Yash Verma
CSE-AI & ML
3RD YEAR
500069950
ROLL NO.110
B2 BATCH
COMPUTATIONAL LINGUISTICS AND NLP LAB

Every file related to this assignment is uploaded on Github, under this link:
Especially for the last question,
https://github.com/yashverma7/user_song_play

Colab Notebook:
https://colab.research.google.com/drive/1IFevluuTgRXmaGoUwXuuKGqwvc-G5xRE

Lab 1

Attached Files:

- train_triplets_sample.txt (123.113 KB)

Attached is a .txt file bearing three columns: ['user','song','play_count'].

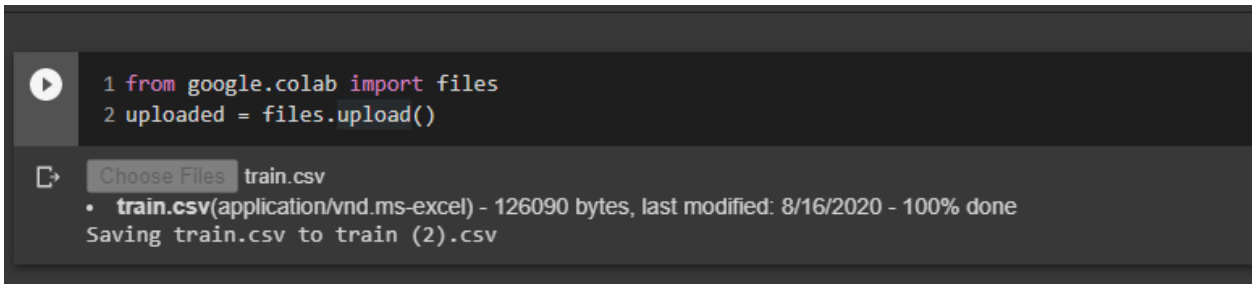
1) Read it as a .csv file and create a dataframe and display it.

2) Display per unique user play_count.

3) Display per unique song play_count.

4) Display top users and songs with respect to play_count and save as different .csv files.

1) Read it as a .csv file and create a dataframe and display it.



```
1 from google.colab import files
2 uploaded = files.upload()
```

```
Choose Files train.csv
• train.csv(application/vnd.ms-excel) - 126090 bytes, last modified: 8/16/2020 - 100% done
Saving train.csv to train (2).csv
```

```
1 import pandas as pd
2 import io
3
4 df = pd.read_csv(io.BytesIO(uploaded['train.csv']))
5 triplet_dataset = pd.read_csv('train.csv',nrows = 10000, sep = ',', names = ['user','song','play_count'], header=1)
```

```
output_dict = {}
with open('train.csv') as f:
    for line_number,line in enumerate(f):
        user = line.split(',')[0]
        play_count = int(line.split(',')[2])
        if user in output_dict:
            play_count += output_dict[user]
            output_dict.update({user:play_count})
        output_dict.update({user:play_count})


output_list = [{'user': k, 'play_count':v} for k,v in output_dict.items()]
play_count_df = pd.DataFrame(output_list)
```

```
[28]  print (df)
```

```
                                       user                song  play_count
0     b80344d063b5ccb3212f76538f3d9e43d87dca9e  SOAKIMP12A8C130995           1
1     b80344d063b5ccb3212f76538f3d9e43d87dca9e  SOAPDEY12A81C210A9           1
2     b80344d063b5ccb3212f76538f3d9e43d87dca9e  SOBBMDR12A8C13253B           2
3     b80344d063b5ccb3212f76538f3d9e43d87dca9e  SOBFNSP12AF72A0E22           1
4     b80344d063b5ccb3212f76538f3d9e43d87dca9e  SOBFOVM12A58A7D494           1
...                                        ...                 ...         ...
1995  951945330eb5df161ac4f97729647514001cd102  SOSBQSS12AB0189C58           4
1996  951945330eb5df161ac4f97729647514001cd102  SOSJTOL12A6D4FB4CD           2
1997  951945330eb5df161ac4f97729647514001cd102  SOSYKTA12AF72A3686           2
1998  951945330eb5df161ac4f97729647514001cd102  SOTIEEP12A6701C779           3
1999  951945330eb5df161ac4f97729647514001cd102  SOUQFCH12AB0184983          10

[2000 rows x 3 columns]
```

2) Display per unique user play_count.

