Management and Pre-Processing Assessment

In this assessment you will go through the process of obtaining data, cleaning it, and then querying it from a database. We are using data about food hygiene from UK open data. The data stored is a copy of the official data.

To provide a solution for each task, you might like to do the practice exercises: "HTML and Page Scraping", and "Using MongoDB to Retrieve Information" first.

You may validate your answers by clicking "Validate" on the "Assignments" tab for this exercise. These will be done automatically, using the tests in this notebook. The final submission will be both machine checked and human marked.

Question 0: Setup [1 mark]

Run the following cell to import the core dependencies required for this exercise

In [5]:

```
# You don't need to write anything here
import requests
import json
from bs4 import BeautifulSoup
from urllib.robotparser import RobotFileParser
from nose.tools import assert_equal, assert_raises
from pymongo import MongoClient
```

In [6]:

```
# Check that the required libraries and functions have been imported
# You don't need to write anything here

try:
    imports = [requests, BeautifulSoup, RobotFileParser, assert_equal, assert_raises
except NameError as e:
    print(e)
    raise AssertionError('You appear to be missing one of the required libraries or
assert True
print('Successfully imported libraries and functions')
```

```
Successfully imported libraries and functions
```

Question 1: Web APIs and Page Scraping

Question 1(a) [2 marks]

Write a function `get_establishment_by_id` which accepts a parameter `id`, and returns the name of that business as a string. It should obtain the data from the food hygeine ratings API (http://ratings.food.gov.uk/open-data/en-GB), and use version 2 of the API.

- You may **assume that the ID exists**
- You should use the **`Establishments`** endpoint

To complete this question you may wish to look at the information found here (http://docs.python-requests.org/en/master/user/quickstart/).

N.B. The version of requests installed on the server is relatively recent. In a previous update, there was a breaking change which meant that only strings or byte-like objects could be passed as headers. As such, if you wish to pass an integer, you will have to do it as e.g., `{'header_name': '4321'}`.

Hint: Week 3, Guided Exercise 2, Scraping With Requests and Beautiful Soup

In [104]:

```
def get_establishment_by_id(id):
    # YOUR CODE HERE
    url = 'https://api.ratings.food.gov.uk/Establishments/%d' %id
    headers = { "x-api-version" : "2" }
    r = requests.get(url , headers = headers )
    return r.json()["BusinessName"]

    raise NotImplementedError()
```

In [105]:

```
assert_equal(get_establishment_by_id(990000), '1N1 Fashion N Pizza')
assert_equal(get_establishment_by_id(511819), 'Star Karahi')
assert_equal(get_establishment_by_id(692630), 'Baldiesburn Bed & Breakfast')
print('All tests successfully passed')
```

```
All tests successfully passed
```

Question 1(b) [2 marks]

Data stored at http://138.68.148.20/ (http://138.68.148.20/), in HTML format will be used for this question. Use the Python `requests` library for any requests to the server:

Write a function `check_robots`, which accepts a **parameter** `url` which tells you whether the server at http://138.68.148.20/ (http://138.68.148.20/) will permit you to scrape that page.

Hint: Week 3, Guided Exercise 2, Robots.txt

In [7]:

```
def check_robots(url):
    """
    Use the RobotFileParser to check if a page on the server can be visited
    """
    # YOUR CODE HERE
    rp = RobotFileParser()
    rp.set_url('http://138.68.148.20//robots.txt')
    rp.read()
    return rp.can_fetch("*", url)


    raise NotImplementedError()
# You don't need to write anything here
```

In [8]:

```
# Testing whether your code works correctly.
# You don't need to write anything here
# Confirm an allowed page returns True
assert check_robots('http://138.68.148.20/index.html')
# Confirm a disallowed page returns False
assert not check_robots('http://138.68.148.20/data/scotland/glasgow_city')
print('Passed all the tests')
```

```
Passed all the tests
```

Question 1(c) [3 marks]

Write a function which takes a URL as a **parameter**, and reads the **XML** on the page it goes to. The function should **return** a `dict` with the amount of records in `EstablishmentCollection`, and the name of the first business.
HINT: You can use `BeautifulSoup` for parsing XML as well as HTML. The function should behave as follows:

- The function should use the Python **`requests`** library.
- **If** the page is banned by robots.txt, then it should not be visited, and should return **`None`**
- **If** the page does not return a **200 status code** in response, then it should not attempt to parse the result, and return **`None`**
- If the page is an **XML** file, it should return a dict in the following format: `{'first_business': 'business name', 'amount_of_records': 1234}`

N.B. The order of a Python `dict` is not guaranteed, so we will not take into account which key appears first.

