SIT742 Modern Data Science

Assignment 1

Student name: Tithra Chap

Student ID: 220051603

Date: 16-April-2021

Table of Contents

Part I - Tulip Hotel Web Logs Exploratory Data Analysis

Hotel TULIP a five-star hotel located at Deakin University, and its CIO Dr Bear Guts has asked the Team-SIT742 team to analyse the weblogs files. As an employee for Hotel Tulip, working in the Information Technology Division, it is required to prepare a set of documentation for Team-SIT742 to allow them to understand the data being dealt with. Throughout this report, some source codes are to explore the weblog, which afterwards the information is presented to Dr Bear Guts in the format of a report.

1. Data ETL

1.1. Data Loading

Fill the DataDictionary.xlsx with discovery from the result of 1.1 Data Loading from your notebook.

1.1.1. Dataset Description

Please add a screenshot of Dataset Description from your DataDictionary.xlsx.

DATA SET NAME	*HTWebLog_p1.zip*
DATA SIZE	*1,012,671,728 bytes (indluded 128 bytes of index column)*
DATE OF RELEASE	*from 01-Nov-2006 to 28-Feb-2007*
NO. OF FILES	*120*
NO. OF ATTRIBUTES	*15*
NO. OF DATA RECORDS	*8438930 (included NaN records)*
DATA SOURCE PROVIDER	*Information Technology Division, Hotel TULIP*
DATA PRIVACY	*Exclusively for SIT742 educational purpose only. Further distribution is not permitted.*
NOTES	2 of the files (i.e. ex061228 & ex061229) contains spaces at the beginning of the files which results in additional 2 records of NaN. This issue can be resolved after removing NaN records from the dataset.

Prepared by:	*Tithra Chap*
Point of Contact:	*tchap@deakin.edu.au*
Team Members:	*[list team members if applicable]*

Data Type Name Convention	Main Type	Subtype
MD	Metric Discrete	BIN - Binary
MC	Metric Continous	DATE - Date/time
		CURR - Currency
CO	Categorical Ordinal	
CN	Categorical Nomin	DI - Dichotomous
		STR - Free String
		ID - Identification
		URL - links such as URLs
		ADDR - Address

1.1.2. Attribute Dictionary

Please add a screenshot of Attribute Dictionary from your DataDictionary.xlsx.

Attribute Name	Data Type	Data Subtype	Description	Examples	Additional Notes
date	Metric Continous	DATE - date	The date on which the activity occurred	2006-11-28	This log file entry was recorded on Nov 28, 2006
time	Metric Continous	DATE - time	The time, in coordinated universal time (UTC), at which the activity occurred	17:42:15	This log file entry was recorded at 5:42 P.M. UTC. Entries are recorded to the log file when the send completion for the last IIS send occurs
s-sitename	Categorical Nominal	STR - Free String	The Internet service name and instance number that was running on the client	W3SVC1	
s-ip	Categorical Nominal	ID - Identification	The IP address of the server on which the log file entry was generated	172.22.255.255	The IP address of the client.
cs-method	Categorical Nominal	STR - Free String	The requested action, for example, a GET method	GET	The user issued a GET, or download, command
cs-uri-stem	Categorical Nominal	URL - links such as URLs	The target of the action, for example, Default.htm	/images/picture.jpg	The user wanted to download the picture.jpg file from the Images folder
cs-uri-query	Categorical Nominal	STR - Free String	The query, if any, that the client was trying to perform. A Universal Resource Identifier (URI) query is necessary only for dynamic	lang=en-us	The URI query requests for web content in English language
s-port	Metric Discrete	BIN - Binary	The server port number that is configured for the service	80	The server port 80 is used for the service
cs-username	Categorical Nominal	ID - Identification	The name of the authenticated user who accessed your server. Anonymous users are indicated by a hyphen	-	The user was anonymous
c-ip	Categorical Nominal	ADDR - Address	The IP address of the client that made the request	70.80.84.76	The IP address of the client
cs(User-Agent)	Categorical Nominal	STR - Free String	The browser type that the client used		
cs(Referer)	Categorical Nominal	URL - links such as URLs	The site that the user last visited. This site provided a link to the current site	Mozilla/4.0+ (compatible;MSIE+5.5; +	The type of browser that the client used, as represented by the browser
sc-status	Metric Discrete	BIN - Binary	The HTTP status code	200	The request was fulfilled with no errors
sc-substatus	Metric Discrete	BIN - Binary	The substatus error code	1	Access is denied due to invalid credentials (if the sc-status is 401) or execute access is denied (if sc-status is
sc-win32-status	Metric Discrete	BIN - Binary	The Windows status code	64	The specified network is no longer available