	user	play_count
0	b80344d063b5ccb3212f76538f3d9e43d87dca9e	125
1	85c1f87fea955d09b4bec2e36aee110927aedf9a	38
2	bd4c6e843f00bd476847fb75c47b4fb430a06856	14
3	8937134734f869debcab8f23d77465b4caaa85df	78
4	969cc6fb74e076a68e36a04409cb9d3765757508	94
5	4bd88bfb25263a75bbdd467e74018f4ae570e5df	95
6	e006b1a48f466bf59feefed32bec6494495a4436	73
7	9d6f0ead607ac2a6c2460e4d14fb439a146b7dec	24
8	9bb911319fbc04f01755814cb5edb21df3d1a336	58
9	b64cdd1a0bd907e5e00b39e345194768e330d652	164
10	17aa9f6dbdf753831da8f38c71b66b64373de613	208
11	d6589314c0a9bcbca4fee0c93b14bc402363afea	354
12	5a905f000fc1ff3df7ca807d57edb608863db05d	1422
13	c737ec8c1b16ce8e39115f4432c9a7fc21ec47a1	16
14	45544491ccfcdc0b0803c34f201a6287ed4e30f8	39
15	ed7d4c476013b1c3dd91982b61494bf7436083ba	22
16	baf47ed8da24d607e50d8684cde78b923538640f	413
17	169f9f4c68b62d1887c7c0ac99d10a79cfca5daf	34
18	a820d2d4f16bbd53be9e41e0417dfb234bfdfba8	105
19	bd8475385f0aa78830fa6dfce9e7242164b035c8	228
20	0afaa5d9d04bf85af720fe8cc566a41ca3e41c97	107
21	81bde1c3a845c64f1677bd9d28f2da85dfefcf30	28
22	403b3b867fc71dfdcc12652f30e88bdc7ccd9aa4	58
23	f84f5b5a5c5d1d9fb4866f6488e0d2661b54c192	128
24	a1380d458c15706b9d5282304db81a5a78352e96	120
25	405d396ea64d75b5eaefaaf8ac836f45fa56af4d	134
26	2c42e6551311710ca5a839d62058820a42ead493	135
27	8deca80c3da6024a1456e123308eb94fee1b439f	12
28	bf30441e24ef5326354295723d9fe1edf59b8554	18
29	12768858f6a825452e412deb1df36d2d1d9c6791	41
30	a58de017cbeda1763ea002fe027ed41b4ed53109	606
31	951945330eb5df161ac4f97729647514001cd102	80

3) Display per unique song play_count.

```
output_dict1 = {}
with open('train.csv') as f:
    for line_number,line in enumerate(f):
        song = line.split(',')[1]
        play_count = int(line.split(',')[2])
        if song in output_dict1:
            play_count += output_dict1[song]
            output_dict1.update({song:play_count})
        output_dict1.update({song:play_count})


output_list1 = [{'song': k, 'play_count':v} for k,v in output_dict1.items()]
play_song_count_df = pd.DataFrame(output_list1)
```

```
[39]  print(play_song_count_df)
```

```
                    song  play_count
0     SOAKIMP12A8C130995           1
1     SOAPDEY12A81C210A9           1
2     SOBBMDR12A8C13253B           2
3     SOBFNSP12AF72A0E22           1
4     SOBFOVM12A58A7D494           1
...                  ...         ...
1787  SORVZLT12AB0189C7B           2
1788  SOSBEAB12A6D4FACF3           5
1789  SOSBQSS12AB0189C58           4
1790  SOSYKTA12AF72A3686           2
1791  SOUQFCH12AB0184983          10

[1792 rows x 2 columns]
```

4) Display top users and songs with respect to play_count and save as different .csv files.

(Files are uploaded on Github, link at the start of the assignment)

```
[47]  1 topuser_song_df= play_count_df.sort_values(by='play_count')
      2 print (topuser_song_df)
```

	user	play_count
27	8deca80c3da6024a1456e123308eb94fee1b439f	12
2	bd4c6e843f00bd476847fb75c47b4fb430a06856	14
13	c737ec8c1b16ce8e39115f4432c9a7fc21ec47a1	16
28	bf30441e24ef5326354295723d9fe1edf59b8554	18
15	ed7d4c476013b1c3dd91982b61494bf7436083ba	22
7	9d6f0ead607ac2a6c2460e4d14fb439a146b7dec	24
21	81bde1c3a845c64f1677bd9d28f2da85dfefcf30	28
17	169f9f4c68b62d1887c7c0ac99d10a79cfca5daf	34
1	85c1f87fea955d09b4bec2e36aee110927aedf9a	38
14	45544491ccfcdc0b0803c34f201a6287ed4e30f8	39
29	12768858f6a825452e412deb1df36d2d1d9c6791	41
8	9bb911319fbc04f01755814cb5edb21df3d1a336	58
22	403b3b867fc71dfdcc12652f30e88bdc7ccd9aa4	58
6	e006b1a48f466bf59feefed32bec6494495a4436	73
3	8937134734f869debcab8f23d77465b4caaa85df	78
31	951945330eb5df161ac4f97729647514001cd102	80
4	969cc6fb74e076a68e36a04409cb9d3765757508	94
5	4bd88bfb25263a75bbdd467e74018f4ae570e5df	95
18	a820d2d4f16bbd53be9e41e0417dfb234bfdfba8	105
20	0afaa5d9d04bf85af720fe8cc566a41ca3e41c97	107
24	a1380d458c15706b9d5282304db81a5a78352e96	120
0	b80344d063b5ccb3212f76538f3d9e43d87dca9e	125
23	f84f5b5a5c5d1d9fb4866f6488e0d2661b54c192	128
25	405d396ea64d75b5eaefaaf8ac836f45fa56af4d	134
26	2c42e6551311710ca5a839d62058820a42ead493	135
9	b64cdd1a0bd907e5e00b39e345194768e330d652	164
10	17aa9f6dbdf753831da8f38c71b66b64373de613	208
19	bd8475385f0aa78830fa6dfce9e7242164b035c8	228
11	d6589314c0a9bcbca4fee0c93b14bc402363afea	354
16	baf47ed8da24d607e50d8684cde78b923538640f	413
30	a58de017cbeda1763ea002fe027ed41b4ed53109	606
12	5a905f000fc1ff3df7ca807d57edb608863db05d	1422

```
[46]  1 topsong_play_df= play_song_count_df.sort_values(by='play_count')
      2 print (topsong_play_df)
```

```
                     song  play_count
0     SOAKIMP12A8C130995           1
1046  SOEUDBC12A8C140BEA           1
1045  SOETKSY12A8C13C666           1
1044  SOEJNVA12A67AE225C           1
1686  SOUXEOI12A6D4FB18E           1
...                  ...         ...
182   SOAUWYT12A81C206F1          25
1240  SOGKEGN12AB0185355          26
740   SOMVTRL12A67AE0921          28
198   SONYKOW12AB01849C9          29
185   SOBONKR12A58A7A7E0          36

[1792 rows x 2 columns]
```

```
# -*- coding: utf-8 -*-
"""Untitled6.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1IFevluuTgRXmaGoUwXuuKGqwvc-G5xRE
"""

from google.colab import files
uploaded = files.upload()
```

```
import pandas as pd
import io

df = pd.read_csv(io.BytesIO(uploaded['train.csv']))
triplet_dataset = pd.read_csv('train.csv',nrows =
10000, sep = ',', names = ['user','song','play_count'],
header=1)

output_dict = {}
with open('train.csv') as f:
    for line_number,line in enumerate(f):
        user = line.split(',')[0]
        play_count = int(line.split(',')[2])
        if user in output_dict:
            play_count += output_dict[user]
            output_dict.update({user:play_count})
        output_dict.update({user:play_count})


output_list = [{'user': k, 'play_count':v} for k,v in
output_dict.items()]
play_count_df = pd.DataFrame(output_list)

print (df)

print (play_count_df)

output_dict1 = {}
with open('train.csv') as f:
    for line_number,line in enumerate(f):
        song = line.split(',')[1]
        play_count = int(line.split(',')[2])
        if song in output_dict1:
            play_count += output_dict1[song]
            output_dict1.update({song:play_count})
        output_dict1.update({song:play_count})
```

```
output_list1 = [{'song': k, 'play_count':v} for k,v in
output_dict1.items()]
play_song_count_df = pd.DataFrame(output_list1)

print(play_song_count_df)

topuser_song_df=
play_count_df.sort_values(by='play_count')
print (topuser_song_df)

topsong_play_df=
play_song_count_df.sort_values(by='play_count')
print (topsong_play_df)
```