Data Wrangling with Python: Activity 9, page 294

In [22]:
```
#Import necessary libraries including regex, and beautifulsoup
import urllib.request, urllib.parse, urllib.error
import requests
from bs4 import BeautifulSoup
import ssl
import re
```

In [23]:
```
ctx = ssl.create_default_context()
ctx.check_hostname = False
ctx.verify_mode = ssl.CERT_NONE
```

In [24]:
```
#Read HTML from the URL
top100url = 'https://www.gutenberg.org/browse/scores/top'
response = requests.get(top100url)
```

In [25]:
```
#Write a small function to check the status of web request
def status_check(r):
    if r.status_code==200:
        print("Success!")
        return 1
    else:
        print("Failed!")
        return -1
```

In [26]:
```
status_check(response)
```

Out[26]:
```
Success!
1
```

In [27]:
```
#Decode the response and pass on to BeautifulSoup for HTML parsing
contents = response.content.decode(response.encoding)
```

In [28]:
```
soup = BeautifulSoup(contents, 'html.parser')
```

In [29]:
```
#Find all the href tags and store them in the list of links. Check how the list looks like - print first 30 elements
lst_links=[]
for link in soup.find_all('a'):
    #print(link.get('href'))
    lst_links.append(link.get('href'))
```

In [30]:
```
lst_links[:30]
```

Out[30]:
```
['/',
 '/about/',
 '/about/',
 '/policy/collection_development.html',
 '/about/contact_information.html',
 '/about/background/',
 '/policy/permission.html',
 '/policy/privacy_policy.html',
 '/policy/terms_of_use.html',
 '/ebooks/',
 '/ebooks/',
 '/ebooks/bookshelf/',
 '/browse/scores/top',
 '/ebooks/offline_catalogs.html',
 '/help/',
 '/help/',
 '/help/copyright.html',
 '/help/errata.html',
 '/help/file_formats.html',
 '/help/faq.html',
 '/policy/',
 '/help/public_domain_ebook_submission.html',
 '/help/submitting_your_own_work.html',
 '/help/mobile.html',
 '/attic/',
 '/donate/',
 '/donate/',
 '#books-last1',
 '#authors-last1',
 '#books-last7']
```

In [31]:
```
#Use regular expression to find the numeric digits in these links.These are the file number for the Top 100 books.
#Initialize empty list to hold the file numbers
booknum=[]
```

In [32]:
```
for i in range(19,119):
    link=lst_links[i]
    link=link.strip()
    # Regular expression to find the numeric digits in the link (href) string
    n=re.findall('[0-9]+',link)
    if len(n)==1:
        # Append the filenumber casted as integer
        booknum.append(int(n[0]))
```

In [33]:
```
#Print the file numbers
print ("\nThe file numbers for the top 100 ebooks on Gutenberg are shown below\n"+"-"*70)
print(booknum)
```

```
The file numbers for the top 100 ebooks on Gutenberg are shown below
----------------------------------------------------------------------
[1, 1, 7, 7, 30, 30, 1513, 2641, 145, 37106, 16389, 67979, 2701, 100, 394, 6761, 2160, 4085, 6593, 1259, 5197, 84, 1
342, 11, 69984, 1952, 64317, 98, 1080, 174, 25344, 2542, 408, 1661, 23, 1400, 345, 5200, 1260, 43, 69983, 844, 76, 4
300, 46, 20228, 2591, 219, 6130, 1727, 28054, 1232, 205, 2814, 2554, 42108, 1184, 69988, 1497, 36, 15399, 2600, 74,
16, 768, 2852, 7370, 5740, 3206, 45, 11030, 69989, 3207, 69985, 69976, 215, 514, 996, 120, 69979, 2680, 4363, 5827,
55, 3825, 30254, 48438, 67098, 1250, 851, 16328, 35]
```

In [34]:
```
#How does the soup object's text look like? Use .text() method and print only first 2000 characters (i.e. do not prin
print(soup.text[:2000])
```

Top 100 | Project Gutenberg

Menu▾

About
▾

▾

About Project Gutenberg
Collection Development
Contact Us

History & Philosophy
Permissions & License
Privacy Policy
Terms of Use

Search and Browse
▾

▾

Book Search
Bookshelves
Frequently Downloaded
Offline Catalogs

Help
▾

▾

All help topics →
Copyright Procedures
Errata, Fixes and Bug Reports
File Formats
Frequently Asked Questions
Policies →
Public Domain eBook Submission
Submitting Your Own Work
Tablets, Phones and eReaders
The Attic →

Donate

Donation

Frequently Viewed or Downloaded
These listings are based on the number of times each eBook gets downloaded.
Multiple downloads from the same Internet address on the same day count as one download, and addresses that do
wnload more than 100 eBooks in a day are considered robots and are not counted.

