STEAM VIDEO GAMES RECOMMENDATION SYSTEM

we need to recommend the similar games to the user based on their behaviour

#About dataset This dataset is a list of user behaviors, with columns: user-id, game-title, behavior-name, value. The behaviors included are 'purchase' and 'play'. The value indicates the degree to which the behavior was performed - in the case of 'purchase' the value is always 1, and in the case of 'play' the value represents the number of hours the user has played the game.

steam-200k - (199999, 5)

Columns in dataset

user-id

game-title

behavior-name

value

0

In [1]:

```
```

```
Mounted at /content/drive
```

In [2]:

```
# import all necessary libraries

# import cosine similarity
```

In [3]:

```
# Read csv file using pandas
```

Out[3]:

	151603712	The Elder Scrolls V Skyrim	purchase	1.0	0
0	151603712	The Elder Scrolls V Skyrim	play	273.0	0
1	151603712	Fallout 4	purchase	1.0	0
2	151603712	Fallout 4	play	87.0	0
3	151603712	Spore	purchase	1.0	0
4	151603712	Spore	play	14.9	0

EDA

In [4]:
```
# remane the column name as games user_id, hoursplay and status
```

Out[4]:

	User_ID	games	Status	Hoursplay	0
0	151603712	The Elder Scrolls V Skyrim	play	273.0	0
1	151603712	Fallout 4	purchase	1.0	0
2	151603712	Fallout 4	play	87.0	0
3	151603712	Spore	purchase	1.0	0
4	151603712	Spore	play	14.9	0
...	...	...	...	...	...
199994	128470551	Titan Souls	play	1.5	0
199995	128470551	Grand Theft Auto Vice City	purchase	1.0	0
199996	128470551	Grand Theft Auto Vice City	play	1.5	0
199997	128470551	RUSH	purchase	1.0	0
199998	128470551	RUSH	play	1.4	0

199999 rows × 5 columns

In [5]:
```
#drop 0 column
```

Out[5]:

	User_ID	games	Status	Hoursplay
0	151603712	The Elder Scrolls V Skyrim	play	273.0
1	151603712	Fallout 4	purchase	1.0
2	151603712	Fallout 4	play	87.0
3	151603712	Spore	purchase	1.0
4	151603712	Spore	play	14.9

In [ ]:
```
#drop duplicate user_id and games keep the last one
```

In [ ]:
```
#check the shape of the dataset
```

Out[53]:
```
(199999, 4)
```

In [ ]:
```
#check columns of the dataset
```

Out[54]:
```
Index(['User_ID', 'games', 'Status', 'Hoursplay'], dtype='object')
```

In [ ]:
```
# Check which columns are having categorical, numerical or boolean values of
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 199999 entries, 0 to 199998
Data columns (total 4 columns):
 #   Column     Non-Null Count   Dtype  
---  ------     --------------   -----  
 0   User_ID    199999 non-null  int64  
 1   games      199999 non-null  object 
 2   Status     199999 non-null  object 
 3   Hoursplay  199999 non-null  float64
dtypes: float64(1), int64(1), object(2)
memory usage: 6.1+ MB
```

In [ ]:
```
# Check for missing values in all the columnns of the train_dataset
```

Out[56]:
```
User_ID      0
games        0
Status       0
Hoursplay    0
dtype: int64
```

In [ ]:
```
# get how many unique values are in games column of dataset
```

Out[57]:
```
Dota 2                                  9682
Team Fortress 2                         4646
Counter-Strike Global Offensive         2789
Unturned                                2632
Left 4 Dead 2                           1752
                                        ... 
Trainz Classic Cabon City                  1
Flashout 2                                 1
Dark Forester                              1
Deep Dungeons of Doom                      1
Musclecar Online                           1
Name: games, Length: 5155, dtype: int64
```

In [ ]:
```
# get the total count of play and total count of purchase
```

Out[58]:
```
Status
play         70489
purchase    129510
Name: Status, dtype: int64
```

In [ ]:
```
# For more information on the dataset like the total count in all the columns
# min, max values and more information of the respective columns
```