1.2. Data Cleaning

A. The number NAs for each column:

- `date 2`
- `time 2`
- `s-sitename 2`
- `s-ip 2`
- `cs-method 2`
- `cs-uri-stem 2`
- `cs-uri-query 7886534`
- `s-port 2`
- `cs-username 8438930`
- `c-ip 2`
- `cs(User-Agent) 3529`
- `cs(Referer) 1309658`
- `sc-status 758`
- `sc-substatus 758`
- `sc-win32-status 758`

B. The number of rows before removal NAs:

- **8438930**

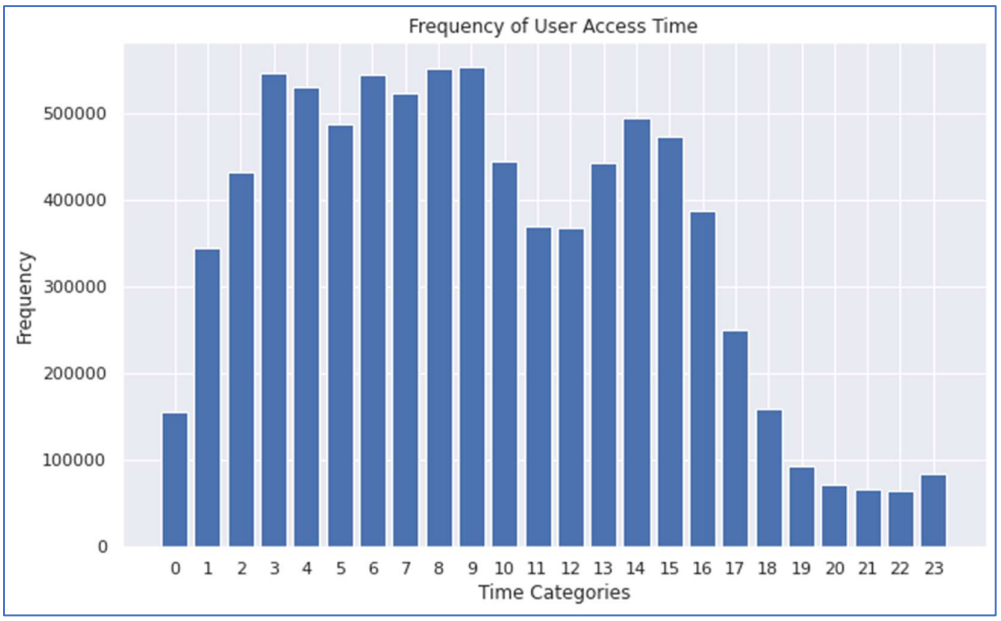
C. The number of rows after removal NAs:

- **8434645**
- NOTE: The last row belongs to [2007-03-01] section, which is out of scope of this study, so I decided to remove it. Now the final records: **8434644**

2. Data Statistics Description

2.1. Traffic Analysis

A. *Please add a figure of Hourly Requests Bar Chart from your Notebook and elaborate the findings from the figure.*



Based on the figure, the peak point of user requests is at frequency of 553564 at 9:00:00 and the lowest point is 65136 at 22:00:00. We also observe that most of the requests occurs between 1:00:00 to 16:00:00 universal time (UTC) with a short fallback at 11:00:00 and 12:00:00. From 19:00:00 to 23:00:00 the requests seem to slow down remarkably.