Downloaded Books
2023-02-08321465
last 7 days1979137
last 30 days7987892

Top 100 EBooks yesterday
Top 100 Authors yesterday
Top 100 EBooks last 7 days
Top 100 Authors last 7 days
Top 100 EBooks last 30 days
Top 100 Authors last 30 days

Top 100 EBooks yesterday

Romeo and Juliet by William Shakespeare (7331)
A Room with a View by E. M. Forster (6198)
Middlemarch by George Eliot (6057)
Little Women; Or, Meg, Jo, Beth, and Amy by Louisa May Alcott (5688)
The Enchanted April by Elizabeth Von Arnim (5598)
The Blue Castle: a novel by L. M. Montgomery (5469)
Moby Dick; Or, The Whale by Herman Melville (5408)
The Complete Works of William Shakespeare by William Shakespeare (5368)
Cranford by Elizabeth Cleghorn Gaskell (5283)

```
The Adventures of Ferdinand Count Fathom — Complete by T.  Smollett (5270)
The Expedition of Humphry Clinker by T.  Smollett (5131)
The Adventures of Roderick Random by T.  Smollett (5088)
History of Tom Jones, a Foundling by Henry Fielding (4838)
Twenty Years After by Alexandre Dumas (4691)
My Life — Volume 1 by Richard Wagner (4687)
Frankenstein; Or, The Moder
```

In [35]:
```
#Search in the extracted text (using regular expression) from the soup object to find the names of top 100 Ebooks (Ye
lst_titles_temp=[]
```

In [36]:
```
#Create a starting index. It should point at the text "Top 100 Ebooks yesterday". Hint: Use splitlines() method of th
start_idx=soup.text.splitlines().index('Top 100 EBooks yesterday')
```

In [37]:
```
#Loop 1-100 to add the strings of next 100 lines to this temporary list. Hint: splitlines() method
for i in range(100):
    lst_titles_temp.append(soup.text.splitlines()[start_idx+2+i])
```

In [38]:
```
#Use regular expression to extract only text from the name strings and append to an empty list
lst_titles=[]
for i in range(100):
    id1,id2=re.match('^[a-zA-Z ]*',lst_titles_temp[i]).span()
    lst_titles.append(lst_titles_temp[i][id1:id2])
```

In [39]:
```
#Print the list of title
for l in lst_titles:
    print(l)
```

```
Top
Top
Top
Top


Top

Romeo and Juliet by William Shakespeare
A Room with a View by E
Middlemarch by George Eliot
Little Women
The Enchanted April by Elizabeth Von Arnim
The Blue Castle
Moby Dick
The Complete Works of William Shakespeare by William Shakespeare
Cranford by Elizabeth Cleghorn Gaskell
The Adventures of Ferdinand Count Fathom
The Expedition of Humphry Clinker by T
The Adventures of Roderick Random by T
History of Tom Jones
Twenty Years After by Alexandre Dumas
My Life
Frankenstein
Pride and Prejudice by Jane Austen
Alice
The mystery of Central Park by Nellie Bly
The Yellow Wallpaper by Charlotte Perkins Gilman
The Great Gatsby by F
A Tale of Two Cities by Charles Dickens
A Modest Proposal by Jonathan Swift
The Picture of Dorian Gray by Oscar Wilde
The Scarlet Letter by Nathaniel Hawthorne
A Doll
The Souls of Black Folk by W
The Adventures of Sherlock Holmes by Arthur Conan Doyle
Narrative of the Life of Frederick Douglass
Great Expectations by Charles Dickens
Dracula by Bram Stoker
Metamorphosis by Franz Kafka
Jane Eyre
The Strange Case of Dr
An original theory or new hypothesis of the Universe by Thomas Wright
The Importance of Being Earnest
```

```
Adventures of Huckleberry Finn by Mark Twain
Ulysses by James Joyce
A Christmas Carol in Prose
Noli Me Tangere by Jos
Grimms
Heart of Darkness by Joseph Conrad
The Iliad by Homer
The Odyssey by Homer
The Brothers Karamazov by Fyodor Dostoyevsky
The Prince by Niccol
Walden
Dubliners by James Joyce
Crime and Punishment by Fyodor Dostoyevsky
The Slang Dictionary
The Count of Monte Cristo
The house on the cliff by Franklin W
The Republic by Plato
The War of the Worlds by H
The Interesting Narrative of the Life of Olaudah Equiano
War and Peace by graf Leo Tolstoy
The Adventures of Tom Sawyer
Peter Pan by J
Wuthering Heights by Emily Bront
The Hound of the Baskervilles by Arthur Conan Doyle
Second Treatise of Government by John Locke
Tractatus Logico
Moby Multiple Language Lists of Common Words by Grady Ward
Anne of Green Gables by L
Incidents in the Life of a Slave Girl
Handicraft for boys by A
Leviathan by Thomas Hobbes
Ilex cassine
All about Little Boy Blue by Emma Gelders Sterne
The Call of the Wild by Jack London
Little Women by Louisa May Alcott
Don Quixote by Miguel de Cervantes Saavedra
Treasure Island by Robert Louis Stevenson
Star Book No
Meditations by Emperor of Rome Marcus Aurelius
Beyond Good and Evil by Friedrich Wilhelm Nietzsche
The Problems of Philosophy by Bertrand Russell
The Wonderful Wizard of Oz by L
Pygmalion by Bernard Shaw
The Romance of Lust
```

```
Rizal
Winnie
Anthem by Ayn Rand
Narrative of the Captivity and Restoration of Mrs
Beowulf
The Time Machine by H
Emma by Jane Austen
Essays of Michel de Montaigne
Gulliver
Candide by Voltaire
The botanist
The Reign of Greed by Jos
```

