

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})
```

```
databases> db.books.find({"Description": null})
[
  {
    _id: ObjectId("618db60e3e83197d6aa89eb3"),
    Title: "Alice in Wonderland (Alice's Adventures in Wonderland #1)",
    UPC: 'cd2a2a70dd5d176d',
    Product_Type: 'Books',
    Price_excl_tax: '£55.53',
    Price_incl_tax: '£55.53',
    Tax: '£0.00',
    Availability: '1',
    Number_of_reviews: '0',
    Stars: 'One'
  },
  {
    _id: ObjectId("618db6423e83197d6aa89fe6"),
    Title: "The Bridge to Consciousness: I'm Writing the Bridge Between Science and Our Old and New Beliefs.",
    UPC: 'efc3768127714ec3',
    Product_Type: 'Books',
    Price_excl_tax: '£32.00',
    Price_incl_tax: '£32.00',
    Tax: '£0.00',
    Availability: '15',
    Number_of_reviews: '0',
    Stars: 'Three'
  }
]
```

Figure 2: Books with no description

Query 3

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

```

databases> db.books.find({"Description": {"$regex": ".*FBI.*"}})
[
  {
    _id: ObjectId("618db5f73e83197d6aa89e1f"),
    Title: 'A Murder in Time',
    Description: "Beautiful and brilliant, Kendra Donovan is a rising star at the FBI. Yet her path to professional success hits a speed bump during a disastrous raid where half her team is murdered, a mole in the FBI is uncovered and she herself is severely wounded. As soon as she recovers, she goes rogue and travels to England to assassinate the man responsible for the deaths of her teammates. Beautiful and brilliant, Kendra Donovan is a rising star at the FBI. Yet her path to professional success hits a speed bump during a disastrous raid where half her team is murdered, a mole in the FBI is uncovered and she herself is severely wounded. As soon as she recovers, she goes rogue and travels to England to assassinate the man responsible for the deaths of her teammates. While fleeing from an unexpected assassin herself, Kendra escapes into a stairwell that promises sanctuary but when she stumbles out again, she is in the same place - Aldrich Castle - but in a different time: 1815, to be exact. Mistaken for a lady's maid hired to help with weekend guests, Kendra is forced to quickly adapt to the time period until she can figure out how she got there; and, more importantly, how to get back home. However, after the body of a young girl is found on the extensive grounds of the county estate, she starts to feel there's some purpose to her bizarre circumstances. Stripped of her twenty-first century tools, Kendra must use her wits alone in order to unmask a cunning madman. ...more",
    UPC: 'f733e8c19d40ec2e',
    Product_Type: 'Books',
    Price_excl_tax: '£16.64',
    Price_incl_tax: '£16.64',
    Tax: '£0.00',
    Availability: '16',
    Number_of_reviews: '0',
    Stars: 'One'
  },
  {
    _id: ObjectId("618db5f73e83197d6aa89e21"),
    Title: 'The Last Mile (Amos Decker #2)',
    Description: "In his #1 New York Times bestseller Memory Man, David Baldacci introduced the extraordinary d

```

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

```

1 from typing import Any, defaultdict
2
3 import requests
4 import re
5 import time
6 import pymongo
7
8 from pymongo import MongoClient
9 from collections import defaultdict
10 from bs4 import BeautifulSoup
11
12 # Used to get DB connection

```

```

13 from db_info import Info
14
15
16 url_to_scrape = 'http://books.toscrape.com/'
17
18
19 def get_website_data(url: str, verbose: bool = False) ->
    ↳ BeautifulSoup:
20     """Sends an HTTP GET request to the specified URL and
    ↳ returns the response inside a BeautifulSoup
    ↳ object."""
21     if verbose:
22         print("Scraping data from: ", file_url)
23     response = requests.get(url)
24     return BeautifulSoup(response.text, 'html.parser')
25
26
27 def get_num_stars(book_soup: BeautifulSoup) -> str:
28     # Need to do .next_sibling twice b/c it will grab white
    ↳ text
29     star_content = book_soup.find('p', {'class': 'instock
    ↳ availability'}).next_sibling.next_sibling
30
31     # Finds the first match and extracts the string matching
    ↳ .*
32     # E.g. <p class="star-rating Three"> -----> Three
33     return re.search('<p class="star-rating (.*)">|$',
    ↳ str(star_content)).group(1)
34
35
36 def get_book_data(file_url: str, book_info: BeautifulSoup) ->
    ↳ DefaultDict[str, Any]:
37     # Store product information as a dictionary
38     book_information_dict = defaultdict(str)
39     book_information_dict['Title'] = book_info['title']
40
41     # Go to book product page
42     book_url = file_url + '/../' + book_info['href']

```

