﻿import numpy as np

import xarray as xr

import matplotlib.pyplot as plt

import matplotlib as mpl

#STARTING EXERCISE

Lat = np.arange(38, 43, .01)

Lon = np.arange(-108, -103, .01)

Lon,Lat = np.meshgrid(Lon,Lat)

﻿H = -100\*(Lat - 40)\*\*2 - 400\*(Lon + 106)\*\*2 +8000

H

Out[4]:

array([[6000. , 6015.96 , 6031.84 , ..., 4071.63999999,

4047.83999999, 4023.95999999],

[6003.99 , 6019.95 , 6035.83 , ..., 4075.62999999,

4051.82999999, 4027.94999999],

[6007.96 , 6023.92 , 6039.8 , ..., 4079.59999999,

4055.79999999, 4031.91999999],

...,

[5517.91 , 5533.87 , 5549.75 , ..., 3589.54999999,

3565.74999999, 3541.86999999],

[5511.96 , 5527.92 , 5543.8 , ..., 3583.59999999,

3559.79999999, 3535.91999999],

[5505.99 , 5521.95 , 5537.83 , ..., 3577.62999999,

3553.82999999, 3529.94999999]])

﻿plt.pcolormesh(Lon,Lat,H, cmap = 'Purples')

Out[5]: <matplotlib.collections.QuadMesh at 0x3204417f0>



﻿#This function describes the Colorado Rockies

﻿﻿plt.contour(Lon, Lat, H)

Out[6]: <matplotlib.contour.QuadContourSet at 0x3216c97b8>



﻿plt.contourf(Lon, Lat, H)

Out[7]: <matplotlib.contour.QuadContourSet at 0x32179b1d0>



﻿plt.pcolormesh(Lon,Lat,H, cmap = 'Purples')

plt.colorbar()

plt.contour(Lon, Lat, H)

plt.title("Colorado Rockies Altitudes", fontsize=14)

Out[8]: Text(0.5, 1.0, 'Colorado Rockies Altitudes')



#FILE INPUT/OUTPUT

﻿﻿tdata = np.loadtxt('/Users/roryeggleston/Downloads/populations.txt')

tdata

Out[9]:

array([[ 1900., 30000., 4000., 48300.],

[ 1901., 47200., 6100., 48200.],

[ 1902., 70200., 9800., 41500.],

[ 1903., 77400., 35200., 38200.],

[ 1904., 36300., 59400., 40600.],

[ 1905., 20600., 41700., 39800.],

[ 1906., 18100., 19000., 38600.],

[ 1907., 21400., 13000., 42300.],

[ 1908., 22000., 8300., 44500.],

[ 1909., 25400., 9100., 42100.],

[ 1910., 27100., 7400., 46000.],

[ 1911., 40300., 8000., 46800.],

[ 1912., 57000., 12300., 43800.],

[ 1913., 76600., 19500., 40900.],

[ 1914., 52300., 45700., 39400.],

[ 1915., 19500., 51100., 39000.],

[ 1916., 11200., 29700., 36700.],

[ 1917., 7600., 15800., 41800.],

[ 1918., 14600., 9700., 43300.],

[ 1919., 16200., 10100., 41300.],

[ 1920., 24700., 8600., 47300.]])

﻿#EXERCISE 2

﻿np.savetxt('test.txt', tdata)

#The data is the same, but the data has been saved in scientific notation to the 14th decimal place

#EXERCISE 3

﻿np.savetxt('test.txt', tdata, fmt = '%.0f', header = 'Year Hare Lynx Carrot')

﻿﻿file = '/Users/roryeggleston/Downloads/CESM.003.SST.1980.nc'

data = xr.open\_dataset(file)

﻿data

Out[13]:

<xarray.Dataset>

Dimensions: (lat: 180, lon: 360, time: 12)

Coordinates:

\* lon (lon) float64 0.0 1.0 2.0 3.0 4.0 ... 355.0 356.0 357.0 358.0 359.0

\* lat (lat) float64 -89.5 -88.5 -87.5 -86.5 -85.5 ... 86.5 87.5 88.5 89.5

\* time (time) datetime64[ns] 1980-01-31 1980-02-29 ... 1980-12-31

Data variables:

SST (time, lat, lon) float32 ...

