DATABASES
PSET 2

Derek Lee
Professor Sokolov
ECE-464

November 20, 2021

Example Queries

Query 1

Based on: `https://stackoverflow.com/a/31189502`

```
db.books.aggregate(
    {
        "$group": {
            "_id": {"Stars": "$Stars" },
            "count": { "$sum": 1 }
        }
    },
    {
        "$project": {
            "count": 1,
            "percentage": {
                "$multiply": [
                    { "$divide": [100, totalNum] },
                    "$count"
                ]
            }
        }
    }
)
```

```
databases> db.books.aggregate({ "$group": { "_id": { "Stars": "$Stars" }, "count": { "$sum": 1 } } },
{ "$project": { "count": 1, "percentage": { "$multiply": [{ "$divide": [100, totalNum] }, "$count"] }
} })
[
  { _id: { Stars: 'Three' }, count: 203, percentage: 20.3 },
  {
    _id: { Stars: 'Four' },
    count: 179,
    percentage: 17.900000000000002
  },
  { _id: { Stars: 'One' }, count: 226, percentage: 22.6 },
  { _id: { Stars: 'Five' }, count: 196, percentage: 19.6 },
  { _id: { Stars: 'Two' }, count: 196, percentage: 19.6 }
]
```

Figure 1: Percentages for each number of stars

Query 2

```
db.books.find({"Description": null})
```

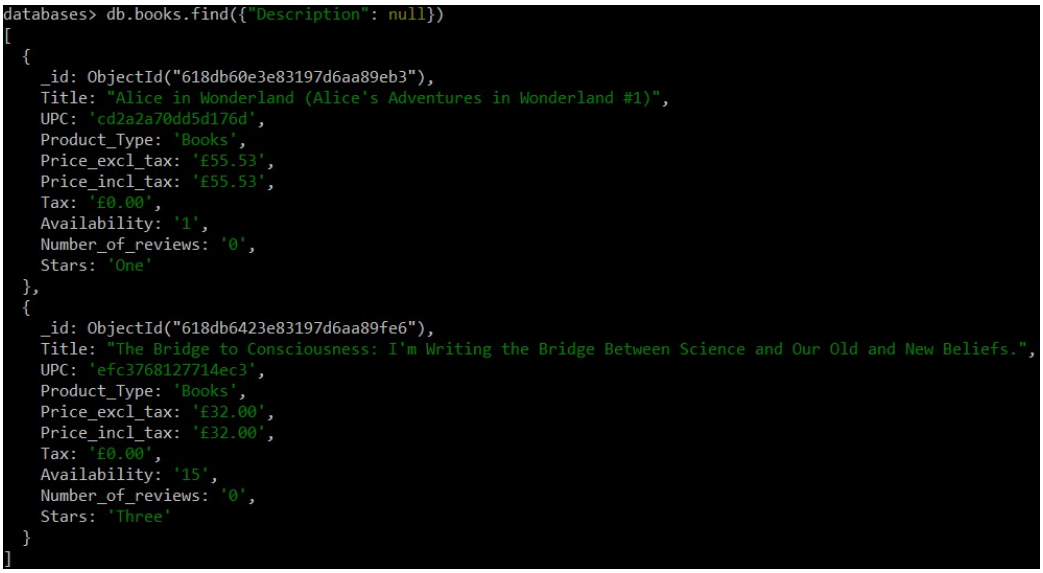

Figure 2: Books with no description

Query 3

```