B. *Please add a table of filter result (hourly_request_amount >= 400000 & hourly_request_amount <= 490000)*

Table below is the result found after executing filtering code: (As the result: 5 rows were eliminated)

Time	Frequency
0	155875.0
1	344287.0
3	546655.0
4	529352.0
6	544711.0
7	522365.0
8	550744.0
9	553564.0
11	368947.0
12	367635.0
14	493693.0
16	387276.0
17	250269.0
18	159375.0
19	93829.0
20	72559.0
21	66474.0
22	65136.0
23	83633.0

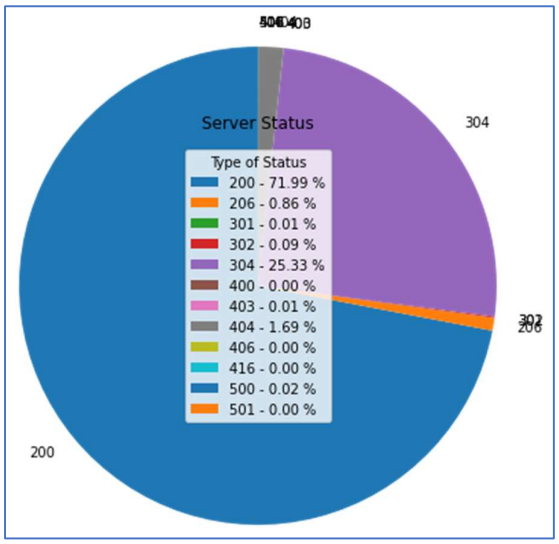
2.2. Server Analysis

A. Please elaborate how many types of reported server status.

- There were **12** types of server status reported

B. Please add a figure of Server Error Pie Chart from your Notebook, and elaborate the findings from the figure.

Pie chart representing the proportion of server status types



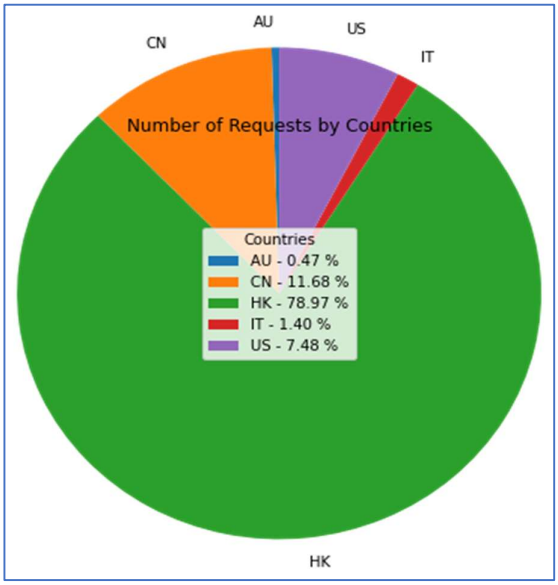
As seen in the figure, the majority of the proportion (about 72% of all statuses) belongs to status code 200 which represents the requests were being served successfully. The second top rank taken by status code 304 which covers up to 25%. This means that about 25% of the requests were served without the need of downloading content from server, instead the pages loaded by using cache information on their local machines. Other noticeable status codes including 404 and 206 which share about 1% each of all server statuses. In particular, code 404 requires extension to specify its meaning, although it can be inferred in general that those users either requested for non-existent contents or restricted contents from server. Code 206, on the other hand, explains that only partial contents were served to the users.

2.3. Geographics Analysis

A. Please add a figure of Country distribution and list top 3 with the number of requests.

1. There were **412** requests raised on 2007-Jan-01 between 20:00:00 to 20:59:59 UTC.
2. There were **5** countries involved