Hint: Week 3, Guided Exercise 2, Parsing HTML - Scraping with Requests and Beautiful Soup

In [157]:

```
def parse_xml(url):
    """
    This function should parse the XML file, for example http://138.68.148.20/west_r
    NOTE: Unlike for HTML, you need to use 'xml' as the second parameter for Beautif
    You may use any of Python's core libraries, or other libraries installed if you
    """

    if check_robots(url) is True: #2번
        pass
    else:
        return None

    res = requests.get(url)                          #1번
    sc = res.status_code

    if sc is not 200: ##r.status_code가 밴되었다면      #3번
        return None

    soup = BeautifulSoup(res.content, 'xml') #res에서 얻은 url을 xml 파일 가져오기

    j = soup.ItemCount.string
    k = soup.BusinessName.string

    dic = { 'amount_of_records' : int(j), 'first_business' : k}
    return dic

    raise NotImplementedError()
```

In [158]:

```
# You don't need to write anything here
# Confirm that the function calls the check_robots function
tmp_check_robots = check_robots
del check_robots

try:
    parse_xml('http://138.68.148.20/data/west_midlands/cannock_chase')
except NameError:
    pass
else:
    raise AssertionError("get_urls does not call check_robots")
finally:
    check_robots = tmp_check_robots

# TEST NOT VISITING PAGES PROHIBITED BY ROBOTS
# THIS SHOULD NOT CALL requests.get

tmp_requests = requests
del requests

try:
    parse_xml('http://138.68.148.20/data/scotland/glasgow_city')
    parse_xml('http://138.68.148.20/data/scotland/clackmannanshire')
except NameError:
    raise AssertionError("The function should not be using requests on this URL")
finally:
    requests = tmp_requests
# TEST OUTPUT RESPONSE
assert_equal(parse_xml('http://138.68.148.20/data/west_midlands/cannock_chase'),
             {'amount_of_records': 731, 'first_business': '1st Choice Pizza/Fish & C
assert_equal(parse_xml('http://138.68.148.20/data/wales/swansea'),
                       {'amount_of_records': 1700, 'first_business': '360 Beach and
# TEST HANDLING 404
assert_equal(parse_xml('http://138.68.148.20/data/calderdale'), None)

print('All test successfully passed')
```

```
All test successfully passed
```

Question 2: Retrieving Data from MongoDB

We will assume that you have successfully cleaned the data, and have stored it in the MongoDB database. Using the following PyMongo configuration, answer the following questions about the data:

In [7]:

```
# These are the credentials to connect to the database
# You don't need to write anything here, but you need to run this cell

client = MongoClient('mongodb://cpduser:M13pV5woDW@mongodb/health_data')
db = client.health_data
```

Question 2(a) [1 mark]

Write a **function** `get_count`, which takes a PyMongo collection object as a parameter and **returns** the amount of businesses in the collection.

Hint: Week 3, Guided Exercise 4, Using MongoDB to Retrieve Information

In [313]:

```
def get_count(collection):
    """
    Return an integer which gives the amount of unique businesses in the given colle
    """
    # YOUR CODE HERE
    count = 0
    for c in collection.find():
        count += 1
    return(count)


    raise NotImplementedError()
```

In [314]:

```
# You don't need to write anything here
assert_equal(get_count(db.uk), 511819)
assert_equal(get_count(db.swansea), 1700)
assert_equal(get_count(db.westminster), 4315)
assert_equal(get_count(db.newcastle_upon_tyne), 2308)
print('Passed all the tests')
```

```
Passed all the tests
```

Question 2(b) [3 marks]

Write a **function** `get_rating_value_percentage` which **returns** the **percentage** of businesses which were awarded an overall `RatingValue` of 5. The function should accept a parameter `collection` of type `Collection`, for which it should return the percentage as a **float** between 0 and 1.

Hint: Week 3, Guided Exercise 4, Cursors

In [366]:

```
def get_rating_value_percentage(collection):
    """
    Return a float between 0 and 1 of the amount with a RatingValue of 5
    """
    count2 = 0.0
    for c in collection.find({'RatingValue': 5}):
        count2 += 1 #5 값

    child = count2
    parent = get_count(collection)


    return child/parent

    raise NotImplementedError()
```

In [367]:

```
# You don't need to write anything here
assert_equal(get_rating_value_percentage(db.uk), 0.5287240215779406)
assert_equal(get_rating_value_percentage(db.swansea), 0.6688235294117647)
assert_equal(get_rating_value_percentage(db.westminster), 0.4600231749710313)
assert_equal(get_rating_value_percentage(db.newcastle_upon_tyne), 0.596620450606585
print('Passed all the tests')
```

```
Passed all the tests
```

Question 2(c) [3 marks]

Write a **function** `get_no_geocode` which will find establishments with region Scotland which do not have a `Geocode` recorded. The parameter `establishment_type` is a string, which will indicate the type of establishment to search for. All queries should be run on the `uk` collection.

