In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
ount("/content/drive", force_remount=True).
```

In [ ]:
```
netflix_dataset1 = pd.read_csv('/content/drive/MyDrive/Netflix csv/combined_data_1.txt
netflix_dataset1.head()
```

Out[ ]:

		1:
1488844	**3.0**	2005-09-06
822109	**5.0**	2005-05-13
885013	**4.0**	2005-10-19
30878	**4.0**	2005-12-26
823519	**3.0**	2004-05-03

In [ ]:
```
# Reading dataset file
netflix_dataset = pd.read_csv('/content/drive/MyDrive/Netflix csv/combined_data_1.txt'
netflix_dataset.head()
```

Out[ ]:

	Cust_Id	Rating
0	1:	NaN
1	1488844	3.0
2	822109	5.0
3	885013	4.0
4	30878	4.0

In [ ]:
```
netflix_dataset
```

Out[ ]:

	Cust_Id	Rating
0	1:	NaN
1	1488844	3.0
2	822109	5.0
3	885013	4.0
4	30878	4.0
...	...	...
24058258	2591364	2.0
24058259	1791000	2.0
24058260	512536	5.0
24058261	988963	3.0
24058262	1704416	3.0

24058263 rows × 2 columns

In [ ]:
```
netflix_dataset.tail()
```

Out[ ]:

	Cust_Id	Rating
24058258	2591364	2.0
24058259	1791000	2.0
24058260	512536	5.0
24058261	988963	3.0
24058262	1704416	3.0

In [ ]:
```
netflix_dataset.dtypes
```

Out[ ]:
```
Cust_Id     object
Rating     float64
dtype: object
```

In [ ]:
```
netflix_dataset.isnull()
```

Out[ ]:

	Cust_Id	Rating
0	False	True
1	False	False
2	False	False
3	False	False
4	False	False
...	...	...
24058258	False	False
24058259	False	False
24058260	False	False
24058261	False	False
24058262	False	False

24058263 rows × 2 columns

In [ ]:
```
netflix_dataset.isnull().sum()
```

Out[ ]:
```
Cust_Id       0
Rating     4499
dtype: int64
```

In [ ]:
```
netflix_dataset.shape
```

Out[ ]:
```
(24058263, 2)
```

In [ ]:
```
#Get the customer count with NaN values
movie_count=netflix_dataset.isnull().sum()
movie_count=movie_count["Rating"]
```

In [ ]:
```
movie_count
```

Out[ ]:
```
4499
```

In [ ]:
```
#To calculate how many customers we are having in the dataset
customer_count=netflix_dataset['Cust_Id'].nunique()
```

In [ ]:
```
customer_count  #This count also inculdes uniques movies id's
```

Out[ ]:
```
475257
```

In [ ]:
```
#Count Without NaN values
customer_count= customer_count-movie_count
customer_count
```

Out[ ]:
```
470758
```

In [ ]:
```
netflix_dataset['Cust_Id'].count() # these are the number of times ratings given
```

Out[ ]: 24058263

In [ ]:
```
#get the total number of ratings given by the customers
rating_count=netflix_dataset['Cust_Id'].count()-movie_count
rating_count
```

Out[ ]: 24053764

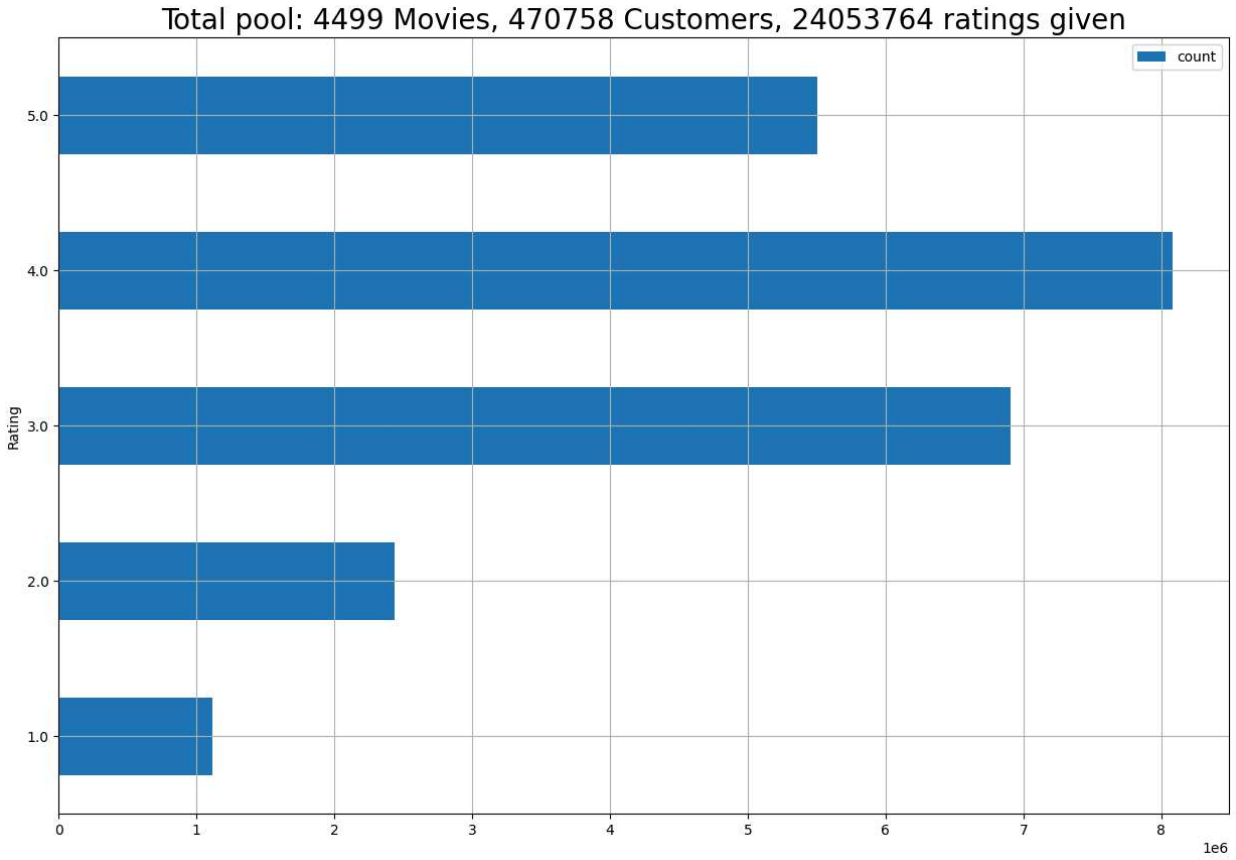
In [ ]:
```
#To find out how many people have rated the movies as 1,2,3,4,5 stars ratings to the m