```

43 book_soup = get_website_data(book_url)
44 book_soup = book_soup.find('article', {'class':
↳ 'product_page'})
45
46 # Some books won't have descriptions
47 try:
48     # Need to do .next_sibling twice b/c it will grab
↳ white text
49     book_description = book_soup.find('div', {'id':
↳ 'product_description'}).next_sibling.next_sibling.get_text(strip=True)
50     book_information_dict['Description'] = book_description
51 except:
52     pass
53
54 book_information_rows =
↳ book_soup.find('table').find_all('tr')
55 for row in book_information_rows:
56     row_name = row.find('th').get_text(strip=True)
57     row_data = row.find('td').get_text(strip=True)
58     book_information_dict[row_name] = row_data
59
60 # Store other information
61 book_information_dict['Stars'] = get_num_stars(book_soup)
62
63 # Finds the first match and extracts the string matching
↳ .*
64 # E.g. In stock (19 available) -----> 19
65 book_information_dict['Availability'] = re.search('In stock
↳ \((.*) available\)|$',
↳ book_information_dict['Availability']).group(1)
66
67 # Drop the extra character at the beginning of the string
68 book_information_dict['Price (excl. tax)'] =
↳ book_information_dict['Price (excl. tax)'][1:]
69 book_information_dict['Price (incl. tax)'] =
↳ book_information_dict['Price (incl. tax)'][1:]
70 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:
84         ↳ x.translate(str.maketrans(chars_to_replace))
85     return {func(k):v for k,v in info.items()}
86
87 if __name__ == '__main__':
88     # Get database info
89     client = MongoClient(Info.connection_string,
90         ↳ authSource=Info.db_name)
91     db_name = client[Info.db_name]
92     collection_name = db_name[Info.collection_name]
93
94     # Scrape main website
95     main_soup = get_website_data(url_to_scrape)
96
97     # Get links to book categories
98     book_categories = main_soup.find('div', {'class':
99         ↳ 'side_categories'})
100
101     # Iterate through each book category
102     # Skip the first one, which contains all books
103     for link in book_categories.find_all('a', href=True)[1:]:
104         file_url = url_to_scrape + link['href']

```



```

104     # Iterate through each page
105     has_more_pages = True
106     while has_more_pages:
107         category_soup = get_website_data(file_url,
108             ↪ verbose=True)
109
110         # Iterate through each book
111         for book in
112             ↪ category_soup.find('body').find('ol').find_all('article',
113             ↪ {'class': 'product_pod'}):
114             book_info = book.find('h3').find('a',
115                 ↪ href=True)
116             book_info_dict = get_book_data(file_url,
117                 ↪ book_info)
118             book_info_dict = post_process(book_info_dict)
119
120             # Insert into the database collection
121             collection_name.insert_one(book_info_dict)
122
123             # Check if there is another page in the category
124             try:
125                 next_url =
126                 ↪ category_soup.find('section').find('ol',
127                 ↪ {'class':
128                 ↪ 'row'}).next_sibling.next_sibling.find('ul',
129                 ↪ {'class': 'pager'}).find('li', {'class':
130                 ↪ 'next'}).find('a', href=True)['href']
131                 file_url += '/../' + next_url
132                 has_more_pages = True
133                 time.sleep(0.1)
134             except:
135                 print("No more pages.\n")
136                 has_more_pages = False
137
138         time.sleep(0.5)
139
140     print("Finished scraping.")

```