P4 – Webscraping General Conference Instructions

Overview:
This is a group project. For this project you are going to web scrape the talks from a recent general conference of The Church of Jesus Christ of Latter-day Saints which you can find here. For those that aren't members, general conference is when leaders of the church give talks (speeches basically) about different topics to the whole world. Talks have a title and a speaker, and typically there are many references to books of scripture (among other things). Your group's task is to scrape the name of the talk, the speaker, as well as the number of references to each book of scripture in each talk.

Note that currently, the project is requiring scraping the October 2023 general conference. This is because there are a couple of quirks (some speakers not making any references to external sources) that make the scraping task a little more complicated, which is great practice for other realistic scraping tasks.

Libraries Required:
- from bs4 import BeautifulSoup
- import requests
- import pandas as pd
- import sqlalchemy
- import matplotlib.pyplot as plot

Logical Flow:
Similar to your other projects, there are multiple ways to do this, and all that matters is that you fulfill the requirements. After the logical flow section, I give recommendations and hints for completing the project, but you do not need to necessarily follow those hints.

You are only required to scrape the talks from the October 2023 conference, but your code should still work even if you switched the url to a different conference session. That means no hardcoding things that wouldn't generally work for different conference sessions. Feel free to ask the professor or the TAs if you aren't sure about this.

Your program will ask the user:
- *"If you want to scrape data, enter 1. If you want to see summaries of stored data, enter 2. Enter any other value to exit the program: "*

PART 1: If they enter 1, do the following:

1. Run a sql query to drop the table "general_conference" if it exists in your postgres database.
2. Using the requests and BeautifulSoup libraries, load https://www.churchofjesuschrist.org/study/general-conference/2023/10?lang=eng into a beautifulsoup object.
3. You need to access the pages for all individual talks
    a. However, you should ignore all the pages that just have video recordings, like "Saturday Morning Session"
    b. You should also ignore the page called "Sustaining of General Authorities, Area Seventies, and General Officers" since that page isn't a talk.

4. For each talk you should store the speaker's name, title of the talk, the "kicker" (the quote in different font before the main text), and the number of times each book of scripture is listed in the references.
5. Save the data as a table called 'general_conference' in your is303 postgres database.
6. Print out the message: *"You've saved the scraped data to your postgres database."*

PART 2: If they enter 2, do the following:

1. Print out: *"You selected to see summaries. Enter 1 to see a summary of all talks. Enter 2 to select a specific talk. Enter anything else to exit: "*
2. If the user enters 1, show them a bar chart with all standard works books on the x-axis that have more than 2 references, and the y-axis being the count of the references.
    a. The title of the chart should be: Standard Works Referenced in General Conference
    b. The x-label should be: Standard Works Books
    c. The y-label should be: # Times Referenced
3. If the user enters 2, display *"The following are the names of speakers and their talks:"* then display a number, the speaker name, and name of the talk, for of all the talks. For example:
    1: Elder David A. Bednar - In the Path of Their Duty
    2: Sister Amy A. Wright - Abide the Day in Christ
    3: etc.....
4. Print out: *"Please enter the number of the talk you want to see summarized: "*
5. Then, for the entered talk, display a bar chart just like the previous bar chart, but this time only for the selected talk, and only show books that have at least 1 reference.
    a. The title of the chart should be "Standard Works Referenced in: {Talk_Name}"
    b. The x-label should be: Standard Works Books
    c. The y-label should be: # Times Referenced

If they enter anything other than 1 or 2:

- Print: "Closing the program."