3. Plot the pie chart of the countries and their request frequencies:

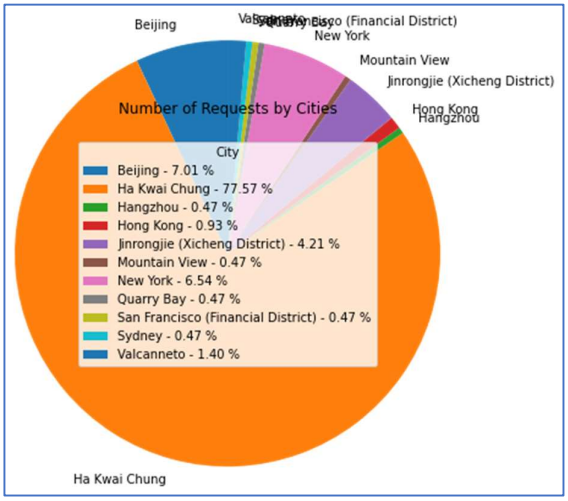


4. The top 3 countries with highest request rates:

Country names	Request rates
HK	**169**
CN	**25**
US	**16**

B. *Please add a figure of City distribution and list top 3 with the number of requests.*

1. There were **11** cities involved
2. Plot the pie chart of the cities and their request frequencies:



3. The top 3 cities with highest request rates:

City names	Request rates
Ha Kwai Chung	**166**
Beijing	**15**
New York	**14**

Part II - School of IT Professor Citation Information

To better introduce all the professors including the emeritus professor, the professor and also associate professor in Deakin University School of IT, faculty will need to know all the citation information on all professors. Google Scholar is a web search engine that freely indexes the metadata of articles on many authors. Majority of the professors choose to use google scholar to track their publications and research works. Therefore, the web crawling on google scholar will be able to have the citation information obtained across all the professors (who have the google scholar profile).

3. Professor List Generation

3.1. Import Web Crawling Library

Please fill this part with the screenshot of your code for import your own web crawling library.

```
# write your import and necessary web crawling libary here
print('Task 3.1: Import and install web crawling library')
!apt-get update
!apt install chromium-chromedriver
!cp /usr/lib/chromium-browser/chromedriver /usr/bin
!pip install selenium
```

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
```

3.2. Find all professors in School of IT

Please fill this part with the screenshot of your code for generating the professor name list csv. The screen shot will also include the results of the running on the code.

Screenshots of the codes:

```
#split string function
def parse_name(stringtext):
  return " ".join(stringtext.split(" ")[-2:])," ".join(stringtext.split(" ")[:-2])

# Establish chrome driver and go to report site URL
# set options to be headless, ..
options = webdriver.ChromeOptions()
options.add_argument('--headless')
options.add_argument('--no-sandbox')
options.add_argument('--disable-dev-shm-usage')

# open it, go to a website, and get results
url = 'https://www.deakin.edu.au/information-technology/staff-listing'
driver = webdriver.Chrome('chromedriver',options=options)
driver.get(url)

#get the table id that contain the staff info
table = driver.find_elements_by_id('table09355')
```

```
#Each staff name is wrapped within an anchor inside the above tables
#So we can target the anchor
anchor = list()
for i in table:
  anchor.append(i.find_elements_by_tag_name('a'))

#collect all staffs from the <tables><anchors> and put into a professor_list
professor_list = list()
for i in anchor:
  for j in i:
    professor_list.append(j.get_attribute('innerHTML'))

#Remove names that have no relation with professor title
professor_list = [i for i in professor_list if "Professor" in i]
```

```
import pandas as pd
# the crawling information will be stored in pandas dataframe and then save as csv
# below you are required to use the parse_name method to crawl the professor's full name and title
# The column name must be same as the provided professor-list.csv
print('Task 3.2: Find all professors in School of IT and save it as csv.')
print('-'*60)

#Add all professors into dataframe [professors]
professors = pd.DataFrame(columns=['Name', 'Title', 'University'])
for i in professor_list:
  new_row = {'University':'Deakin University','Title':parse_name(i)[1],'Name':parse_name(i)[0]}
  professors = professors.append(new_row,ignore_index=True)

#Write the dataframe into file
professors.to_csv('Professor-name-list.csv',index=False)
print('Professor-name-list has been created ...')
print('The content of the list: ')
professors
```

Screenshots of the results:

```
Task 3.2: Find all professors in School of IT and save it as csv.
------------------------------------------------------------
Professor-name-list has been created ...
The content of the list:
```

	Name	Title	University
0	Lynn Batten	Emeritus Professor	Deakin University
1	Andrzej Goscinski	Emeritus Professor	Deakin University
2	Jemal Abawajy	Professor	Deakin University
3	Maia Angelova	Professor	Deakin University
4	Gleb Beliakov	Professor	Deakin University
5	Terry Caelli	Professor	Deakin University
6	Jinho Choi	Professor	Deakin University
7	Chang-Tsun Li	Professor	Deakin University
8	Robin Doss	Professor	Deakin University
9	Peter Eklund	Professor	Deakin University
10	Seng Loke	Professor	Deakin University

…

39	William Moran	Honorary Professor	Deakin University
40	Ana Novak	Honorary Associate Professor	Deakin University
41	Zahir Quettawala	Adjunct Professor	Deakin University
42	Anthony Robertshaw	Adjunct Associate Professor	Deakin University
43	Jamie Rossato	Adjunct Professor	Deakin University
44	Chadi Saliby	Adjunct Professor	Deakin University
45	Malcolm Shore	Adjunct Professor	Deakin University
46	EJ Wise	Adjunct Professor	Deakin University
47	Abbas Kudrati	Professor	Deakin University

3.2.1. Professor Name List CSV

Please fill this part with the screenshot of your csv.

	A	B	C	D	E
1	Name	Title	University		
2	Lynn Batten	Emeritus Professor	Deakin University		
3	Andrzej Goscinski	Emeritus Professor	Deakin University		
4	Jemal Abawajy	Professor	Deakin University		
5	Maia Angelova	Professor	Deakin University		
6	Gleb Beliakov	Professor	Deakin University		
7	Terry Caelli	Professor	Deakin University		
8	Jinho Choi	Professor	Deakin University		
9	Chang-Tsun Li	Professor	Deakin University		
10	Robin Doss	Professor	Deakin University		
11	Peter Eklund	Professor	Deakin University		
12	Seng Loke	Professor	Deakin University		
13	Antonio Robles-Kelly	Professor	Deakin University		
14	Jean-Guy Schneider	Professor	Deakin University		
15	Yong Xiang	Professor	Deakin University		
16	John Yearwood	Professor	Deakin University		
17	Arkady Zaslavsky	Professor	Deakin University		
18	Mohamed Abdelrazek	Associate Professor	Deakin University		
19	Andrew Cain	Associate Professor	Deakin University		
20	Richard Dazeley	Associate Professor	Deakin University		
21	Guangyan Huang	Associate Professor	Deakin University		
22	Gang Li	Associate Professor	Deakin University		
23	Jianxin Li	Associate Professor	Deakin University		