db.books.find({"Description": {"$regex": ".*FBI.*"}})
```

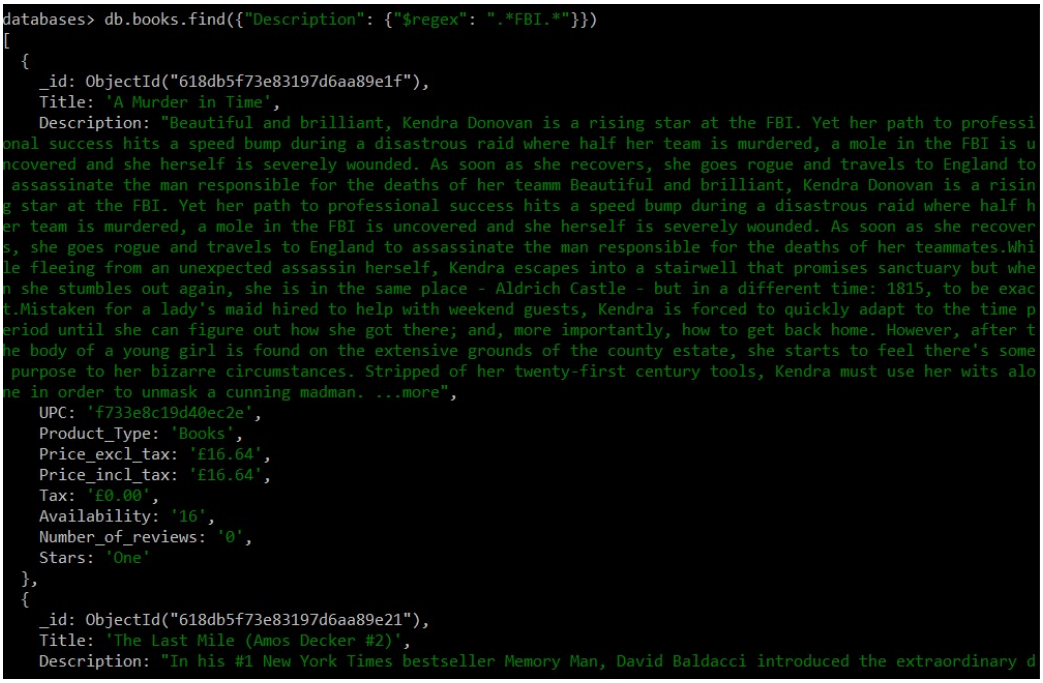

Figure 3: Books about the FBI

Query 4

```
db.books.aggregate(
    {
        "$project": {
            "Title": 1,
            "count": {
                "$convert": {
                    "input":"$Number_of_reviews", "to": "int"
                }
            }
        }
    },
    {
        "$match": {
            "count": {"$eq": 0}
```

```
        }
    }
)
```

```
databases> db.books.aggregate( { "$project": { "Title": 1, "count": { "$convert": { "input": "$Number_of_reviews", "to"
: "int" } } } }, { "$match": { "count": { "$eq": 0 } } })
[
  {
    _id: ObjectId("618db5f43e83197d6aa89e11"),
    Title: "It's Only the Himalayas",
    count: 0
  },
  {
    _id: ObjectId("618db5f43e83197d6aa89e12"),
    Title: 'Full Moon over Noahâ\x80\x99s Ark: An Odyssey to Mount Ararat and Beyond',
    count: 0
  },
  {
    _id: ObjectId("618db5f43e83197d6aa89e13"),
    Title: 'See America: A Celebration of Our National Parks & Treasured Sites',
    count: 0
  },
  {
    _id: ObjectId("618db5f43e83197d6aa89e14"),
    Title: 'Vagabonding: An Uncommon Guide to the Art of Long-Term World Travel',
    count: 0
  },
  {
    _id: ObjectId("618db5f43e83197d6aa89e15"),
    Title: 'Under the Tuscan Sun',
    count: 0
  },
```

Figure 4: Books with no reviews

Code

```python
from typing import Any, DefaultDict

import requests
import re
import time
import pymongo

from pymongo import MongoClient
from collections import defaultdict
from bs4 import BeautifulSoup

# Used to get DB connection
```

```python
from db_info import Info


url_to_scrape = 'http://books.toscrape.com/'


def get_website_data(url: str, verbose: bool = False) -> BeautifulSoup:
    """Sends an HTTP GET request to the specified URL and returns the response inside a BeautifulSoup object."""
    if verbose:
        print("Scraping data from: ", file_url)
    response = requests.get(url)
    return BeautifulSoup(response.text, 'html.parser')


def get_num_stars(book_soup: BeautifulSoup) -> str:
    # Need to do .next_sibling twice b/c it will grab white text
    star_content = book_soup.find('p', {'class': 'instock availability'}).next_sibling.next_sibling

    # Finds the first match and extracts the string matching .*
    # E.g. <p class="star-rating Three"> -----> Three
    return re.search('<p class="star-rating (.*)">|$', str(star_content)).group(1)


def get_book_data(file_url: str, book_info: BeautifulSoup) -> DefaultDict[str, Any]:
    # Store product information as a dictionary
    book_information_dict = defaultdict(str)
    book_information_dict['Title'] = book_info['title']

    # Go to book product page
    book_url = file_url + '/../' + book_info['href']
```

```
    book_soup = get_website_data(book_url)
    book_soup = book_soup.find('article', {'class':
    ↪  'product_page'})

    # Some books won't have descriptions
    try:
        # Need to do .next_sibling twice b/c it will grab
        ↪  white text
        book_description = book_soup.find('div', {'id':
        ↪  'product_description'}).next_sibling.next_sibling.get_text(strip=True)
        book_information_dict['Description'] = book_description
    except:
        pass

    book_information_rows =
    ↪  book_soup.find('table').find_all('tr')
    for row in book_information_rows:
        row_name = row.find('th').get_text(strip=True)
        row_data = row.find('td').get_text(strip=True)
        book_information_dict[row_name] = row_data

    # Store other information
    book_information_dict['Stars'] = get_num_stars(book_soup)

    # Finds the first match and extracts the string matching
    ↪  .*
    # E.g. In stock (19 available) -----> 19
    book_information_dict['Availability'] = re.search('In stock
    ↪  \((.*) available\)|$',
    ↪  book_information_dict['Availability']).group(1)

    # Drop the extra character at the beginning of the string
    book_information_dict['Price (excl. tax)'] =
    ↪  book_information_dict['Price (excl. tax)'][1:]
    book_information_dict['Price (incl. tax)'] =
    ↪  book_information_dict['Price (incl. tax)'][1:]
    book_information_dict['Tax'] =
    ↪  book_information_dict['Tax'][1:]
```

```
71
72      return book_information_dict
73
74
75  def post_process(info: DefaultDict[str, Any]) ->
    ↪  DefaultDict[str, Any]:
76      """Replace spaces with underscores and removes periods
        ↪  and parentheses."""
77      chars_to_replace = {
78          ' ': '_',
79          '.': '',
80          '(': '',
81          ')': ''
82      }
83      func = lambda x:
        ↪  x.translate(str.maketrans(chars_to_replace))
84      return {func(k):v for k,v in info.items()}
85
86
87  if __name__ == '__main__':
88      # Get database info
89      client = MongoClient(Info.connection_string,
        ↪  authSource=Info.db_name)
90      db_name = client[Info.db_name]
91      collection_name = db_name[Info.collection_name]
92
93      # Scrape main website
94      main_soup = get_website_data(url_to_scrape)
95
96      # Get links to book categories
97      book_categories = main_soup.find('div', {'class':
        ↪  'side_categories'})
98
99      # Iterate through each book category
100     # Skip the first one, which contains all books
101     for link in book_categories.find_all('a', href=True)[1:]:
102         file_url = url_to_scrape + link['href']
103
```

```
104         # Iterate through each page
105         has_more_pages = True
106         while has_more_pages:
107             category_soup = get_website_data(file_url,
                ↪  verbose=True)
108 
109             # Iterate through each book
110             for book in
                ↪  category_soup.find('body').find('ol').find_all('article',
                ↪  {'class': 'product_pod'}):
111                 book_info = book.find('h3').find('a',
                    ↪  href=True)
112                 book_info_dict = get_book_data(file_url,
                    ↪  book_info)
113                 book_info_dict = post_process(book_info_dict)
114 
115                 # Insert into the database collection
116                 collection_name.insert_one(book_info_dict)
117 
118             # Check if there is another page in the category
119             try:
120                 next_url =
                    ↪  category_soup.find('section').find('ol',
                    ↪  {'class':
                    ↪  'row'}).next_sibling.next_sibling.find('ul',
                    ↪  {'class': 'pager'}).find('li', {'class':
                    ↪  'next'}).find('a', href=True)['href']
121                 file_url += '/../' + next_url
122                 has_more_pages = True
123                 time.sleep(0.1)
124             except:
125                 print("No more pages.\n")
126                 has_more_pages = False
127 
128         time.sleep(0.5)
129 
130     print("Finished scraping.")
```