**Introduction**

Our project included two tasks, the first being scraping the store details from united websites including 4 different stores, i.e. Amigos, United Supermarkets, Market Street and Albertsons. Our task was to extract store details which included Store ID, Store Name, Location Name and Services provided at the store. This information would further be processed by the ETL team to MongoDB.

Our second task was to extract barcodes from the united dataset, followed by extracting and parsing the product information of that particular barcode scraped from barcode lookup websites.

**Task 1 – Store Information**

Steps for scraping Store details

Parse the Json to extract store details

Get the Json Object

Scrape the website

Task 1: Scraping and Parsing United Website for Store Service Details.



Figure 1: Store Information Location on United Website

*First Step*

The first Step in this Task was scraping the united website: -

from selenium import webdriver

from time import sleep

browser = webdriver.Chrome("C:/webdrivers/chromedriver.exe")

url = 'https://www.unitedsupermarkets.com/rs/StoreLocator'

abc = browser.get(url)

sleep(15)

browser.find\_element\_by\_id('main-body').click()

a = browser.page\_source

ab = open("scraping.txt", 'w')

ab.write(a)

ab.close()

browser.close()

We used the selenium package to scrape the website using the chrome webdriver. Our store details were scripted in java script function in a key-value pair format. This was a bit tricky to workout with, as we initially attempted to scrape the data using Beautiful Soup html scraping, which failed. Scraping using selenium was challenging until we made major change in our code, by including the sleep (15) function. This allowed the website to load completely and hence we could capture entire function of javascript.



Figure 2: Javascript scarping using selenium

*Second Step*

Now that we have extracted the contents of the java script function, we had to extract the JSON data embedded. This is the second step in our task, i.e. extracting the JSON data which contain information about all the stores:

import json

import csv

with open('scraping.csv', 'w') as csvfile:

fieldnames = [""]

writer = csv.writer(csvfile,delimiter=',', quoting=csv.QUOTE\_MINIMAL)

filepath = 'scraping.txt'

with open(filepath) as fp:

line = fp.readline()

cnt = 1

while line:

line = fp.readline()

cnt += 1

if "var stores" in line:

line\_v = line

var = line\_v.find('=')

print(var)

var\_2 = line\_v.find(';')

print(var\_2)

json\_data = line\_v[var + 1 : var\_2 ]

data=json.loads(json\_data)

*Third Step*

Following is the python code used to parse the JSON object to extract the store details:

for d in data:

services=json.loads(data[0]['Services'])

for s in services['Services']:

print("StoreName "+":"+d['StoreName']+","+" StoreID "+":"+str(d['StoreID'])+","+"LocationName "+":"+d['LocationName']+","+" State "+":"+d['State']+","+" Zipcode "+":"+d['Zipcode']+","+"Service Name "+":"+s['Name']+","+"Service Value "+":"+s['Value'])

writer.writerow(["StoreName "+":"+d['StoreName']," StoreID "+":"+str(d['StoreID']),"LocationName "+":"+d['LocationName']," State "+":"+d['State']," Zipcode "+":"+d['Zipcode'],"Service Name "+":"+s['Name'],"Service Value "+":"+s['Value']])