﻿In [14]: lat = np.array(data.lat)

lon = np.array(data.lon)

SST = np.array(data.SST)

﻿data.lon

Out[15]:

<xarray.DataArray 'lon' (lon: 360)>

array([ 0., 1., 2., ..., 357., 358., 359.])

Coordinates:

\* lon (lon) float64 0.0 1.0 2.0 3.0 4.0 ... 355.0 356.0 357.0 358.0 359.0

Attributes:

standard\_name: longitude

long\_name: longitude

units: degrees\_east

axis: X

﻿import netCDF4 as nc

﻿data2 = nc.Dataset(file, 'r')

data2

Out[17]:

<class 'netCDF4.\_netCDF4.Dataset'>

root group (NETCDF4 data model, file format HDF5):

dimensions(sizes): lon(360), lat(180), time(12)

variables(dimensions): float64 lon(lon), float64 lat(lat), int64 time(time), float32 SST(time,lat,lon)

groups:

﻿data2.variables

Out[18]:

OrderedDict([('lon', <class 'netCDF4.\_netCDF4.Variable'>

float64 lon(lon)

\_FillValue: nan

standard\_name: longitude

long\_name: longitude

units: degrees\_east

axis: X

unlimited dimensions:

current shape = (360,)

filling on), ('lat', <class 'netCDF4.\_netCDF4.Variable'>

float64 lat(lat)

\_FillValue: nan

standard\_name: latitude

long\_name: latitude

units: degrees\_north

axis: Y

unlimited dimensions:

current shape = (180,)

filling on), ('time', <class 'netCDF4.\_netCDF4.Variable'>

int64 time(time)

units: days since 1970-01-31

calendar: proleptic\_gregorian

unlimited dimensions:

current shape = (12,)

filling on, default \_FillValue of -9223372036854775806 used), ('SST',

<class 'netCDF4.\_netCDF4.Variable'>

float32 SST(time, lat, lon)

\_FillValue: 9.96921e+36

long\_name: Potential Temperature

units: degC

grid\_loc: 3111

cell\_methods: time: mean

unlimited dimensions:

current shape = (12, 180, 360)

filling on)])

﻿data2.dimensions

Out[19]:

OrderedDict([('lon',

<class 'netCDF4.\_netCDF4.Dimension'>: name = 'lon', size = 360),

('lat',

<class 'netCDF4.\_netCDF4.Dimension'>: name = 'lat', size = 180),

('time',

<class 'netCDF4.\_netCDF4.Dimension'>: name = 'time', size = 12)])

﻿SST2 = data2.variables['SST'][:]

﻿#EXERCISE 4

#SST2 is a masked array, which is different from SST, which is simply an array

﻿np.array(SST2)

#Removing the mask makes the values that are "nan" in the SST array into a repeating printout of the same very large value, not sure where this value came from specifically

Out[21]:

array([[[ 9.9692100e+36, 9.9692100e+36, 9.9692100e+36, ...,

9.9692100e+36, 9.9692100e+36, 9.9692100e+36],

[ 9.9692100e+36, 9.9692100e+36, 9.9692100e+36, ...,

9.9692100e+36, 9.9692100e+36, 9.9692100e+36],

[ 9.9692100e+36, 9.9692100e+36, 9.9692100e+36, ...,

9.9692100e+36, 9.9692100e+36, 9.9692100e+36],

...,

[-1.8010110e+00, -1.8009781e+00, -1.8009448e+00, ...,

-1.8011017e+00, -1.8010732e+00, -1.8010437e+00],

[-1.8005607e+00, -1.8005456e+00, -1.8005308e+00, ...,

-1.8006089e+00, -1.8005923e+00, -1.8005762e+00],

[-1.8000543e+00, -1.8000516e+00, -1.8000489e+00, ...,

-1.8000630e+00, -1.8000600e+00, -1.8000572e+00]],

[[ 9.9692100e+36, 9.9692100e+36, 9.9692100e+36, ...,

9.9692100e+36, 9.9692100e+36, 9.9692100e+36],

[ 9.9692100e+36, 9.9692100e+36, 9.9692100e+36, ...,

9.9692100e+36, 9.9692100e+36, 9.9692100e+36],

[ 9.9692100e+36, 9.9692100e+36, 9.9692100e+36, ...,

9.9692100e+36, 9.9692100e+36, 9.9692100e+36],

...,

[-1.8003076e+00, -1.8002812e+00, -1.8002526e+00, ...,

-1.8003770e+00, -1.8003554e+00, -1.8003322e+00],

[-1.8003292e+00, -1.8003246e+00, -1.8003204e+00, ...,

-1.8003466e+00, -1.8003403e+00, -1.8003345e+00],

[-1.8001331e+00, -1.8001313e+00, -1.8001295e+00, ...,

-1.8001386e+00, -1.8001367e+00, -1.8001349e+00]],

[[ 9.9692100e+36, 9.9692100e+36, 9.9692100e+36, ...,

9.9692100e+36, 9.9692100e+36, 9.9692100e+36],

[ 9.9692100e+36, 9.9692100e+36, 9.9692100e+36, ...,

9.9692100e+36, 9.9692100e+36, 9.9692100e+36],

[ 9.9692100e+36, 9.9692100e+36, 9.9692100e+36, ...,

9.9692100e+36, 9.9692100e+36, 9.9692100e+36],

...,

[-1.7999357e+00, -1.7998886e+00, -1.7998408e+00, ...,

-1.8000666e+00, -1.8000252e+00, -1.7999823e+00],

[-1.7992343e+00, -1.7992030e+00, -1.7991716e+00, ...,

-1.7993257e+00, -1.7992955e+00, -1.7992651e+00],

[-1.7991197e+00, -1.7991090e+00, -1.7990983e+00, ...,

-1.7991520e+00, -1.7991413e+00, -1.7991306e+00]],

...,

[[ 9.9692100e+36, 9.9692100e+36, 9.9692100e+36, ...,

9.9692100e+36, 9.9692100e+36, 9.9692100e+36],

[ 9.9692100e+36, 9.9692100e+36, 9.9692100e+36, ...,

9.9692100e+36, 9.9692100e+36, 9.9692100e+36],

[ 9.9692100e+36, 9.9692100e+36, 9.9692100e+36, ...,

9.9692100e+36, 9.9692100e+36, 9.9692100e+36],

...,

[-1.7971301e+00, -1.7972082e+00, -1.7972833e+00, ...,

-1.7968988e+00, -1.7969744e+00, -1.7970490e+00],

[-1.7991402e+00, -1.7991787e+00, -1.7992169e+00, ...,

-1.7990228e+00, -1.7990623e+00, -1.7991014e+00],

[-1.8009756e+00, -1.8009845e+00, -1.8009934e+00, ...,

-1.8009490e+00, -1.8009579e+00, -1.8009667e+00]],

[[ 9.9692100e+36, 9.9692100e+36, 9.9692100e+36, ...,

9.9692100e+36, 9.9692100e+36, 9.9692100e+36],

[ 9.9692100e+36, 9.9692100e+36, 9.9692100e+36, ...,

9.9692100e+36, 9.9692100e+36, 9.9692100e+36],

[ 9.9692100e+36, 9.9692100e+36, 9.9692100e+36, ...,

9.9692100e+36, 9.9692100e+36, 9.9692100e+36],

...,

[-1.8003566e+00, -1.8002627e+00, -1.8001657e+00, ...,

-1.8006001e+00, -1.8005269e+00, -1.8004475e+00],

[-1.7980388e+00, -1.7980016e+00, -1.7979645e+00, ...,

-1.7981507e+00, -1.7981133e+00, -1.7980760e+00],

[-1.7970190e+00, -1.7970107e+00, -1.7970022e+00, ...,

-1.7970442e+00, -1.7970358e+00, -1.7970275e+00]],

[[ 9.9692100e+36, 9.9692100e+36, 9.9692100e+36, ...,

9.9692100e+36, 9.9692100e+36, 9.9692100e+36],

[ 9.9692100e+36, 9.9692100e+36, 9.9692100e+36, ...,

9.9692100e+36, 9.9692100e+36, 9.9692100e+36],

[ 9.9692100e+36, 9.9692100e+36, 9.9692100e+36, ...,

9.9692100e+36, 9.9692100e+36, 9.9692100e+36],

...,

[-1.8001564e+00, -1.8001677e+00, -1.8001820e+00, ...,

-1.8001393e+00, -1.8001423e+00, -1.8001478e+00],

[-1.8004109e+00, -1.8004327e+00, -1.8004551e+00, ...,

-1.8003485e+00, -1.8003688e+00, -1.8003895e+00],

[-1.8015555e+00, -1.8015622e+00, -1.8015690e+00, ...,

-1.8015368e+00, -1.8015429e+00, -1.8015491e+00]]], dtype=float32)

﻿n [22]: SST3 = np.ma.masked\_invalid(SST)

SST3

Out[22]:

masked\_array(

data=[[[--, --, --, ..., --, --, --],

[--, --, --, ..., --, --, --],

[--, --, --, ..., --, --, --],

...,

[-1.8010109663009644, -1.8009780645370483, -1.8009448051452637,

..., -1.8011016845703125, -1.8010731935501099,

-1.8010437488555908],

[-1.800560712814331, -1.800545573234558, -1.8005307912826538,

..., -1.8006088733673096, -1.800592303276062,

-1.8005762100219727],

[-1.8000543117523193, -1.8000515699386597, -1.8000489473342896,

..., -1.8000630140304565, -1.8000600337982178,

-1.8000571727752686]],

[[--, --, --, ..., --, --, --],

[--, --, --, ..., --, --, --],

[--, --, --, ..., --, --, --],

...,

[-1.8003076314926147, -1.8002811670303345, -1.8002525568008423,

..., -1.8003770112991333, -1.8003554344177246,

-1.8003321886062622],

[-1.8003292083740234, -1.800324559211731, -1.8003203868865967,

..., -1.8003466129302979, -1.8003402948379517,

-1.8003344535827637],

[-1.8001331090927124, -1.8001313209533691, -1.8001295328140259,

..., -1.8001385927200317, -1.800136685371399,

-1.8001348972320557]],

[[--, --, --, ..., --, --, --],

[--, --, --, ..., --, --, --],

[--, --, --, ..., --, --, --],

...,

[-1.7999356985092163, -1.7998886108398438, -1.7998408079147339,

..., -1.800066590309143, -1.800025224685669,

-1.7999823093414307],

[-1.7992342710494995, -1.7992030382156372, -1.7991715669631958,

..., -1.799325704574585, -1.7992955446243286,

-1.7992651462554932],

[-1.7991197109222412, -1.7991089820861816, -1.799098253250122,

..., -1.7991520166397095, -1.79914128780365,

-1.7991305589675903]],

...,

[[--, --, --, ..., --, --, --],

[--, --, --, ..., --, --, --],

[--, --, --, ..., --, --, --],

...,

[-1.7971301078796387, -1.7972081899642944, -1.7972832918167114,

..., -1.7968988418579102, -1.7969744205474854,

-1.7970490455627441],

[-1.799140214920044, -1.7991787195205688, -1.799216866493225,

..., -1.7990227937698364, -1.7990622520446777,

-1.7991013526916504],

[-1.8009755611419678, -1.800984501838684, -1.8009934425354004,

..., -1.800948977470398, -1.8009579181671143,

-1.800966739654541]],

[[--, --, --, ..., --, --, --],

[--, --, --, ..., --, --, --],

[--, --, --, ..., --, --, --],

...,

[-1.8003566265106201, -1.800262689590454, -1.8001656532287598,

..., -1.8006000518798828, -1.8005268573760986,

-1.8004474639892578],

[-1.7980388402938843, -1.7980016469955444, -1.7979644536972046,

..., -1.798150658607483, -1.7981133460998535,

-1.7980760335922241],

[-1.7970190048217773, -1.7970106601715088, -1.7970021963119507,

..., -1.7970441579818726, -1.797035813331604,

-1.7970274686813354]],

[[--, --, --, ..., --, --, --],

[--, --, --, ..., --, --, --],

[--, --, --, ..., --, --, --],

...,

[-1.8001563549041748, -1.8001676797866821, -1.8001819849014282,

..., -1.800139307975769, -1.8001422882080078,