The function should **return** a PyMongo **`Cursor`** object, with only the following fields:

- `BusinessName`, `BusinessType`, and `LocalAuthorityName`.
- `_id` should not be included

Hint: Week 3, Guided Exercise 4, Returning Part of a Document

In [12]:

```
def get_no_geocode(establishment_type):
    # YOUR CODE HERE
    for c in db.uk.find():
        resultt = db['uk'].find({'Geocode': None, 'BusinessType' : establishment_ty
                              {'BusinessName' : 1, 'BusinessType' : 1, 'LocalAuthorityN

    return resultt
    raise NotImplementedError()
```

In [473]:

```
# You don't need to write anything here

cursor = get_no_geocode('Restaurant/Cafe/Canteen' )
for cur in cursor:

    assert '_id' not in cur
    assert 'BusinessType' in cur
    assert_equal(cur['BusinessType'], 'Restaurant/Cafe/Canteen')
    assert 'BusinessName' in cur
    assert 'LocalAuthorityName' in cur

assert_equal(len(list(get_no_geocode('Takeaway/sandwich shop'))), 405)
assert_equal(len(list(get_no_geocode('Retailers - other'))), 1079)
print('Passed all the tests')
```

```
Passed all the tests
```

Question 2(d) [5 marks]

Question 2(d) [5 marks]

What was the earliest and latest dates that an inspection was carried out? Write a **function** which returns a dict in the form `{'earliest_date': 'YYYY-MM-DD', 'latest_date': 'YYYY-MM-DD'}`.

Hint: Week 3, Guided Exercise 4, MongoDB Aggregation Framework

In [525]:

```
from datetime import datetime
def get_earliest_and_latest_dates(collection):
    # YOUR CODE HERE

    coll = collection.aggregate(
        [
            {"$group": { "_id" : "latest_date", "RatingDate" : {'$max' : "$RatingDat
        ]
    )
    for dot in coll:
        ld = datetime.date(dot['RatingDate'])

    coll2 = collection.aggregate(
        [
            {"$group": { "_id" : "earliest_date", "RatingDate" : {'$min' : "$RatingD
        ]
    )
    for dot in coll2:
        ed = datetime.date(dot['RatingDate'])

    dic2 = {'earliest_date' : str(ed), 'latest_date' : str(ld)}
    return dic2

    raise NotImplementedError()
```

In [526]:

```
# You don't need to write anything here
assert_equal(get_earliest_and_latest_dates(db.uk),{'earliest_date': '1989-01-01', 'l
assert_equal(get_earliest_and_latest_dates(db.swansea),{'earliest_date': '2010-10-0
assert_equal(get_earliest_and_latest_dates(db.westminster),
                {'earliest_date': '1999-01-27', 'latest_date': '2016-09-13'})
assert_equal(get_earliest_and_latest_dates(db.newcastle_upon_tyne),
                {'earliest_date': '2005-07-08', 'latest_date': '2016-09-06'})
print('Passed all the tests')
```

```
Passed all the tests
```

Question 3 Exploring and fixing data [5 marks]

During this week Huw has talked about issues which may arise when integrating data. For this task, consider the data described in this notebook, and any other source you wish.

- Provide two concise examples of possible issues, and their mitigation in relation to these data
- Each example should be approximately one paragraph

1 우선, 데이터 통합시 키워드 통일의 문제가 생길 수 있다. 예를 들어 영국을 표기를 할때 UK를 UnitedKingdom로 표기된 데이터가 있어서 서로 데이터로 간주할 수 있으므로, 이를 완화시키기 위하여 한쪽의 데이터(UK)로 통일시켜 하나로 만들어 주는 것이 좋다.

2 두번째로 데이터가 편향되어 있지 않아야 한다. 예를 들어 UK와 USA 데이터를 비교하기 위하여 데이터를 통합 시킬 때, UK의 데이터 양은 충분히 많지만 내가 추출한 USA 데이터 양이 충분히 확보가 되지 않는다면 데이터가 통합되는 것이 의미가 없으므로 이를 완화시키기 위하여 사전에 충분히 양질의 데이터가 있는지 확인해보아야 한다.

```
In [ ]:
```