24	Xiao Liu	Associate Professor	Deakin University		
25	Vicky Mak	Associate Professor	Deakin University		
26	Tim Wilkin	Associate Professor	Deakin University		
27	Abid Adam	Adjunct Professor	Deakin University		
28	Catherine Buhler	Adjunct Professor	Deakin University		
29	Eshan Dissanayake	Adjunct Professor	Deakin University		
30	Patrick Fair	Adjunct Professor	Deakin University		
31	David Fairman	Adjunct Professor	Deakin University		
32	Mark Fitzgerald	Honorary Professor	Deakin University		
33	Praveen Gauravaram	Adjunct Professor	Deakin University		
34	Nigel Hedges	Adjunct Professor	Deakin University		
35	Darren Kane	Adjunct Professor	Deakin University		
36	Len Kleinman	Adjunct Professor	Deakin University		
37	James Kotsias	Adjunct Professor	Deakin University		
38	Berin Lautenbach	Adjunct Professor	Deakin University		
39	Phillip Magness	Adjunct Professor	Deakin University		
40	Joseph Mathew	Adjunct Associate Professor	Deakin University		
41	William Moran	Honorary Professor	Deakin University		
42	Ana Novak	Honorary Associate Professor	Deakin University		
43	Zahir Quettawala	Adjunct Professor	Deakin University		
44	Anthony Robertshaw	Adjunct Associate Professor	Deakin University		
45	Jamie Rossato	Adjunct Professor	Deakin University		
46	Chadi Saliby	Adjunct Professor	Deakin University		
47	Malcolm Shore	Adjunct Professor	Deakin University		
48	EJ Wise	Adjunct Professor	Deakin University		
49	Abbas Kudrati	Professor	Deakin University		
50					

Professor-name-list

4. Professor Citation Information Generation

4.1. Search the google scholar for all

Please fill this part with the screenshot of the code for generating the professor citation information (include the actual crawling steps).

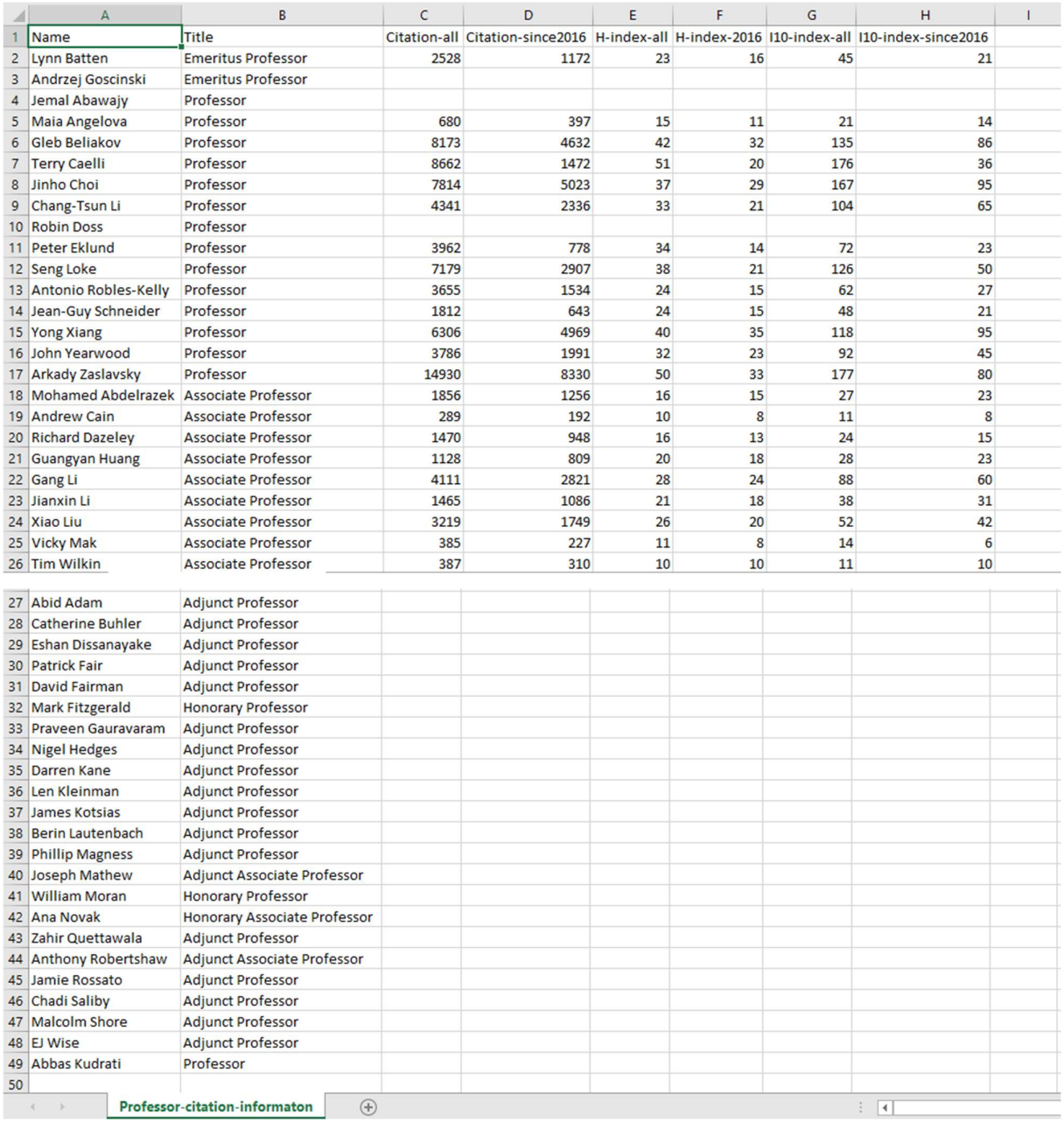
```
print('Task 4.1: Search the google scholar for all professors')
print('-'*60)
#Function to check whether a professor has google scholar profile
def get_check(body_div):
    try:
      return body_div.find_element_by_class_name('gs_ai_name')
    except:
      return False
#Function add citation into dataframe
def new_row(name,title,args):
  row = {'Name':name,
        'Title':title,
        'Citation-all':float(args[0]),
        'Citation-since2016':float(args[1]),
        'H-index-all':float(args[2]),
        'H-index-2016':float(args[3]),
        'I10-index-all':float(args[4]),
        'I10-index-since2016':float(args[5])
        }
  return row

#Create new dataframe
col_names=['Name','Title','Citation-all','Citation-since2016','H-index-all',
           'H-index-2016','I10-index-all','I10-index-since2016']
citations = pd.DataFrame(columns=col_names)
```

```
#Search loop for all professors
for i,j in zip(professors.Name,professors.Title):
  url = 'https://scholar.google.com/citations?hl=en&view_op=search_authors&mauthors='+i+'+Deakin'
  driver.get(url)
  #When the name has google scholar profile, the page shows the person
  #in its main body <div> with id ='gs_bdy'
  body = driver.find_element_by_id('gs_bdy')
  result = get_check(body)

  #when the <div id='gs_bdy'> is not there, it means the professor
  # has no profile with google scholar,and dataframe receive NaN
  if not(result):
    items = np.empty(6)
    items.fill(np.nan)
    citations = citations.append(new_row(i,j,items),ignore_index=True)
    continue

  #Click one the link on the name of the finding professor
  result.find_element_by_tag_name('a').click()
  #Wait at least 5s before table id='gsc_rsb_st' loaded, and get its content
  table = WebDriverWait(driver,5).until(EC.presence_of_element_located((By.ID,'gsc_rsb_st')))
```

```
  #Add the td contents into dataframe
  items = []
  for ele in td:
    items.append(ele.get_attribute('innerHTML'))
  citations = citations.append(new_row(i,j,items),ignore_index=True)
print('\nBelow is the dataframe to be written into csv file...\n')
print(citations)
citations.to_csv('Professor-citation-informaton.csv',index=False)
```

4.1.1. Professor Citation Information CSV

Please fill this part with the screenshot of the professor citation information CSV.



	A	B	C	D	E	F	G	H	I
1	Name	Title	Citation-all	Citation-since2016	H-index-all	H-index-2016	I10-index-all	I10-index-since2016	
2	Lynn Batten	Emeritus Professor	2528	1172	23	16	45	21	
3	Andrzej Goscinski	Emeritus Professor							
4	Jemal Abawajy	Professor							
5	Maia Angelova	Professor	680	397	15	11	21	14	
6	Gleb Beliakov	Professor	8173	4632	42	32	135	86	
7	Terry Caelli	Professor	8662	1472	51	20	176	36	
8	Jinho Choi	Professor	7814	5023	37	29	167	95	
9	Chang-Tsun Li	Professor	4341	2336	33	21	104	65	
10	Robin Doss	Professor							
11	Peter Eklund	Professor	3962	778	34	14	72	23	
12	Seng Loke	Professor	7179	2907	38	21	126	50	
13	Antonio Robles-Kelly	Professor	3655	1534	24	15	62	27	
14	Jean-Guy Schneider	Professor	1812	643	24	15	48	21	
15	Yong Xiang	Professor	6306	4969	40	35	118	95	
16	John Yearwood	Professor	3786	1991	32	23	92	45	
17	Arkady Zaslavsky	Professor	14930	8330	50	33	177	80	
18	Mohamed Abdelrazek	Associate Professor	1856	1256	16	15	27	23	
19	Andrew Cain	Associate Professor	289	192	10	8	11	8	
20	Richard Dazeley	Associate Professor	1470	948	16	13	24	15	
21	Guangyan Huang	Associate Professor	1128	809	20	18	28	23	
22	Gang Li	Associate Professor	4111	2821	28	24	88	60	
23	Jianxin Li	Associate Professor	1465	1086	21	18	38	31	
24	Xiao Liu	Associate Professor	3219	1749	26	20	52	42	
25	Vicky Mak	Associate Professor	385	227	11	8	14	6	
26	Tim Wilkin	Associate Professor	387	310	10	10	11	10	
27	Abid Adam	Adjunct Professor							
28	Catherine Buhler	Adjunct Professor							
29	Eshan Dissanayake	Adjunct Professor							
30	Patrick Fair	Adjunct Professor							
31	David Fairman	Adjunct Professor							
32	Mark Fitzgerald	Honorary Professor							
33	Praveen Gauravaram	Adjunct Professor							
34	Nigel Hedges	Adjunct Professor							
35	Darren Kane	Adjunct Professor							
36	Len Kleinman	Adjunct Professor							
37	James Kotsias	Adjunct Professor							
38	Berin Lautenbach	Adjunct Professor							
39	Phillip Magness	Adjunct Professor							
40	Joseph Mathew	Adjunct Associate Professor							
41	William Moran	Honorary Professor							
42	Ana Novak	Honorary Associate Professor							
43	Zahir Quettawala	Adjunct Professor							
44	Anthony Robertshaw	Adjunct Associate Professor							
45	Jamie Rossato	Adjunct Professor							
46	Chadi Saliby	Adjunct Professor							
47	Malcolm Shore	Adjunct Professor							
48	EJ Wise	Adjunct Professor							
49	Abbas Kudrati	Professor							
50									

Professor-citation-informaton

4.2. Find the Professor with most citations

Please fill this part with the screenshot of the code (include the results of the code running).

```
# find out the professor name having the most citations (please remove those professor who does not have google scholar
# write your code here
print('Task 4.2: Find out the professor name having the most citations')
print('-'*60)

print('Drop the professor name who dose not have google scholar profile...')
top_professor = citations.copy().dropna()

top_professor = top_professor [top_professor ['Citation-all']==np.max(top_professor ['Citation-all'])]['Name']
print('\nProfessor name has the most citations is: {}'.format(top_professor.values))
```

```
Task 4.2: Find out the professor name having the most citations
------------------------------------------------------------
Drop the professor name who dose not have google scholar profile...

Professor name has the most citations is: ['Arkady Zaslavsky']
```

4.3. Find the Associate Professor with most i10-index since 2016

Please fill this part with the screenshot of the code (include the results of the code running).

```
# find out the row for associate professor having the most i10_index since 2016 (please remove those professor who does
# write your code here
print('Task 4.3: Find out the row for associate professor having the most i10-index since 2016')
print('-'*60)

print('Drop the professor name who dose not have google scholar profile...')
top_i10_2006 = citations.copy().dropna()

top_i10_2006 = top_i10_2006[top_i10_2006['Title']=='Associate Professor']
top_i10_2006 = top_i10_2006[top_i10_2006['I10-index-since2016']==top_i10_2006['I10-index-since2016'].max()]

print('\nThe associate professor who has recorded the most i10-index since 2006 is: ')
top_i10_2006
```

```
Task 4.3: Find out the row for associate professor having the most i10-index since 2016
------------------------------------------------------------
Drop the professor name who dose not have google scholar profile...

The associate professor who has recorded the most i10-index since 2006 is:
```

	Name	Title	Citation-all	Citation-since2016	H-index-all	H-index-2016	I10-index-all	I10-index-since2016
20	Gang Li	Associate Professor	4111.0	2821.0	28.0	24.0	88.0	60.0

4.4. Find all Professors name who have the citations since 2016 > 2500

Please fill this part with the screenshot of the code (include the results of the code running).

```
# find out all the professors name who has the citations_since2016 > 2500
# write your code here
print('Task 4.4: Find out all the professors name who has the citations_since2016 > 2500')
print('-'*60)

print('Drop the professor name who dose not have google scholar profile...')
target_professor = citations.copy().dropna()

target_professor = target_professor[target_professor['Citation-since2016']>2500]['Name']
print('\nProfessor names with citations_since2016 > 2500:')
for i in target_professor:
  print('\t- {}'.format(i))
```

```
Task 4.4: Find out all the professors name who has the citations_since2016 > 2500
------------------------------------------------------------
Drop the professor name who dose not have google scholar profile...

Professor names with citations_since2016 > 2500:
        - Gleb Beliakov
        - Jinho Choi
        - Seng Loke
        - Yong Xiang
        - Arkady Zaslavsky
        - Gang Li
```