Data Wrangling with Python: Activity 10, page 295

In [ ]:
```
    #Retrieves and prints basic data about a movie (title entered by user) from the web (OMDB database)
    #If a poster of the movie could be found, it downloads the file and saves at a user-specified location
```

In [2]:
```
import urllib.request, urllib.parse, urllib.error
import json
```

In [3]:
```
#Load the secret API key (you have to get one from OMDB website and use that, 1000 daily limit) from a JSON file, sto
```

In [7]:
```
with open('APIkeys.json') as f:
    keys = json.load(f)
    omdbapi = keys['keys']
```

In [9]:
```
#print(omdbapi)
```

The final URL to be passed should look like: http://www.omdbapi.com/?i=tt3896198&apikey=ca392c03

Create a variable apikey with the last portion of the URL ("&apikey=secretapikey"), where secretapikey is your own API key (an actual code)

In [10]:
```
serviceurl = 'http://www.omdbapi.com/?'
apikey = '&apikey='+omdbapi
```

Write a utility function print_json to print nicely the movie data from a JSON file (which we will get from the portal)

In [11]:
```
def print_json(json_data):
    list_keys=['Title', 'Year', 'Rated', 'Released', 'Runtime', 'Genre', 'Director', 'Writer',
               'Actors', 'Plot', 'Language', 'Country', 'Awards', 'Ratings',
               'Metascore', 'imdbRating', 'imdbVotes', 'imdbID']
    print("="*50)
    for k in list_keys:
        if k in list(json_data.keys()):
            print(f"        {k}: {json_data[k]}")
    print("="*50)
```

Write a utility function to download a poster of the movie based on the information from the jason dataset and save in your local folder

Use os module
The poster data is stored in the JSON key 'Poster'
You may want to split the name of the Poster file and extract the file extension only. Let's say the extension is *'jpg'*.
Then later join this extension to the movie name and create a filename like *movie.jpg*
Use the Python command open to open a file and write the poster data. Close the file after done.
This function may not return anything. It just saves the poster data as an image file.

In [12]:
```
def save_poster(json_data):
    import os
    title = json_data['Title']
    poster_url = json_data['Poster']
    # Splits the poster url by '.' and picks up the last string as file extension
    poster_file_extension=poster_url.split('.')[-1]
    # Reads the image file from web
    poster_data = urllib.request.urlopen(poster_url).read()

    savelocation=os.getcwd()+'\\'+'Posters'+'\\'
    # Creates new directory if the directory does not exist. Otherwise, just use the existing path.
    if not os.path.isdir(savelocation):
        os.mkdir(savelocation)

    filename=savelocation+str(title)+'.'+poster_file_extension
    f=open(filename,'wb')
    f.write(poster_data)
    f.close()
```

Write a utility function search_movie to search a movie by its name, print the downloaded JSON data (use the print_json function for this) and save the movie poster in the local folder (use save_poster function for this)

```
Use try-except loop for this i.e. try to connect to the web portal, if successful proceed but if
not (i.e. exception raised) then just print an error message
Here use the previously created variables serviceurl and apikey
You have to pass on a dictionary with a key t and the movie name as the corresponding value to
urllib.parse.urlencode() function and then add the serviceurl and apikey to the output of the
function to construct the full URL
This URL will be used for accessing the data
The JSON data has a key called Response. If it is True, that means the read was successful. Check
this before processing the data. If not successful, then print the JSON key Error, which will
contain the appropriate error message returned by the movie database.
```

In [13]:
```
def search_movie(title):
    try:
        url = serviceurl + urllib.parse.urlencode({'t': str(title)})+apikey
        print(f'Retrieving the data of "{title}" now... ')
        print(url)
        uh = urllib.request.urlopen(url)
        data = uh.read()
        json_data=json.loads(data)

        if json_data['Response']=='True':
            print_json(json_data)
            # Asks user whether to download the poster of the movie
            if json_data['Poster']!='N/A':
                save_poster(json_data)
        else:
            print("Error encountered: ",json_data['Error'])

    except urllib.error.URLError as e:
        print(f"ERROR: {e.reason}")
```

In [15]:
```
#Testing of search move function
search_movie("Enter The Dragon")
```

```
Retrieving the data of "Enter The Dragon" now...
http://www.omdbapi.com/?t=Enter+The+Dragon&apikey=ca392c03
================================================
        Title: Enter the Dragon
        Year: 1973
        Rated: R
        Released: 19 Aug 1973
        Runtime: 102 min
        Genre: Action, Crime, Drama
        Director: Robert Clouse
        Writer: Michael Allin, Bruce Lee
        Actors: Bruce Lee, John Saxon, Jim Kelly
        Plot: A Shaolin martial artist travels to an island fortress to spy on an opium lord - who is also a former
monk from his temple - under the guise of attending a fighting tournament.
        Language: English, Cantonese
        Country: Hong Kong, United States
        Awards: 1 win
        Ratings: [{'Source': 'Internet Movie Database', 'Value': '7.6/10'}, {'Source': 'Rotten Tomatoes', 'Value': '
95%'}, {'Source': 'Metacritic', 'Value': '83/100'}]
        Metascore: 83
        imdbRating: 7.6
        imdbVotes: 106,190
        imdbID: tt0070034
================================================
```

Test search_movie function by entering "Random_error" (obviously this will not be found and you should be able to check whether your error catching code is working properly)

In [16]:
```
search_movie("Random_error")
```

```
Retrieving the data of "Random_error" now...
http://www.omdbapi.com/?t=Random_error&apikey=ca392c03
Error encountered:  Movie not found!
```

Connect to the Twitter API and do a simple data pull

Connect to the Twitter API and do a simple data pull a. If you don't have a twitter account – create one at twitter.com/signup (you can delete the account after this assignment) b. Sign in to apps.twitter.com c. Click "Create New App" d. Give your app a name and description e. Agree to the developer agreement – you will want to make sure to indicate this is for a class project, and this step can take several days to get through, so don't wait until last minute to complete this portion of the assignment f. Create an access token g. You should receive a consumer key and a token h. Using either the instructions from the book on connecting to an API or for help look here – pull back data searching for "Bellevue University" and "Data

Science" (or something else you are interested in) i. How to Create a Twitter App and API Interface via Python. (Grogan, 2016) ii. Welcome Python-Twitter's Documentation! (The Python-Twitter Developers, 2016)

In [18]:
```
pip install tweepy
```

```
Collecting tweepyNote: you may need to restart the kernel to use updated packages.

  Downloading tweepy-4.12.1-py3-none-any.whl (101 kB)
     ------------------------------------ 101.6/101.6 kB 5.7 MB/s eta 0:00:00
Collecting requests-oauthlib<2,>=1.2.0
  Downloading requests_oauthlib-1.3.1-py2.py3-none-any.whl (23 kB)
Requirement already satisfied: requests<3,>=2.27.0 in c:\users\atanu\anaconda3\lib\site-packages (from tweepy) (2.2
8.1)
Collecting oauthlib<4,>=3.2.0
  Downloading oauthlib-3.2.2-py3-none-any.whl (151 kB)
     ------------------------------------ 151.7/151.7 kB 4.6 MB/s eta 0:00:00
Requirement already satisfied: idna<4,>=2.5 in c:\users\atanu\anaconda3\lib\site-packages (from requests<3,>=2.27.0-
>tweepy) (3.3)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\atanu\anaconda3\lib\site-packages (from requests<3,
>=2.27.0->tweepy) (1.26.11)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\atanu\anaconda3\lib\site-packages (from requests<3,>=
2.27.0->tweepy) (2022.9.14)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\atanu\anaconda3\lib\site-packages (from requests
<3,>=2.27.0->tweepy) (2.0.4)
Installing collected packages: oauthlib, requests-oauthlib, tweepy
Successfully installed oauthlib-3.2.2 requests-oauthlib-1.3.1 tweepy-4.12.1
```

In [19]:
```
# I am using tweepy library to get data from twitter.
import tweepy
```

In [21]:
```
# Lets save the credentials in the variable
consumer_key = "f6ooxq3L2J1GLn1ZTQfq6ZP7a"  #same as api key
consumer_secret = "PGIMSOVMGw4P3ffowHfjFo0Q9iAUt6ExzhFRjllrlt8nhj4Gwx"  #same as api secret
access_key = "1021444442-zXCvSGERjJZ7m8e8nTde3Ic2D7RDqC9uE3r7uN9"
access_secret = "8pbfIHZcYRksoCJDk8JlyzCtTgy3hhYBp4nEr9Uhtp4fu"
```

In [25]:
```
#Lets create the AUTH and create api object

# Twitter authentication
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_key, access_secret)

# Creating an API object
api = tweepy.API(auth)
```

In [29]:
```
username_tweets = tweepy.Cursor(api.search_tweets, q="@elonmusk", tweet_mode='extended').items(5)
```

In [30]:
```
for tweet in username_tweets:
    text = tweet._json["full_text"]
    print(text)    #using different attributes
    print(tweet.favorite_count)
    print(tweet.retweet_count)
    print(tweet.created_at)
```

```
RT @elonmusk: True
0
4798
2023-02-16 00:39:25+00:00
@elonmusk @POTUS @Tesla @siddharthkara
0
0
2023-02-16 00:39:25+00:00
RT @elonmusk: And has 🔥🔥 style https://t.co/9rcEtu9w1Z
0
22544
2023-02-16 00:39:24+00:00
@elonmusk anj lu
0
0
2023-02-16 00:39:24+00:00
@elonmusk @Teslaconomics If it's off-grid, something has to fuel the generator.
0
0
2023-02-16 00:39:24+00:00
```

In [32]:
```
#below is the example of particular tweet.
hashtag_tweets = tweepy.Cursor(api.search_tweets, q="#VaccinationDrive", tweet_mode='extended').items(5)
for tweet in hashtag_tweets:
    text = tweet._json["full_text"]
    print(text)
```

```
#VaccinationDrive Karnataka cumulatively upto 3.30pm on February 15 has administered 1/2 dose vaccine to 117556416 p
eople aged 12-14; 15-17; 18-44; 45-60 &amp;amp; 60+. Total 122142043 doses administered: @DHFWKA. @MoHFW_INDIA @sputnikv
accine @mangalurucorp @SerumInstIndia @BharatBiotech https://t.co/1sL9itkySr
MR campaign running smoothly at West District Delhi #MRCampaign #VaccinationDrive #healthcare
@iasdanishashraf
@MoHFW_INDIA
@CMODelhi
 @LtGovDelhi https://t.co/mR5dAJ17i9
Health worker sensitizing people visited Hospital #MRCampaign #VaccinationDrive #healthcare  @iasdanishashraf  @MoHF
W_INDIA  @CMODelhi   @LtGovDelhi https://t.co/qR1LGlMYjO
RT @DelhiDfw: After receiving the MR vaccine, children aged 9 months to 5 years pose in front of the camera, their s
hining eyes motivate us…
After receiving the MR vaccine, children aged 9 months to 5 years pose in front of the camera, their shining eyes mo
tivate us to make MR a success.
#MRCampaign #VaccinationDrive #healthcare
@iasdanishashraf
@MoHFW_INDIA
@CMODelhi
@LtGovDelhi https://t.co/cEc8FAmiq8
```

4. Using one of the datasets provided in Weeks 7 & 8, or a dataset of your own, choose 3 of the following visualizations to complete. You must submit via PDF along with your code. You are free to use Matplotlib, Seaborn or another package if you prefer. a. Line b. Scatter c. Bar d. Histogram e. Density Plot f. Pie Chart

In [1]:
```
import pandas as pd
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
candy = pd.read_csv("candyhierarchy2017.csv", encoding='ISO-8859-1')
candy.head(2)
```

Out[1]:

	Internal ID	Q1: GOING OUT?	Q2: GENDER	Q3: AGE	Q4: COUNTRY	Q5: STATE, PROVINCE, COUNTY, ETC	Q6 | 100 Grand Bar	Q6 | Anonymous brown globs that come in black and orange wrappers\t(a.k.a. Mary Janes)	Q6 | Any full-sized candy bar	Q6 | Black Jacks	Q6 | Bonkers (the candy)	Q6 | Bonkers (the board game)	Q6 | Bottle Caps	Box'o'Ra
0	90258773	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	90272821	No	Male	44	USA	NM	MEH	DESPAIR	JOY	MEH	DESPAIR	DESPAIR	DESPAIR	DES

