In [69]:
```
import urllib.request, urllib.parse, urllib.error
import requests
from bs4 import BeautifulSoup
import ssl
import re
```

Activity 9

In [96]:
```
ctx = ssl.create_default_context()
ctx.check_hostname = False
ctx.verify_mode = ssl.CERT_NONE
```

In [97]:
```
# Read HTML
top100url = 'https://www.gutenberg.org/browse/scores/top'
response = requests.get(top100url)
```

In [98]:
```
#Check Web status
def status_check(r):
    if r.status_code==200:
        print("Success!")
        return 1
    else:
        print("Failed!")
        return -1
```

In [99]:
```
#Display response
status_check(response)
```

```
Success!
```

Out[99]:
```
1
```

In [100]:
```
#Beging parsing with beautifulsoup
contents = response.content.decode(response.encoding)
```

In [101]:
```
soup = BeautifulSoup(contents, 'html.parser')
```

In [102]:
```
lst_links=[]
```

In [103]:
```
#Look for href tags
for link in soup.find_all('a'):
    lst_links.append(link.get('href'))
```

```
In [104]:  #Display tags
           lst_links[:30]
```

```
Out[104]: ['/',
            '/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 [105]:  booknum=[]
```

```
In [106]:  # Find numbers in link and append to list
           for i in range(19,119):
               link=lst_links[i]
               link=link.strip()
               n=re.findall('[0-9]+',link)
               if len(n)==1:
                   booknum.append(int(n[0]))
```

In [107]:
```
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, 84, 1342, 25344, 1952, 46, 1080, 1250, 2701, 11, 2542, 844, 5200, 98, 76, 23, 1232, 16328, 43, 1
661, 12345, 63737, 205, 345, 160, 174, 6130, 3825, 2591, 1260, 74, 408, 16, 63719, 1064, 63749, 1400, 2500, 215, 320
7, 41, 4300, 42324, 2852, 42108, 63742, 120, 1497, 19942, 219, 1404, 203, 58585, 2600, 20203, 209, 55, 63745, 5740, 2
7827, 63729, 1727, 28054, 4934, 63741, 63751, 376, 2554, 514, 158, 34901, 2148, 1184, 36, 135, 2814, 996, 11030, 45,
4363, 38269, 63733, 113, 3600, 22381, 7370, 140]
```

In [108]:
```
print(soup.text[:2000])
```

```




Top 100 | Project Gutenberg
```

In [109]:
```
lst_titles_temp=[]
```

In [110]:
```
#Index
start_idx=soup.text.splitlines().index('Top 100 EBooks yesterday')
```

In [111]:
```
#Add 100 lines to list
for i in range(100):
    lst_titles_temp.append(soup.text.splitlines()[start_idx+2+i])
```

In [112]:

```
#Pull text from 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 [113]:
```
for l in lst_titles:
    print(l)
```

```
Top
Top
Top
Top


Top

Frankenstein
Pride and Prejudice by Jane Austen
The Scarlet Letter by Nathaniel Hawthorne
The Yellow Wallpaper by Charlotte Perkins Gilman
A Christmas Carol in Prose
A Modest Proposal by Jonathan Swift
Anthem by Ayn Rand
Moby Dick
Alice
Et dukkehjem
The Importance of Being Earnest
Metamorphosis by Franz Kafka
A Tale of Two Cities by Charles Dickens
Adventures of Huckleberry Finn by Mark Twain
Narrative of the Life of Frederick Douglass
Il Principe
Beowulf
The Strange Case of Dr
The Adventures of Sherlock Holmes by Arthur Conan Doyle
Friday
Egypt and its Monuments by Robert Hichens and Jules Gue
Walden
Dracula by Bram Stoker
The Awakening
The Picture of Dorian Gray by Oscar Wilde
The Iliad by Homer
Pygmalion by Bernard Shaw
Grimms
Jane Eyre
The Adventures of Tom Sawyer by Mark Twain
The Souls of Black Folk by W
Peter Pan by J
The Louvre
The Masque of the Red Death by Edgar Allan Poe
Mimsy
Great Expectations by Charles Dickens
```

Siddhartha by Hermann Hesse
The Call of the Wild by Jack London
Leviathan by Thomas Hobbes
The Legend of Sleepy Hollow by Washington Irving
Ulysses by James Joyce
Frankenstein
The Hound of the Baskervilles by Arthur Conan Doyle
The Slang Dictionary
Sixty
Treasure Island by Robert Louis Stevenson
The Republic by Plato
Candide by Voltaire
Heart of Darkness by Joseph Conrad
The Federalist Papers by Alexander Hamilton and John Jay and James Madison
Uncle Tom
The Prophet by Kahlil Gibran
War and Peace by graf Leo Tolstoy
Autobiography of Benjamin Franklin by Benjamin Franklin
The Turn of the Screw by Henry James
The Wonderful Wizard of Oz by L
Beyond Rope and Fence by David Grew
Tractatus Logico
The Kama Sutra of Vatsyayana by Vatsyayana
In the Garden of Delight by Lily Hardy Hammond
The Odyssey by Homer
The Brothers Karamazov by Fyodor Dostoyevsky
The Natural History of Wiltshire by John Aubrey
The Galactic Ghost by Mack Reynolds
The Derelict by William J
A Journal of the Plague Year by Daniel Defoe
Prestuplenie i nakazanie
Little Women by Louisa May Alcott
Emma by Jane Austen
On Liberty by John Stuart Mill
The Works of Edgar Allan Poe
The Count of Monte Cristo
The War of the Worlds by H
Les Mis
Dubliners by James Joyce
Don Quixote by Miguel de Cervantes Saavedra
Incidents in the Life of a Slave Girl
Anne of Green Gables by L
Beyond Good and Evil by Friedrich Wilhelm Nietzsche
A History of the Philippines by David P
China and the Chinese by Edmond Plauchut
The Secret Garden by Frances Hodgson Burnett
Essays of Michel de Montaigne
Myths and Legends of Ancient Greece and Rome by E

```
Second Treatise of Government by John Locke
The Jungle by Upton Sinclair
Wuthering Heights by Emily Bront
The Autobiography of Benjamin Franklin by Benjamin Franklin
The Nursery Rhymes of England
The Confessions of St
Oliver Twist by Charles Dickens
Mary Boyle
```

Activity 10

In [114]:
```python
import urllib.request, urllib.parse, urllib.error
import json
```

In [139]:
```python
#Pull json file with API key.
# Json file wasn't saving new api key so I had to pivot
omdbapi = 'fc0ebc46'
```

In [140]:
```python
#create url for API access
serviceurl = 'http://www.omdbapi.com/?'
apikey = '&apikey='+omdbapi
```

In [141]:
```python
#print json data
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)
```

In [142]:
```
#Dowload poster
def save_poster(json_data):
    import os
    title = json_data['Title']
    poster_url = json_data['Poster']
    poster_file_extension=poster_url.split('.')[-1]
    poster_data = urllib.request.urlopen(poster_url).read()

    savelocation=os.getcwd()+'\\'+'Posters'+'\\'
    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()
```

In [143]:
```
#Search movie by name
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)
            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 [148]:
```
search_movie("Titanic")
```

```
Retrieving the data of "Titanic" now...
http://www.omdbapi.com/?t=Titanic&apikey=fc0ebc46 (http://www.omdbapi.com/?t=Titanic&apikey=fc0ebc46)
ERROR: Unauthorized
```

In [149]:
```
search_movie("Random_error")
```

```
Retrieving the data of "Random_error" now...
http://www.omdbapi.com/?t=Random_error&apikey=fc0ebc46 (http://www.omdbapi.com/?t=Random_error&apikey=fc0ebc46)
ERROR: Unauthorized
```

Twitter Assignment

In [60]:
```
import twitter
import tweepy as tw

#Setting up API connection
consumer_key = 'WrGNuVNNamkwXrzRyUvk6UU8O'
consumer_secret = 'K0UlOFcUGELMQK1GYBCam3QnVQArHFnaFi1rN0IsbGDTcJmFBu'
access_token_key = '1326395879360704512-nJBWIS1hpOmwexzym42exW5YlCYeKu'
access_token_secret = 'DgloHloWwbuow8JMoJOXHsWcuLkm37VIqjwnSYDZBiXCK'
auth = tw.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token_key, access_token_secret)
api = tw.API(auth, wait_on_rate_limit = True)
```

In [61]:
```
#Assigning parameters
keyword = "#datascience"
date = "2013-01-01"
```

In [63]:
```
#Pull 1000 tweets
tweets = tw.Cursor(api.search, q=keyword, lang="en", since=date).items(1000)
```

In [64]:
```
#Print tweet data
for tweet in tweets:
    print(tweet.text)
```

```
RT @Udemy_Coupons1: The Machine Learning Certification | 100%OFF #udemycoupon https://t.co/kSm5O11JAd (https://t.c
o/kSm5O11JAd)

#MachineLearning. #BigData #Analytic…
RT @Udemy_Coupons1: 100%OFF #UdemyCoupon Code For Today 14/11/2020 Daily Update https://t.co/4zNv91cLJj (https://
t.co/4zNv91cLJj)

#MachineLearning. #BigData #Analyt…
RT @Udemy_Coupons1: Vulnerability Analysis Course For Ethical Hacking | 100%OFF #udemycoupon https://t.co/lg9BbkLg
QG (https://t.co/lg9BbkLgQG)

#MachineLearning. #Bi…
RT @ProgrammingHero: Top Online Code Editors You Should Know

What do you prefer? Online or Local?
.
.
.
.
```

Visualizations

In [45]:
```
#Uploading old dataset from statistics class

df = pd.read_csv('C:/Users/nneam/OneDrive/Documents/540Assignments/heights.csv')
```

In [46]:

```
df.head(10)
```

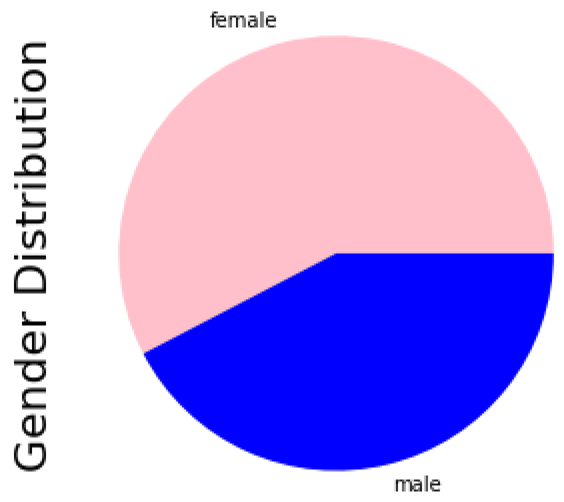
Out[46]:

	earn	height	sex	ed	age	race
0	50000.0	74.424439	male	16	45	white
1	60000.0	65.537543	female	16	58	white
2	30000.0	63.629198	female	16	29	white
3	50000.0	63.108562	female	16	91	other
4	51000.0	63.402484	female	17	39	white
5	9000.0	64.399508	female	15	26	white
6	29000.0	61.656326	female	12	49	white
7	32000.0	72.698544	male	17	46	white
8	2000.0	72.039467	male	15	21	hispanic
9	27000.0	72.234933	male	12	26	white

In [47]:
```
# Pie chart on gender

from matplotlib import pyplot as plt

fig, (ax1) = plt.subplots(ncols=1, figsize=(10, 5))
df.groupby('sex').size().plot(kind='pie', colors =['pink', 'blue'], ax=ax1)
ax1.set_ylabel('Gender Distribution', size=22)
plt.show()
```



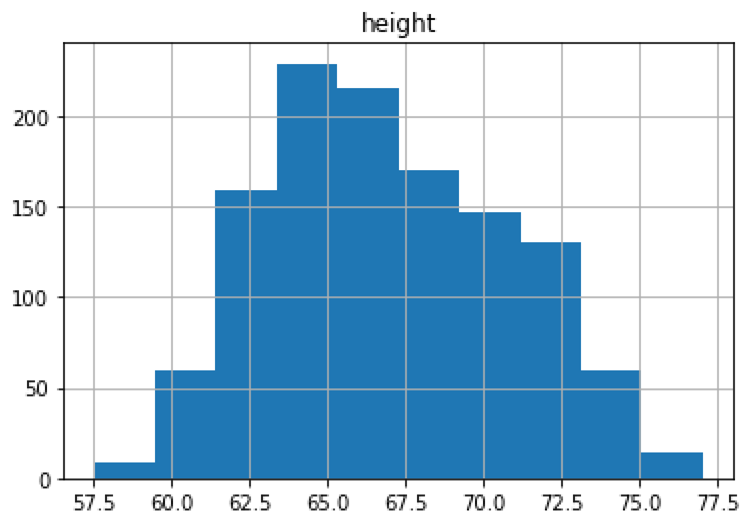
In [65]:

```python
#Histogram on height

df.hist(column='height')
```

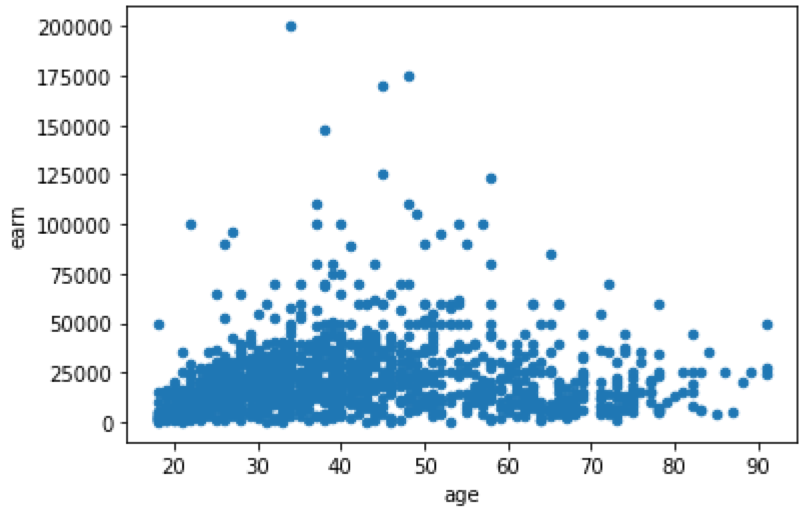
Out[65]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000023592994FD0>]],
      dtype=object)
```



In [64]:

```python
#Scatter plot on earnings by age
df.plot(x ='age', y='earn', kind = 'scatter')
plt.show()
```


earn
age