stars=netflix_dataset.groupby('Rating')['Rating'].agg(['count'])
```

In [ ]:
```
stars
```

Out[ ]:

Rating	count
1.0	1118186
2.0	2439073
3.0	6904181
4.0	8085741
5.0	5506583

In [ ]:
```
ax=stars.plot(kind='barh',figsize=(15,10))
plt.title(f'Total pool: {movie_count} Movies, {customer_count} Customers, {rating_cour
plt.grid(True)
```



In [ ]:
```
pd.isnull(netflix_dataset.Rating)
```

Out[ ]:
```
0           True
1          False
2          False
3          False
4          False
           ...
24058258   False
24058259   False
24058260   False
24058261   False
24058262   False
Name: Rating, Length: 24058263, dtype: bool
```

In [ ]:
```
#Add another column that will have movie id
#First of all we have to calculate how many null values we are having in the ratings c
df_nan=pd.DataFrame(pd.isnull(netflix_dataset.Rating))
```

In [ ]:
```
df_nan
```

Out[ ]:

	Rating
0	True
1	False
2	False
3	False
4	False
...	...
24058258	False
24058259	False
24058260	False
24058261	False
24058262	False

24058263 rows × 1 columns

In [ ]:
```
df_nan=df_nan[df_nan['Rating']==True]
df_nan
```

Out[ ]:

	Rating
0	True
548	True
694	True
2707	True
2850	True
...	...
24046714	True
24047329	True
24056849	True
24057564	True
24057834	True

4499 rows × 1 columns

In [ ]:
```
df_nan.shape
```

Out[ ]:
```
(4499, 1)
```

In [ ]:
```
df_nan.head()
```

Out[ ]:

	Rating
0	True
548	True
694	True
2707	True
2850	True

In [ ]:
```
#now we have to reset the index as the column
df_nan=df_nan.reset_index()
```

In [ ]:
```
df_nan
```

Out[ ]:

	index	Rating
0	0	True
1	548	True
2	694	True
3	2707	True
4	2850	True
...	...	...
4494	24046714	True
4495	24047329	True
4496	24056849	True
4497	24057564	True
4498	24057834	True

4499 rows × 2 columns

In [ ]:
```
#we have to extract all the records from the index column except for the last index--
df_nan['index'][:-1]
```

Out[ ]:
```
0              0
1            548
2            694
3           2707
4           2850
          ...
4493    24046583
4494    24046714
4495    24047329
4496    24056849
4497    24057564
Name: index, Length: 4498, dtype: int64
```

In [ ]:
```
#This sytax will extract records from the index column from the 1st index
df_nan['index'][1:]
```

Out[ ]:
```
1            548
2            694
3           2707
4           2850
5           3991
          ...
4494    24046714
4495    24047329
4496    24056849
4497    24057564
4498    24057834
Name: index, Length: 4498, dtype: int64
```

In [ ]:
```
#Working of full function
np.full((2,4), '2.0')
```

Out[ ]:
```
array([['2.0', '2.0', '2.0', '2.0'],
       ['2.0', '2.0', '2.0', '2.0']], dtype='<U3')
```

In [ ]:
```
#working
x=zip(df_nan['index'][1:], df_nan['index'][:-1])
x
```

Out[ ]:
```
<zip at 0x79891929d0c0>
```

In [ ]:
```
tuple(x)
```

Out[ ]:
```
((548, 0),
 (694, 548),
 (2707, 694),
 (2850, 2707),
 (3991, 2850),
 (5011, 3991),
 (5105, 5011),
 (20016, 5105),
 (20112, 20016),
 (20362, 20112),
 (20561, 20362),
 (21108, 20561),
 (21234, 21108),
 (21353, 21234),
 (21644, 21353),
 (24344, 21644),
 (31453, 24344),
 (42176, 31453),
 (42716, 42176),
 (42833, 42716),
 (43052, 42833),
 (43256, 43052),
 (43872, 43256),
 (45206, 43872),
 (46414, 45206),
 (52276, 46414),
 (52550, 52276),
 (92303, 52550),
 (92827, 92303),
 (211241, 92827),
 (211463, 211241),
 (213318, 211463),
 (220209, 213318),
 (220318, 220209),
 (221158, 220318),
 (222098, 221158),
 (222380, 222098),
 (223183, 222380),
 (223936, 223183),
 (224428, 223936),
 (224522, 224428),
 (224651, 224522),
 (224757, 224651),
 (233259, 224757),
 (235879, 233259),
 (242438, 235879),
 (244789, 242438),
 (248381, 244789),
 (248613, 248381),
 (248942, 248613),
 (249033, 248942),
 (254181, 249033),
 (254296, 254181),
 (254456, 254296),
 (256863, 254456),
 (258697, 256863),
 (262260, 258697),
 (279666, 262260),
 (279871, 279666),
 (280167, 279871),
```

```
 (280315, 280167),
 (280606, 280315),
 (280791, 280606),
 (280887, 280791),
 (281146, 280887),
 (281313, 281146),
 (281603, 281313),
 (283820, 281603),
 (283937, 283820),
 (284281, 283937),
 (285843, 284281),
 (286023, 285843),
 (286771, 286023),
 (287160, 286771),
 (287380, 287160),
 (290335, 287380),
 (303934, 290335),
 (308785, 303934),
 (312183, 308785),
 (312458, 312183),
 (313708, 312458),
 (313828, 313708),
 (332553, 313828),
 (335306, 332553),
 (335573, 335306),
 (335750, 335573),
 (335868, 335750),
 (336847, 335868),
 (337726, 336847),
 (338352, 337726),
 (338920, 338352),
 (339082, 338920),
 (339257, 339082),
 (339600, 339257),
 (340536, 339600),
 (341025, 340536),
 (352534, 341025),
 (352665, 352534),
 (352792, 352665),
 (352871, 352792),
 (353202, 352871),
 (353533, 353202),
 (353743, 353533),
 (357244, 353743),
 (357437, 357244),
 (358378, 357437),
 (359208, 358378),
 (378227, 359208),
 (378773, 378227),
 (380933, 378773),
 (411061, 380933),
 (411301, 411061),
 (412695, 411301),
 (413124, 412695),
 (413791, 413124),
 (414918, 413791),
 (415900, 414918),
 (435141, 415900),
 (435238, 435141),
 (435578, 435238),
```

```
(435678, 435578),
(439061, 435678),
(439338, 439061),
(439508, 439338),
(441238, 439508),
(442423, 441238),
(445438, 442423),
(445856, 445438),
(446613, 445856),
(446700, 446613),
(447406, 446700),
(448237, 447406),
(450801, 448237),
(450901, 450801),
(451286, 450901),
(451407, 451286),
(451612, 451407),
(457620, 451612),
(457717, 457620),
(457837, 457717),
(457964, 457837),
(458291, 457964),
(496654, 458291),
(496944, 496654),
(497400, 496944),
(497767, 497400),
(497899, 497767),
(520094, 497899),
(520157, 520094),
(520423, 520157),
(520600, 520423),
(529763, 520600),
(530085, 529763),
(530244, 530085),
(531225, 530244),
(539405, 531225),
(540111, 539405),
(540486, 540111),
(540651, 540486),
(541551, 540651),
(542201, 541551),
(542676, 542201),
(542870, 542676),
(543523, 542870),
(548112, 543523),
(557548, 548112),
(566700, 557548),
(568054, 566700),
(568390, 568054),
(568606, 568390),
(574797, 568606),
(575120, 574797),
(577357, 575120),
(577549, 577357),
(668000, 577549),
(668211, 668000),
(668453, 668211),
(670385, 668453),
(670672, 670385),
(673091, 670672),
```

```
(676650, 673091),
(676885, 676650),
(677049, 676885),
(677449, 677049),
(679747, 677449),
(680039, 679747),
(700089, 680039),
(702639, 700089),
(716777, 702639),
(716882, 716777),
(815603, 716882),
(815879, 815603),
(816001, 815879),
(816648, 816001),
(816834, 816648),
(816989, 816834),
(898250, 816989),
(898445, 898250),
(933955, 898445),
(934085, 933955),
(936810, 934085),
(937193, 936810),
(937290, 937193),
(937521, 937290),
(938035, 937521),
(939132, 938035),
(939391, 939132),
(941588, 939391),
(945962, 941588),
(946650, 945962),
(947305, 946650),
(948196, 947305),
(951679, 948196),
(951804, 951679),
(957622, 951804),
(966518, 957622),
(966944, 966518),
(967060, 966944),
(967172, 967060),
(967394, 967172),
(967556, 967394),
(967656, 967556),
(991782, 967656),
(992869, 991782),
(1000265, 992869),
(1000455, 1000265),
(1001786, 1000455),
(1001882, 1001786),
(1002535, 1001882),
(1002662, 1002535),
(1003431, 1002662),
(1008408, 1003431),
(1008827, 1008408),
(1008925, 1008827),
(1009368, 1008925),
(1009494, 1009368),
(1009843, 1009494),
(1012179, 1009843),
(1014370, 1012179),
(1019555, 1014370),
```

```
            (1062678, 1019555),
            (1065054, 1062678),
            (1065174, 1065054),
            (1065280, 1065174),
            (1065372, 1065280),
            (1065804, 1065372),
            (1066458, 1065804),
            (1072620, 1066458),
            (1072818, 1072620),
            (1073592, 1072818),
            (1074797, 1073592),
            (1099185, 1074797),
            (1104236, 1099185),
            (1104427, 1104236),
            (1109014, 1104427),
            (1121279, 1109014),
            (1148637, 1121279),
            (1148802, 1148637),
            (1149679, 1148802),
            (1149885, 1149679),
            (1150412, 1149885),
            (1154791, 1150412),
            (1155692, 1154791),
            (1155827, 1155692),
            (1156136, 1155827),
            (1156263, 1156136),
            (1157426, 1156263),
            (1162972, 1157426),
            (1184440, 1162972),
            (1220872, 1184440),
            (1221351, 1220872),
            (1221414, 1221351),
            (1253442, 1221414),
            (1254217, 1253442),
            (1264821, 1254217),
            (1264937, 1264821),
            (1265095, 1264937),
            (1265179, 1265095),
            (1265917, 1265179),
            (1266457, 1265917),
            (1269675, 1266457),
            (1270069, 1269675),
            (1276446, 1270069),
            (1276544, 1276446),
            (1290553, 1276544),
            (1290745, 1290553),
            (1290996, 1290745),
            (1292145, 1290996),
            (1296558, 1292145),
            (1344210, 1296558),
            (1344319, 1344210),
            (1344854, 1344319),
            (1345263, 1344854),
            (1345726, 1345263),
            (1374681, 1345726),
            (1374798, 1374681),
            (1376755, 1374798),
            (1377211, 1376755),
            (1454156, 1377211),
            (1454269, 1454156),
```

```
            (1454445, 1454269),
            (1455325, 1454445),
            (1455551, 1455325),
            (1458979, 1455551),
            (1471800, 1458979),
            (1471998, 1471800),
            (1472121, 1471998),
            (1472322, 1472121),
            (1472776, 1472322),
            (1473222, 1472776),
            (1498058, 1473222),
            (1563536, 1498058),
            (1663349, 1563536),
            (1663864, 1663349),
            (1663974, 1663864),
            (1666495, 1663974),
            (1666590, 1666495),
            (1666686, 1666590),
            (1666806, 1666686),
            (1666931, 1666806),
            (1667076, 1666931),
            (1667975, 1667076),
            (1668688, 1667975),
            (1669080, 1668688),
            (1670202, 1669080),
            (1670368, 1670202),
            (1670547, 1670368),
            (1670745, 1670547),
            (1737012, 1670745),
            (1764051, 1737012),
            (1811219, 1764051),
            (1811369, 1811219),
            (1811763, 1811369),
            (1851639, 1811763),
            (1851789, 1851639),
            (1851989, 1851789),
            (1854283, 1851989),
            (1855686, 1854283),
            (1856240, 1855686),
            (1856419, 1856240),
            (1858282, 1856419),
            (1859641, 1858282),
            (1859734, 1859641),
            (1862339, 1859734),
            (1866515, 1862339),
            (1867001, 1866515),
            (1867110, 1867001),
            (1879850, 1867110),
            (1880434, 1879850),
            (1881586, 1880434),
            (1881716, 1881586),
            (1881836, 1881716),
            (1902297, 1881836),
            (1902440, 1902297),
            (1902591, 1902440),
            (1904462, 1902591),
            (1981777, 1904462),
            (1984786, 1981777),
            (1989367, 1984786),
            (1991599, 1989367),
```

```
(2027449, 1991599),
(2031386, 2027449),
(2033030, 2031386),
(2033154, 2033030),
(2033415, 2033154),
(2033766, 2033415),
(2046204, 2033766),
(2047600, 2046204),
(2048563, 2047600),
(2049048, 2048563),
(2051053, 2049048),
(2051202, 2051053),
(2051288, 2051202),
(2051532, 2051288),
(2053597, 2051532),
(2053791, 2053597),
(2053904, 2053791),
(2062090, 2053904),
(2062249, 2062090),
(2062779, 2062249),
(2069440, 2062779),
(2069599, 2069440),
(2069823, 2069599),
(2071262, 2069823),
(2080631, 2071262),
(2082101, 2080631),
(2082690, 2082101),
(2082908, 2082690),
(2083300, 2082908),
(2088543, 2083300),
(2089116, 2088543),
(2089177, 2089116),
(2101071, 2089177),
(2102689, 2101071),
(2104512, 2102689),
(2104717, 2104512),
(2105466, 2104717),
(2109032, 2105466),
(2109158, 2109032),
(2112852, 2109158),
(2116099, 2112852),
(2116222, 2116099),
(2116663, 2116222),
(2117122, 2116663),
(2124244, 2117122),
(2172862, 2124244),
(2173082, 2172862),
(2175904, 2173082),
(2176352, 2175904),
(2177362, 2176352),
(2177566, 2177362),
(2177887, 2177566),
(2207797, 2177887),
(2209226, 2207797),
(2209353, 2209226),
(2220610, 2209353),
(2222076, 2220610),
(2224138, 2222076),
(2224740, 2224138),
(2229236, 2224740),
```