-1.8001477718353271],

[-1.8004108667373657, -1.8004326820373535, -1.800455093383789,

..., -1.8003485202789307, -1.8003687858581543,

-1.8003895282745361],

[-1.8015555143356323, -1.8015621900558472, -1.8015689849853516,

..., -1.8015367984771729, -1.80154287815094,

-1.8015490770339966]]],

mask=[[[ True, True, True, ..., True, True, True],

[ True, True, True, ..., True, True, True],

[ True, True, True, ..., True, True, True],

...,

[False, False, False, ..., False, False, False],

[False, False, False, ..., False, False, False],

[False, False, False, ..., False, False, False]],

[[ True, True, True, ..., True, True, True],

[ True, True, True, ..., True, True, True],

[ True, True, True, ..., True, True, True],

...,

[False, False, False, ..., False, False, False],

[False, False, False, ..., False, False, False],

[False, False, False, ..., False, False, False]],

[[ True, True, True, ..., True, True, True],

[ True, True, True, ..., True, True, True],

[ True, True, True, ..., True, True, True],

...,

[False, False, False, ..., False, False, False],

[False, False, False, ..., False, False, False],

[False, False, False, ..., False, False, False]],

...,

[[ True, True, True, ..., True, True, True],

[ True, True, True, ..., True, True, True],

[ True, True, True, ..., True, True, True],

...,

[False, False, False, ..., False, False, False],

[False, False, False, ..., False, False, False],

[False, False, False, ..., False, False, False]],

[[ True, True, True, ..., True, True, True],

[ True, True, True, ..., True, True, True],

[ True, True, True, ..., True, True, True],

...,

[False, False, False, ..., False, False, False],

[False, False, False, ..., False, False, False],

[False, False, False, ..., False, False, False]],

[[ True, True, True, ..., True, True, True],

[ True, True, True, ..., True, True, True],

[ True, True, True, ..., True, True, True],

...,

[False, False, False, ..., False, False, False],

[False, False, False, ..., False, False, False],

[False, False, False, ..., False, False, False]]],

fill\_value=1e+20,

dtype=float32)

﻿import pandas as pd

﻿file = '/Users/roryeggleston/Downloads/movies.xls'

movies = pd.read\_excel(file)

﻿movies2 = xr.Dataset.from\_dataframe(movies)

﻿movies2

Out[26]:

<xarray.Dataset>

Dimensions: (index: 1338)

Coordinates:

\* index (index) int64 0 1 2 3 4 ... 1334 1335 1336 1337

Data variables:

Title (index) object "Intolerance: Love's Struggle Throughout the Ages\xa0" ... 'Wing Commander\xa0'

Year (index) int64 1916 1920 1925 ... 1999 1999 1999

Genres (index) object 'Drama|History|War' ... 'Action|Adventure|Sci-Fi'

Language (index) object nan nan ... 'English' 'English'

Country (index) object 'USA' 'USA' ... 'USA' 'USA'