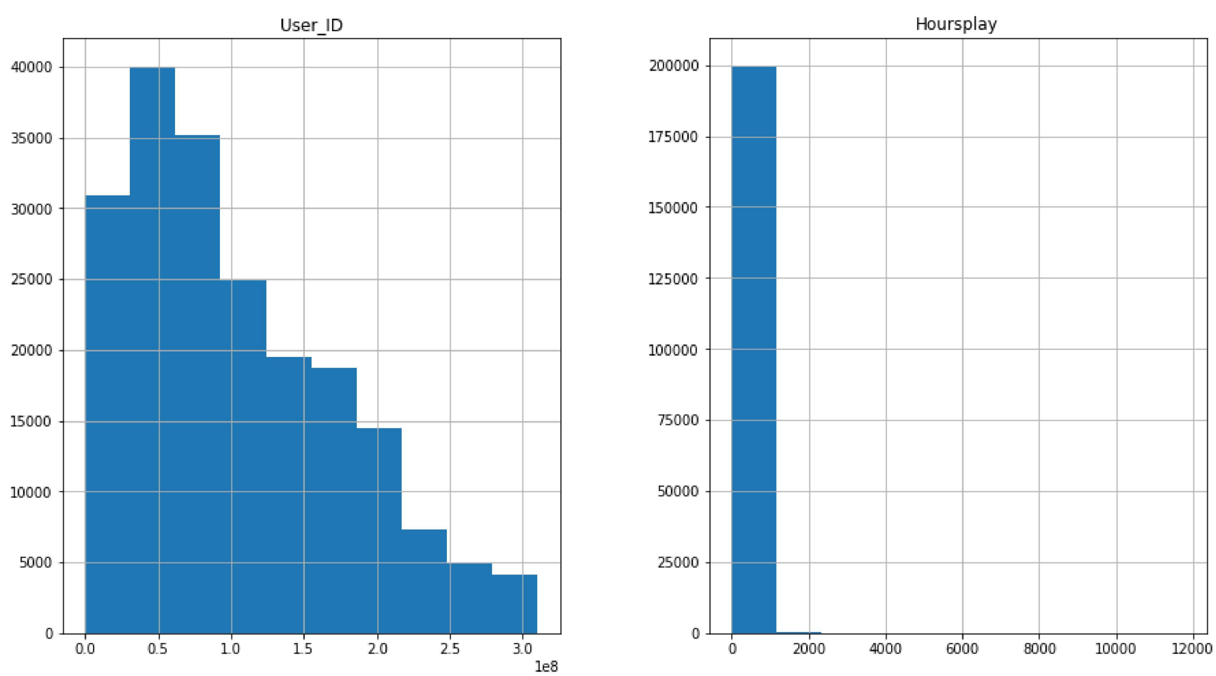
Out[59]:

	User_ID	Hoursplay
count	1.999990e+05	199999.000000
mean	1.036556e+08	17.874468
std	7.208084e+07	138.057292
min	5.250000e+03	0.100000
25%	4.738420e+07	1.000000
50%	8.691201e+07	1.000000
75%	1.542309e+08	1.300000
max	3.099031e+08	11754.000000