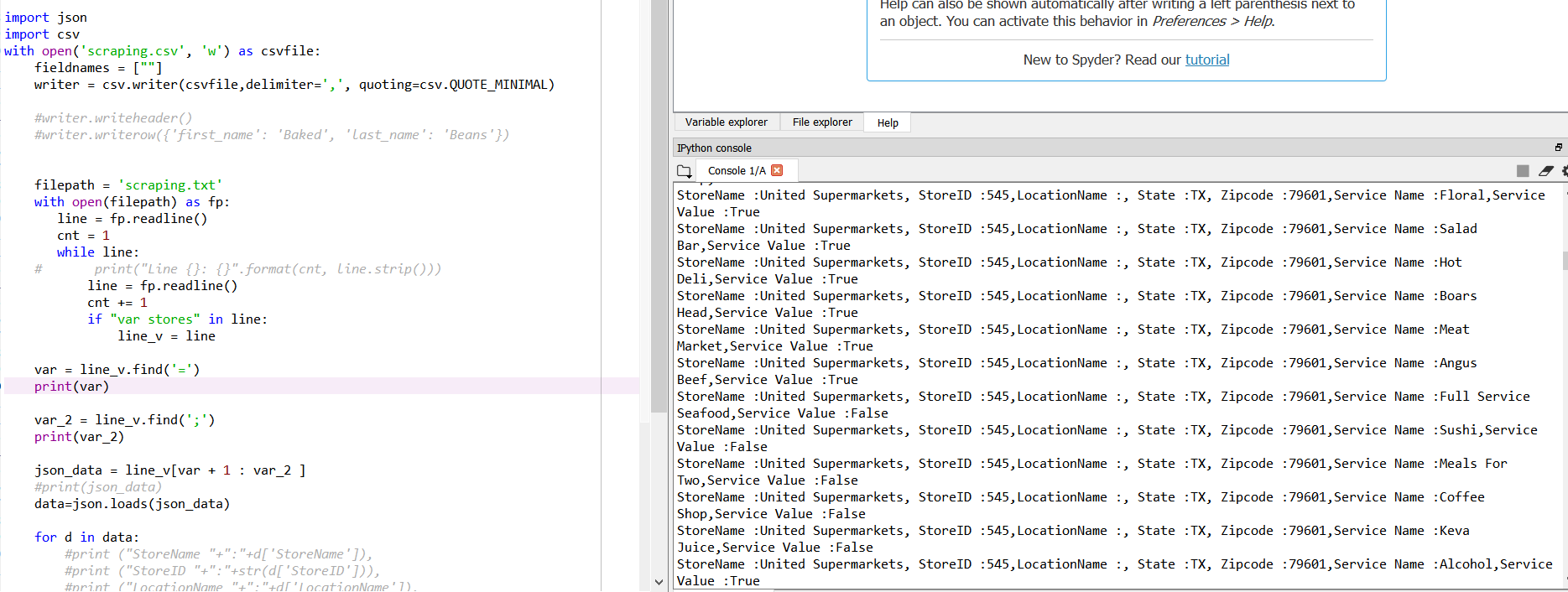


Figure 3: Parsing JSON data

We first extracted the var stores variable from the JavaScript which contain our json data for the service details of different stores. Then we used json.loads() function to convert json object, which is in key-value pair format to python dictionary. Using this function, we obtained the necessary details regarding the store, which we then we printed the required details like Store Name, StoreID, LocationName, State, Zipcode and the Service Name and Service values of the respective stores.

We printed all these information into a csv file which would further be processed by the ETL team to MongoDB.

These steps were repeated for all the stores of United – United Supermarkets, Albertson, Amigos, MarketStreet.

**Task 2 – Barcodes**

Steps involved for scraping barcode related product information

Parse the html text to extract barcode details

Get the html text

Scrape the website

*Task 2: Steps for Scraping and parsing the barcode lookup website to extract information about the product details*

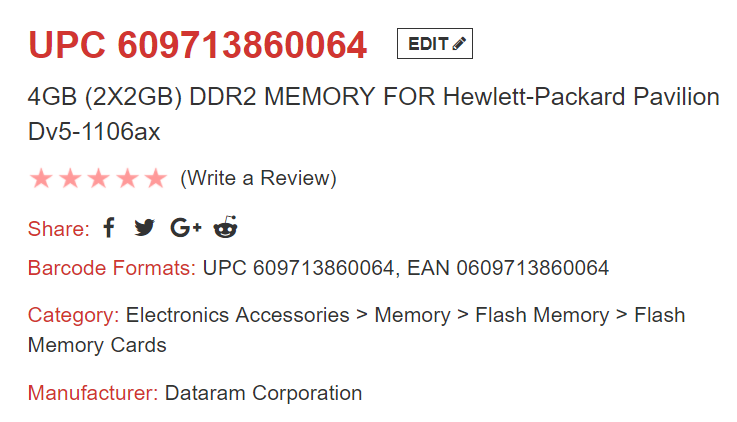


Figure 4: Barcode Related Product Information from Look Up Website

We continued using the selenium package as in task 1, to extract the product information corresponding to a barcode. Following is the python code used:

from selenium import webdriver

import re

import csv

with open('barcode.csv', 'wb') as csvfile:

spamwriter = csv.writer(csvfile, delimiter=',', quoting=csv.QUOTE\_MINIMAL)

list=["Barcode Formats","Category","Manufacturer"]

spamwriter.writerow(list)

browser = webdriver.Chrome("D:/webdriver/chromedriver.exe")

browser.get("https://www.barcodelookup.com/609713860064")

nav = browser.find\_element\_by\_id("body-container")

print(nav.text)

data=nav.text.split('\n')

#print "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

myData=[]

for i in data:

if re.compile('|'.join(list),re.IGNORECASE).search(i):

#print i

myData.append(i.encode('utf-8').split(":")[1])

print myData

spamwriter.writerow(myData)

Here we scraped the product details from the barcode lookup website. In this task, the required data was stored in html format, hence it was much simple than the first case. We made a list of the required data which we wanted to extract from all the text data. Then filtered it and store it into a csv file.

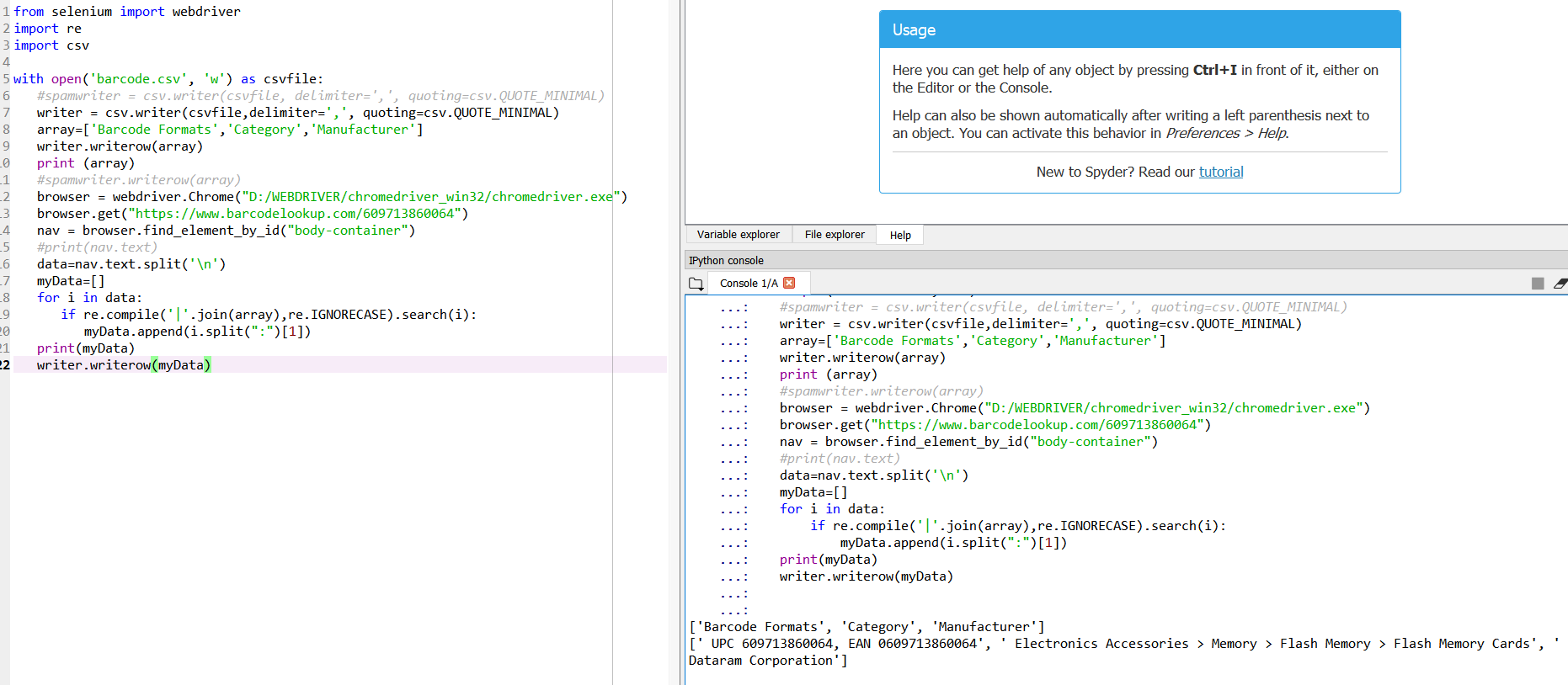


Figure 5: Extracting and Parsing Barcode Lookup Websites

**Code Details**

1. BI\_united\_1.py :- Creates scrapping.txt to capture JavaScript for united website.
2. BI\_united\_2.py :- Uses scrapping.txt as input and writes the service details of United into csv file called scrapping.csv in the format where there is a new row for every service.
3. BI\_united\_2\_new.py :- Uses scrapping.txt as input and writes the service details of United into csv file called scrapping\_new.csv in the format where there is a single row for all the service details of a particular store.
4. BI\_united\_3.py :- Creates scrapping\_2.txt to capture JavaScript for market street website.
5. BI\_united\_4.py :- Uses scrapping\_2.txt as input and writes the service details of market street into csv file called scrapping\_2.csv in the format where there is a new row for every service.
6. BI\_united\_4\_new.py :- Uses scrapping\_2.txt as input and writes the service details of market street into csv file called scrapping\_2\_new.csv in the format where there is a single row for all the service details of a particular store.
7. BI\_united\_5.py :- Creates scrapping\_3.txt to capture JavaScript for amigos website.
8. BI\_united\_6.py :- Uses scrapping\_3.txt as input and writes the service details of amigos into csv file called scrapping\_3.csv in the format where there is a new row for every service.
9. BI\_united\_6\_new.py :- Uses scrapping\_3.txt as input and writes the service details of amigos into csv file called scrapping\_3\_new.csv in the format where there is a single row for all the service details of a particular store.
10. BI\_united\_7.py :- Creates scrapping\_4.txt to capture JavaScript for Albertson website.
11. BI\_united\_8.py :- Uses scrapping\_4.txt as input and writes the service details of Albertson into csv file called scrapping\_4.csv in the format where there is a new row for every service.
12. BI\_united\_8\_new.py :- Uses scrapping\_4.txt as input and writes the service details of alberston into csv file called scrapping\_4\_new.csv in the format where there is a single row for all the service details of a particular store.
13. Barcodescraping.py :- Calls a function barcode\_call to extract information about UPC from barcodelookup.com and writes into a csv file called barcode.csv.