Hints/Suggestions:

PART 1: If they enter 1, do the following:

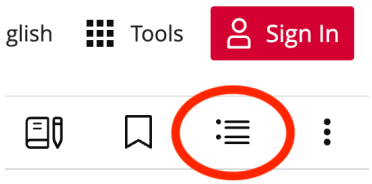
1. Run a sql query to drop the table "general_conference" if it exists.
    a. Since we didn't really go over SQL much in this class I'll just give this code to you. Dropping in SQL means deleting. Do this before you start web scraping.
    b.
```
# drop the general_conference table if it exists. This prevents appending duplicate data
# if you run the program multiple times.
drop_table_query = sqlalchemy.text("drop table if exists general_conference;")
conn = engine.connect()
conn.execute(drop_table_query)
conn.commit()
conn.close()
```
    c. Note, that depending on how you write your code later on, this might actually be unnecessary, but I'm requiring everyone to put this in (just because it's a good thing to know) and it makes grading consistent.
2. Using the requests and BeautifulSoup libraries, load https://www.churchofjesuschrist.org/study/general-conference/2023/10?lang=eng into a beautifulsoup object.

    a. See the class examples of this. Make sure you've imported the requests and BeautifulSoup libraries otherwise it won't work.
3. You need to access the pages for all individual talks, but ignore pages that only have video recordings of an entire session and "Sustaining of General Authorities"
    a. Remember, the urls for the individual talk pages are usually in "a" elements, but you may find it useful to grab some other element that contains only the links you want to avoid also grabbing other "a" elements that you don't need (like links to other parts of the church website, etc.).
    b. You can use if statements to leave out visiting the pages that don't have actual talks in them. Here are some possible ways to do that:
        i. You could look at the URLs (the hrefs) of the pages that you don't want. You can filter out pages that have "Session" in them for example.
        ii. For the "Sustaining of General Authorities" the url doesn't have unique info to filter it out, but you can skip looping through it by checking the title of the talk and seeing if it has "Sustaining" in the title.
    c. You'll then want to load each of these URLs using requests and BeautifulSoup. Remember that these will be relative links, so you need to add 'https://www.churchofjesuschrist.org' to the beginning of the urls for them to work.
    d. This is optional, but I recommend printing out "trying to scrape url: {url}" before loading a specific page. This is useful because if your code breaks on a specific page, you know where to check it. It is also a good visual indicator when your code is running. This probably isn't a great idea for webscrapers visiting thousands of pages since printing slows things down, but it is good for smaller projects.
4. For each talk you should store the speaker's name, title of the talk, the "kicker" (the quote in different font before the main text), and the number of times each book of scripture is listed in the references.
    a. When counting the references to the specific books, there are 87 books of scripture in the LDS standard works. Lots of things to study! That's a pain to type out though, so I made a dictionary for you to copy into your code with all the books, as well as a 3 other spots for the Speaker_Name, Talk_Name, and Kicker. Note you can copy the code below, it's not a screenshot.
        i. Notice that the value of each key is an empty list. That means you can just append the values for each talk to the list.

```
standard_works_dict = {'Speaker_Name' : [], 'Talk_Name' : [], 'Kicker' : [],
'Matthew': [], 'Mark': [], 'Luke': [], 'John': [], 'Acts': [], 'Romans': [], '1
Corinthians': [], '2 Corinthians': [], 'Galatians': [], 'Ephesians': [],
'Philippians': [], 'Colossians': [], '1 Thessalonians': [], '2 Thessalonians': [],
'1 Timothy': [], '2 Timothy': [], 'Titus': [], 'Philemon': [], 'Hebrews': [],
'James': [], '1 Peter': [], '2 Peter': [], '1 John': [], '2 John': [], '3 John':
[], 'Jude': [], 'Revelation': [], 'Genesis': [], 'Exodus': [], 'Leviticus': [],
'Numbers': [], 'Deuteronomy': [], 'Joshua': [], 'Judges': [], 'Ruth': [], '1
Samuel': [], '2 Samuel': [], '1 Kings': [], '2 Kings': [], '1 Chronicles': [], '2
Chronicles': [], 'Ezra': [], 'Nehemiah': [], 'Esther': [], 'Job': [], 'Psalm': [],
'Proverbs': [], 'Ecclesiastes': [], 'Song of Solomon': [], 'Isaiah': [],
'Jeremiah': [], 'Lamentations': [], 'Ezekiel': [], 'Daniel': [], 'Hosea': [],
'Joel': [], 'Amos': [], 'Obadiah': [], 'Jonah': [], 'Micah': [], 'Nahum': [],
'Habakkuk': [], 'Zephaniah': [], 'Haggai': [], 'Zechariah': [], 'Malachi': [], '1
Nephi': [], '2 Nephi': [], 'Jacob': [], 'Enos': [], 'Jarom': [], 'Omni': [], 'Words
```

```
of Mormon': [], 'Mosiah': [], 'Alma': [], 'Helaman': [], '3 Nephi': [], '4 Nephi':
[], 'Mormon': [], 'Ether': [], 'Moroni': [], 'Doctrine and Covenants': [], 'Moses':
[], 'Abraham': [], 'Joseph Smith—Matthew': [], 'Joseph Smith—History': [],
'Articles of Faith': []}
```

b. For all of these you probably just want to use .find()
c. The Speaker Name
    i. All the speaker names have "By " before their name. Look up how to leave off the first few characters of a string. You can leave titles like "Elder" or "Sister" in their name.
    ii. You can append the speaker name to the list associated with the "Speaker_Name" key in the dictionary above.
d. Title of the talk and "kicker"
    i. The kicker is just the name of the quote at the beginning before the actual text of the talk begins.
    ii. You can append the title and kicker to the list associated with the using the associated keys in the dictionary above.
e. References to each book of scripture:
    i. This part is much trickier. I recorded a video explaining this portion to help you out: Scraping references tips
    ii. The footnotes are normally accessed by pressing this button:



1.
    iii. BUT, if you try to access the footnotes through beautifulsoup this way, it won't actually work, since that footnotes button loads separately from the main webpage. The html is only visible if you actually click the button, which is something requests / beautifulsoup can't easily do unless you understand javascript or combine beautifulsoup with another scraping tool called selenium. Feel free to look those up if you're interested for future reference, but there's an easier way around this problem for now.
    iv. Luckily, all the reference information is also stored in the footer element with the "class" attribute called "notes". You can get at it with code like this (your specific variable names might be different though):
        1.
```
footnotes_section = specific_talk_page.find('footer', attrs={'class' : 'notes'})
```
    v. BUT, there is at least one talk that doesn't have any references at all, meaning it doesn't have a footer element. Your code should handle this situation (a very common and realistic situation when web scraping). If the above code can't find a footer element, then the "footnotes_section" variable will be equal to "None". That means you can use an if statement to check if "footnotes_section" is equal to None. This can help you avoid errors such as using .get_text() on an element that doesn't exist.
    vi. With the dictionary given to you, you can do the following :

1. Loop through the dictionary. By default, when doing a for loop with a dictionary, it gives you the keys. Or you can use .keys() to get just the keys from a dictionary
2. Check how many times a key appears in the footnotes_section text using .count(your_variable_for_dictionary_key)
    a. But be sure to skip the Speaker_Name, Talk_Name, and Kicker when counting instances of each, otherwise you'll be trying to count how often "Speaker_Name" appears in the references and be adding an extra row of 0 to your dictionary.
    b. You may notice that when you try to count books like "1 Nephi" or "2 Kings", etc. it always returns 0. That is because in the HTML the spaces between the number and the book name are special "non-breaking" spaces that prevent the "1" and "Nephi" from ever appearing on separate lines. To fix this, you can run code like this before counting, which will replace the non-breaking spaces with regular spaces. See the video I posted (in section e.i) for a demonstration of this.
    c. `footnotes_section_text = footnotes_section_text.replace('\u00A0', ' ')`
3. Update the value of the dictionary for that key to append the number that .count() returns.
    a. As an example, as you loop through the dictionary, it will eventually come across the key "Matthew". .count() can count how many times "Matthew" appears in the footnotes section text. Then append that number to the list associated with "Matthew" in the dictionary. Then it moves on to the next key, etc.

5. Save the data as a table called 'general_conference' in your is303 postgres database.
    a. Easiest way to do this is to first transform your dictionary into a pandas DataFrame.
    b. Then use the .to_sql method. If you did it like I describe in the hints, you'll probably want to use "replace" for the if_exists parameter, but if you did it a different way, "append" could work too.
    c. As a side note, if you've taken more database classes, you'll know that storing all this data in one table isn't the best way to structure things, but it is a simple way that works well enough for this context.
6. Print out the message: *"You've saved the scraped data to your postgres database."*

PART 2: If they enter 2, do the following:

6. Print out: *"You selected to see summaries. Enter 1 to see a summary of all talks. Enter 2 to select a specific talk. Enter anything else to exit: "):"*
7. If the user enters 1, show them a bar chart with all standard works books on the x-axis that have more than 2 references, and the y-axis being the count of the references.
    a. Because we haven't covered charts and prepping data for charts. I'm just going to give you the code here in a screenshot. The main idea is that I'm dropping the columns that aren't related to reference counts, then summing up everything else, then filtering it to results that are greater than 2.

b.

```
if user_selection_2 == "1":
    sql_query = 'select * from general_conference'
    df_from_postgres = pd.read_sql_query(sql_query, engine)

    df_sums = df_from_postgres.drop(['Speaker_Name', 'Talk_Name', "Kicker"], axis=1).sum()
    df_sums_filtered = df_sums[df_sums > 2]

    df_sums_filtered.plot(kind='bar')
    plot.title('Standard Works Referenced in General Conference')
    plot.xlabel("Standard Works Books")
    plot.ylabel("# Times Referenced")
    plot.show()
```

8. If the user enters 2, display a number, the speaker name, and name of the talk of all the talks.
    a. This is very similar to what you did in project 2. This time, if you get all the data from your 'general_conference' table in postgres, I recommend looping through it with .iterrows(). That way you get an index (a number), and then can access the specific columns of row["Speaker_Name"] and row["Talk_Name"]. You can do whatever works best for you though.
        -
```
for index, row in df_from_postgres.iterrows(): # loops through each row, but also gives an index variable
```
9. Print out: *"Please enter the number of the talk you want to see summarized: "*
    a. You'll need a way to translate a number (e.g. 1) into a talk name. A dictionary is often a good way to do this.
10. Then, for the entered talk, display a bar chart just like the previous bar chart, but this time only for the selected talk, and only show books that have at least 1 reference.
    a. Again, I'll give you the code for the charts. The exact implementation may change depending on how you have the user select the talk and what you call your variables.
    b.

```
# filtering the dataframe to only have rows from the selected talk
df_filtered = df_from_postgres.query(f"Talk_Name == '{requested_talk}'")

df_filtered = df_filtered.drop(['Speaker_Name', 'Talk_Name', "Kicker"], axis=1).sum()
df_filtered = df_filtered[df_filtered > 0]

df_filtered.plot(kind='bar')
plot.title(f'Standard works Referenced in: {requested_talk}')
plot.xlabel("Standard Works Books")
plot.ylabel("# Times Referenced")
plot.show()
```

If they enter anything other than 1 or 2:

- Print: "Closing the program."

Upload just the python file to Learning Suite. Only one person per group needs to upload.

Example Output:

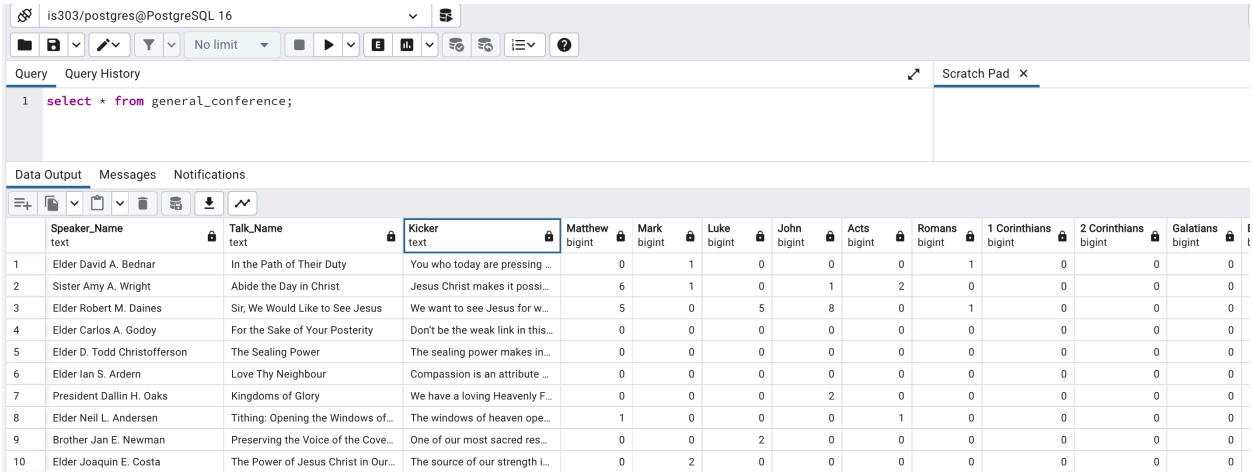
Part 1:

If you want to scrape data, enter 1. If you want to see summaries of stored data, enter 2. Enter any other value to exit the program: 1
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/11bednar?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/12wright?lang=eng

trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/13daines?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/14godoy?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/15christofferson?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/16ardern?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/17oaks?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/21eyring?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/22andersen?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/23newman?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/24costa?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/25stevenson?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/26choi?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/27phillips?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/28rasband?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/31sabin?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/32koch?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/33runia?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/34soares?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/41ballard?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/42freeman?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/43parrella?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/44cook?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/45uchtdorf?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/46waddell?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/47eyring?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/57renlund?lang=eng

trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/52pingree?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/53cordon?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/55esplin?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/54gong?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/56giraud-carrier?lang=eng
trying to scrape url: https://www.churchofjesuschrist.org/study/general-conference/2023/10/51nelson?lang=eng
You've saved the scraped data to your postgres database

Note, you don't need to upload anything from postgres/pgadmin4, but if you did part 1 correctly, if you do select * from general_conference; in pgAdmin4 it should look like this:

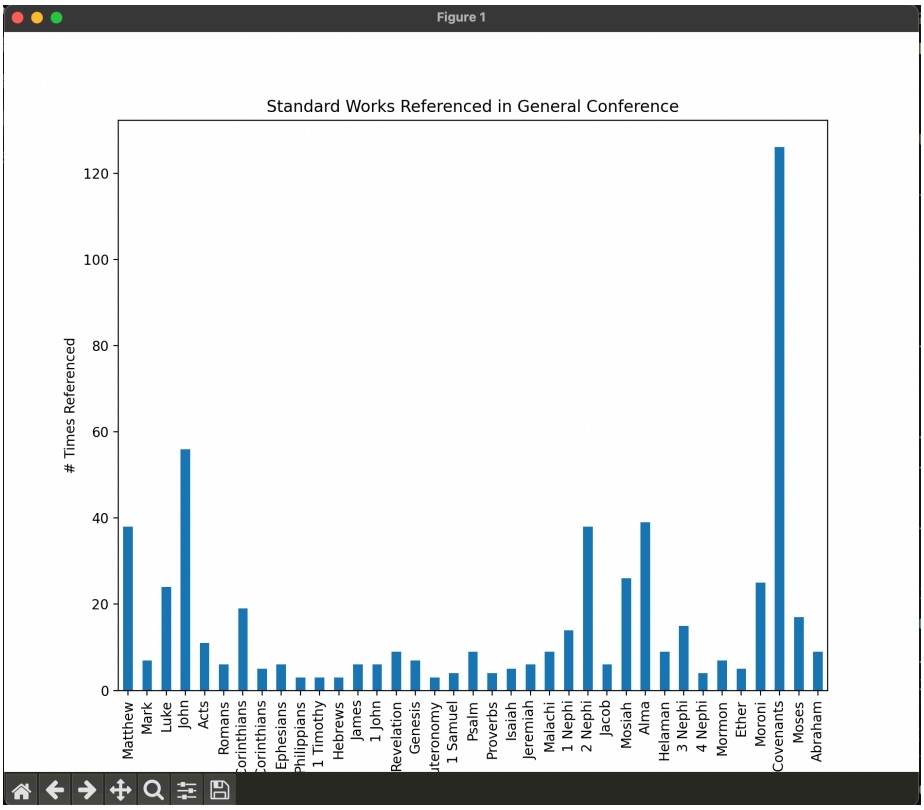


	Speaker_Name text	Talk_Name text	Kicker text	Matthew bigint	Mark bigint	Luke bigint	John bigint	Acts bigint	Romans bigint	1 Corinthians bigint	2 Corinthians bigint	Galatians bigint
1	Elder David A. Bednar	In the Path of Their Duty	You who today are pressing ...	0	1	0	0	0	1	0	0	0
2	Sister Amy A. Wright	Abide the Day in Christ	Jesus Christ makes it possi...	6	1	0	1	2	0	0	0	0
3	Elder Robert M. Daines	Sir, We Would Like to See Jesus	We want to see Jesus for w...	5	0	5	8	0	1	0	0	0
4	Elder Carlos A. Godoy	For the Sake of Your Posterity	Don't be the weak link in this...	0	0	0	0	0	0	0	0	0
5	Elder D. Todd Christofferson	The Sealing Power	The sealing power makes in...	0	0	0	0	0	0	0	0	0
6	Elder Ian S. Ardern	Love Thy Neighbour	Compassion is an attribute ...	0	0	0	0	0	0	0	0	0
7	President Dallin H. Oaks	Kingdoms of Glory	We have a loving Heavenly F...	0	0	0	2	0	0	0	0	0
8	Elder Neil L. Andersen	Tithing: Opening the Windows of...	The windows of heaven ope...	1	0	0	0	1	0	0	0	0
9	Brother Jan E. Newman	Preserving the Voice of the Cove...	One of our most sacred res...	0	0	2	0	0	0	0	0	0
10	Elder Joaquin E. Costa	The Power of Jesus Christ in Our...	The source of our strength i...	0	2	0	0	0	0	0	0	0

Part 2.1:

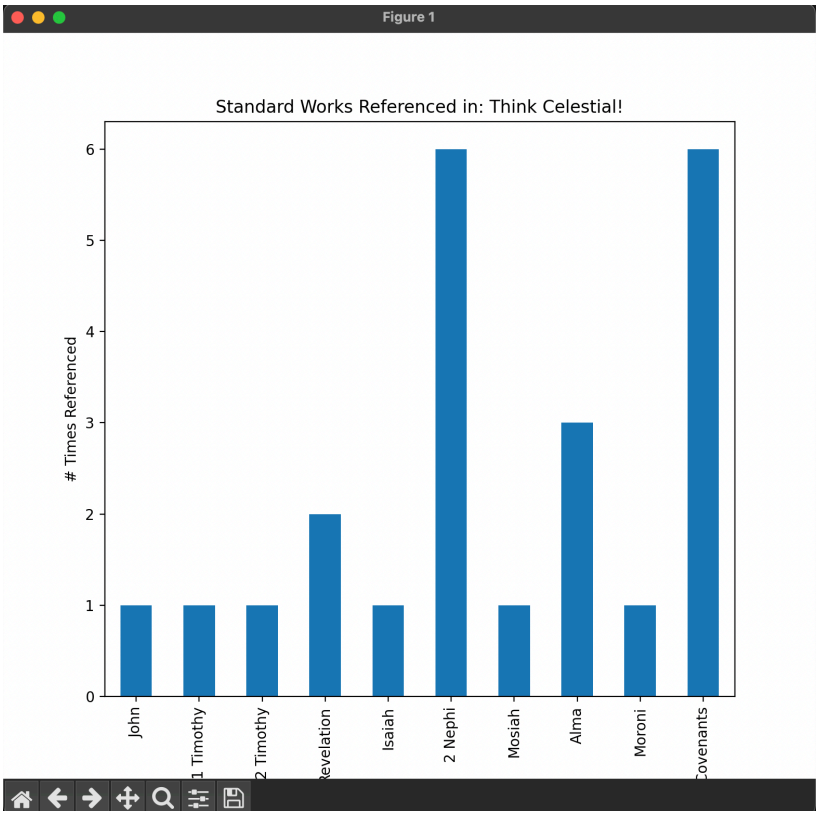
If you want to scrape data, enter 1. If you want to see summaries of stored data, enter 2. Enter any other value to exit the program: 2
You selected to see summaries. Enter 1 to see a summary of all talks. Enter 2 to select a specific talk. Enter anything else to exit: 1

Then this should pop up:



Part 2.2:
If you want to scrape data, enter 1. If you want to see summaries of stored data, enter 2. Enter any other value to exit the program: 2
You selected to see summaries. Enter 1 to see a summary of all talks. Enter 2 to select a specific talk. Enter anything else to exit: 2
The following are the names of speakers and their talks:
1: Elder David A. Bednar - In the Path of Their Duty
2: Sister Amy A. Wright - Abide the Day in Christ
3: Elder Robert M. Daines - Sir, We Would Like to See Jesus
4: Elder Carlos A. Godoy - For the Sake of Your Posterity
5: Elder D. Todd Christofferson - The Sealing Power
6: Elder Ian S. Ardern - Love Thy Neighbour
7: President Dallin H. Oaks - Kingdoms of Glory
8: Elder Neil L. Andersen - Tithing: Opening the Windows of Heaven
9: Brother Jan E. Newman - Preserving the Voice of the Covenant People in the Rising Generation
10: Elder Joaquin E. Costa - The Power of Jesus Christ in Our Lives Every Day
11: Elder Gary E. Stevenson - Promptings of the Spirit
12: Elder Yoon Hwan Choi - Do You Want to Be Happy?
13: Elder Alan T. Phillips - God Knows and Loves You
14: Elder Ronald A. Rasband - How Great Will Be Your Joy
15: Elder Gary B. Sabin - Hallmarks of Happiness
16: Elder Joni L. Koch - Humble to Accept and Follow
17: Sister Tamara W. Runia - Seeing God's Family through the Overview Lens
18: Elder Ulisses Soares - Brothers and Sisters in Christ
19: President M. Russell Ballard - Praise to the Man

20: President Emily Belle Freeman - Walking in Covenant Relationship with Christ
21: Elder Adilson de Paula Parrella - Bearing Witness of Jesus Christ in Word and Actions
22: Elder Quentin L. Cook - Be Peaceable Followers of Christ
23: Elder Dieter F. Uchtdorf - The Prodigal and the Road That Leads Home
24: Bishop W. Christopher Waddell - More Than a Hero
25: President Henry B. Eyring - Our Constant Companion
26: Elder Dale G. Renlund - Jesus Christ Is the Treasure
27: Elder John C. Pingree Jr. - Eternal Truth
28: Elder Valeri V. Cordón - Divine Parenting Lessons
29: Elder J. Kimo Esplin - The Savior's Healing Power upon the Isles of the Sea
30: Elder Gerrit W. Gong - Love Is Spoken Here
31: Elder Christophe G. Giraud-Carrier - We Are His Children
32: President Russell M. Nelson - Think Celestial!
Please enter the number of the talk you want to see summarized: 32



Don't worry if the sizing of the chart cuts off some of the text.

Requirements:

Requirement	Sub Requirements
asks the user to enter 1 or 2	
Runs sql query to drop the general_conference table if it exists	

Starts off on the general conference page that lists all the talks in all the sessions.	
Loads all the pages for the talks, but leaves out the pages specified in the instructions.	- You will lose some points if you hard code the urls to avoid. You should be as general as possible in your logic, as suggested in the hints. Hard coding some text, such as looking for "session" is good though.
Stores the speaker name, talk name, kicker, and # of references to books of scripture.	
Saves the data to a table called general_conference in postgres.	
When selecting option 2, allows for another option of selecting 1 or 2	-
Displays a chart showing the reference counts across all the talks with more than 2 references.	- The titles and axis labels should be as described
If user enters 2 a second time, displays a number, the name of the speaker, and name of the talk for all talks.	- Will lose points if the names of the talks are hardcoded.
Displays bar chart based on the talk selected by the user.	- The titles and axis labels should be as described.
includes useful comments	- Should include a name and description at the top - Should also include comments throughout