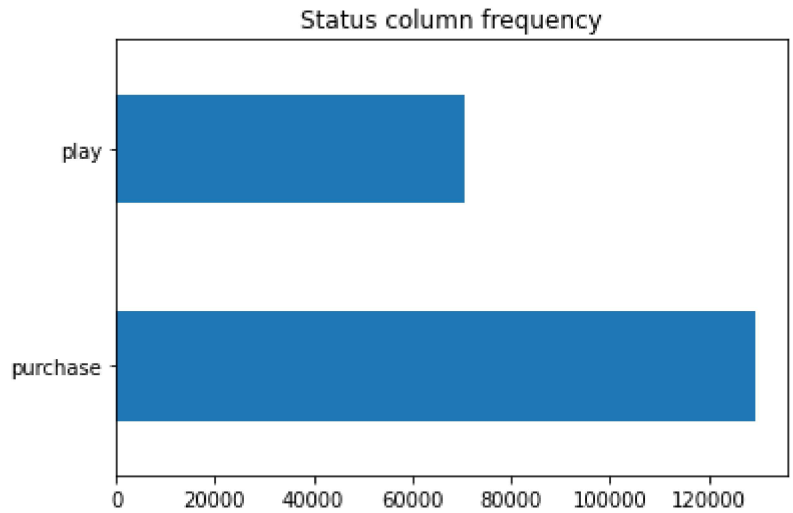
visualizing data

In [ ]:
```
# Histogram using pandas
```

Out[60]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fc1cd81e8d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7fc1cd7dfb10>]],
      dtype=object)
```

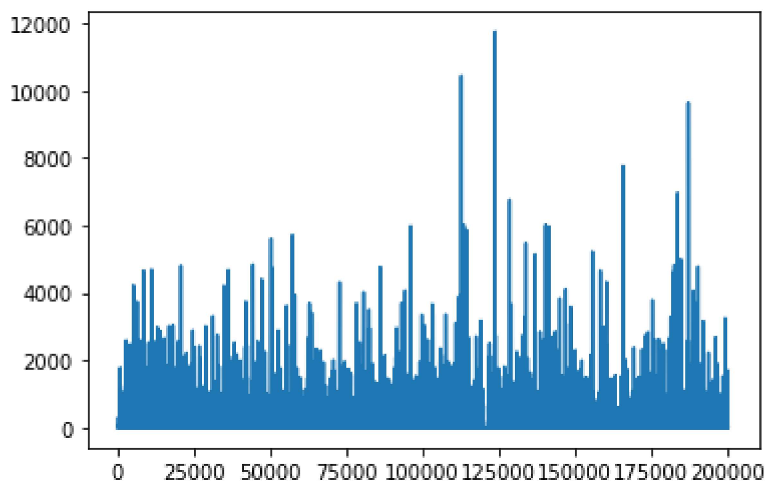


In [ ]:
```
# plot a horizontal bar plot of column status
```



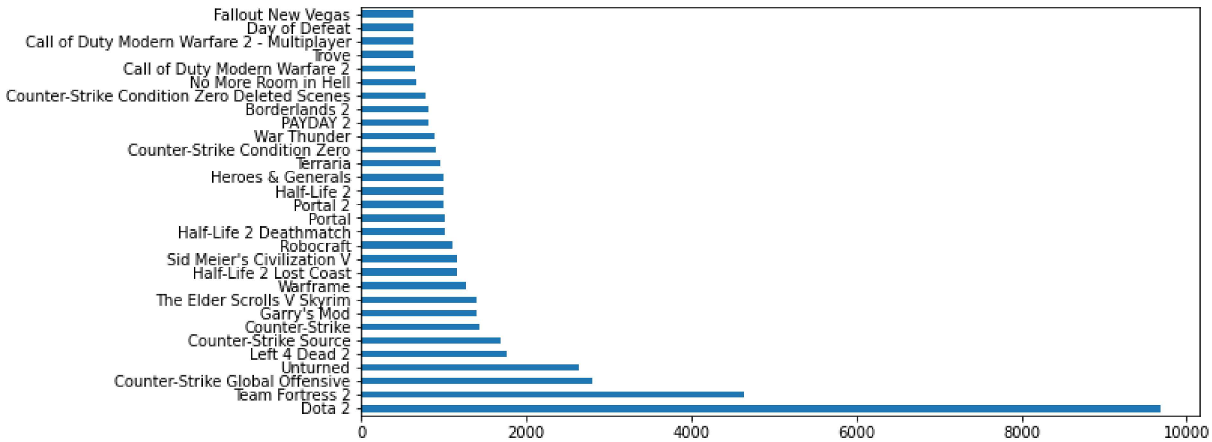
In [ ]:
```
#plot a count plot of hoursplay column
```

Out[63]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc1cd083bd0>



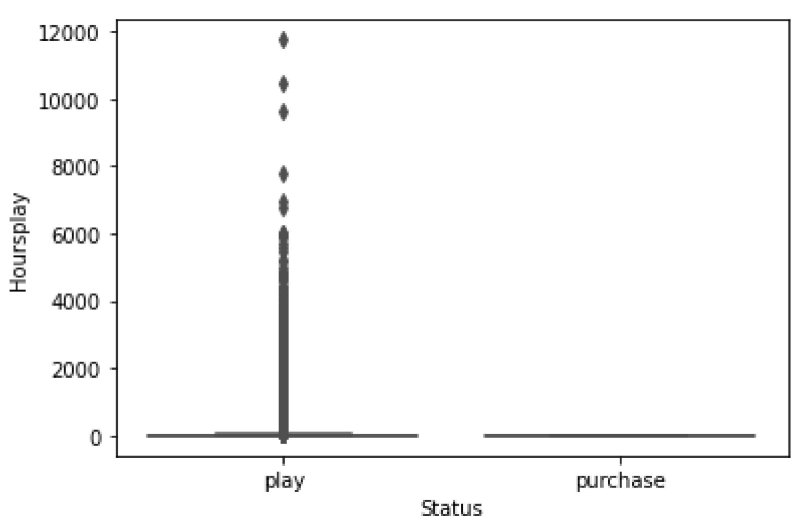
In [ ]:
```
# plot a horizontal bar plot of games column for top 30 games
```

Out[64]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc1c87f3910>



In [ ]:
```
# plot a boxplot of status as x-axis and hoursplay as y-axis
```

Out[65]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc1c8731710>



converting hours to rating

In [7]:
```
# comvert the hoursplay into rating
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: SettingWithC
opyWarning:
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/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  import sys
```

Out[7]:

	User_ID	games	Status	Hoursplay	avg_Hoursplay	rating
0	151603712	The Elder Scrolls V Skyrim	play	273.0	115.351792	5
1	59945701	The Elder Scrolls V Skyrim	play	58.0	115.351792	3
2	92107940	The Elder Scrolls V Skyrim	play	110.0	115.351792	5
3	250006052	The Elder Scrolls V Skyrim	play	465.0	115.351792	5
4	11373749	The Elder Scrolls V Skyrim	play	220.0	115.351792	5
...	...	...	...	...	...	...
36415	51822361	Warhammer 40,000 Dawn of War Soulstorm	play	23.0	14.109091	5
36416	38317154	Warhammer 40,000 Dawn of War Soulstorm	play	5.5	14.109091	2
36417	36404933	Warhammer 40,000 Dawn of War Soulstorm	play	5.8	14.109091	3
36418	87201181	Warhammer 40,000 Dawn of War Soulstorm	play	24.0	14.109091	5
36419	34901647	Warhammer 40,000 Dawn of War Soulstorm	play	15.4	14.109091	5

36420 rows × 6 columns

In [8]:
```
# keep only important columns( user_id, games,rating ) drop everthing else
```

Out[8]:

	User_ID	games	rating
0	151603712	The Elder Scrolls V Skyrim	5
1	59945701	The Elder Scrolls V Skyrim	3
2	92107940	The Elder Scrolls V Skyrim	5
3	250006052	The Elder Scrolls V Skyrim	5
4	11373749	The Elder Scrolls V Skyrim	5
...	...	...	...
36415	51822361	Warhammer 40,000 Dawn of War Soulstorm	5
36416	38317154	Warhammer 40,000 Dawn of War Soulstorm	2
36417	36404933	Warhammer 40,000 Dawn of War Soulstorm	3
36418	87201181	Warhammer 40,000 Dawn of War Soulstorm	5
36419	34901647	Warhammer 40,000 Dawn of War Soulstorm	5

36420 rows × 3 columns

MEMORY BASED COLLABORATIVE FILTERING

Memory-based algorithms approach the collaborative filtering problem by using the entire database.It tries to find users that are similar to the active user (i.e. the users we want to make predictions for), and uses their preferences to predict ratings for the active user.

In [9]:
```
#import pairwise_distances, cosine, corelation
```

In [10]:
```
# create pivot table containing user_id as index, games as columns, ratings a
```

In [11]:
```
#check shape of pivot table

#check first five rows of pivot table
```

(8309, 427)

Out[11]:

games User_ID	7 Days to Die	APB Reloaded	ARK Survival Evolved	Ace of Spades	AdVenture Capitalist	Aftermath	Age of Chivalry	Age of Empires II HD Edition	Age of Empires III Complete Collection
100053304	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
100057229	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
100070732	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
100096071	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
100168166	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 427 columns

In [12]:
```
## Note: As we are subtracting the mean from each rating to standardize
##all users with only one rating or who had rated everything the same will be
# Normalize the values in pivot table



# Drop all columns containing only zeros representing users who did not rate
```

In [13]:
```
# import scipy, operator
```

In [14]:
```
# convert the data into sparse matrix format to be read by the following func
```

In [15]:
```
# create matrices to show the computed cosine similarity values between each
```

In [16]:
```
# Inserting the similarity matricies into dataframe objects

#item similarity dataframe

#user similarity dataframe
```

In [17]:
```
# write a function which will return the top 10 games with the highest cosine
```

In [18]:
```

```

```
Similar games to Aftermath include:

No. 1: Alice Madness Returns
No. 2: Shadow Warrior
No. 3: Brtal Legend
No. 4: Resident Evil 5 / Biohazard 5
No. 5: Infestation Survivor Stories
No. 6: Call of Juarez Gunslinger
No. 7: The Walking Dead Season Two
No. 8: Counter-Strike Nexon Zombies
No. 9: Star Conflict
No. 10: L.A. Noire
```

In [ ]:
```
# check the column of pivot table
```

Out[32]:
```
Index(['100057229', '100096071', '100311267', '100322840', '100351493',
       '100359523', '100431715', '100444456', '100519466', '100630947',
       ...
       '994489', '9946133', '99484728', '99640715', '99704390', '99711581',
       '99713453', '99723205', '99766416', '99802512'],
      dtype='object', name='User_ID', length=3056)
```

In [19]:
```
# write a function which will return the top 5 users with the highest similar
```

In [ ]:

```
Most Similar Users:

User #40289887, Similarity value: 0.73
User #185494712, Similarity value: 0.71
User #16710264, Similarity value: 0.71
User #20566124, Similarity value: 0.67
User #49769103, Similarity value: 0.67
User #15702351, Similarity value: 0.65
User #161139120, Similarity value: 0.59
User #202057920, Similarity value: 0.58
User #57271785, Similarity value: 0.58
User #33993318, Similarity value: 0.58
```

In [20]:

```
# write a function which constructs a list of lists containing the highest ra
# and returns the name of the game along with the frequency it appears in the
```

In [21]:

Out[21]:

```
[('Robocraft', 6),
 ('BLOCKADE 3D', 2),
 ("Garry's Mod", 2),
 ('ARK Survival Evolved', 1),
 ('Dino D-Day', 1)]
```

COLLABORATIVE FILTERING USING KNN

Collaborative Filtering Using k-Nearest Neighbors (kNN). kNN is a machine learning algorithm to find clusters of similar users based on common ratings, and make predictions using the average rating of top-k nearest neighbors.

[https://datascienceplus.com/building-a-book-recommender-system-the-basics-knn-and-matrix-factorization/ (https://datascienceplus.com/building-a-book-recommender-system-the-basics-knn-and-matrix-factorization/)](https://datascienceplus.com/building-a-book-recommender-system-the-basics-knn-and-matrix-factorization/)

In [30]:

```
# import NearestNeighbors

#make an object for the NearestNeighbors Class.

# fit the dataset
```

Out[30]:

```
NearestNeighbors(algorithm='brute', leaf_size=30, metric='cosine',
                 metric_params=None, n_jobs=-1, n_neighbors=20, p=2,
                 radius=1.0)
```

Test model and make some recommendations:

In [39]:
```
# choose random game

# print the name of random game


# use kNN algorithm to measures distance to determine the closeness of instan


# pick most popular games among the neighbors and print their names
```

```
Choosen game is:  Counter-Strike Condition Zero Deleted Scenes
Recommendations for Counter-Strike Condition Zero Deleted Scenes:

1: Fistful of Frags, with distance of 0.9569464262825205:
2: Nosgoth, with distance of 0.9581430523545201:
3: Counter-Strike Source, with distance of 0.958229847541747:
4: Tom Clancy's Splinter Cell Conviction, with distance of 0.959762675184064
9:
5: Orcs Must Die!, with distance of 0.9647249846492842:
```

#Conclusion

We can use different different methods based on our problem statement and dataset. Here we used collaborative filtering technique to recommend games.We can use this method to recommend alot of other things as well such as music, movies, books, news etc.

#Congratulation for completing the assignment. You have learned a lot while doing this assignment.