VYAKYA

Technical Assessment

Name: Indhu Sree S

Roll no: 1832023

M.Sc Data Science

Task 1:

1. Perform an EDA of this books dataset & share insights

2. Build a system to recommend more books to a reader based on a book already selected.

**Work Flow with Code and Insights:**

* **Importing needed Libraries**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.cluster import KMeans

from sklearn import neighbors

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import MinMaxScaler

* **Dataset**

book = pd.read\_csv("books.csv", error\_bad\_lines = 0)

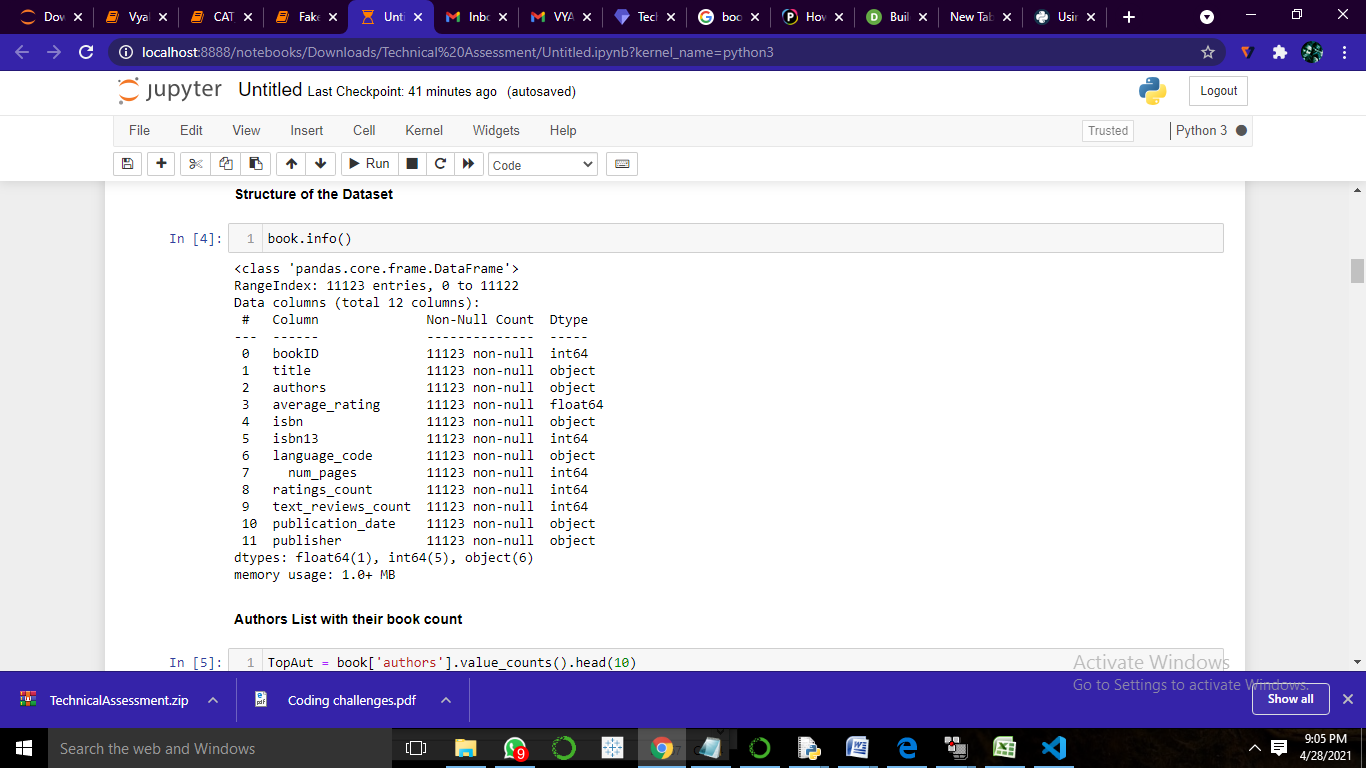
book.head()

* **Size of Dataset**

book.shape

* Information about the dataset

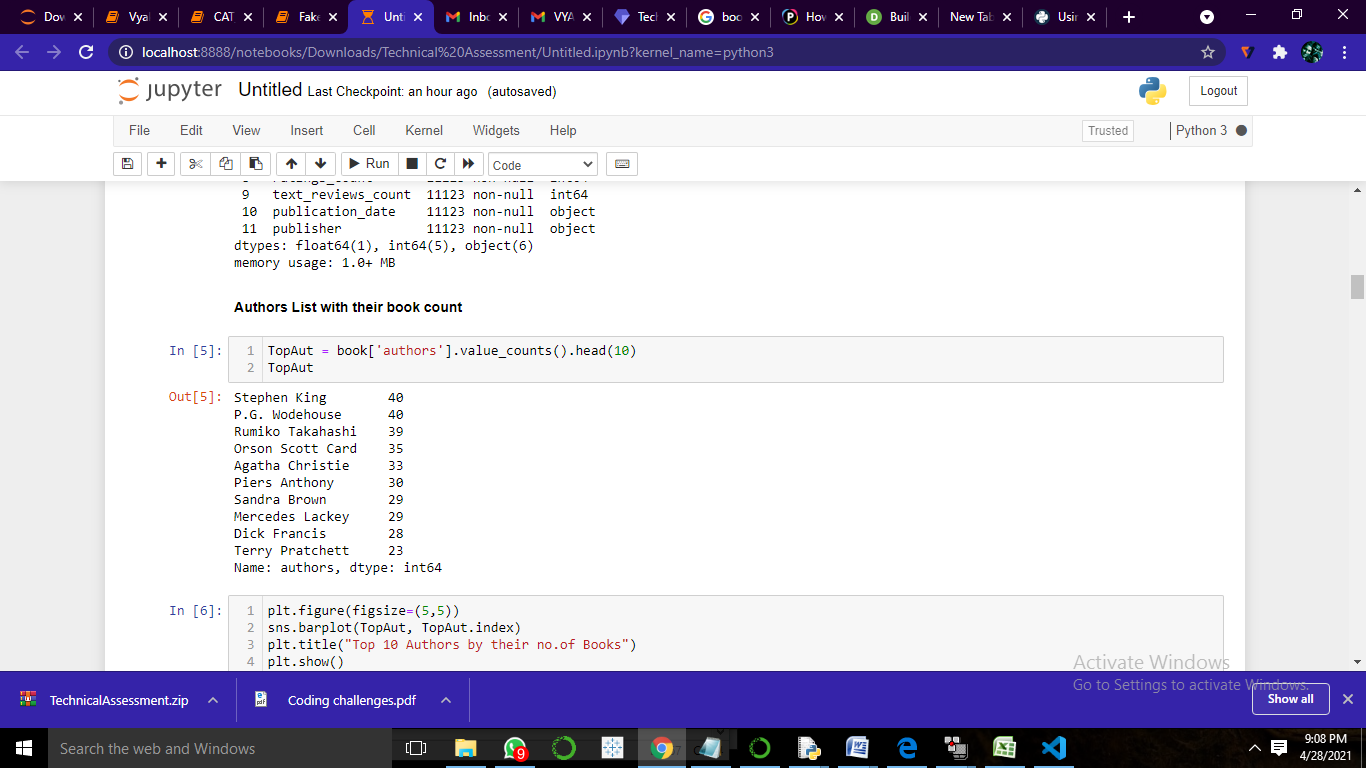
book.info()