In [2]:
```
candy.shape
```

Out[2]:
```
(2460, 120)
```

Histogram

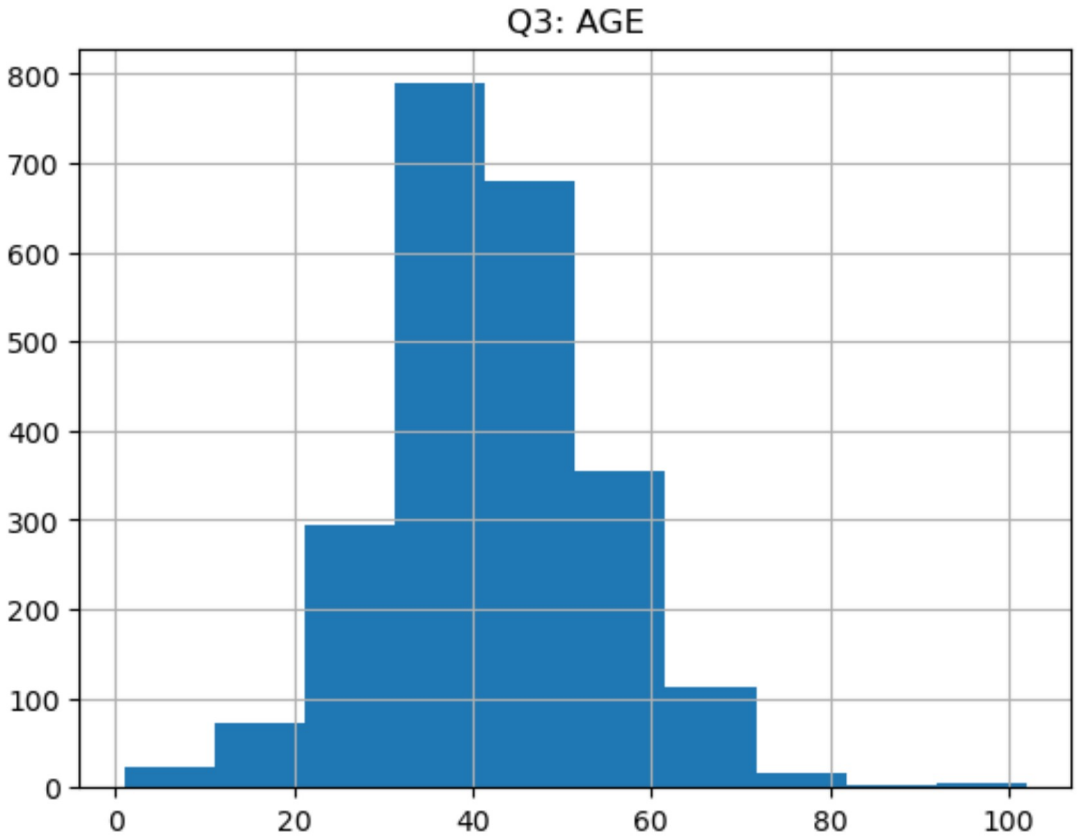
In [7]:
```
candy = candy.dropna(subset=['Q3: AGE'])
candy = candy[candy["Q3: AGE"].str.contains(r'^\d+$')]
```

In [12]:
```
candy['Q3: AGE']=candy['Q3: AGE'].astype(int)
```

In [15]:
```
candy = candy[candy["Q3: AGE"] <= 150]
```

In [16]:
```
candy.hist(column = 'Q3: AGE')
```

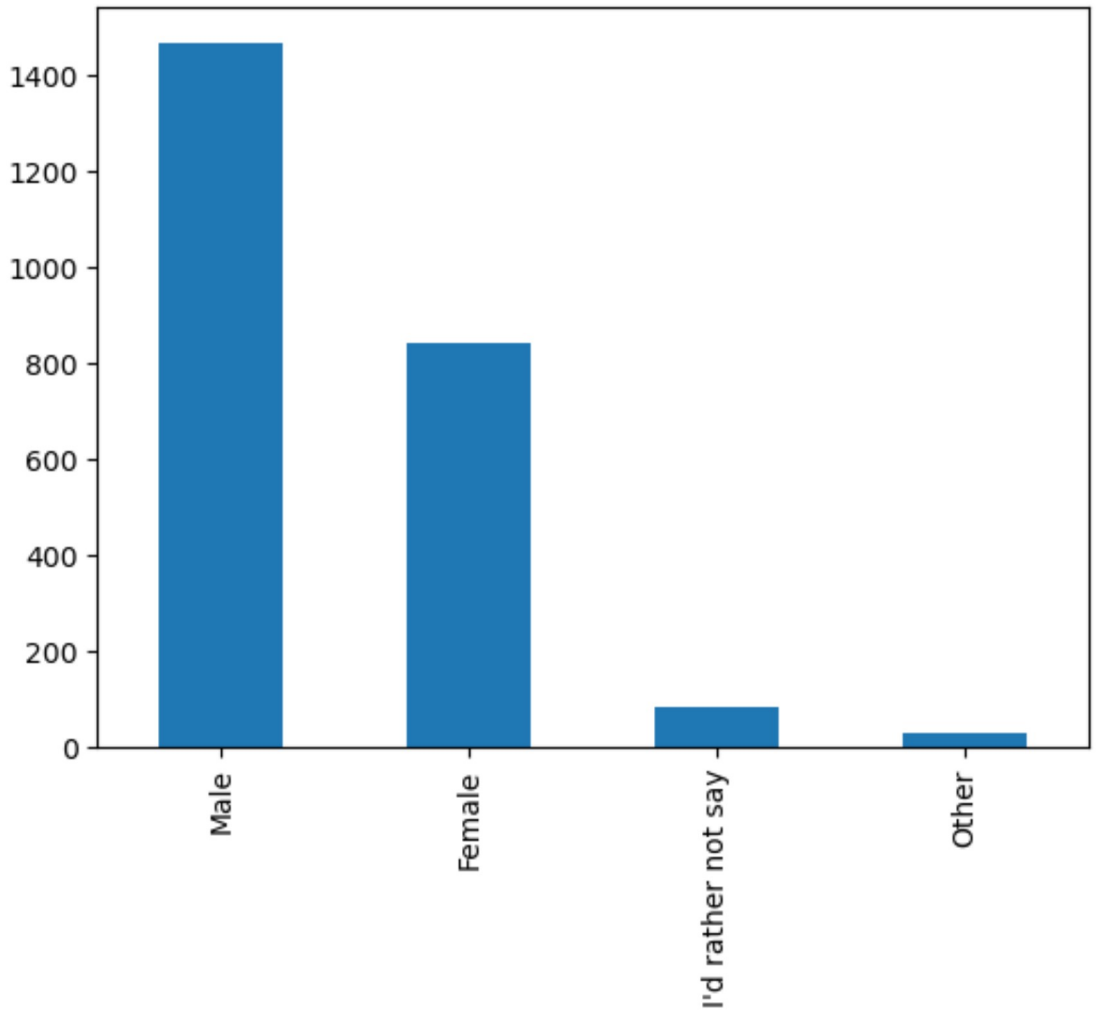
Out[16]:
```
array([[<AxesSubplot:title={'center':'Q3: AGE'}>]], dtype=object)
```



Bar

In [57]:
```
candy['Q2: GENDER'].value_counts().plot(kind='bar')
```

Out[57]:
```
<AxesSubplot:>
```



Pie-Chart

In [18]:
```
candy['Q6 | 100 Grand Bar'].value_counts()
```

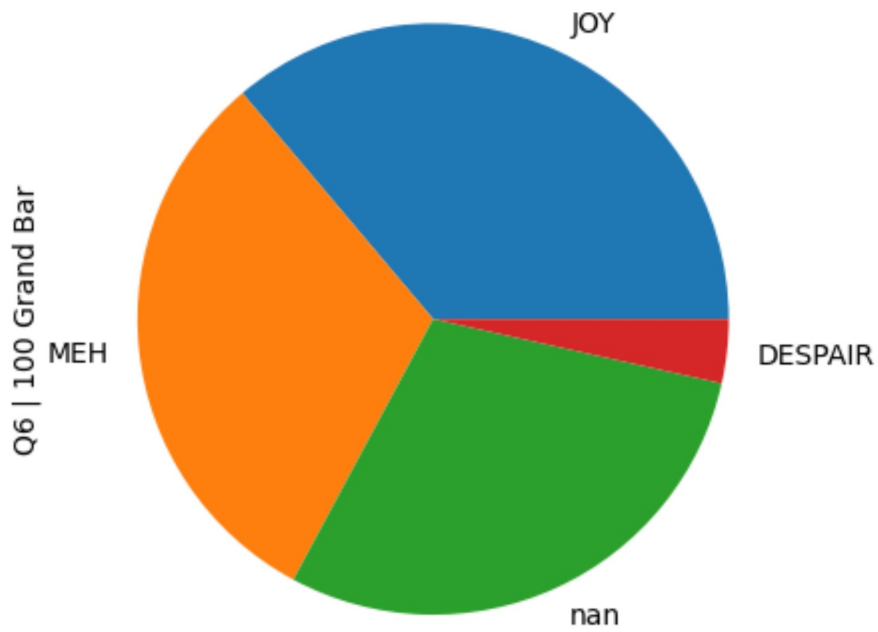
Out[18]:
```
JOY        849
MEH        728
DESPAIR     82
Name: Q6 | 100 Grand Bar, dtype: int64
```

In [22]:
```
candy = candy.dropna(subset=['Q6 | 100 Grand Bar'])
```

In [26]:
```
candy['Q6 | 100 Grand Bar']=candy['Q6 | 100 Grand Bar'].astype(pd.StringDtype())
```