Content Rating (index) object 'Not Rated' nan ... 'PG-13'

Duration (index) int64 123 110 151 145 ... 106 106 100

Aspect Ratio (index) float64 1.33 1.33 1.33 ... 1.85 2.35

Budget (index) float64 3.859e+05 1e+05 ... 3e+07

Gross Earnings (index) float64 nan 3e+06 ... 1.158e+07

Director (index) object 'D.W. Griffith' ... 'Chris Roberts'

Actor 1 (index) object 'Lillian Gish' ... 'Saffron Burrows'

Actor 2 (index) object 'Mae Marsh' ... 'David Suchet'

Actor 3 (index) object 'Walter Long' ... 'Jürgen Prochnow'

Facebook Likes - Director (index) int64 204 0 54 756 21 ... 7 48 188 38

Facebook Likes - Actor 1 (index) int64 436 2 81 136 ... 23000 10000 811

Facebook Likes - Actor 2 (index) int64 22 2 12 23 ... 577 255 4000 586

Facebook Likes - Actor 3 (index) float64 9.0 0.0 6.0 ... 582.0 362.0

Facebook Likes - cast Total (index) int64 481 4 108 ... 23369 15870 2497

Facebook likes - Movie (index) int64 691 0 226 12000 ... 401 0 0 858

Facenumber in posters (index) int64 1 1 0 1 1 8 1 1 ... 3 1 0 0 0 2 3

User Votes (index) int64 10718 5 4849 ... 129601 14747

Reviews by Users (index) int64 88 1 45 413 ... 162 267 648 338

Reviews by Crtiics (index) float64 69.0 1.0 48.0 ... 85.0 85.0

IMDB Score (index) float64 8.0 4.8 8.3 8.3 ... 6.4 4.8 4.1

﻿movies.Year

Out[27]:

0 1916

1 1920

2 1925

3 1927

4 1929

5 1929

6 1930

7 1932

8 1933

9 1933

10 1934

11 1935

12 1936

13 1936

14 1937

15 1937

16 1938

17 1938

18 1939

19 1939

20 1939

21 1940

22 1940

23 1940

24 1940

25 1940

26 1941

27 1942

28 1942

29 1943

1308 1999

1309 1999

1310 1999

1311 1999

1312 1999

1313 1999

1314 1999

1315 1999

1316 1999

1317 1999

1318 1999

1319 1999

1320 1999

1321 1999

1322 1999

1323 1999

1324 1999

1325 1999

1326 1999

1327 1999

1328 1999

1329 1999

1330 1999

1331 1999

1332 1999

1333 1999

1334 1999

1335 1999

1336 1999

1337 1999

Name: Year, Length: 1338, dtype: int64

﻿movies = pd.read\_excel(file)

movies.head()

Out[28]:

Title ... IMDB Score

0 Intolerance: Love's Struggle Throughout the Ages ... 8.0

1 Over the Hill to the Poorhouse ... 4.8

2 The Big Parade ... 8.3

3 Metropolis ... 8.3

4 Pandora's Box ... 8.0

[5 rows x 25 columns]

﻿

movies\_sheet1 = pd.read\_excel(file, sheetname=0, index\_col=0)

movies\_sheet1.head()

/anaconda3/lib/python3.7/site-packages/pandas/io/excel.py:329: FutureWarning: The `sheetname` keyword is deprecated, use `sheet\_name` instead

\*\*kwds)

Out[29]:

Year ... IMDB Score

Title ...

Intolerance: Love's Struggle Throughout the Ages 1916 ... 8.0

Over the Hill to the Poorhouse 1920 ... 4.8

The Big Parade 1925 ... 8.3

Metropolis 1927 ... 8.3

Pandora's Box 1929 ... 8.0

[5 rows x 24 columns]

movies\_sheet2 = pd.read\_excel(file, sheetname=1, index\_col=0)

movies\_sheet2.head()

Out[30]:

Year ... IMDB Score

Title ...

102 Dalmatians 2000 ... 4.8

28 Days 2000 ... 6.0

3 Strikes 2000 ... 4.0

Aberdeen 2000 ... 7.3

All the Pretty Horses 2000 ... 5.8

[5 rows x 24 columns]

movies\_sheet3 = pd.read\_excel(file, sheetname=2, index\_col=0)

movies\_sheet3.head()

Out[31]:

Year ... IMDB Score

Title ...

127 Hours 2010.0 ... 7.6

3 Backyards 2010.0 ... 5.2

3 2010.0 ... 6.8

8: The Mormon Proposition 2010.0 ... 7.1

A Turtle's Tale: Sammy's Adventures 2010.0 ... 6.1

[5 rows x 24 columns]

﻿movies = pd.concat([movies\_sheet1, movies\_sheet2, movies\_sheet3])

﻿movies.shape

Out[33]: (5042, 24)

﻿xlsx = pd.ExcelFile(file)

movies\_sheets = []

for sheet in xlsx.sheet\_names:

movies\_sheets.append(xlsx.parse(sheet))

movies = pd.concat(movies\_sheets)

﻿movies.shape

Out[37]: (5042, 25)

﻿movies.tail()

Out[38]:

Title ... IMDB Score

1599 War & Peace ... 8.2

1600 Wings ... 7.3

1601 Wolf Creek ... 7.1

1602 Wuthering Heights ... 7.7

1603 Yu-Gi-Oh! Duel Monsters ... 7.0

[5 rows x 25 columns]

﻿sorted\_by\_gross = movies.sort\_values(['Gross Earnings'], ascending=False)

﻿sorted\_by\_gross["Gross Earnings"].head(10)

Out[40]:

1867 760505847.0

1027 658672302.0

1263 652177271.0

610 623279547.0

611 623279547.0

1774 533316061.0

1281 474544677.0

226 460935665.0

1183 458991599.0

618 448130642.0

Name: Gross Earnings, dtype: float64

﻿sorted\_by\_gross['Gross Earnings'].head(10).plot(kind='barh')

plt.show()



﻿n [42]: movies['IMDB Score'].plot(kind="hist")

plt.show()



﻿movies.describe()

Out[43]:

Year Duration ... Reviews by Crtiics IMDB Score

count 4935.000000 5028.000000 ... 4993.000000 5042.000000

mean 2002.470517 107.201074 ... 140.194272 6.442007

std 12.474599 25.197441 ... 121.601675 1.125189

min 1916.000000 7.000000 ... 1.000000 1.600000

25% 1999.000000 93.000000 ... 50.000000 5.800000

50% 2005.000000 103.000000 ... 110.000000 6.600000

75% 2011.000000 118.000000 ... 195.000000 7.200000

max 2016.000000 511.000000 ... 813.000000 9.500000

[8 rows x 16 columns]

﻿movies["Gross Earnings"].mean()

Out[44]: 48468407.52680933

﻿movies\_skip\_rows = pd.read\_excel("/Users/roryeggleston/Downloads/movies.xls", header=None, skiprows=4)

movies\_skip\_rows.head(5)

Out[45]:

0 1 2 ... 22 23 24

0 Metropolis 1927 Drama|Sci-Fi ... 413 260.0 8.3

1 Pandora's Box 1929 Crime|Drama|Romance ... 84 71.0 8.0

2 The Broadway Melody 1929 Musical|Romance ... 71 36.0 6.3

3 Hell's Angels 1930 Drama|War ... 53 35.0 7.8

4 A Farewell to Arms 1932 Drama|Romance|War ... 46 42.0 6.6

[5 rows x 25 columns]

﻿movies\_skip\_rows.columns = ['Title', 'Year', 'Genres', 'Language', 'Country', 'Content Rating', 'Duration', 'Aspect Ratio', 'Budget', 'Gross Earnings', 'Director', 'Actor 1', 'Actor 2', 'Actor 3', 'Facebook Likes - Director', 'Facebook Likes - Actor 1', 'Facebook Likes - Actor 2', 'Facebook Likes - Actor 3', 'Facebook Likes - cast Total', 'Facebook Likes - Movie', 'Facenumber in posters', 'User Votes', 'Reviews by Users', 'Reviews by Critics', 'IMDB Score']

movies\_skip\_rows.head()

Out[46]:

Title Year ... Reviews by Critics IMDB Score

0 Metropolis 1927 ... 260.0 8.3

1 Pandora's Box 1929 ... 71.0 8.0

2 The Broadway Melody 1929 ... 36.0 6.3

3 Hell's Angels 1930 ... 35.0 7.8

4 A Farewell to Arms 1932 ... 42.0 6.6

[5 rows x 25 columns]

﻿n [47]: movies\_subset\_columns = pd.read\_excel(file, parse\_cols=6)

movies\_subset\_columns.head()

\_\_main\_\_:1: FutureWarning: the 'parse\_cols' keyword is deprecated, use 'usecols' instead

Out[47]:

Title ... Duration

0 Intolerance: Love's Struggle Throughout the Ages ... 123

1 Over the Hill to the Poorhouse ... 110

2 The Big Parade ... 151

3 Metropolis ... 145

4 Pandora's Box ... 110

[5 rows x 7 columns]

﻿movies["Net Earnings"] = movies["Gross Earnings"] - movies["Budget"]

﻿sorted\_movies = movies[["Net Earnings"]].sort\_values(["Net Earnings"], ascending=[False])

sorted\_movies.head(10)["Net Earnings"].plot.barh()

plt.show()



﻿movies\_subset = movies[["Year", "Gross Earnings"]]

movies\_subset.head()

Out[50]:

Year Gross Earnings

0 1916.0 NaN

1 1920.0 3000000.0

2 1925.0 NaN

3 1927.0 26435.0

4 1929.0 9950.0

﻿earnings\_by\_year = movies\_subset.pivot\_table(index=["Year"])

earnings\_by\_year.head()

Out[51]:

Gross Earnings

Year

1920.0 3000000.0

1927.0 26435.0

1929.0 1408975.0

1933.0 2300000.0

1935.0 3000000.0

﻿earnings\_by\_year.plot()

plt.show()



﻿movies\_subset = movies[['Country', 'Language', 'Gross Earnings']]

movies\_subset.head()

Out[53]:

Country Language Gross Earnings

0 USA NaN NaN

1 USA NaN 3000000.0

2 USA NaN NaN

3 Germany German 26435.0

4 Germany German 9950.0

﻿earnings\_by\_co\_lang = movies\_subset.pivot\_table(index=['Country', 'Language'])

earnings\_by\_co\_lang.head()

Out[54]:

Gross Earnings

Country Language

Afghanistan Dari 1.127331e+06

Argentina Spanish 7.230936e+06

Aruba English 1.007614e+07

Australia Aboriginal 6.165429e+06

Dzongkha 5.052950e+05

﻿n [55]: earnings\_by\_co\_lang.head(20).plot(kind='bar', figsize=(20,8))

plt.show()



﻿movies.to\_excel('output.xlsx')

﻿n [57]: movies.head()

Out[57]:

Title ... Net Earnings

0 Intolerance: Love's Struggle Throughout the Ages ... NaN

1 Over the Hill to the Poorhouse ... 2900000.0

2 The Big Parade ... NaN

3 Metropolis ... -5973565.0

4 Pandora's Box ... NaN

[5 rows x 26 columns]

﻿movies.to\_excel('output.xlsx', index=False)

﻿writer = pd.ExcelWriter('output.xlsx', engine='xlsxwriter')

movies.to\_excel(writer, index=False, sheet\_name='report')

workbook = writer.book

worksheet = writer.sheets['report']

﻿header\_fmt = workbook.add\_format({'bold': True})

worksheet.set\_row(0, None, header\_fmt)

﻿writer.save()