****

* Top Authors based on their Book Publication

TopAut = book['authors'].value\_counts().head(10)

TopAut

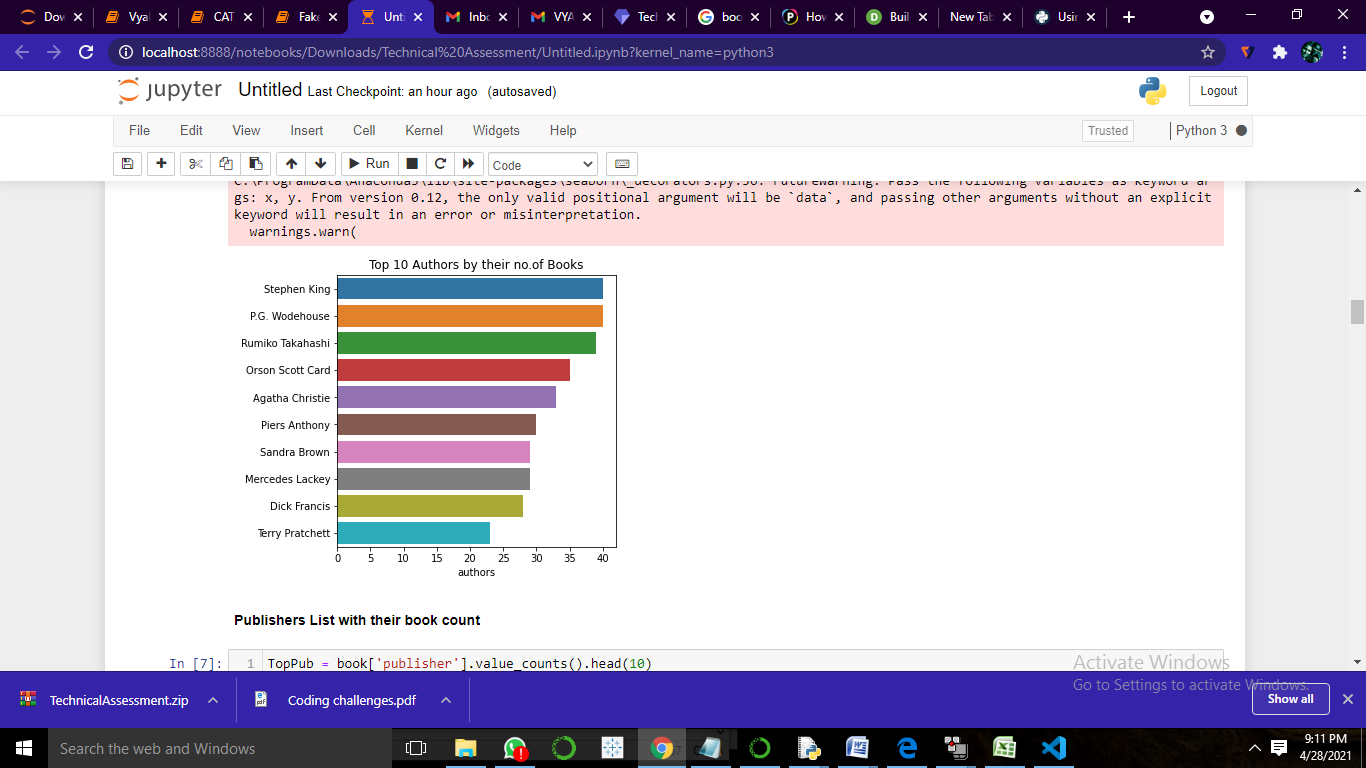
****

plt.figure(figsize=(5,5))

sns.barplot(TopAut, TopAut.index)

plt.title("Top 10 Authors by their no.of Books")

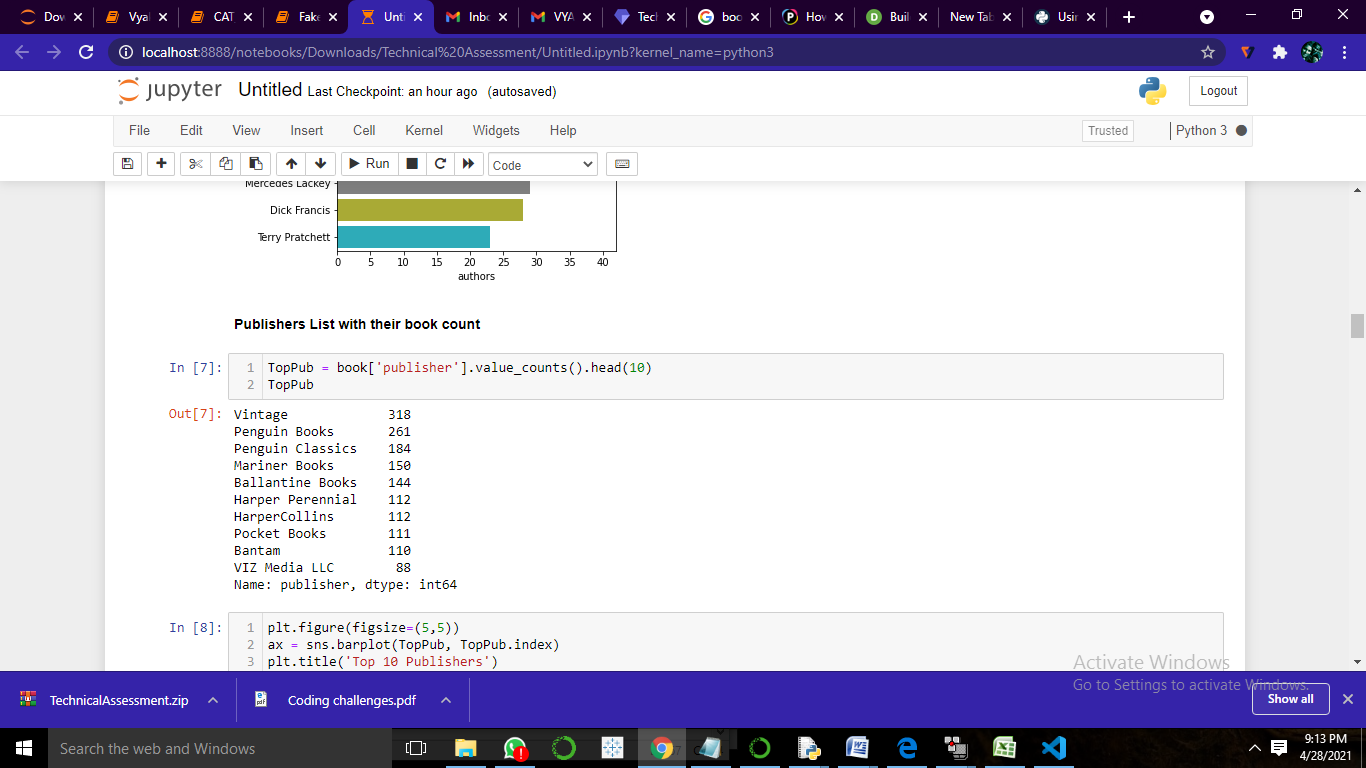
plt.show()

****

* Top Publishers list by their no. of Books Published

TopPub = book['publisher'].value\_counts().head(10)

TopPub

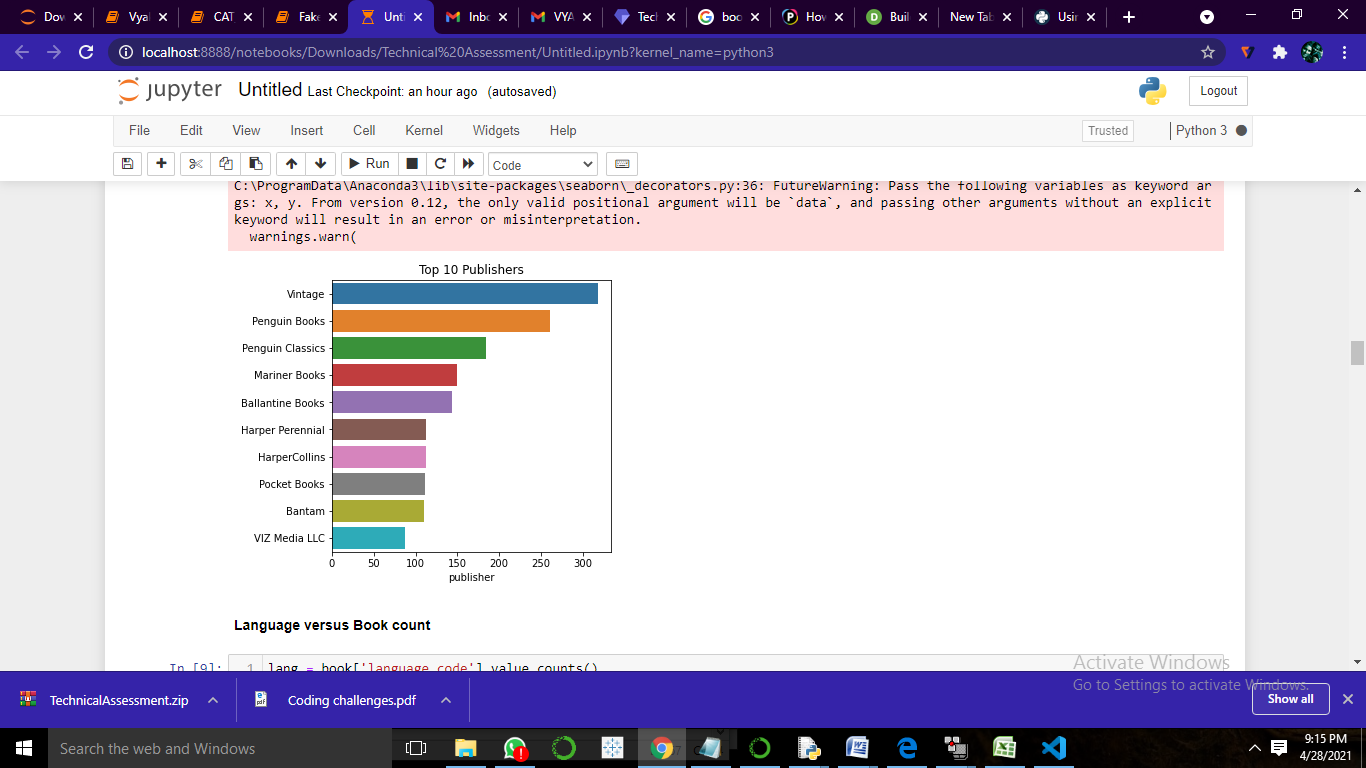
****

plt.figure(figsize=(5,5))

ax = sns.barplot(TopPub, TopPub.index)

plt.title('Top 10 Publishers')

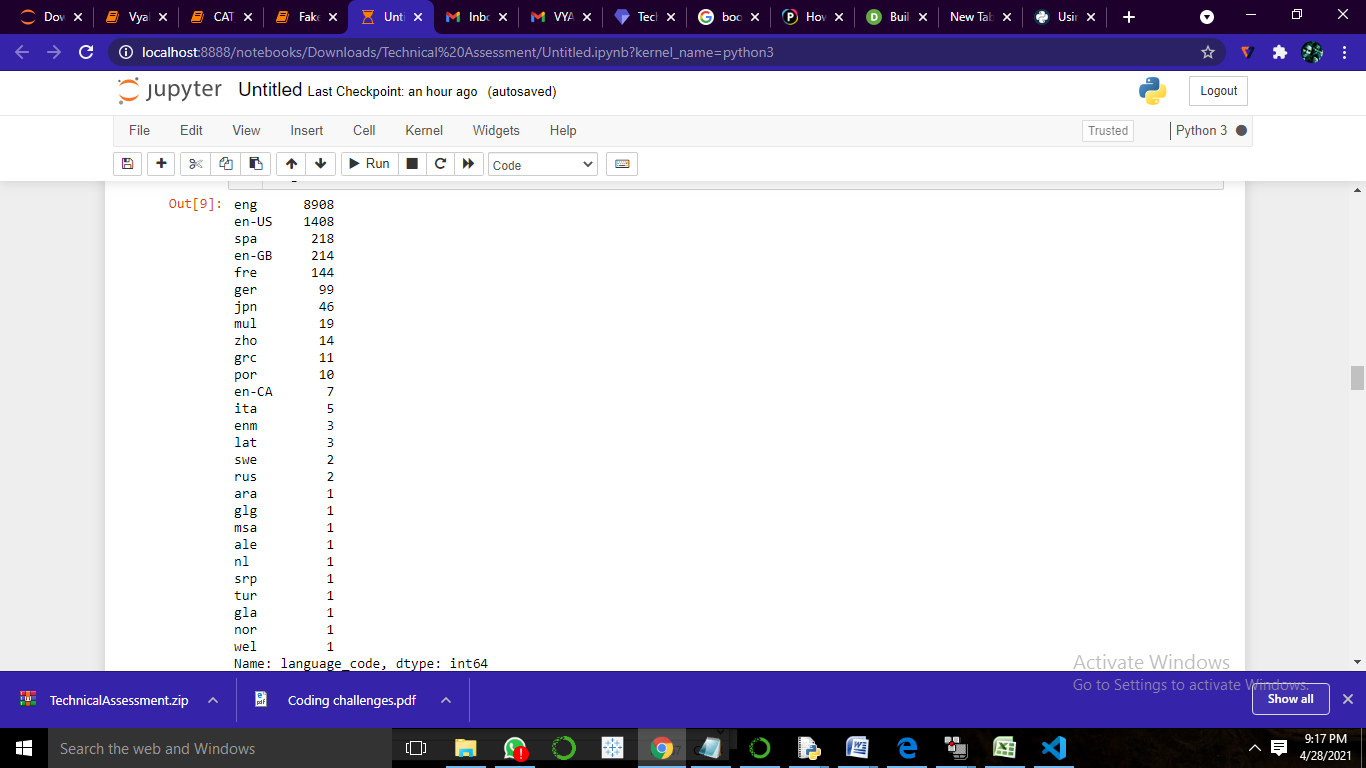
plt.show()

****

* Languages versus Book Count

lang = book['language\_code'].value\_counts()

lang

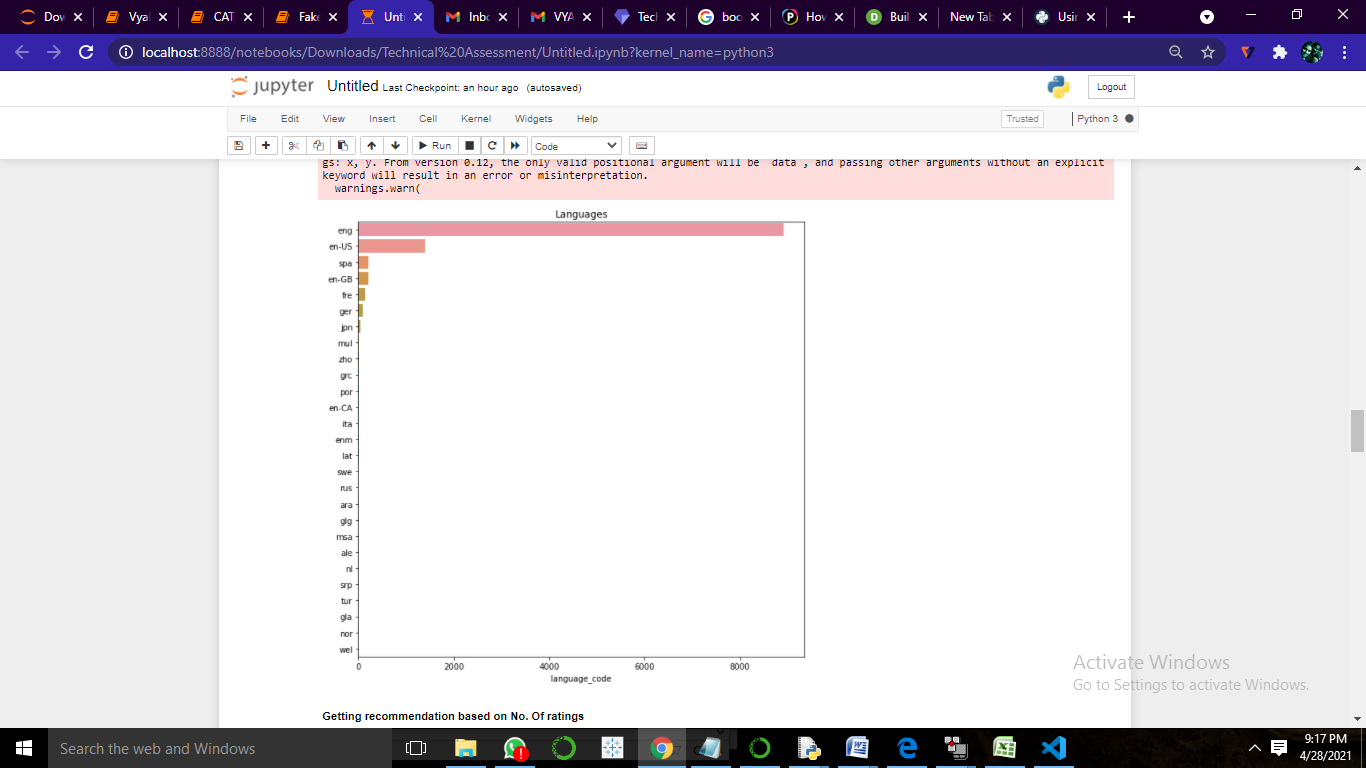
****

plt.figure(figsize=(10,10))

sns.barplot(lang, lang.index)

plt.title('Languages')

plt.show()

****

* Getting Recommendation Based on **No. of Ratings**

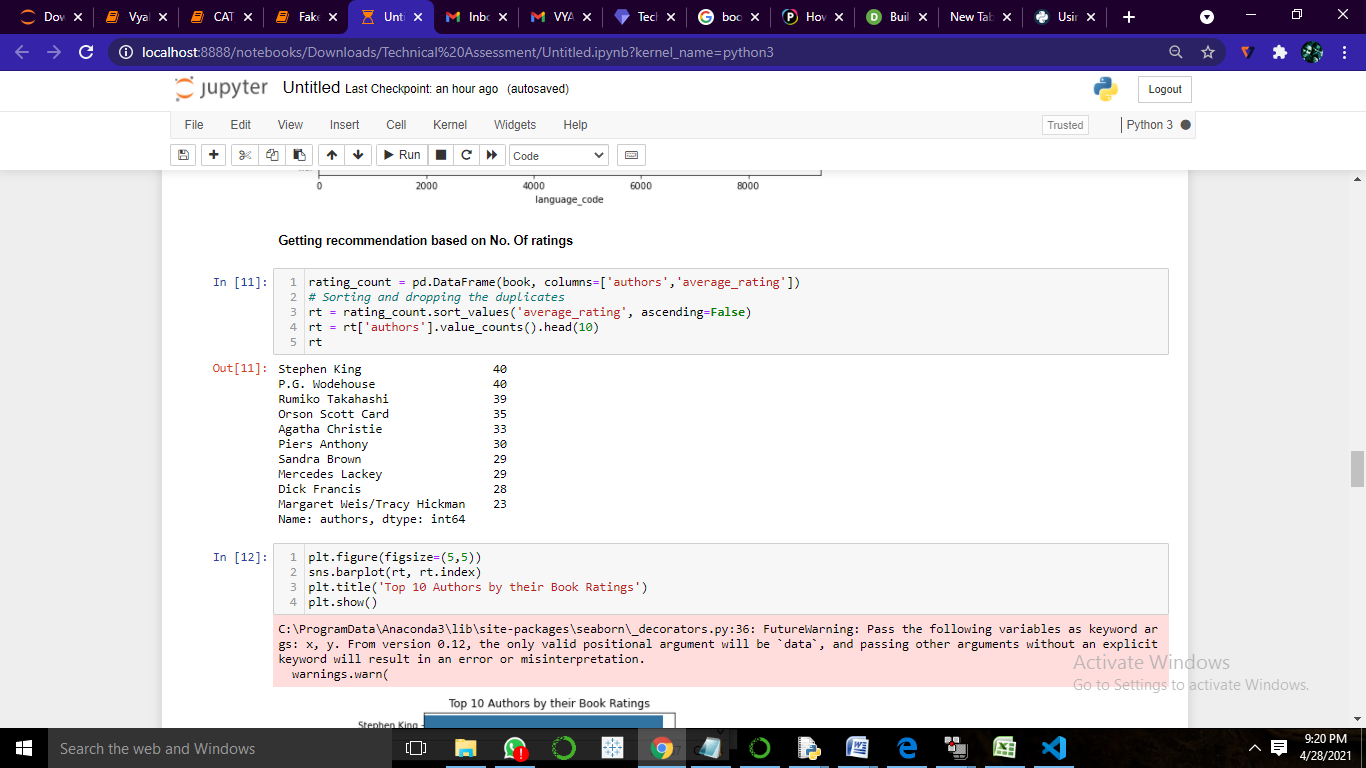
rating\_count = pd.DataFrame(book, columns=['authors','average\_rating'])

# Sorting and dropping the duplicates

rt = rating\_count.sort\_values('average\_rating', ascending=False)

rt = rt['authors'].value\_counts().head(10)

rt

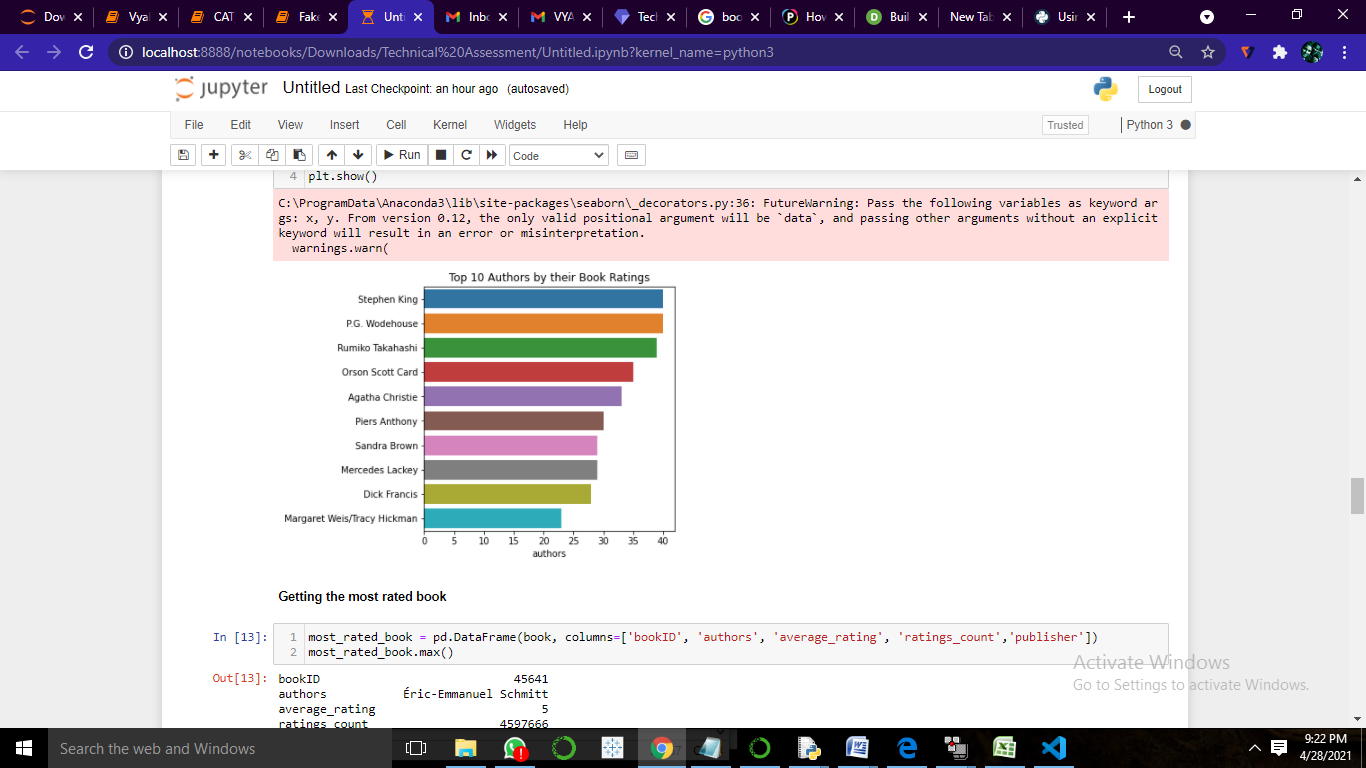
****

plt.figure(figsize=(5,5))

sns.barplot(rt, rt.index)

plt.title('Top 10 Authors by their Book Ratings')

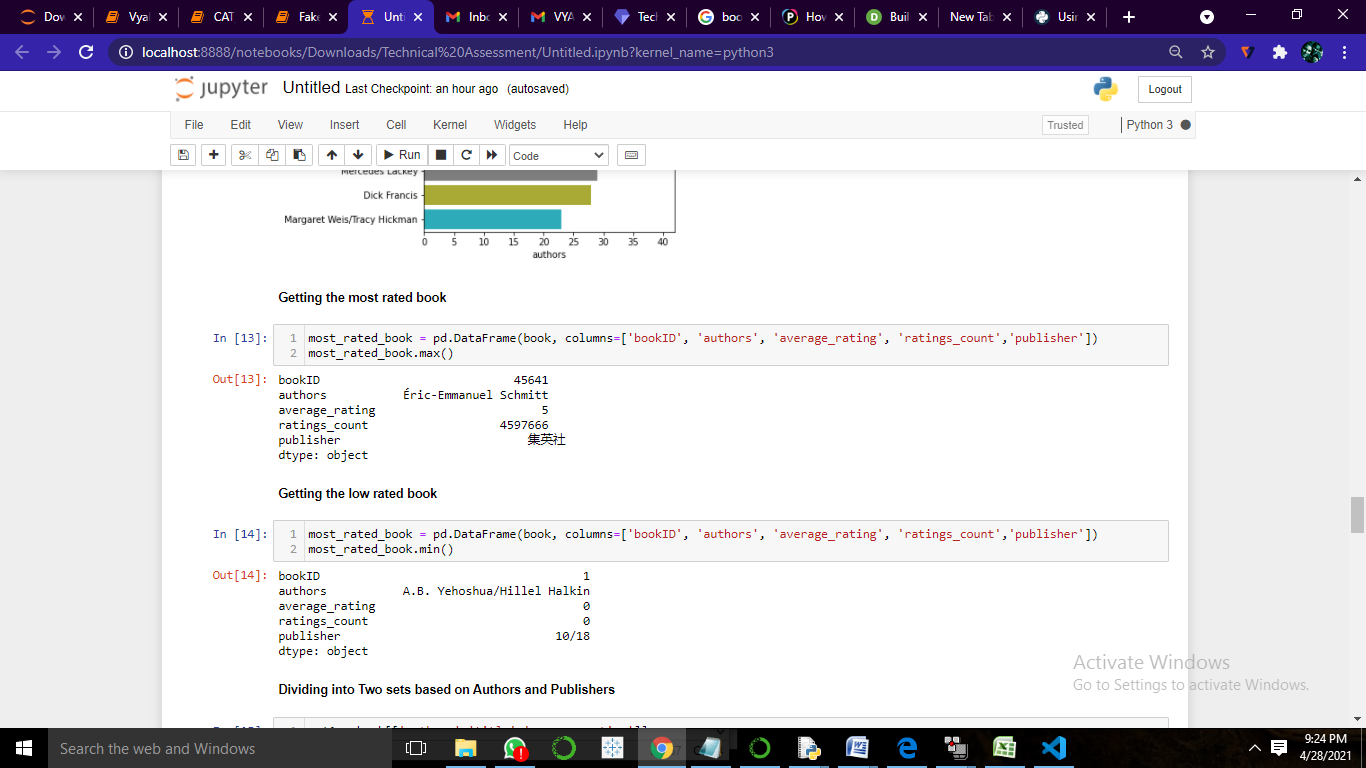
plt.show()

****

* Getting the most rated book

most\_rated\_book = pd.DataFrame(book, columns=['bookID', 'authors', 'average\_rating', 'ratings\_count','publisher'])

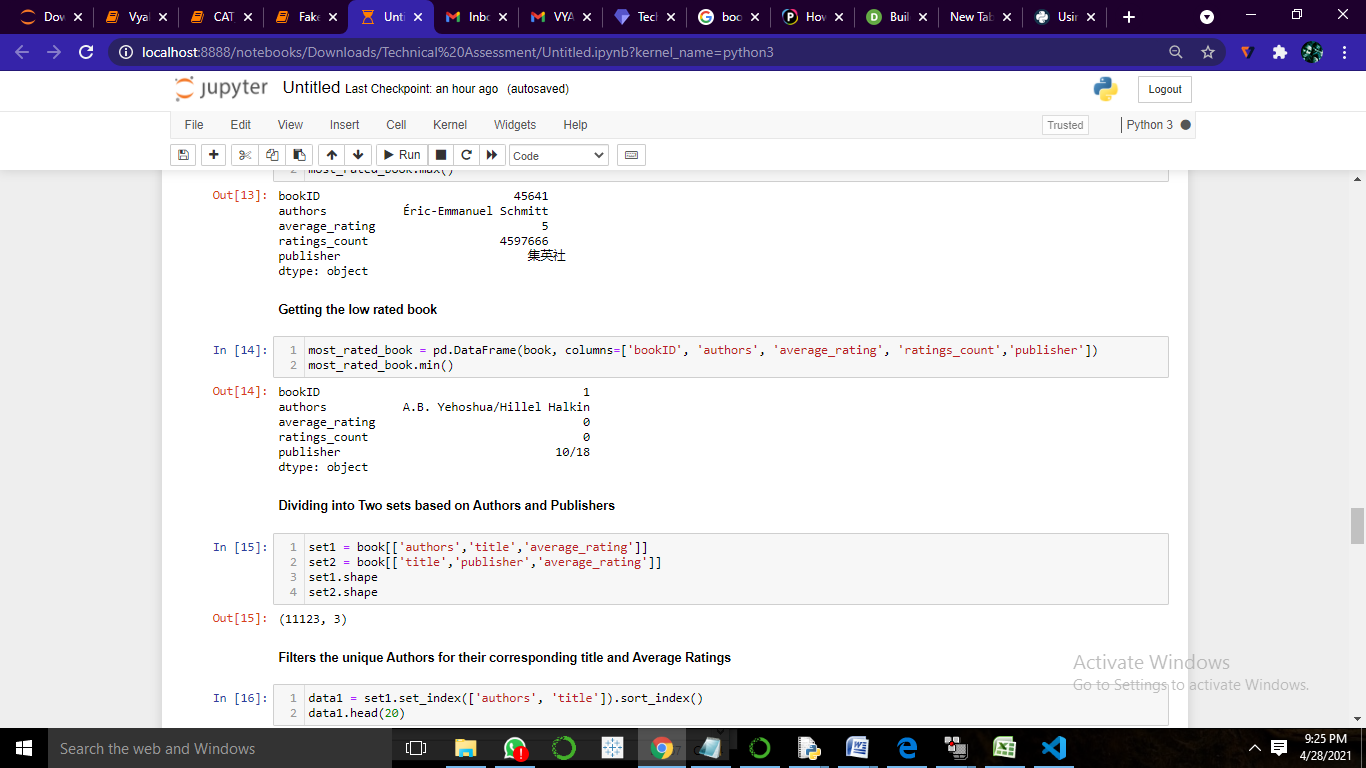
most\_rated\_book.max()



* Getting the low rated book

most\_rated\_book = pd.DataFrame(book, columns=['bookID', 'authors', 'average\_rating', 'ratings\_count','publisher'])

most\_rated\_book.min()

****

set1 = book[['authors','title','average\_rating']]

set2 = book[['title','publisher','average\_rating']]

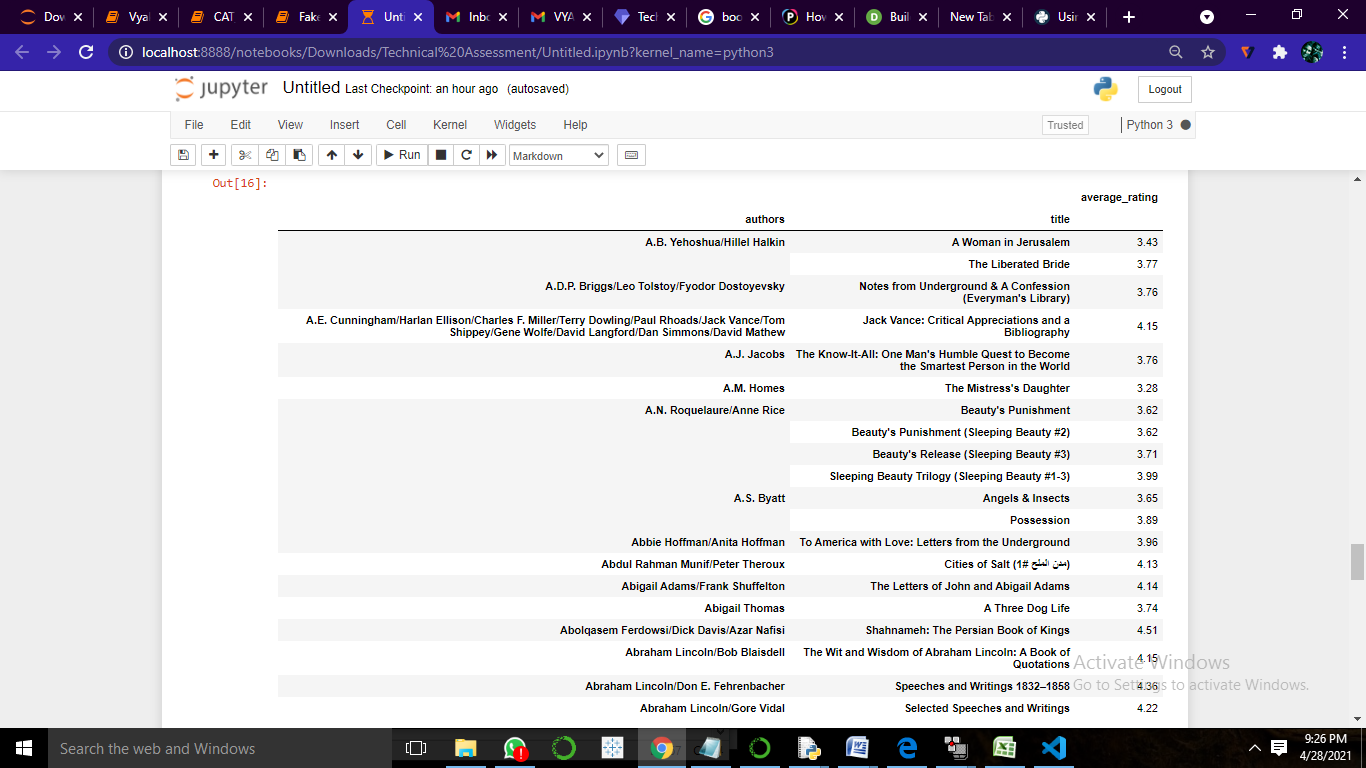
set1.shape

set2.shape

* **Filters the unique Authors for their corresponding title and Average Ratings**

**data1 = set1.set\_index(['authors', 'title']).sort\_index()**

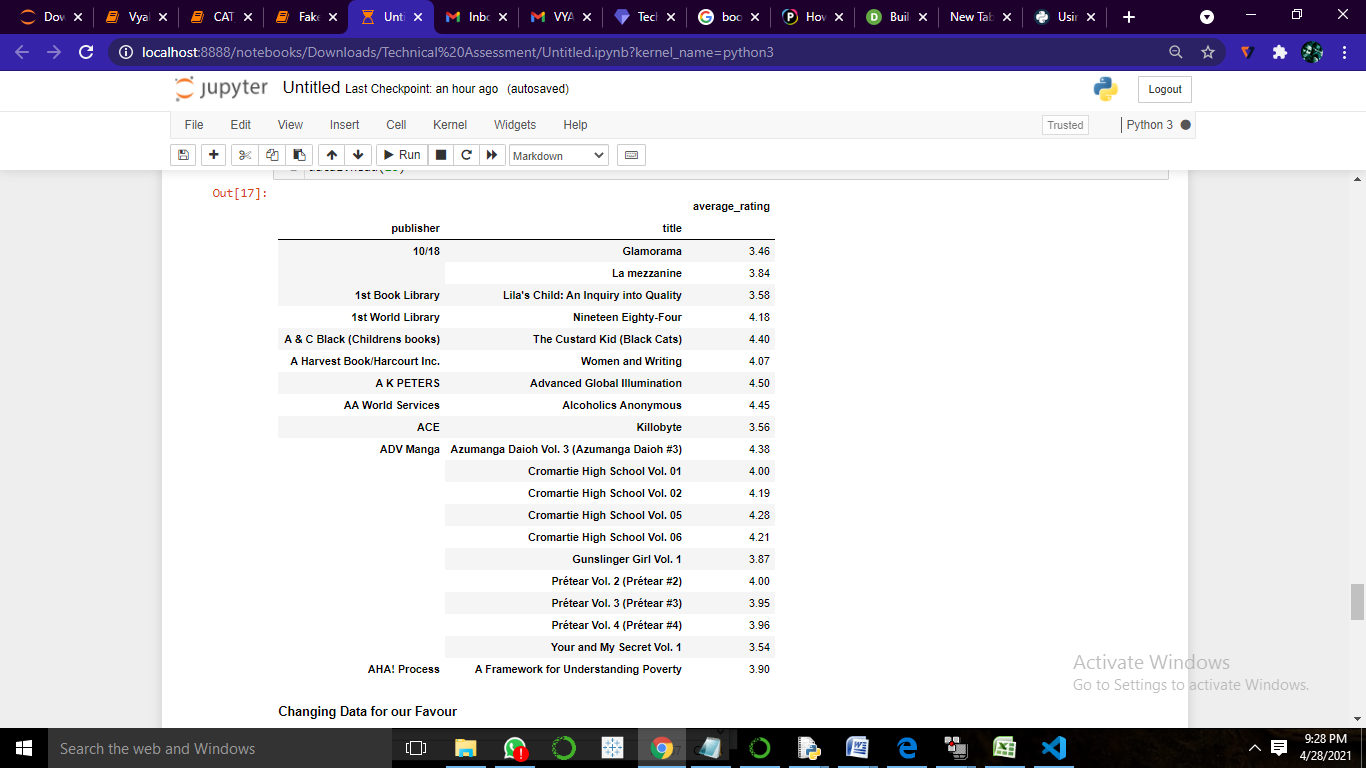
**data1.head(20)**

****

* **Filters the unique Publishers for their corresponding title and Average Ratings**

**data1 = set2.set\_index(['publisher', 'title']).sort\_index()**

**data1.head(20)**

****

# Returns a distance-based similarity score for person1 and person2

def sim\_distance(prefs,person1,person2):

si = {}

for item in prefs[person1]:

if item in prefs[person2]:

si[item]=1

if len(si) == 0:

return 0

sum\_of\_squares = sum([pow(prefs[person1][item] - prefs[person2][item],2)

for item in prefs[person1] if item in prefs[person2]])

return 1/(1+sum\_of\_squares)

# Checks for similarity to Maureen using Euclidean distance score

def top\_matches(prefs, person, n=10, similarity = sim\_distance):

scores = [(similarity(prefs,person,other), other)

for other in prefs if other!=person]

scores.sort()

scores.reverse()

return scores[0:n]

* **Convertion of Average Rating**

df=book.copy()

df.loc[ (df['average\_rating'] >= 0) & (df['average\_rating'] <= 1), 'rating'] = "0-1"

df.loc[ (df['average\_rating'] > 1) & (df['average\_rating'] <= 2), 'rating'] = "1-2"

df.loc[ (df['average\_rating'] > 2) & (df['average\_rating'] <= 3), 'rating'] = "2-3"

df.loc[ (df['average\_rating'] > 3) & (df['average\_rating'] <= 4), 'rating'] = "3-4"

df.loc[ (df['average\_rating'] > 4) & (df['average\_rating'] <= 5), 'rating'] = "4-5"

rating\_df = pd.get\_dummies(df['rating'])

rating\_df.head()

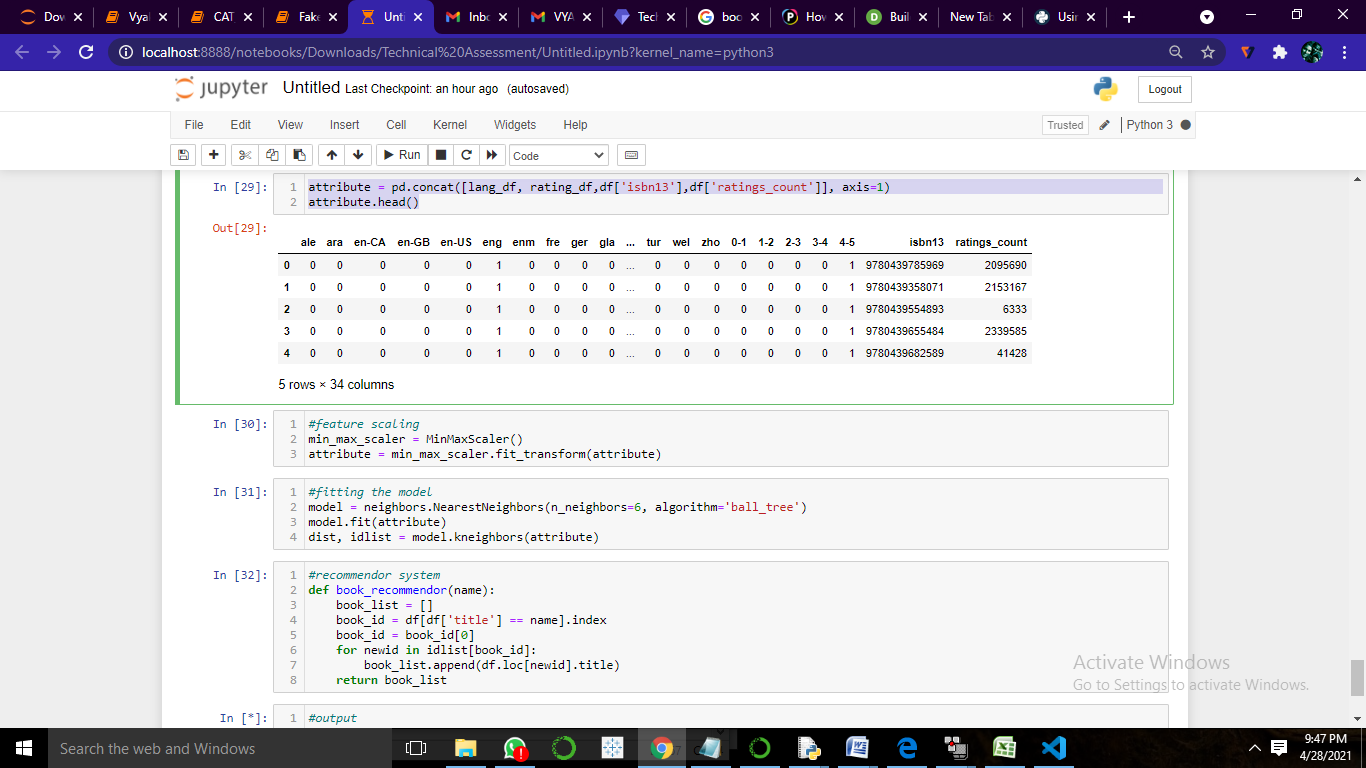
lang\_df = pd.get\_dummies(df['language\_code'])

lang\_df.head()

attribute = pd.concat([lang\_df, rating\_df,df['isbn13'],df['ratings\_count']], axis=1)

attribute.head()

* **Features**

****

* **feature scaling**

min\_max\_scaler = MinMaxScaler()

attribute = min\_max\_scaler.fit\_transform(attribute)

* **fitting the model**

model = neighbors.NearestNeighbors(n\_neighbors=6, algorithm='ball\_tree')

model.fit(attribute)

dist, idlist = model.kneighbors(attribute)

* **recommendor system**

def book\_recommendor(name):

book\_list = []

book\_id = df[df['title'] == name].index

book\_id = book\_id[0]

for newid in idlist[book\_id]:

book\_list.append(df.loc[newid].title)

return book\_list

* **output**

Author=input("The Author name:")

# Top 10 similar users to Author

top\_matches(d, Author)

**Output:**

**The Author name:A.S. Byatt**

**[(0, 'Éric-Emmanuel Schmitt'),**

**(0, 'Émile Zola/Roger Pearson'),**

**(0, 'Émile Zola/Robin Buss/Brian Nelson'),**

**(0, 'Émile Zola/Robert Lethbridge/Elinor Dorday'),**

**(0, 'Émile Zola/Henri Mitterand'),**

**(0, 'Émile Zola/Ernest Alfred Vizetelly/Henry Vizetelly'),**

**(0, 'Émile Zola/Ernest Alfred Vizetelly'),**

**(0, 'Émile Zola/Douglas Parmée'),**

**(0, 'Émile Zola/Andrew Moore/Ernest Alfred Vizetelly'),**

**(0, 'Åsne Seierstad/Ingrid Christopherson')]**