EDS THEORY ACTIVITY NO-1

NAME- SAMRUDDHI DEEEPAK PHULARI

ROLL NO – CS4-34

PRN NO – 202401040129

DATASET – MOVIELENS LATEST DATASET

	A	B	C	D	E
1	userId	movieId	tag	timestamp	
2	2	60756	funny	1.45E+09	
3	2	60756	Highly quo	1.45E+09	
4	2	60756	will ferrell	1.45E+09	
5	2	89774	Boxing sto	1.45E+09	
6	2	89774	MMA	1.45E+09	
7	2	89774	Tom Hardy	1.45E+09	
8	2	106782	drugs	1.45E+09	
9	2	106782	Leonardo	1.45E+09	
10	2	106782	Martin Scc	1.45E+09	
11	7	48516	way too lc	1.17E+09	
12	18	431	Al Pacino	1.46E+09	
13	18	431	gangster	1.46E+09	
14	18	431	mafia	1.46E+09	
15	18	1221	Al Pacino	1.46E+09	
16	18	1221	Mafia	1.46E+09	
17	18	5995	holocaust	1.46E+09	
18	18	5995	true story	1.46E+09	
19	18	44665	twist endir	1.46E+09	
20	18	52604	Anthony H	1.46E+09	
21	18	52604	courtroom	1.46E+09	
22	18	52604	twist endir	1.46E+09	
23	18	88094	britpop	1.46E+09	
24	18	88094	indie recor	1.46E+09	
25	18	88094	music	1.46E+09	
26	18	144210	dumpster (	1.46E+09	

7	18	144210	Sustainabil	1.46E+09
8	21	1569	romantic c	1.42E+09
9	21	1569	wedding	1.42E+09
0	21	118985	painter	1.42E+09
1	21	119141	bloody	1.42E+09
2	49	109487	black hole	1.49E+09
3	49	109487	sci-fi	1.49E+09
4	49	109487	time-trave	1.49E+09
5	62	2	fantasy	1.53E+09
6	62	2	magic boa	1.53E+09
7	62	2	Robin Willi	1.53E+09
8	62	110	beautiful s	1.53E+09
9	62	110	epic	1.53E+09
0	62	110	historical	1.53E+09
1	62	110	inspiration	1.53E+09
2	62	110	Medieval	1.53E+09
3	62	110	mel gibson	1.53E+09
4	62	110	Oscar (Bes	1.53E+09
5	62	110	revenge	1.53E+09
6	62	110	sword figh	1.53E+09
7	62	410	black com	1.53E+09
8	62	410	Christina R	1.53E+09
9	62	410	Christophe	1.53E+09
0	62	410	dark come	1.53E+09
1	62	410	family	1.53E+09
2	62	410	gothic	1.53E+09

76	260	action	1.44E+09
76	260	sci-fi	1.44E+09
103	260	EPIC	1.43E+09
103	260	great soun	1.43E+09
103	296	good dialo	1.43E+09
103	296	great soun	1.43E+09
103	296	non-linear	1.43E+09
106	4896	Everything	1.47E+09
106	106489	adventure	1.47E+09
112	260	classic sci-	1.44E+09
112	260	engrossing	1.44E+09
112	260	EPIC	1.44E+09
119	260	classic	1.44E+09
119	260	Nerd	1.44E+09
119	101142	animation	1.44E+09
119	101142	funny	1.44E+09
119	101142	stone age	1.44E+09
119	115149	action	1.44E+09
119	115149	killer	1.44E+09
119	115149	widows/w	1.44E+09
119	115617	animation	1.44E+09
119	115617	kids	1.44E+09
119	115617	robots	1.44E+09
119	120635	action	1.44E+09
119	120635	murder	1.44E+09
119	120635	police	1.44E+09

Files

Analyze your files with code written by Gemini Upload

..

sample_data

tags.csv

```
import pandas as pd
df=pd.read_csv('/content/tags.csv')
print(df)
```

```
      userId  movieId              tag   timestamp
0          2    60756            funny  1445714994
1          2    60756  Highly quotable  1445714996
2          2    60756     will ferrell  1445714992
3          2    89774     Boxing story  1445715207
4          2    89774              MMA  1445715200
...      ...      ...              ...         ...
3678     606     7382        for katie  1171234019
3679     606     7936          austere  1173392334
3680     610     3265           gun fu  1493843984
3681     610     3265  heroic bloodshed  1493843978
3682     610   168248  Heroic Bloodshed  1493844270

[3683 rows x 4 columns]
```

```
sum_id=df['userId'].sum()
print(sum_id)
```

```
1587923
```

[14]

```
2. # Find total number of rows (tags) in the dataset.#
print(len(df))
```

```
3683
```

[15]

```
3.# Find the total number of unique users.#
print(df['userId'].nunique())
```

```
58
```

[16]

```
4.#Find the total number of unique movies.#
print(df['movieId'].nunique())
```

```
1572
```

[18]

```
5.# Find the total number of unique tags.#
print(df['tag'].nunique())
```

```
1589
```

```
6.# Find the most common tag used.#
print(df['tag'].value_counts().idxmax())
```

```
In Netflix queue
```

[19]

```
7.#Find the least common tag#
least_common_tag = df['tag'].value_counts().idxmin()
print(least_common_tag)
```

```
Heroic Bloodshed
```

```
[20] 8.#Find how many tags contain the word "funny".#
     funny_tags_count = df['tag'].str.contains('funny', case=False).sum()
     print(funny_tags_count)
```

```
28
```

```
9.#Find the number of tags created each year.#
df['year'] = df['timestamp'].dt.year
tag_counts_by_year = df.groupby('year').size()
print(tag_counts_by_year)
```

```
year
2006    1533
2007      46
2008       9
2009     166
2010     133
2011      13
2012      47
2013      10
2014       7
2015     191
2016     355
2017     329
2018     844
dtype: int64
```

```
[24] 10.#Find the average number of tags per user.#
     average_tags_per_user = df.groupby('userId').size().mean()
     print(average_tags_per_user)
```

```
63.5
```

```
[25] 11.#Find the user who assigned the most tags.#
     user_with_most_tags = df['userId'].value_counts().idxmax()
     print(user_with_most_tags)
```

```
474
```

```
[26] 12.#Find the movie that received the most tags.#
     movie_with_most_tags = df['movieId'].value_counts().idxmax()
     print(movie_with_most_tags)
```

```
296
```

```
[27] 13.#Find the earliest tagging time.#
     earliest_tagging_time = df['timestamp'].min()
     print(earliest_tagging_time)
```

```
2006-01-13 19:09:12
```

[28]
```python
14.# Find the latest tagging time.#
latest_tagging_time = df['timestamp'].max()
print(latest_tagging_time)
```

```
2018-09-16 11:50:03
```

[29]
```python
15.# Find users who tagged more than 1000 movies.#
users_with_more_than_1000_movies = df.groupby('userId').filter(lambda x: len(x) > 1000)
print(users_with_more_than_1000_movies)
```

```
      userId  movieId               tag           timestamp  year
981      474        1             pixar 2006-01-14 02:47:05  2006
982      474        2              game 2006-01-16 01:39:12  2006
983      474        5         pregnancy 2006-01-16 01:11:43  2006
984      474        5            remake 2006-01-16 01:11:43  2006
985      474        7            remake 2006-01-16 01:40:42  2006
...      ...      ...               ...                 ...   ...
2483     474    40819      Johnny Cash 2006-01-14 01:03:15  2006
2484     474    41566        C.S. Lewis 2006-01-13 19:46:57  2006
2485     474    41997  In Netflix queue 2006-01-13 19:13:23  2006
2486     474    42002  In Netflix queue 2006-01-14 01:29:10  2006
2487     474    42740  In Netflix queue 2006-02-01 14:30:37  2006

[1507 rows x 5 columns]
```

```python
16.# Find tags used by more than 10 users.#
tags_used_by_more_than_10_users = df.groupby('tag').filter(lambda x: len(x) > 10)
print(tags_used_by_more_than_10_users)
```

```
      userId  movieId                 tag           timestamp  year
0          2    60756               funny 2015-10-24 19:29:54  2015
17        18    44665        twist ending 2016-03-02 19:51:23  2016
20        18    52604        twist ending 2016-03-10 22:58:02  2016
23        18    88094               music 2016-03-08 13:43:29  2016
31        49   109487              sci-fi 2017-04-25 04:08:52  2017
...      ...      ...                 ...                 ...   ...
3659     599     2959              quirky 2017-06-26 06:01:58  2017
3665     599     2959             surreal 2017-06-26 06:01:40  2017
3667     599     2959  thought-provoking 2017-06-26 06:01:41  2017
3669     599     2959        twist ending 2017-06-26 06:01:28  2017
3673     606     1357               music 2007-04-16 23:16:33  2007

[855 rows x 5 columns]
```

[31]
```
17.#Find the number of different tags assigned to each movie.#
tags_per_movie = df.groupby('movieId')['tag'].nunique()
print(tags_per_movie)
```

```
movieId
1         2
2         4
3         2
5         2
7         1
         ..
183611    3
184471    3
187593    3
187595    2
193565    4
Name: tag, Length: 1572, dtype: int64
```

[ ]
```
18.#Find the average time difference between two tags.#
time_diff = df['timestamp'].sort_values().diff().mean()
print(time_diff)
```

```
1 days 06:10:14.679793590
```

[33]
```
19.# Find movies that were tagged in 2015.#
movies_in_2015 = df[df['timestamp'].dt.year == 2015]['movieId'].unique()
print(movies_in_2015)
```

```
[ 60756  89774 106782    260    296 101142 115149 115617 120635   4878
   7147   7361  97938 126548   6711  71494  90769  96084 118784 127172
   5952  98809 112552  80834    356    527   1625   1721   2329   2571
   3949   5989   8533  55765  58295  69122  69481  72998  79132  80489
  80906  89745  90439  92259  95067  95441  97304  99114 104218 104841
 105504]
```

```
20.# List all tags containing "Al Pacino".#
al_pacino_tags = df[df['tag'].str.contains('Al Pacino', case=False)]['tag'].unique()
print(al_pacino_tags)
```

```
['Al Pacino']
```