In [32]:
```
candy['Q6 | 100 Grand Bar'].value_counts().plot(kind='pie')
```

Out[32]:
```
<AxesSubplot:ylabel='Q6 | 100 Grand Bar'>
```



In [33]:
```
pip install nbconvert
```

```
Requirement already satisfied: nbconvert in c:\users\atanu\anaconda3\lib\site-packages (6.4.4)
Requirement already satisfied: traitlets>=5.0 in c:\users\atanu\anaconda3\lib\site-packages (from nbconvert) (5.1.1)
Requirement already satisfied: nbformat>=4.4 in c:\users\atanu\anaconda3\lib\site-packages (from nbconvert) (5.5.0)
Requirement already satisfied: beautifulsoup4 in c:\users\atanu\anaconda3\lib\site-packages (from nbconvert) (4.11.
1)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in c:\users\atanu\anaconda3\lib\site-packages (from nbconvert)
(0.5.13)
Requirement already satisfied: jupyterlab-pygments in c:\users\atanu\anaconda3\lib\site-packages (from nbconvert)
(0.1.2)
Requirement already satisfied: jinja2>=2.4 in c:\users\atanu\anaconda3\lib\site-packages (from nbconvert) (2.11.3)
Requirement already satisfied: pygments>=2.4.1 in c:\users\atanu\anaconda3\lib\site-packages (from nbconvert) (2.11.
2)
Requirement already satisfied: entrypoints>=0.2.2 in c:\users\atanu\anaconda3\lib\site-packages (from nbconvert) (0.
4)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\atanu\anaconda3\lib\site-packages (from nbconvert)
(1.5.0)
Requirement already satisfied: jupyter-core in c:\users\atanu\anaconda3\lib\site-packages (from nbconvert) (4.11.1)
Requirement already satisfied: bleach in c:\users\atanu\anaconda3\lib\site-packages (from nbconvert) (4.1.0)
Requirement already satisfied: defusedxml in c:\users\atanu\anaconda3\lib\site-packages (from nbconvert) (0.7.1)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\users\atanu\anaconda3\lib\site-packages (from nbconvert) (0.
8.4)
Requirement already satisfied: testpath in c:\users\atanu\anaconda3\lib\site-packages (from nbconvert) (0.6.0)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\atanu\anaconda3\lib\site-packages (from jinja2>=2.4->nbc
onvert) (2.0.1)
Requirement already satisfied: nest-asyncio in c:\users\atanu\anaconda3\lib\site-packages (from nbclient<0.6.0,>=0.
5.0->nbconvert) (1.5.5)
Requirement already satisfied: jupyter-client>=6.1.5 in c:\users\atanu\anaconda3\lib\site-packages (from nbclient<0.
6.0,>=0.5.0->nbconvert) (7.3.4)
Requirement already satisfied: fastjsonschema in c:\users\atanu\anaconda3\lib\site-packages (from nbformat>=4.4->nbc
onvert) (2.16.2)
Requirement already satisfied: jsonschema>=2.6 in c:\users\atanu\anaconda3\lib\site-packages (from nbformat>=4.4->nb
convert) (4.16.0)
Requirement already satisfied: soupsieve>1.2 in c:\users\atanu\anaconda3\lib\site-packages (from beautifulsoup4->nbc
onvert) (2.3.1)
Requirement already satisfied: packaging in c:\users\atanu\anaconda3\lib\site-packages (from bleach->nbconvert) (21.
3)
Requirement already satisfied: webencodings in c:\users\atanu\anaconda3\lib\site-packages (from bleach->nbconvert)
(0.5.1)
Requirement already satisfied: six>=1.9.0 in c:\users\atanu\anaconda3\lib\site-packages (from bleach->nbconvert) (1.
16.0)
Requirement already satisfied: pywin32>=1.0 in c:\users\atanu\anaconda3\lib\site-packages (from jupyter-core->nbconv
ert) (302)
Requirement already satisfied: attrs>=17.4.0 in c:\users\atanu\anaconda3\lib\site-packages (from jsonschema>=2.6->nb
format>=4.4->nbconvert) (21.4.0)
```

```
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in c:\users\atanu\anaconda3\lib\site-pa
ckages (from jsonschema>=2.6->nbformat>=4.4->nbconvert) (0.18.0)
Requirement already satisfied: pyzmq>=23.0 in c:\users\atanu\anaconda3\lib\site-packages (from jupyter-client>=6.1.5
->nbclient<0.6.0,>=0.5.0->nbconvert) (23.2.0)
Requirement already satisfied: tornado>=6.0 in c:\users\atanu\anaconda3\lib\site-packages (from jupyter-client>=6.1.
5->nbclient<0.6.0,>=0.5.0->nbconvert) (6.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\atanu\anaconda3\lib\site-packages (from jupyter-cl
ient>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (2.8.2)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\atanu\anaconda3\lib\site-packages (from packagin
g->bleach->nbconvert) (3.0.9)
Note: you may need to restart the kernel to use updated packages.
```

In [ ]: