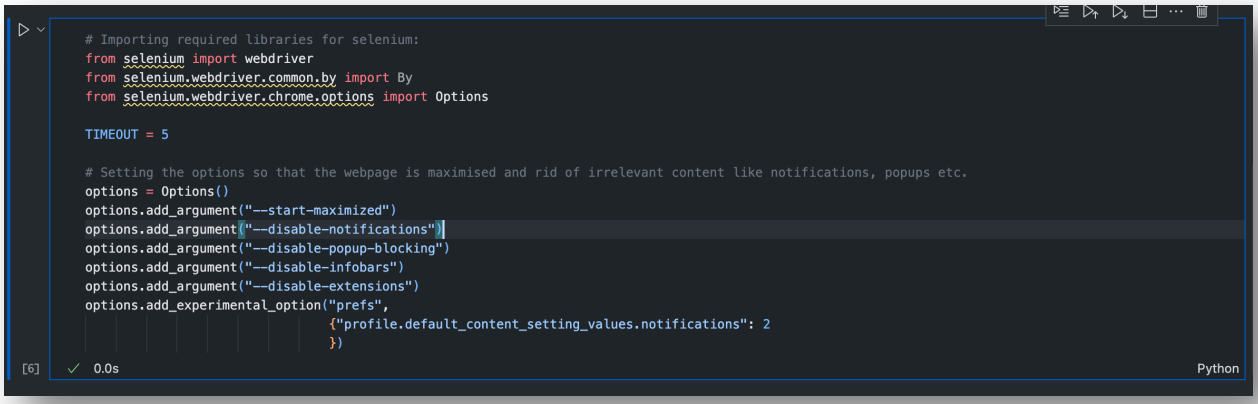


Image 5:

Based on the above response to the GET request, we can see that it says, "***You need to enable JavaScript to run this app***.".

Let us try to run this through the Selenium library - which is a web automation testing software. I have decided to still use selenium library because it opens the website on the Google Chrome browser and allows me to see the webpage which is being scraped. First, let us import the required libraries for selenium and set up different options. By setting these options we can

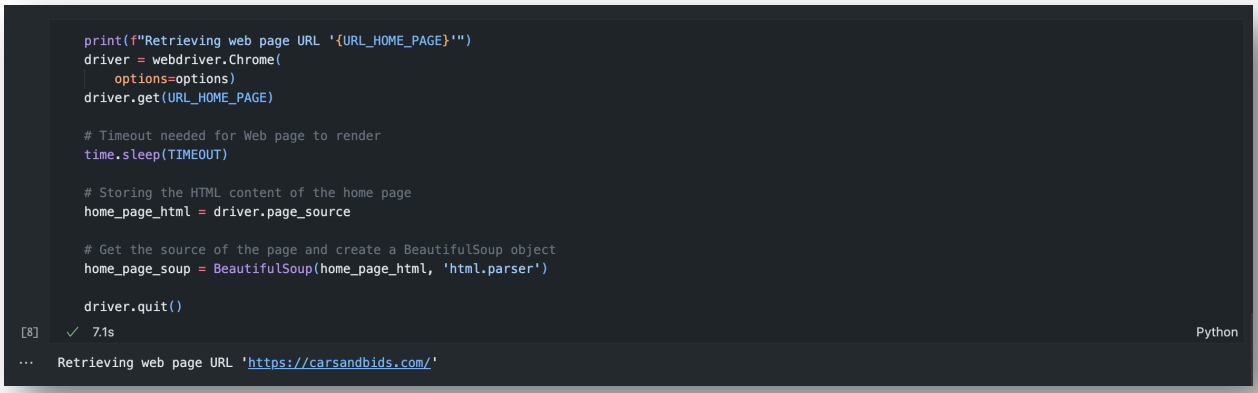
maximize the window and get rid of irrelevant content like notifications, popups, and even disable extensions, to name a few.



```python
# Importing required libraries for selenium:
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options

TIMEOUT = 5

# Setting the options so that the webpage is maximised and rid of irrelevant content like notifications, popups etc.
options = Options()
options.add_argument("--start-maximized")
options.add_argument("--disable-notifications")
options.add_argument("--disable-popup-blocking")
options.add_argument("--disable-infobars")
options.add_argument("--disable-extensions")
options.add_experimental_option("prefs",
                                {"profile.default_content_setting_values.notifications": 2
                                })
```

[6] ✓ 0.0s Python

Let us now try to retrieve the webpage:

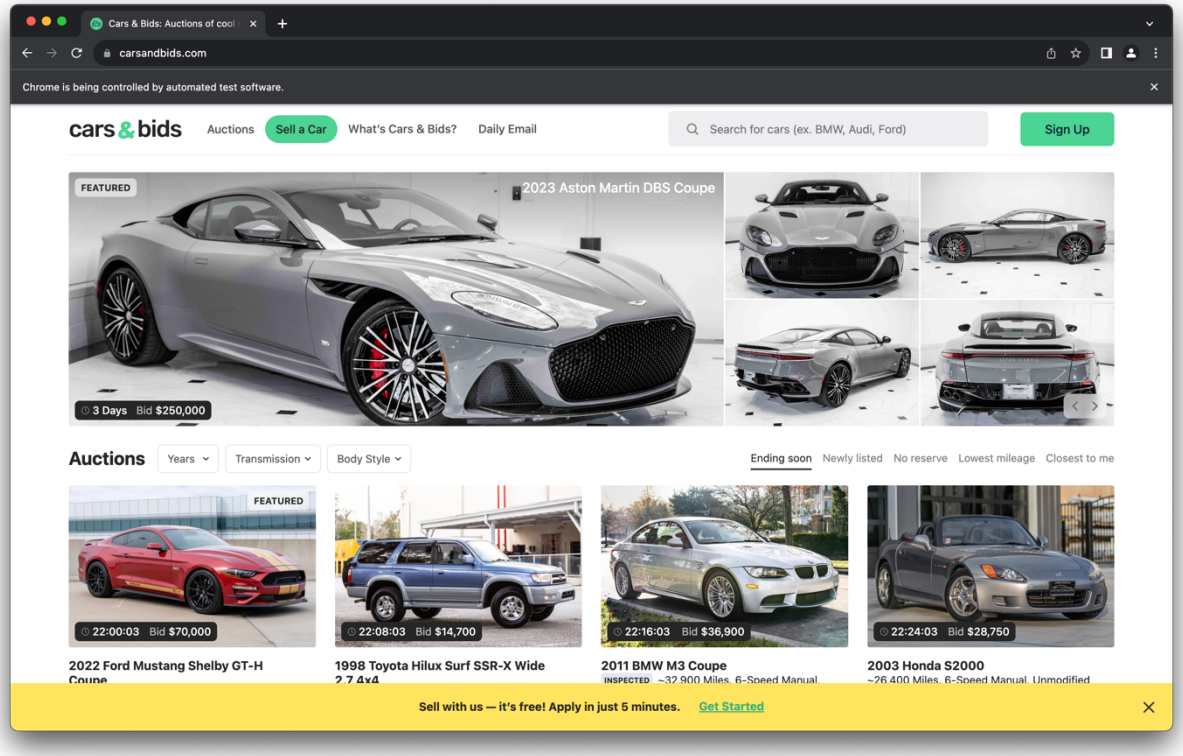


```python
print(f"Retrieving web page URL '{URL_HOME_PAGE}'")
driver = webdriver.Chrome(
    options=options)
driver.get(URL_HOME_PAGE)

# Timeout needed for Web page to render
time.sleep(TIMEOUT)

# Storing the HTML content of the home page
home_page_html = driver.page_source

# Get the source of the page and create a BeautifulSoup object
home_page_soup = BeautifulSoup(home_page_html, 'html.parser')

driver.quit()
```

[8] ✓ 7.1s Python

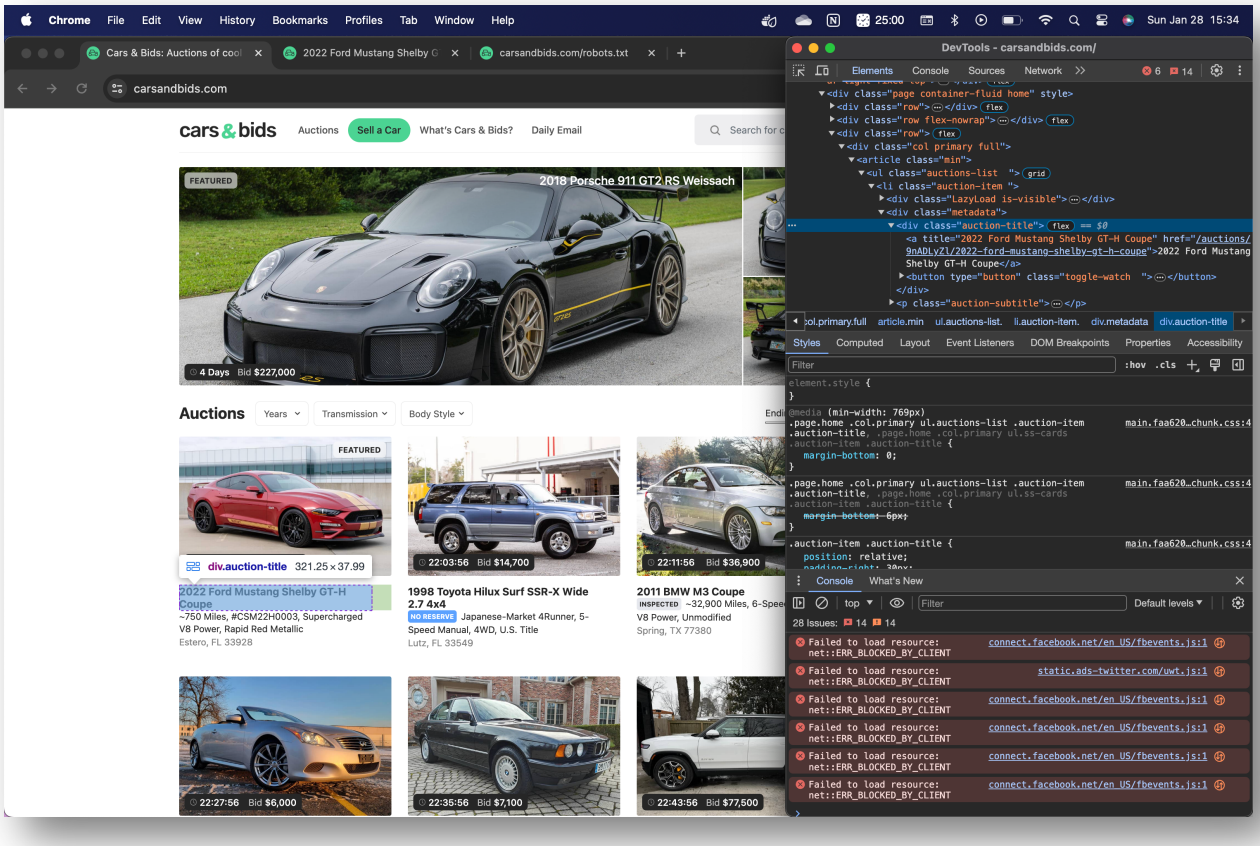
```
Retrieving web page URL 'https://carsandbids.com/'
```

Running the above code will open up the Chrome browser with a note which says: "***Chrome is being controlled by an automated test software***". Based on the webpage which has opened, the HTML page source will be extracted, and a BeautifulSoup object will be created.

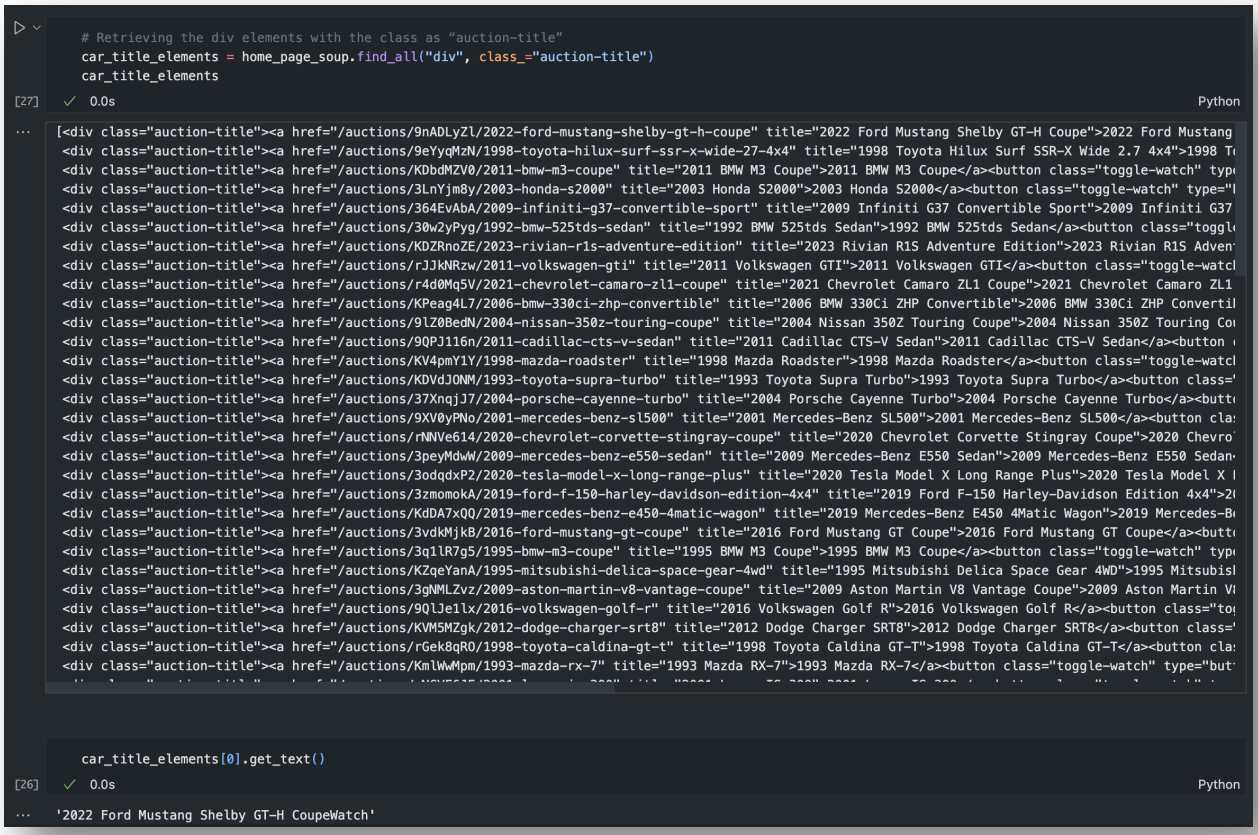


Inspecting Elements & Understanding Tags

Before we get into coding, let us inspect the webpage and navigate to the elements which contains the tile of the car and locate the link. From the image below, we can see that the title is in the div tag with class = “auction-title”. This tag also has a nested <a> tag containing the link.



This is good information, now let us try to obtain all the div tags with the class as “auction-title” from the python code. Once we do this we will also extract the HREF URL from the nested <a> tag:



```
# Retrieving the div elements with the class as "auction-title"
car_title_elements = home_page_soup.find_all("div", class_="auction-title")
car_title_elements
```

[27] ✓ 0.0s Python

```
[<div class="auction-title"><a href="/auctions/9nADLyZl/2022-ford-mustang-shelby-gt-h-coupe" title="2022 Ford Mustang Shelby GT-H Coupe">2022 Ford Mustang
 <div class="auction-title"><a href="/auctions/9eYyqMzN/1998-toyota-hilux-surf-ssr-x-wide-27-4x4" title="1998 Toyota Hilux Surf SSR-X Wide 2.7 4x4">1998 T
 <div class="auction-title"><a href="/auctions/KDbdMZV0/2011-bmw-m3-coupe" title="2011 BMW M3 Coupe">2011 BMW M3 Coupe</a><button class="toggle-watch" typ
 <div class="auction-title"><a href="/auctions/3LnYjm8y/2003-honda-s2000" title="2003 Honda S2000">2003 Honda S2000</a><button class="toggle-watch" type="
 <div class="auction-title"><a href="/auctions/364EvAbA/2009-infiniti-g37-convertible-sport" title="2009 Infiniti G37 Convertible Sport">2009 Infiniti G37
 <div class="auction-title"><a href="/auctions/30w2yPyg/1992-bmw-525tds-sedan" title="1992 BMW 525tds Sedan">1992 BMW 525tds Sedan</a><button class="toggl
 <div class="auction-title"><a href="/auctions/KDZRnoZE/2023-rivian-r1s-adventure-edition" title="2023 Rivian R1S Adventure Edition">2023 Rivian R1S Adven
 <div class="auction-title"><a href="/auctions/rJJkNRzw/2011-volkswagen-gti" title="2011 Volkswagen GTI">2011 Volkswagen GTI</a><button class="toggle-watch
 <div class="auction-title"><a href="/auctions/r4d0Mq5V/2021-chevrolet-camaro-zl1-coupe" title="2021 Chevrolet Camaro ZL1 Coupe">2021 Chevrolet Camaro ZL1
 <div class="auction-title"><a href="/auctions/KPeag4L7/2006-bmw-330ci-zhp-convertible" title="2006 BMW 330Ci ZHP Convertible">2006 BMW 330Ci ZHP Convertib
 <div class="auction-title"><a href="/auctions/9lZ0BedN/2004-nissan-350z-touring-coupe" title="2004 Nissan 350Z Touring Coupe">2004 Nissan 350Z Touring Cou
 <div class="auction-title"><a href="/auctions/9QPJ116n/2011-cadillac-cts-v-sedan" title="2011 Cadillac CTS-V Sedan">2011 Cadillac CTS-V Sedan</a><button
 <div class="auction-title"><a href="/auctions/KV4pmY1Y/1998-mazda-roadster" title="1998 Mazda Roadster">1998 Mazda Roadster</a><button class="toggle-watch
 <div class="auction-title"><a href="/auctions/KDVdJONM/1993-toyota-supra-turbo" title="1993 Toyota Supra Turbo">1993 Toyota Supra Turbo</a><button class="
 <div class="auction-title"><a href="/auctions/37XnqjJ7/2004-porsche-cayenne-turbo" title="2004 Porsche Cayenne Turbo">2004 Porsche Cayenne Turbo</a><butt
 <div class="auction-title"><a href="/auctions/9XV0yPNo/2001-mercedes-benz-sl500" title="2001 Mercedes-Benz SL500">2001 Mercedes-Benz SL500</a><button cla
 <div class="auction-title"><a href="/auctions/rNNVe614/2020-chevrolet-corvette-stingray-coupe" title="2020 Chevrolet Corvette Stingray Coupe">2020 Chevro
 <div class="auction-title"><a href="/auctions/3peyMdwW/2009-mercedes-benz-e550-sedan" title="2009 Mercedes-Benz E550 Sedan">2009 Mercedes-Benz E550 Sedan
 <div class="auction-title"><a href="/auctions/3odqdxP2/2020-tesla-model-x-long-range-plus" title="2020 Tesla Model X Long Range Plus">2020 Tesla Model X
 <div class="auction-title"><a href="/auctions/3zmomokA/2019-ford-f-150-harley-davidson-edition-4x4" title="2019 Ford F-150 Harley-Davidson Edition 4x4">20
 <div class="auction-title"><a href="/auctions/KdDA7xQQ/2019-mercedes-benz-e450-4matic-wagon" title="2019 Mercedes-Benz E450 4Matic Wagon">2019 Mercedes-Be
 <div class="auction-title"><a href="/auctions/3vdkMjkB/2016-ford-mustang-gt-coupe" title="2016 Ford Mustang GT Coupe">2016 Ford Mustang GT Coupe</a><butt
 <div class="auction-title"><a href="/auctions/3q1lR7g5/1995-bmw-m3-coupe" title="1995 BMW M3 Coupe">1995 BMW M3 Coupe</a><button class="toggle-watch" typ
 <div class="auction-title"><a href="/auctions/KZqeYanA/1995-mitsubishi-delica-space-gear-4wd" title="1995 Mitsubishi Delica Space Gear 4WD">1995 Mitsubis
 <div class="auction-title"><a href="/auctions/3gNMLZvz/2009-aston-martin-v8-vantage-coupe" title="2009 Aston Martin V8 Vantage Coupe">2009 Aston Martin V
 <div class="auction-title"><a href="/auctions/9QlJe1lx/2016-volkswagen-golf-r" title="2016 Volkswagen Golf R">2016 Volkswagen Golf R</a><button class="to
 <div class="auction-title"><a href="/auctions/KVM5MZgk/2012-dodge-charger-srt8" title="2012 Dodge Charger SRT8">2012 Dodge Charger SRT8</a><button class="
 <div class="auction-title"><a href="/auctions/rGek8qRO/1998-toyota-caldina-gt-t" title="1998 Toyota Caldina GT-T">1998 Toyota Caldina GT-T</a><button cla
 <div class="auction-title"><a href="/auctions/KmlWwMpm/1993-mazda-rx-7" title="1993 Mazda RX-7">1993 Mazda RX-7</a><button class="toggle-watch" type="but
```

```
car_title_elements[0].get_text()
```

[26] ✓ 0.0s Python

```
'2022 Ford Mustang Shelby GT-H CoupeWatch'
```

From the above we can see that we have retrieved multiple (total 107 cars) car title div tags. I have verified that all of these tags denote a car, and none of them belong to any advertisements. Let us dive into the first car and try to retrieve the URL link:

```
car_title_elements[0].get_text()
```
[29] ✓ 0.0s

```
'2022 Ford Mustang Shelby GT-H CoupeWatch'
```

```
# Now let us try to get the <a> tag and retrieve the link:
print(car_title_elements[0].find('a').get('href'))
```
[30] ✓ 0.0s

```
/auctions/9nADLyZl/2022-ford-mustang-shelby-gt-h-coupe
```

We have tested out this in theory to get the title of the car as well as the link. However, we can see that the link is relative. To visit the website with the car details we need to append this with the URL_HOME_PAGE (https://carsandbids.com). Therefore, the link to this car is 'https://carsandbids.com/auctions/9nADLyZl/2022-ford-mustang-shelby-gt-h-coupe'.

Now, let us try to retrieve all the titles and links in the entire home page containing 107 cars:

```
# Extracting the car title URLs and storing them in a list
# As seen above, each link must be appended with the URL_HOME_PAGE
car_links = []
car_links = [URL_HOME_PAGE+car_titile.find('a').get('href').strip() for car_titile in car_title_elements]
print(car_links)
```
[25] ✓ 0.0s Python

```
['https://carsandbids.com/auctions/9nADLyZl/2022-ford-mustang-shelby-gt-h-coupe', 'https://carsandbids.com/auctions/9eYyqMzN/1998-toyota-hilux-surf-ssr-x-wi
```

For good programming practice, we can write this logic into a function so that it can be reused.

Defining A Function to Retrieve Car Links from Home Page

```
def get_car_urls(soup):
    car_title_elements = soup.find_all("div", class_="auction-title")
    car_links = []
    car_links = [URL_HOME_PAGE+car_titile.find('a').get('href').strip() for car_titile in car_title_elements]
    return car_links
```
[32] ✓ 0.0s Python

Another useful function would be to be able to get the soup (BeautifulSoup object) based on any provided URL link.

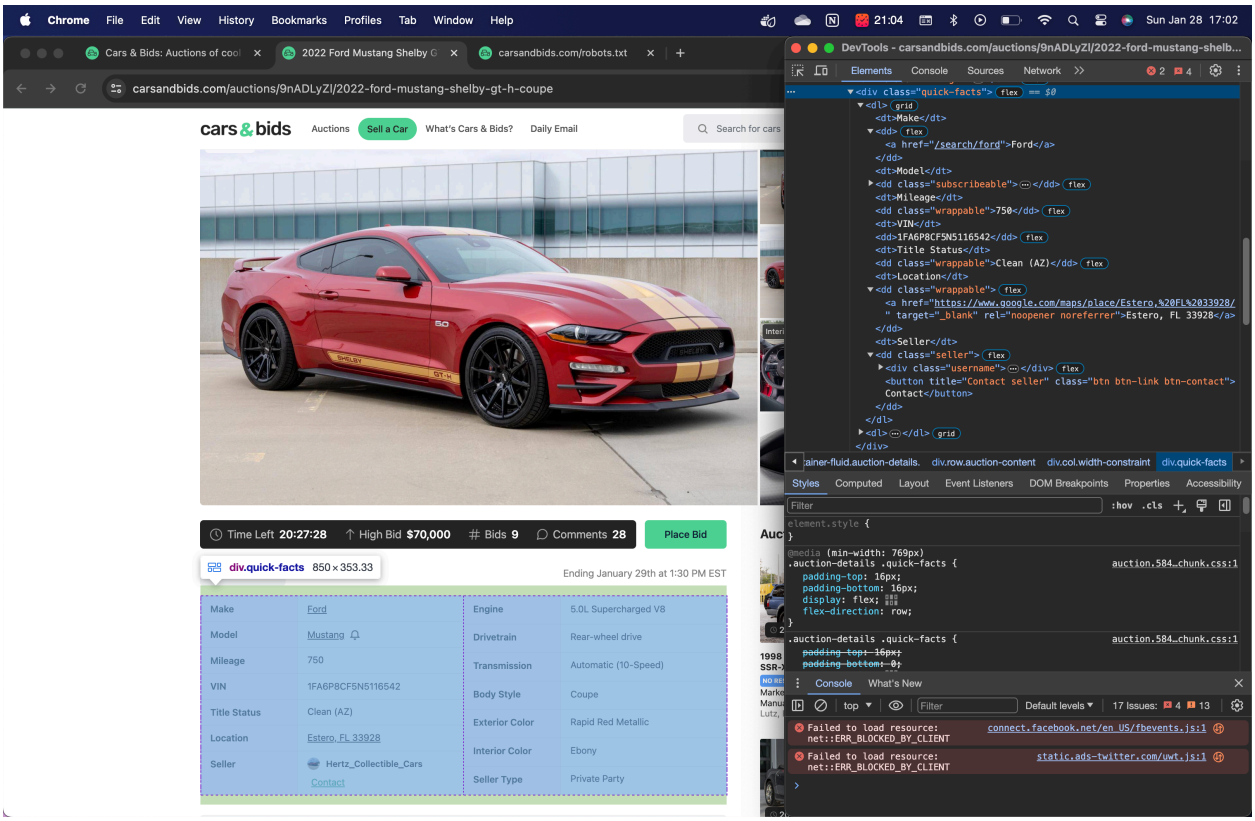
Defining A Function to Get BeautifulSoup Object From a Link

```
def get_soup(url):
    TIMEOUT = 5

    # Setting the options so that the webpage is maximised and rid of irrelevant content like notifications, popups etc.
    options = Options()
    options.add_argument("--start-maximized")
    options.add_argument("--disable-notifications")
    options.add_argument("--disable-popup-blocking")
    options.add_argument("--disable-infobars")
    options.add_argument("--disable-extensions")
    options.add_experimental_option("prefs",
                                    {"profile.default_content_setting_values.notifications": 2
                                    })

    print(f"Retrieving web page URL '{url}'")
    driver = webdriver.Chrome(
        options=options)
    driver.get(url)

    # Timeout needed for Web page to render
    time.sleep(TIMEOUT)

    # Storing the HTML content of the home page
    html = driver.page_source

    # Get the source of the page and create a BeautifulSoup object
    soup = BeautifulSoup(html, 'html.parser')

    driver.quit()
    return soup
```

[33] ✓ 0.0s Python

Getting Data from the Car Page

Now that we have all the links, we will visit each of these websites, and retrieve all the information about these cars. Let us take the example of the first car – '*2022 Ford Mustang Shelby GT-H CoupeWatch*'. We will inspect the element again on this website to be able to understand the different details about the car.



From the above we can see that most of the details are in the div class="quick-facts". Here we have the details such as the make, model, engine, and many more details. Let us try to get individual relevant details from here.

```
# We can get the titles from the 'dt' tag:
quick_facts_element[0].find_all('dt')
keys = [ key.get_text() for key in quick_facts_element[0].find_all('dt')]
print(keys)
```

[47] ✓ 0.0s Python

```
['Make', 'Model', 'Mileage', 'VIN', 'Title Status', 'Location', 'Seller', 'Engine', 'Drivetrain', 'Transmission', 'Body Style', 'Exterior Color', 'Interior
```

```
# We can get the actual data of these from the dd
quick_facts_element[0].find_all('dd')[0].get_text()
values = []
values = [fact.get_text() for fact in quick_facts_element[0].find_all('dd')]
print(values)
```

[48] ✓ 0.0s Python

```
['Ford', 'MustangSave', '750', '1FA6P8CF5N5116542', 'Clean (AZ)', 'Estero, FL 33928', 'Hertz_Collectible_CarsContact', '5.0L Supercharged V8', 'Rear-wheel d
```

From the above, all the data is coming correctly, except for the car model. In the model, there is a bell icon next to the text which has the value “Save” - therefore, we are getting the result as “MustangSave”. However, the correct data is “Mustang”. Similarly, the Seller is 'Hertz_Collectible_Cars' but the value is coming as 'Hertz_Collectible_CarsContact'. Therefore, let us remove the value 'Save' from the Model column and remove the 'Contact' from the Seller column by tweaking the logic.

```python
# In the code below, I am looping through all the 'dd' tags in the quick_facts_element
# if index is 1 (2nd item) then I specifically get the text from the <a> tag - which will have only the Model
# if index is 6 (7th item) then I specifically get the text from the <div> tag with class as "username" - which will have only the Seller username

values = []
for index, fact in enumerate(quick_facts_element[0].find_all('dd')):
    if(index==1):
        values.append(fact.find('a').get_text())
    elif(index==6):
        values.append(fact.find('div', class_="username").get_text())
    else:
        values.append(fact.get_text())
print(values)
```

[55] ✓ 0.0s Python

```
['Ford', 'Mustang', '750', '1FA6P8CF5N5116542', 'Clean (AZ)', 'Estero, FL 33928', 'Hertz_Collectible_Cars', '5.0L Supercharged V8', 'Rear-wheel drive', 'Aut
```

Apart from the quick facts I have also retrieved the values of the Title of the page, bid price, number of bids and number comments, and respectively stored them in the following lists of keys and values:

```
keys
```

[77] ✓ 0.0s

```
['Make',
 'Model',
 'Mileage',
 'VIN',
 'Title Status',
 'Location',
 'Seller',
 'Engine',
 'Drivetrain',
 'Transmission',
 'Body Style',
 'Exterior Color',
 'Interior Color',
 'Seller Type',
 'Highest Bid',
 'Number of Bids',
 'Number of Comments',
 'Title']
```

```
values
```

[78] ✓ 0.0s

```
['Ford',
 'Mustang',
 '750',
 '1FA6P8CF5N5116542',
 'Clean (AZ)',
 'Estero, FL 33928',
 'Hertz_Collectible_Cars',
 '5.0L Supercharged V8',
 'Rear-wheel drive',
 'Automatic (10-Speed)',
 'Coupe',
 'Rapid Red Metallic',
 'Ebony',
 'Private Party',
 '$70,000',
 '9',
 '28',
 '2022 Ford Mustang Shelby GT-H Coupe']
```

For a better way to store these details, let us store them in a Python Dictionary.

```
# Each car will have these facts:
car_facts = {}
for key, value in zip(keys, values):
    car_facts[key] = value
car_facts
```

[79] ✓ 0.0s

```
{'Make': 'Ford',
 'Model': 'Mustang',
 'Mileage': '750',
 'VIN': '1FA6P8CF5N5116542',
 'Title Status': 'Clean (AZ)',
 'Location': 'Estero, FL 33928',
 'Seller': 'Hertz_Collectible_Cars',
 'Engine': '5.0L Supercharged V8',
 'Drivetrain': 'Rear-wheel drive',
 'Transmission': 'Automatic (10-Speed)',
 'Body Style': 'Coupe',
 'Exterior Color': 'Rapid Red Metallic',
 'Interior Color': 'Ebony',
 'Seller Type': 'Private Party',
 'Highest Bid': '$70,000',
 'Number of Bids': '9',
 'Number of Comments': '28',
 'Title': '2022 Ford Mustang Shelby GT-H Coupe'}
```

For each car we will have a dictionary which has the data as shown above. However, since we will have to store details of multiple cars, we will use a list of dictionaries for the final data.

Defining Functions for Different Elements

Now we have the logic to get all these values for one page. Let us write this in multiple functions so that we can reuse the code for other cars web-pages.

Function To Get Title from Car Page

```
def get_car_title(soup):
    return soup.find_all('h1')[0].get_text()
```

[80] ✓ 0.0s

Function To Get Number of Bids

```
def get_number_of_bids(soup):
    return soup.find_all('li', class_="num-bids")[0].find('span', class_="value").get_text()
```

[81] ✓ 0.0s

+ Code + Markdown

Function To Get Number of Comments

```
def get_number_of_comments(soup):
    return soup.find_all('li', class_="num-comments")[0].find('span', class_="value").get_text()
```

[82] ✓ 0.0s

Function To Get Highest Bid

```
def get_highest_bid(soup):
    return soup.find_all('span', class_="bid-value")[0].get_text()
```

[83] ✓ 0.0s

Function To Get All Details of A Car

```python
def get_car_details(car_url):
    car_soup = get_soup(url=car_url)
    keys = []
    values = []

    # Getting the items in the Quick Facts section
    quick_facts_element = car_soup.find_all('div', class_="quick-facts")
    keys = [ key.get_text() for key in quick_facts_element[0].find_all('dt')]
    for index, fact in enumerate(quick_facts_element[0].find_all('dd')):
        if(index==1):
            values.append(fact.find('a').get_text())
        elif(index==6):
            values.append(fact.find('div', class_="username").get_text())
        else:
            values.append(fact.get_text())

    # Getting the items near the heading like car title, bid amount, # of bids, # of comments
    car_title = get_car_title(soup=car_soup)
    keys.append('Title')
    values.append(car_title)

    highest_bid = get_highest_bid(soup=car_soup)
    keys.append('Highest Bid')
    values.append(highest_bid)

    number_of_bids = get_number_of_bids(soup=car_soup)
    keys.append('Number of Bids')
    values.append(number_of_bids)

    number_of_comments = get_number_of_comments(soup=car_soup)
    keys.append('Number of Comments')
    values.append(number_of_comments)

    # Now converting the keys and values list into a dictionary
    car_facts = {}
    for key, value in zip(keys, values):
        car_facts[key] = value
    return car_facts
```

[88] ✓ 0.0s

The above function takes the car webpage url as an input parameter, and retrieves the BeautifulSoup object, and extracts all the relevant fields from the HTML. However, this is only for one webpage, next we will loop over the 107 car website URLs which we have extracted, and we will extract all 18 columns for each car, giving us 1926 total data points.

The below function is used to retrieve all the URLs of all the cars which are shown in the homepage of the website. It will be stored in a list and we will iterate through this list to be able to further get details of each car.

Getting Details of All 107 Cars

Function to Get All Car Links from Home Page

```
def get_all_car_links():
    home_soup = get_soup(url=URL_HOME_PAGE)

    car_title_elements = home_soup.find_all("div", class_="auction-title")
    car_links = []
    car_links = [URL_HOME_PAGE+car_titile.find('a').get('href').strip() for car_titile in car_title_elements]
    return car_links
```

[91] ✓ 0.0s

Issues & Debugging

```
all_cars_list = scrape_website_cars_and_bids()
```

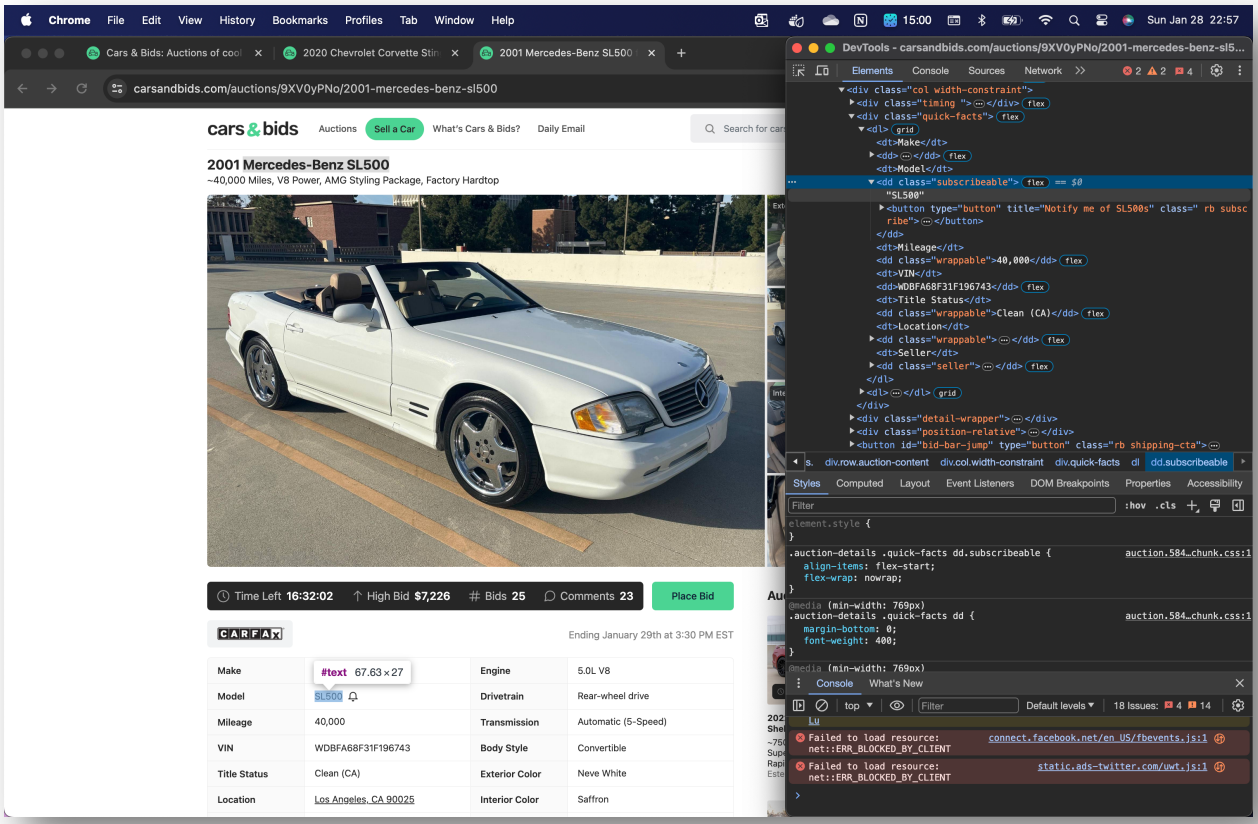
[96] ⊗ 1m 56.6s

```
Retrieving web page URL 'https://carsandbids.com'
Retrieving web page URL 'https://carsandbids.com/auctions/9nADLyZl/2022-ford-mustang-shelby-gt-h-coupe'
Retrieving web page URL 'https://carsandbids.com/auctions/9eYyqMzN/1998-toyota-hilux-surf-ssr-x-wide-27-4x4'
Retrieving web page URL 'https://carsandbids.com/auctions/KDbdMZV0/2011-bmw-m3-coupe'
Retrieving web page URL 'https://carsandbids.com/auctions/3LnYjm8y/2003-honda-s2000'
Retrieving web page URL 'https://carsandbids.com/auctions/364EvAbA/2009-infiniti-g37-convertible-sport'
Retrieving web page URL 'https://carsandbids.com/auctions/30w2yPyg/1992-bmw-525tds-sedan'
Retrieving web page URL 'https://carsandbids.com/auctions/KDZRnoZE/2023-rivian-r1s-adventure-edition'
Retrieving web page URL 'https://carsandbids.com/auctions/rJJkNRzw/2011-volkswagen-gti'
Retrieving web page URL 'https://carsandbids.com/auctions/r4d0Mq5V/2021-chevrolet-camaro-zl1-coupe'
Retrieving web page URL 'https://carsandbids.com/auctions/KPeag4L7/2006-bmw-330ci-zhp-convertible'
Retrieving web page URL 'https://carsandbids.com/auctions/9lZ0BedN/2004-nissan-350z-touring-coupe'
Retrieving web page URL 'https://carsandbids.com/auctions/9QPJ116n/2011-cadillac-cts-v-sedan'
Retrieving web page URL 'https://carsandbids.com/auctions/KV4pmY1Y/1998-mazda-roadster'
Retrieving web page URL 'https://carsandbids.com/auctions/KDVdJONM/1993-toyota-supra-turbo'
Retrieving web page URL 'https://carsandbids.com/auctions/37XnqjJ7/2004-porsche-cayenne-turbo'
Retrieving web page URL 'https://carsandbids.com/auctions/9XV0yPNo/2001-mercedes-benz-sl500'

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[96], line 1
----> 1 all_cars_list = scrape_website_cars_and_bids()

Cell In[94], line 7
      5 for car_url in car_urls:
      6     car_details = {}
----> 7     car_details = get_car_details(car_url)
      8     cars_details.append(car_details)
      9 return cars_details

Cell In[88], line 11
      9 for index, fact in enumerate(quick_facts_element[0].find_all('dd')):
     10     if(index==1):
---> 11         values.append(fact.find('a').get_text())
     12     elif(index==6):
     13         values.append(fact.find('div', class_="username").get_text())

AttributeError: 'NoneType' object has no attribute 'get_text'
```

It seems that while running this query at the 16th record, it get's a NoneType object and the code runs into an error. After some debugging, it looks like it failed while retrieving data for 'Mercedes-Benz SL500' car.

```
merc_details = get_car_details(car_url='https://carsandbids.com/auctions/9XV0yPNo/2001-mercedes-benz-sl500')
[115]  7.7s                                                                                    Python

Retrieving web page URL 'https://carsandbids.com/auctions/9XV0yPNo/2001-mercedes-benz-sl500'
<dd class="subscribeable">SL500<button class="rb subscribe" title="Notify me of SL500s" type="button"><span class="sr-only">Save</span></button></dd>

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[115], line 1
----> 1 merc_details = get_car_details(car_url='https://carsandbids.com/auctions/9XV0yPNo/2001-mercedes-benz-sl500')

Cell In[113], line 12
     10 if(index==1):
     11     print(fact)
---> 12     values.append(fact.find('a').get_text())
     13 elif(index==6):
     14     values.append(fact.find('div', class_="username").get_text())

AttributeError: 'NoneType' object has no attribute 'get_text'
```

Let us look at this webpage:



From the above website, it is running into an error because the model's <dd> tag does not have an <a> tag. Therefore, it is being treated as a NoneType object. Based on this information, let us change the function retrieving the car model.

```python
def get_car_details(car_url):
    car_soup = get_soup(url=car_url)
    keys = []
    values = []

    # Getting the items in the Quick Facts section
    quick_facts_element = car_soup.find_all('div', class_="quick-facts")
    keys = [ key.get_text() for key in quick_facts_element[0].find_all('dt')]
    for index, fact in enumerate(quick_facts_element[0].find_all('dd')):
        if(index==1):
            if(fact.find('a') is None):
                values.append(fact.get_text()[0:-4]) # made changes to take the text if <a> is not found. Also removed the last 4 characters 'Save'
            else:
                values.append(fact.find('a').get_text())
        elif(index==6):
            values.append(fact.find('div', class_="username").get_text())
        else:
            values.append(fact.get_text())

    # Getting the items near the heading like car title, bid amount, # of bids, # of comments
    car_title = get_car_title(soup=car_soup)
    keys.append('Title')
    values.append(car_title)

    highest_bid = get_highest_bid(soup=car_soup)
    keys.append('Highest Bid')
    values.append(highest_bid)

    number_of_bids = get_number_of_bids(soup=car_soup)
    keys.append('Number of Bids')
    values.append(number_of_bids)

    number_of_comments = get_number_of_comments(soup=car_soup)
    keys.append('Number of Comments')
    values.append(number_of_comments)

    # Now converting the keys and values list into a dictionary
    car_facts = {}
    for key, value in zip(keys, values):
        car_facts[key] = value
    return car_facts
```

[125] ✓ 0.0s Python

After making these changes finally extracting all the details.

```python
cars_details = scrape_website_cars_and_bids()
```

[130] ✓ 12m 16.0s Python

```
Retrieving web page URL 'https://carsandbids.com'
Retrieving web page URL 'https://carsandbids.com/auctions/9nADLyZl/2022-ford-mustang-shelby-gt-h-coupe'
Retrieving web page URL 'https://carsandbids.com/auctions/9eYyqMzN/1998-toyota-hilux-surf-ssr-x-wide-27-4x4'
Retrieving web page URL 'https://carsandbids.com/auctions/KDbdMZV0/2011-bmw-m3-coupe'
Retrieving web page URL 'https://carsandbids.com/auctions/3LnYjm8y/2003-honda-s2000'
Retrieving web page URL 'https://carsandbids.com/auctions/364EvAbA/2009-infiniti-g37-convertible-sport'
Retrieving web page URL 'https://carsandbids.com/auctions/30w2yPyg/1992-bmw-525tds-sedan'
Retrieving web page URL 'https://carsandbids.com/auctions/KDZRnoZE/2023-rivian-r1s-adventure-edition'
Retrieving web page URL 'https://carsandbids.com/auctions/rJJkNRzw/2011-volkswagen-gti'
Retrieving web page URL 'https://carsandbids.com/auctions/r4d0Mq5V/2021-chevrolet-camaro-zl1-coupe'
Retrieving web page URL 'https://carsandbids.com/auctions/KPeag4L7/2006-bmw-330ci-zhp-convertible'
Retrieving web page URL 'https://carsandbids.com/auctions/9lZ0BedN/2004-nissan-350z-touring-coupe'
Retrieving web page URL 'https://carsandbids.com/auctions/9QPJ116n/2011-cadillac-cts-v-sedan'
Retrieving web page URL 'https://carsandbids.com/auctions/KV4pmY1Y/1998-mazda-roadster'
Retrieving web page URL 'https://carsandbids.com/auctions/KDVdJQNM/1993-toyota-supra-turbo'
Retrieving web page URL 'https://carsandbids.com/auctions/37XngjJ7/2004-porsche-cayenne-turbo'
Retrieving web page URL 'https://carsandbids.com/auctions/9XV0yPNo/2001-mercedes-benz-sl500'
Retrieving web page URL 'https://carsandbids.com/auctions/rNNVe614/2020-chevrolet-corvette-stingray-coupe'
Retrieving web page URL 'https://carsandbids.com/auctions/3oeyMdwW/2009-mercedes-benz-e550-sedan'
Retrieving web page URL 'https://carsandbids.com/auctions/3odqdxP2/2020-tesla-model-x-long-range-plus'
Retrieving web page URL 'https://carsandbids.com/auctions/3zmomokA/2019-ford-f-150-harley-davidson-edition-4x4'
Retrieving web page URL 'https://carsandbids.com/auctions/KdDA7xQQ/2019-mercedes-benz-e450-4matic-wagon'
Retrieving web page URL 'https://carsandbids.com/auctions/3vdkMjkB/2016-ford-mustang-gt-coupe'
Retrieving web page URL 'https://carsandbids.com/auctions/3q1lR7g5/1995-bmw-m3-coupe'
Retrieving web page URL 'https://carsandbids.com/auctions/KZqeYanA/1995-mitsubishi-delica-space-gear-4wd'
...
Retrieving web page URL 'https://carsandbids.com/auctions/rkJAbWNl/2012-bmw-550i'
Retrieving web page URL 'https://carsandbids.com/auctions/9nVDmyP2/1993-nissan-skyline-gts-25t-type-m-coupe'
Retrieving web page URL 'https://carsandbids.com/auctions/KYg2jXPx/2002-jaguar-x-type-25-sedan'
Retrieving web page URL 'https://carsandbids.com/auctions/30R824v1/2019-mercedes-amg-e63-s-sedan'
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```python
len(cars_details)
```

[131] ✓ 0.0s Python

```
107
```

```
cars_details
```

[132] ✓ 0.0s

```
[{'Make': 'Ford',
  'Model': 'Mustang',
  'Mileage': '750',
  'VIN': '1FA6P8CF5N5116542',
  'Title Status': 'Clean (AZ)',
  'Location': 'Estero, FL 33928',
  'Seller': 'Hertz_Collectible_Cars',
  'Engine': '5.0L Supercharged V8',
  'Drivetrain': 'Rear-wheel drive',
  'Transmission': 'Automatic (10-Speed)',
  'Body Style': 'Coupe',
  'Exterior Color': 'Rapid Red Metallic',
  'Interior Color': 'Ebony',
  'Seller Type': 'Private Party',
  'Title': '2022 Ford Mustang Shelby GT-H Coupe',
  'Highest Bid': '$70,000',
  'Number of Bids': '9',
  'Number of Comments': '30'},
 {'Make': 'Toyota',
  'Model': 'Hilux',
  'Mileage': '62,200',
  'VIN': 'RZN185-0031273',
  'Title Status': 'Clean (FL)',
  'Location': 'Lutz, FL 33549',
  'Seller': 'PrestigeImportsTampa',
...
  'Seller Type': 'Private Party',
  'Title': '2019 Mercedes-AMG E63 S Sedan',
  'Highest Bid': '$45,000',
  'Number of Bids': '9',
  'Number of Comments': '14'}]
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Converting List of Dictionaries into A Spreadsheet/ CSV

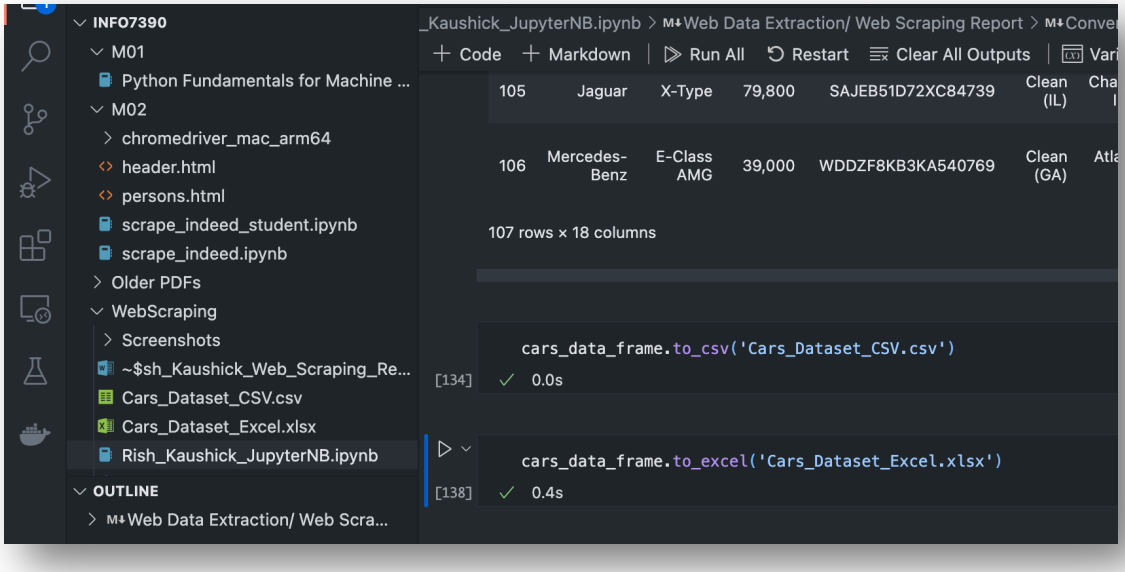
```
cars_data_frame = pd.DataFrame(cars_details)
cars_data_frame
```

[133] ✓ 0.0s Python

	Make	Model	Mileage	VIN	Title Status	Location	Seller	Engine	Drivetrain	Transmission	Body Style	Exterior Color
0	Ford	Mustang	750	1FA6P8CF5N5116542	Clean (AZ)	Estero, FL 33928	Hertz_Collectible_Cars	5.0L Supercharged V8	Rear-wheel drive	Automatic (10-Speed)	Coupe	Rapid Red Metallic
1	Toyota	Hilux	62,200	RZN185-0031273	Clean (FL)	Lutz, FL 33549	PrestigeImportsTampa	2.7L I4	4WD/AWD	Manual (5-Speed)	SUV/Crossover	Horizon Blue Metallic
2	BMW	E9X M3	32,900	WBSKG9C54BE368731	Clean (TX)	Spring, TX 77380	joemc	4.0L V8	Rear-wheel drive	Manual (6-Speed)	Coupe	Silverstone Metallic
3	Honda	S2000	26,400	JHMAP11483T006251	Clean (CA)	Paso Robles, CA 93446	LVSMotorcars	2.0L I4	Rear-wheel drive	Manual (6-Speed)	Convertible	Silverstone Metallic
4	Infiniti	G37	124,700	JNKCV66E39M722152	Clean (PA)	Allentown, PA 18104	autoplex	3.7L V6	Rear-wheel drive	Manual (6-Speed)	Convertible	Brilliant Silver Metallic
...	...	...	...	...	...	...	...	...	...	...	...	...
102	Land Rover	Range Rover	55,700	SALGV5SE3MA424870	Clean (NJ)	Hillsdale, NJ 07642	darricm	5.0L Supercharged V8	4WD/AWD	Automatic (8-Speed)	SUV/Crossover	Santorini Black Metallic
103	BMW	5 Series	85,000	WBAFR9C58CDV59348	Clean (CA)	Lake Elsinore, CA 92530	POWERBID	4.4L Turbocharged V8	Rear-wheel drive	Automatic (8-Speed)	Sedan	Space Gray Metallic
104	Nissan	R33 Skyline	88,100	ECR33007301	Clean (TX)	Austin, TX 78754	Viet_N999	2.5L Turbocharged I6	Rear-wheel drive	Manual (5-Speed)	Coupe	Gray
105	Jaguar	X-Type	79,800	SAJEB51D72XC84739	Clean (IL)	Champaign, IL 61820	jackren	2.5L V6	4WD/AWD	Manual (5-Speed)	Sedan	Phoenix
106	Mercedes-Benz	E-Class AMG	39,000	WDDZF8KB3KA540769	Clean (GA)	Atlanta, GA 30350	ggoodlife	4.0L Turbocharged V8	4WD/AWD	Automatic (9-Speed)	Sedan	Selenite Grey Metallic

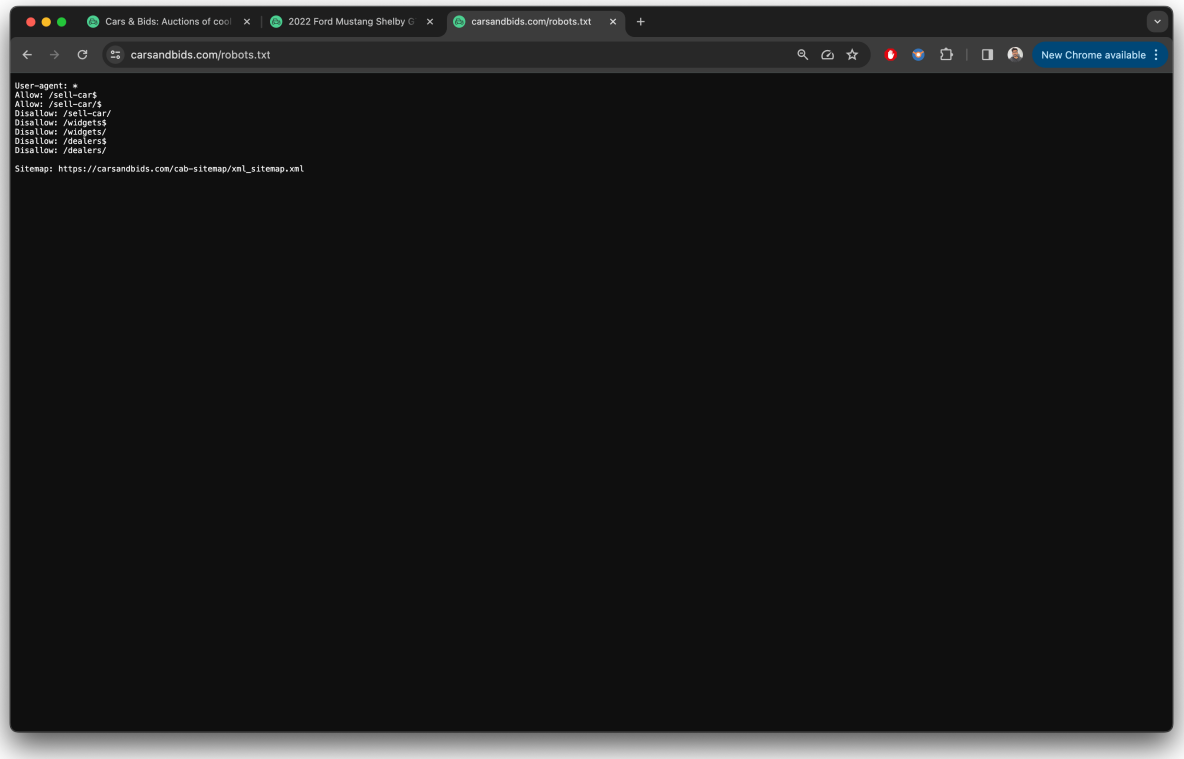
107 rows × 18 columns

We have created a Pandas data frame object from the list of dictionaries. Now we can further convert this into a spreadsheet/ csv format based on our needs as follows:



Disclaimer

This project is for research purposes and personal use only. Following is the Robots.txt file for the carsandbids website, which allows /sell-car.



Conclusion

In this project, I implemented the concept of web-scraping on the website 'https://carsandbids.com/'. From this webpage I extracted the links to 107 cars, and for each link I extracted additional 18 features each. In the end we have received a dataset of 107 rows and 18 columns which have been converted into CSV and Excel format.