```
            (2230224, 2229236),
            (2235723, 2230224),
            (2236020, 2235723),
            (2250141, 2236020),
            (2254674, 2250141),
            (2258892, 2254674),
            (2261574, 2258892),
            (2261690, 2261574),
            (2262910, 2261690),
            (2264457, 2262910),
            (2267907, 2264457),
            (2270622, 2267907),
            (2280666, 2270622),
            (2280805, 2280666),
            (2281149, 2280805),
            (2284635, 2281149),
            (2284801, 2284635),
            (2288514, 2284801),
            (2289203, 2288514),
            (2289311, 2289203),
            (2289652, 2289311),
            (2314945, 2289652),
            (2351461, 2314945),
            (2351552, 2351461),
            (2359087, 2351552),
            (2359807, 2359087),
            (2359899, 2359807),
            (2360089, 2359899),
            (2360180, 2360089),
            (2360318, 2360180),
            (2361431, 2360318),
            (2363862, 2361431),
            (2364289, 2363862),
            (2366123, 2364289),
            (2366908, 2366123),
            (2369359, 2366908),
            (2486122, 2369359),
            (2486478, 2486122),
            (2493750, 2486478),
            (2494095, 2493750),
            (2494640, 2494095),
            (2495419, 2494640),
            (2497706, 2495419),
            (2497789, 2497706),
            (2497890, 2497789),
            (2498253, 2497890),
            (2499309, 2498253),
            (2559725, 2499309),
            (2559831, 2559725),
            (2560411, 2559831),
            (2564912, 2560411),
            (2565428, 2564912),
            (2589031, 2565428),
            (2592067, 2589031),
            (2593109, 2592067),
            (2595137, 2593109),
            (2597379, 2595137),
            (2597811, 2597379),
            (2599584, 2597811),
            (2599944, 2599584),
```

```
(2600321, 2599944),
(2656826, 2600321),
(2746837, 2656826),
(2747070, 2746837),
(2748397, 2747070),
(2752298, 2748397),
(2752576, 2752298),
(2755824, 2752576),
(2760651, 2755824),
(2761525, 2760651),
(2761814, 2761525),
(2763538, 2761814),
(2763645, 2763538),
(2791247, 2763645),
(2791787, 2791247),
(2792023, 2791787),
(2792599, 2792023),
(2792882, 2792599),
(2798671, 2792882),
(2799204, 2798671),
(2814384, 2799204),
(2815962, 2814384),
(2816214, 2815962),
(2818558, 2816214),
(2820934, 2818558),
(2821092, 2820934),
(2822402, 2821092),
(2822597, 2822402),
(2822784, 2822597),
(2822969, 2822784),
(2824683, 2822969),
(2824981, 2824683),
(2825406, 2824981),
(2826274, 2825406),
(2826407, 2826274),
(2846589, 2826407),
(2853157, 2846589),
(2855426, 2853157),
(2855768, 2855426),
(2855944, 2855768),
(2856077, 2855944),
(2856398, 2856077),
(2856693, 2856398),
(2862830, 2856693),
(2862976, 2862830),
(2863558, 2862976),
(2863830, 2863558),
(2892285, 2863830),
(2892775, 2892285),
(2892932, 2892775),
(2893339, 2892932),
(2894397, 2893339),
(2900313, 2894397),
(2905631, 2900313),
(2911742, 2905631),
(2912159, 2911742),
(2912347, 2912159),
(2912550, 2912347),
(2912779, 2912550),
(2912942, 2912779),
```

```
(2913047, 2912942),
(2913219, 2913047),
(2914429, 2913219),
(2914583, 2914429),
(2914922, 2914583),
(2915079, 2914922),
(2915679, 2915079),
(2923419, 2915679),
(2923492, 2923419),
(2957290, 2923492),
(2972113, 2957290),
(2972844, 2972113),
(2973444, 2972844),
(2979335, 2973444),
(2980386, 2979335),
(2981016, 2980386),
(2981190, 2981016),
(2981768, 2981190),
(2981884, 2981768),
(2983591, 2981884),
(2986392, 2983591),
(2986479, 2986392),
(2997326, 2986479),
(3008152, 2997326),
(3008285, 3008152),
(3008937, 3008285),
(3009118, 3008937),
(3009303, 3009118),
(3013654, 3009303),
(3013917, 3013654),
(3168750, 3013917),
(3169311, 3168750),
(3169516, 3169311),
(3169625, 3169516),
(3171748, 3169625),
(3172396, 3171748),
(3174491, 3172396),
(3174726, 3174491),
(3176774, 3174726),
(3185040, 3176774),
(3185125, 3185040),
(3187339, 3185125),
(3187495, 3187339),
(3188504, 3187495),
(3190488, 3188504),
(3192825, 3190488),
(3193354, 3192825),
(3195408, 3193354),
(3196008, 3195408),
(3196186, 3196008),
(3196701, 3196186),
(3196839, 3196701),
(3196932, 3196839),
(3197227, 3196932),
(3198850, 3197227),
(3200012, 3198850),
(3200229, 3200012),
(3200477, 3200229),
(3210224, 3200477),
(3213135, 3210224),
```

```
(3213542, 3213135),
(3214735, 3213542),
(3215137, 3214735),
(3215253, 3215137),
(3215492, 3215253),
(3216337, 3215492),
(3316586, 3216337),
(3316655, 3316586),
(3317426, 3316655),
(3317531, 3317426),
(3317678, 3317531),
(3317933, 3317678),
(3318514, 3317933),
(3318657, 3318514),
(3332446, 3318657),
(3332561, 3332446),
(3333357, 3332561),
(3333719, 3333357),
(3333986, 3333719),
(3334657, 3333986),
(3343210, 3334657),
(3343558, 3343210),
(3343798, 3343558),
(3344893, 3343798),
(3345490, 3344893),
(3345788, 3345490),
(3345887, 3345788),
(3346552, 3345887),
(3358369, 3346552),
(3358615, 3358369),
(3366915, 3358615),
(3367644, 3366915),
(3368066, 3367644),
(3369599, 3368066),
(3369741, 3369599),
(3383461, 3369741),
(3384158, 3383461),
(3388324, 3384158),
(3388516, 3388324),
(3388954, 3388516),
(3390154, 3388954),
(3391391, 3390154),
(3394672, 3391391),
(3394975, 3394672),
(3403661, 3394975),
(3407697, 3403661),
(3408389, 3407697),
(3408560, 3408389),
(3408641, 3408560),
(3409081, 3408641),
(3409226, 3409081),
(3414356, 3409226),
(3414493, 3414356),
(3414579, 3414493),
(3415397, 3414579),
(3415517, 3415397),
(3417448, 3415517),
(3461450, 3417448),
(3464440, 3461450),
(3485242, 3464440),
```

```
(3488684, 3485242),
(3500094, 3488684),
(3500906, 3500094),
(3501264, 3500906),
(3501472, 3501264),
(3501824, 3501472),
(3502929, 3501824),
(3509294, 3502929),
(3509374, 3509294),
(3516942, 3509374),
(3517421, 3516942),
(3521868, 3517421),
(3523201, 3521868),
(3530839, 3523201),
(3531019, 3530839),
(3533408, 3531019),
(3539760, 3533408),
(3540043, 3539760),
(3540132, 3540043),
(3542830, 3540132),
(3546917, 3542830),
(3551684, 3546917),
(3551823, 3551684),
(3551998, 3551823),
(3553062, 3551998),
(3567732, 3553062),
(3567866, 3567732),
(3568191, 3567866),
(3576886, 3568191),
(3577816, 3576886),
(3578005, 3577816),
(3610834, 3578005),
(3613241, 3610834),
(3614127, 3613241),
(3614294, 3614127),
(3621389, 3614294),
(3621599, 3621389),
(3621679, 3621599),
(3622039, 3621679),
(3626724, 3622039),
(3627047, 3626724),
(3627257, 3627047),
(3627452, 3627257),
(3627615, 3627452),
(3666296, 3627615),
(3666964, 3666296),
(3669241, 3666964),
(3741623, 3669241),
(3741987, 3741623),
(3742245, 3741987),
(3757005, 3742245),
(3757904, 3757005),
(3758035, 3757904),
(3758112, 3758035),
(3758616, 3758112),
(3758719, 3758616),
(3760671, 3758719),
(3762419, 3760671),
(3763234, 3762419),
(3789953, 3763234),
```

```
(3790297, 3789953),
(3796253, 3790297),
(3810292, 3796253),
(3811516, 3810292),
(3816649, 3811516),
(3816818, 3816649),
(3816952, 3816818),
(3817135, 3816952),
(3817350, 3817135),
(3821819, 3817350),
(3823814, 3821819),
(3826378, 3823814),
(3829065, 3826378),
(3835781, 3829065),
(3835979, 3835781),
(3836295, 3835979),
(3836403, 3836295),
(3836994, 3836403),
(3837933, 3836994),
(3838060, 3837933),
(3838480, 3838060),
(3838696, 3838480),
(3839156, 3838696),
(3839263, 3839156),
(3839377, 3839263),
(3863505, 3839377),
(3863610, 3863505),
(3867248, 3863610),
(3870554, 3867248),
(3872626, 3870554),
(3882187, 3872626),
(3890857, 3882187),
(3891017, 3890857),
(3893054, 3891017),
(3894063, 3893054),
(3894151, 3894063),
(3894276, 3894151),
(3998639, 3894276),
(4015496, 3998639),
(4024809, 4015496),
(4026873, 4024809),
(4029209, 4026873),
(4041127, 4029209),
(4041782, 4041127),
(4042000, 4041782),
(4042406, 4042000),
(4042721, 4042406),
(4043381, 4042721),
(4043493, 4043381),
(4043662, 4043493),
(4044051, 4043662),
(4044121, 4044051),
(4044654, 4044121),
(4047130, 4044654),
(4047609, 4047130),
(4048966, 4047609),
(4049579, 4048966),
(4049706, 4049579),
(4050134, 4049706),
(4050532, 4050134),
```

```
(4051274, 4050532),
(4051370, 4051274),
(4052574, 4051370),
(4052799, 4052574),
(4053187, 4052799),
(4053696, 4053187),
(4053779, 4053696),
(4112307, 4053779),
(4143845, 4112307),
(4144211, 4143845),
(4144600, 4144211),
(4144721, 4144600),
(4144929, 4144721),
(4145739, 4144929),
(4145951, 4145739),
(4146401, 4145951),
(4146722, 4146401),
(4228001, 4146722),
(4229106, 4228001),
(4229261, 4229106),
(4229420, 4229261),
(4229521, 4229420),
(4229885, 4229521),
(4230495, 4229885),
(4230792, 4230495),
(4231789, 4230792),
(4234851, 4231789),
(4238565, 4234851),
(4242454, 4238565),
(4242560, 4242454),
(4254109, 4242560),
(4254232, 4254109),
(4254375, 4254232),
(4254805, 4254375),
(4255045, 4254805),
(4255147, 4255045),
(4259998, 4255147),
(4274383, 4259998),
(4290170, 4274383),
(4290216, 4290170),
(4290336, 4290216),
(4294149, 4290336),
(4294427, 4294149),
(4295366, 4294427),
(4301239, 4295366),
(4301412, 4301239),
(4301580, 4301412),
(4301798, 4301580),
(4310891, 4301798),
(4311025, 4310891),
(4330589, 4311025),
(4334084, 4330589),
(4342463, 4334084),
(4342836, 4342463),
(4343474, 4342836),
(4344266, 4343474),
(4345511, 4344266),
(4346110, 4345511),
(4347028, 4346110),
(4350585, 4347028),
```

```
            (4350736, 4350585),
            (4351316, 4350736),
            (4362718, 4351316),
            (4362899, 4362718),
            (4363136, 4362899),
            (4366735, 4363136),
            (4366981, 4366735),
            (4367351, 4366981),
            (4368879, 4367351),
            (4370975, 4368879),
            (4396076, 4370975),
            (4396646, 4396076),
            (4397112, 4396646),
            (4397270, 4397112),
            (4397728, 4397270),
            (4398114, 4397728),
            (4399602, 4398114),
            (4400155, 4399602),
            (4411232, 4400155),
            (4411362, 4411232),
            (4411936, 4411362),
            (4431288, 4411936),
            (4431584, 4431288),
            (4431798, 4431584),
            (4432054, 4431798),
            (4432880, 4432054),
            (4433016, 4432880),
            (4433653, 4433016),
            (4433767, 4433653),
            (4433905, 4433767),
            (4435326, 4433905),
            (4467018, 4435326),
            (4468250, 4467018),
            (4468353, 4468250),
            (4468550, 4468353),
            (4468737, 4468550),
            (4469177, 4468737),
            (4469270, 4469177),
            (4469729, 4469270),
            (4471401, 4469729),
            (4472534, 4471401),
            (4472617, 4472534),
            (4477154, 4472617),
            (4477250, 4477154),
            (4477713, 4477250),
            (4586320, 4477713),
            (4586562, 4586320),
            (4586672, 4586562),
            (4589587, 4586672),
            (4589946, 4589587),
            (4590234, 4589946),
            (4590877, 4590234),
            (4591658, 4590877),
            (4596503, 4591658),
            (4598971, 4596503),
            (4629565, 4598971),
            (4641922, 4629565),
            (4643149, 4641922),
            (4643339, 4643149),
            (4654517, 4643339),
```

```
            (4655109, 4654517),
            (4655710, 4655109),
            (4655819, 4655710),
            (4656018, 4655819),
            (4656165, 4656018),
            (4657022, 4656165),
            (4661923, 4657022),
            (4672962, 4661923),
            (4673301, 4672962),
            (4673438, 4673301),
            (4673978, 4673438),
            (4674301, 4673978),
            (4674470, 4674301),
            (4674593, 4674470),
            (4674637, 4674593),
            (4675080, 4674637),
            (4675193, 4675080),
            (4682768, 4675193),
            (4693603, 4682768),
            (4693717, 4693603),
            (4694503, 4693717),
            (4694707, 4694503),
            (4694821, 4694707),
            (4694915, 4694821),
            (4695645, 4694915),
            (4695755, 4695645),
            (4695973, 4695755),
            (4696317, 4695973),
            (4697226, 4696317),
            (4697865, 4697226),
            (4699675, 4697865),
            (4699829, 4699675),
            (4700093, 4699829),
            (4701460, 4700093),
            (4702187, 4701460),
            (4702357, 4702187),
            (4726049, 4702357),
            (4726230, 4726049),
            (4726392, 4726230),
            (4744857, 4726392),
            (4745019, 4744857),
            (4748274, 4745019),
            (4750028, 4748274),
            (4750118, 4750028),
            (4750232, 4750118),
            (4750374, 4750232),
            (4751401, 4750374),
            (4751538, 4751401),
            (4753215, 4751538),
            (4753307, 4753215),
            (4753640, 4753307),
            (4756051, 4753640),
            (4763052, 4756051),
            (4771142, 4763052),
            (4771407, 4771142),
            (4774445, 4771407),
            (4775061, 4774445),
            (4775326, 4775061),
            (4775521, 4775326),
            (4776594, 4775521),
```

```
(4778965, 4776594),
(4800250, 4778965),
(4837561, 4800250),
(4840438, 4837561),
(4840763, 4840438),
(4840960, 4840763),
(4841827, 4840960),
(4841935, 4841827),
(4842068, 4841935),
(4842272, 4842068),
(4845410, 4842272),
(4845639, 4845410),
(4845865, 4845639),
(4846492, 4845865),
(4846623, 4846492),
(4846790, 4846623),
(4848867, 4846790),
(4851123, 4848867),
(4851252, 4851123),
(4852480, 4851252),
(4852795, 4852480),
(4854359, 4852795),
(4854857, 4854359),
(4856571, 4854857),
(4946571, 4856571),
(4950210, 4946571),
(4950306, 4950210),
(4952146, 4950306),
(4970445, 4952146),
(4973404, 4970445),
(4973764, 4973404),
(4979975, 4973764),
(4983313, 4979975),
(4996567, 4983313),
(4996672, 4996567),
(5008531, 4996672),
(5008834, 5008531),
(5009281, 5008834),
(5010438, 5009281),
(5011199, 5010438),
...)
```

In [ ]:
```
temp=np.full((1,547), 1)
print(temp)
```

```
[[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  1 1 1 1 1 1 1]]
```

In [ ]:

```
#now we will create a numpy array that will contain 1 from values 0 to 547, 2 from 548
movie_np=[]
movie_id=1
for i, j in zip(df_nan['index'][1:], df_nan['index'][:-1]):
    temp=np.full((1, i-j-1), movie_id)
    movie_np=np.append(movie_np, temp)
    movie_id+=1

print(movie_np)

#account for last record and corresponding length
#numpy approach
last_record=np.full((1, len(netflix_dataset)-df_nan.iloc[-1,0]-1) , movie_id)#movie id
movie_np=np.append(movie_np, last_record)
```

```
[1.000e+00 1.000e+00 1.000e+00 ... 4.498e+03 4.498e+03 4.498e+03]
```

In [ ]:

```
import pandas as pd

# Create a dictionary with data
data = {
    'index': [10, 30, 50],
    'B': [20, 40, 60]
}

# Create the DataFrame
df_nan1 = pd.DataFrame(data)

# Print the DataFrame
print(df_nan1)
```

```
   index   B
0     10  20
1     30  40
2     50  60
```

In [ ]:

```
df_nan1.iloc[-1, 0]
```

Out[ ]:

```
50
```

In [ ]:

```
netflix_dataset=netflix_dataset[pd.notnull(netflix_dataset['Rating'])]

netflix_dataset['Movie_Id']=movie_np.astype(int)
```

```
netflix_dataset['Cust_Id']=netflix_dataset['Cust_Id'].astype(int)
print("Now the dataset will look like: ")
netflix_dataset
```

```
<ipython-input-39-13537b305b64>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  netflix_dataset['Movie_Id']=movie_np.astype(int)
Now the dataset will look like:
<ipython-input-39-13537b305b64>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  netflix_dataset['Cust_Id']=netflix_dataset['Cust_Id'].astype(int)
```

Out[ ]:

	Cust_Id	Rating	Movie_Id
1	1488844	3.0	1
2	822109	5.0	1
3	885013	4.0	1
4	30878	4.0	1
5	823519	3.0	1
...	...	...	...
24058258	2591364	2.0	4499
24058259	1791000	2.0	4499
24058260	512536	5.0	4499
24058261	988963	3.0	4499
24058262	1704416	3.0	4499

24053764 rows × 3 columns

In [ ]:
```
netflix_dataset.tail()
```

Out[ ]:

	Cust_Id	Rating	Movie_Id
24058258	2591364	2.0	4499
24058259	1791000	2.0	4499
24058260	512536	5.0	4499
24058261	988963	3.0	4499
24058262	1704416	3.0	4499

In [ ]:
```
#now we will remove all the users that have rated less movies and
#also all those movies that has been rated less in numbers
```

In [ ]:
```
netflix_dataset
```

Out[ ]:

	Cust_Id	Rating	Movie_Id
1	1488844	3.0	1
2	822109	5.0	1
3	885013	4.0	1
4	30878	4.0	1
5	823519	3.0	1
...	...	...	...
24058258	2591364	2.0	4499
24058259	1791000	2.0	4499
24058260	512536	5.0	4499
24058261	988963	3.0	4499
24058262	1704416	3.0	4499

24053764 rows × 3 columns

In [ ]:
```
dataset_movie_summary=netflix_dataset.groupby('Movie_Id')['Rating'].agg(["count"])
```

In [ ]:
```
dataset_movie_summary
```

Out[ ]:

Movie_Id	count
1	547
2	145
3	2012
4	142
5	1140
...	...
4495	614
4496	9519
4497	714
4498	269
4499	428

4499 rows × 1 columns

In [ ]:
```
#We have to create Benchmark for Ratings
dataset_movie_summary["count"].quantile(0.7)
```

Out[ ]:
```
1798.6
```

In [ ]:
```
#Now we will create a benchmark
movie_benchmark=round(dataset_movie_summary['count'].quantile(0.7),0)
movie_benchmark
```

Out[ ]:
```
1799.0
```

In [ ]:
```
dataset_movie_summary['count']
```

Out[ ]:
```
Movie_Id
1        547
2        145
3       2012
4        142
5       1140
        ...
4495     614
4496    9519
4497     714
4498     269
4499     428
Name: count, Length: 4499, dtype: int64
```

In [ ]:
```
#We have to drop movies which are less than Benchmark
drop_movie_list=dataset_movie_summary[dataset_movie_summary['count']<movie_benchmark].
drop_movie_list
```

Out[ ]:
```
Int64Index([   1,    2,    4,    5,    6,    7,    9,   10,   11,   12,
            ...
            4484, 4486, 4487, 4489, 4491, 4494, 4495, 4497, 4498, 4499],
           dtype='int64', name='Movie_Id', length=3149)
```

In [ ]:
```
len(drop_movie_list)
```

Out[ ]:
```
3149
```

In [ ]:
```
#Now we will remove all the users that are in-active
dataset_cust_summary=netflix_dataset.groupby('Cust_Id')['Rating'].agg(["count"])
dataset_cust_summary
```

Out[ ]:

	count
Cust_Id	
6	153
7	195
8	21
10	49
25	4
...	...
2649404	12
2649409	10
2649421	3
2649426	74
2649429	62

470758 rows × 1 columns

In [ ]:
```
#We have to create same Benchmark for customer also
cust_benchmark=round(dataset_cust_summary['count'].quantile(0.7),0)
cust_benchmark
```

Out[ ]:
```
52.0
```

In [ ]:
```
#We have to drop Customer which are less than Benchmark
drop_cust_list=dataset_cust_summary[dataset_cust_summary['count']<cust_benchmark].inde
drop_cust_list
```

Out[ ]:
```
Int64Index([      8,      10,      25,      33,      42,      59,      83,
                87,      94,     116,
            ...
            2649343, 2649351, 2649375, 2649376, 2649379, 2649384, 2649401,
            2649404, 2649409, 2649421],
           dtype='int64', name='Cust_Id', length=327300)
```

In [ ]:
```
len(drop_cust_list)
```

Out[ ]:
```
327300
```

In [ ]:
```
#We will remove all the customers and movies that are below the benchmark
print('The original dataframe has: ', netflix_dataset.shape, 'shape')
```

```
The original dataframe has:  (24053764, 3) shape
```

In [ ]:
```
netflix_dataset['Movie_Id'].isin(drop_movie_list)
```

Out[ ]:
```
1           True
2           True
3           True
4           True
5           True
            ...
24058258    True
24058259    True
24058260    True
24058261    True
24058262    True
Name: Movie_Id, Length: 24053764, dtype: bool
```

In [ ]:
```
netflix_dataset=netflix_dataset[~netflix_dataset['Movie_Id'].isin(drop_movie_list)]
netflix_dataset=netflix_dataset[~netflix_dataset['Cust_Id'].isin(drop_cust_list)]
print('After the triming, the shape is: {}'.format(netflix_dataset.shape))
```

```
After the triming, the shape is: (17337458, 3)
```

In [ ]:
```
netflix_dataset.head()
```

Out[ ]:

	Cust_Id	Rating	Movie_Id
696	712664	5.0	3
697	1331154	4.0	3
698	2632461	3.0	3
699	44937	5.0	3
700	656399	4.0	3

In [ ]:
```
#Now we are adding Movies data set to work further
import pandas as pd
```

In [ ]:
```
df_title = pd.read_csv("/content/drive/MyDrive/Netflix csv/movies.csv",  encoding='ISO

df_title.set_index('Movie_Id', inplace=True)
```

In [ ]:
```
df_title.head(10)
```

Out[ ]:

Movie_Id	Year	Name
movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	Father of the Bride Part II (1995)	Comedy
6	Heat (1995)	Action|Crime|Thriller
7	Sabrina (1995)	Comedy|Romance
8	Tom and Huck (1995)	Adventure|Children
9	Sudden Death (1995)	Action

In [ ]:
```
df_title.iloc[:4499,:]
```

Out[ ]:

Movie_Id	Year	Name
movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
...	...	...
4588	Eddie and the Cruisers II: Eddie Lives! (1989)	Drama|Musical
4589	Eddie and the Cruisers (1983)	Drama|Musical|Mystery
4590	Enemies: A Love Story (1989)	Drama
4591	Erik the Viking (1989)	Adventure|Comedy|Fantasy
4592	Experts, The (1989)	Comedy

4499 rows × 2 columns

In [ ]:
```
!pip install scikit-surprise
```

```
Collecting scikit-surprise
  Downloading scikit-surprise-1.1.3.tar.gz (771 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 772.0/772.0 kB 14.6 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: joblib>=1.0.0 in /usr/local/lib/python3.10/dist-packag
es (from scikit-surprise) (1.3.2)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packag
es (from scikit-surprise) (1.25.2)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-package
s (from scikit-surprise) (1.11.4)
Building wheels for collected packages: scikit-surprise
  Building wheel for scikit-surprise (setup.py) ... done
  Created wheel for scikit-surprise: filename=scikit_surprise-1.1.3-cp310-cp310-linux
_x86_64.whl size=3163017 sha256=c31baa39a32f523bff3d984a057b736eedc258ec649198dd3ce32
eabaf0a4ddb
  Stored in directory: /root/.cache/pip/wheels/a5/ca/a8/4e28def53797fdc4363ca4af740db
15a9c2f1595ebc51fb445
Successfully built scikit-surprise
Installing collected packages: scikit-surprise
Successfully installed scikit-surprise-1.1.3
```

In [ ]:
```
#model building

import math
import seaborn as sns
from surprise import Reader, Dataset, SVD
from surprise.model_selection import cross_validate
```

In [ ]:
```
#!pip install scikit-surprise
```

In [ ]:
```
#Help us to read the dataset for svd algo
reader=Reader()
```

In [ ]:
```
#We only work with top 100K rows for quick runtime
data=Dataset.load_from_df(netflix_dataset[['Cust_Id','Movie_Id','Rating']][:100000], r
```

In [ ]:
```
data
```

Out[ ]:
```
<surprise.dataset.DatasetAutoFolds at 0x798901a13fd0>
```

In [ ]:
```
# cross_validate(svd, data, measures=['RMSE','MAE'], cv=3)
#for 1st fold- 1,2,3,4,5
model=SVD()
```

In [ ]:
```
cross_validate(model, data, measures=['RMSE','MAE'], cv=4)
```

Out[ ]:
```
{'test_rmse': array([0.99386251, 0.99324249, 0.99504583, 1.00788747]),
 'test_mae': array([0.79686912, 0.79624673, 0.7950658 , 0.79303899]),
 'fit_time': (2.293118476867676,
  1.4674701690673828,
  1.4886362552642822,
  1.6716058254241943),
 'test_time': (0.2630624771118164,
  0.26549339294433594,
  0.1284322738647461,
  0.26459240913391113)}
```

In [ ]:
```
netflix_dataset.head()
```

Out[ ]:

	Cust_Id	Rating	Movie_Id
696	712664	5.0	3
697	1331154	4.0	3
698	2632461	3.0	3
699	44937	5.0	3
700	656399	4.0	3

In [ ]:
```
#so first we take user 1331154 and we try to recommend some movies based on the past a
#He rated so many movies with 5 *
dataset_1331154=netflix_dataset[(netflix_dataset['Cust_Id'] ==1331154)& (netflix_datas
dataset_1331154
```

Out[ ]:

	Cust_Id	Rating	Movie_Id
458308	1331154	5.0	143
1184450	1331154	5.0	270
1991774	1331154	5.0	361
2369367	1331154	5.0	457
2600328	1331154	5.0	482
3417458	1331154	5.0	658
4029215	1331154	5.0	763
5646194	1331154	5.0	1144
7075510	1331154	5.0	1425
7423467	1331154	5.0	1476
7468997	1331154	5.0	1495
8184220	1331154	5.0	1642
8257180	1331154	5.0	1650
8789450	1331154	5.0	1754
10165725	1331154	5.0	1974
10919877	1331154	5.0	2128
11519840	1331154	5.0	2192
12202244	1331154	5.0	2372
12746620	1331154	5.0	2452
14167769	1331154	5.0	2743
14525287	1331154	5.0	2795
15104095	1331154	5.0	2913
17319044	1331154	5.0	3333
17679853	1331154	5.0	3379
19254925	1331154	5.0	3650
19555314	1331154	5.0	3730
19992284	1331154	5.0	3825
20774457	1331154	5.0	3925
21174535	1331154	5.0	3962
21317452	1331154	5.0	3966
22718437	1331154	5.0	4306

In [ ]:
```
df_title
```

Out[ ]:

Movie_Id	Year	Name
movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
...	...	...
131254	Kein Bund fÃ¼r's Leben (2007)	Comedy
131256	Feuer, Eis & Dosenbier (2002)	Comedy
131258	The Pirates (2014)	Adventure
131260	Rentun Ruusu (2001)	(no genres listed)
131262	Innocence (2014)	Adventure|Fantasy|Horror

27279 rows × 2 columns

In [ ]:
```
#now we will build the recommendation algorithm
#first we will make a shallow copy of the movie_titles.csv file so that we can change
#the values in the copied dataset, not in the actual dataset

user_1331154=df_title.copy()
user_1331154
```

Out[ ]:

Movie_Id	Year	Name
movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
...	...	...
131254	Kein Bund fÃ¼r's Leben (2007)	Comedy
131256	Feuer, Eis & Dosenbier (2002)	Comedy
131258	The Pirates (2014)	Adventure
131260	Rentun Ruusu (2001)	(no genres listed)
131262	Innocence (2014)	Adventure|Fantasy|Horror

27279 rows × 2 columns

In [ ]:
```
user_1331154=user_1331154.reset_index()
user_1331154
```

Out[ ]:

	Movie_Id	Year	Name
0	movieId	title	genres
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	2	Jumanji (1995)	Adventure|Children|Fantasy
3	3	Grumpier Old Men (1995)	Comedy|Romance
4	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
...	...	...	...
27274	131254	Kein Bund fÃ¼r's Leben (2007)	Comedy
27275	131256	Feuer, Eis & Dosenbier (2002)	Comedy
27276	131258	The Pirates (2014)	Adventure
27277	131260	Rentun Ruusu (2001)	(no genres listed)
27278	131262	Innocence (2014)	Adventure|Fantasy|Horror

27279 rows × 3 columns

In [ ]:
```
user_1331154=user_1331154[~user_1331154['Movie_Id'].isin(drop_movie_list)]
user_1331154
```

Out[ ]:

	Movie_Id	Year	Name
0	movieId	title	genres
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	2	Jumanji (1995)	Adventure|Children|Fantasy
3	3	Grumpier Old Men (1995)	Comedy|Romance
4	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
...	...	...	...
27274	131254	Kein Bund fÃ¼r's Leben (2007)	Comedy
27275	131256	Feuer, Eis & Dosenbier (2002)	Comedy
27276	131258	The Pirates (2014)	Adventure
27277	131260	Rentun Ruusu (2001)	(no genres listed)
27278	131262	Innocence (2014)	Adventure|Fantasy|Horror

27279 rows × 3 columns

In [ ]:
```
user_1331154['Estimate_Score']=user_1331154['Movie_Id'].apply(lambda x: model.predict(
```

In [ ]:
```
user_1331154
```

Out[ ]:

	Movie_Id	Year	Name	Estimate_Score
0	movieId	title	genres	3.612475
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	3.612475
2	2	Jumanji (1995)	Adventure|Children|Fantasy	3.612475
3	3	Grumpier Old Men (1995)	Comedy|Romance	3.612475
4	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	3.612475
...	...	...	...	...
27274	131254	Kein Bund fÃ¼r's Leben (2007)	Comedy	3.612475
27275	131256	Feuer, Eis & Dosenbier (2002)	Comedy	3.612475
27276	131258	The Pirates (2014)	Adventure	3.612475
27277	131260	Rentun Ruusu (2001)	(no genres listed)	3.612475
27278	131262	Innocence (2014)	Adventure|Fantasy|Horror	3.612475

27279 rows × 4 columns

In [ ]:
```
user_1331154=user_1331154.sort_values('Estimate_Score',ascending=False)
```

In [ ]:
```
print(user_1331154)
```

```
      Movie_Id                                Year  \
0      movieId                               title
2            2                      Jumanji (1995)
4            4            Waiting to Exhale (1995)
5            5  Father of the Bride Part II (1995)
6            6                         Heat (1995)
...        ...                                 ...
27275   131256       Feuer, Eis & Dosenbier (2002)
27276   131258                  The Pirates (2014)
27277   131260                Rentun Ruusu (2001)
27267   131237          What Men Talk About (2010)
27278   131262                   Innocence (2014)

                            Name  Estimate_Score
0                         genres        3.612475
2      Adventure|Children|Fantasy        3.612475
4            Comedy|Drama|Romance        3.612475
5                         Comedy        3.612475
6           Action|Crime|Thriller        3.612475
...                           ...             ...
27275                     Comedy        3.612475
27276                  Adventure        3.612475
27277          (no genres listed)        3.612475
27267                     Comedy        3.612475
27278    Adventure|Fantasy|Horror        3.612475

[27279 rows x 4 columns]
```

In [ ]: