Due to server space limitations, files were uploaded and separated based on customer information such as account creation, transactions, streaming activity, and website activity (adding to wishlist, deleting, rating content)

Data ranges from 2016 - 2023 so the following code is to pull only 2023 data from each file

Additionally, due to server space limitations, an entire file cannot by loaded into python, therefore we need to separate each file by year or import by chunks

```
import random
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os

pd.set_option('display.max_columns', None)

# to pull from main server

account_creation = pd.read_csv("../acc.csv")
transactions= pd.read_csv("../txns.csv")
ratings = pd.read_csv("../rate.csv")
avod = pd.read_csv("../avod.csv")
wishlist = pd.read_csv("../wish.csv")

import pandas as pd

# ACCOUNTS 2023

# Define the chunk size
chunk_size = 100000  # Adjust the chunk size as needed

# Read the chunks of the CSV file
account_chunks = pd.read_csv("../acc.csv", header=0,
chunksize=chunk_size)

# Initialize an empty list to store filtered chunks
filtered_chunks = []

# Loop through the chunks
for chunk in account_chunks:
    # Convert createtime to datetime
    chunk['createtime'] = pd.to_datetime(chunk['createtime'])

    # Filter for 2023
    chunk_2023 = chunk.loc[chunk['createtime'].dt.year == 2023]

    # Break up createtime into date and time columns
    chunk_2023['Date'] = chunk_2023['createtime'].dt.date
    chunk_2023['Time'] = chunk_2023['createtime'].dt.time
```

```
    # Replace the createtime column with a simpler version
    chunk_2023['date_created'] = pd.to_datetime(chunk_2023['Date'])
    chunk_2023 = chunk_2023.drop(['Date', 'createtime'], axis=1)

    # Create the month, year, day, and quarter columns
    chunk_2023['year'] = chunk_2023['date_created'].dt.year
    chunk_2023['month']

# FILTER ALL TVOD FILES FOR 2023

# Define the directory where the CSV files are located
directory = "../"
# Define the list of CSV files
csv_files = ['tvod_1.csv', 'tvod_2.csv',
             'tvod_3.csv', 'tvod_4.csv',
             'tvod_5.csv', 'tvod_6.csv',
             'tvod_7.csv']

# Initialize an empty list to store filtered DataFrames
filtered_dfs = []

# Loop through each CSV file
for file in csv_files:
    # Read the CSV file in chunks
    chunk_size = 100000
    chunks = pd.read_csv(os.path.join(directory, file),
chunksize=chunk_size)

    # Initialize an empty list to store filtered chunks for this file
    filtered_chunks = []

    # Loop through the chunks in the current file
    for chunk in chunks:
        # Filter for 2023
        chunk_2023 =
chunk.loc[pd.to_datetime(chunk['starttime']).dt.year == 2023]

        # Append the filtered chunk to the list
        filtered_chunks.append(chunk_2023)

    # Concatenate all filtered chunks of this file into one DataFrame
    filtered_df = pd.concat(filtered_chunks, ignore_index=True)

    # Append the filtered DataFrame of this file to the list
    filtered_dfs.append(filtered_df)

# Concatenate all filtered DataFrames from different files into one
DataFrame
tvod_2023 = pd.concat(filtered_dfs, ignore_index=True)
```

```
# Save the DataFrame as a CSV file
tvod_2023.to_csv("tvod_2023.csv", index=False)

# RATINGS 2023

# Read the first chunk of the CSV file
chunk_size = 100000  # Adjust the chunk size as needed
rating_chunks = pd.read_csv("../rate.csv", header=0,
chunksize=chunk_size)

# Initialize an empty list to store filtered chunks
filtered_chunks = []

# Loop through the chunks
for chunk in rating_chunks:
    # Filter for 2023
    rating_2023 = chunk.loc[pd.to_datetime(chunk['modtime']).dt.year
== 2023]

    # Convert modtime to datetime
    rating_2023['modtime'] = pd.to_datetime(rating_2023['modtime'])

    # Break up modtime into date and time columns
    rating_2023['Date'] = rating_2023['modtime'].dt.date
    rating_2023['Time'] = rating_2023['modtime'].dt.time

    # Replace the created time pst column with a simpler version
    rating_2023['date_created'] = pd.to_datetime(rating_2023['Date'])
    rating_2023 = rating_2023.drop(['Date', 'modtime'], axis=1)

    # Created the month, year, day, and quarter columns
    rating_2023['year'] = rating_2023['date_created'].dt.year
    rating_2023['month'] = rating_2023['date_created'].dt.month
    rating_2023['day'] = rating_2023['date_created'].dt.day
    rating_2023['quarter'] = rating_2023['date_created'].dt.quarter

    # Append the filtered chunk to the list
    filtered_chunks.append(rating_2023)

# Concatenate all filtered chunks into one DataFrame
rating_2023 = pd.concat(filtered_chunks, ignore_index=True)

# Save the DataFrame as a CSV file
rating_2023.to_csv("rating_2023.csv", index=False)

import pandas as pd

# WISHLISTS 2023

# Define the chunk size
chunk_size = 100000  # Adjust the chunk size as needed
```

```
# Read the chunks of the CSV file
wishlist_chunks = pd.read_csv("../wish.csv", header=0,
chunksize=chunk_size)

# Initialize an empty list to store filtered chunks
filtered_chunks = []

# Loop through the chunks
for chunk in wishlist_chunks:
    # Filter for rows where either modtime or deltime is in 2023
    wishlist_2023 =
chunk.loc[(pd.to_datetime(chunk['modtime']).dt.year == 2023) |
                              (pd.to_datetime(chunk['deltime']).dt.year
== 2023)]

    # Convert modtime and deltime to datetime
    wishlist_2023['modtime'] =
pd.to_datetime(wishlist_2023['modtime'])
    wishlist_2023['deltime'] =
pd.to_datetime(wishlist_2023['deltime'])

    # Break up modtime and deltime into date and time columns
    wishlist_2023['Mod Date'] = wishlist_2023['modtime'].dt.date
    wishlist_2023['Mod Time'] = wishlist_2023['modtime'].dt.time
    wishlist_2023['Del Date'] = wishlist_2023['deltime'].dt.date
    wishlist_2023['Del Time'] = wishlist_2023['deltime'].dt.time

    # Replace the modtime and deltime columns with simpler versions
    wishlist_2023['date_modified'] = pd.to_datetime(wishlist_2023['Mod
Date'])
    wishlist_2023['date_deleted'] = pd.to_datetime(wishlist_2023['Del
Date'])
    wishlist_2023 = wishlist_2023.drop(['Mod Date', 'modtime', 'Del
Date', 'deltime'], axis=1)

    # Create the month, year, day, and quarter columns for
modification date
    wishlist_2023['mod_year'] = wishlist_2023['date_modified'].dt.year
    wishlist_2023['mod_month'] =
wishlist_2023['date_modified'].dt.month
    wishlist_2023['mod_day'] = wishlist_2023['date_modified'].dt.day
    wishlist_2023['mod_quarter'] =
wishlist_2023['date_modified'].dt.quarter

    # Create the month, year, day, and quarter columns for deletion
date
    wishlist_2023['del_year'] = wishlist_2023['date_deleted'].dt.year
    wishlist_2023['del_month'] =
wishlist_2023['date_deleted'].dt.month
```

```
    wishlist_2023['del_day'] = wishlist_2023['date_deleted'].dt.day
    wishlist_2023['del_quarter'] =
wishlist_2023['date_deleted'].dt.quarter

    # Append the filtered chunk to the list
    filtered_chunks.append(wishlist_2023)

# Concatenate all filtered chunks into one DataFrame
wishlist_2023 = pd.concat(filtered_chunks, ignore_index=True)

# Save the DataFrame as a CSV file
wishlist_2023.to_csv("wishlist_2023.csv", index=False)

# TRANSACTIONS 2023

# Read the first chunk of the CSV file
chunk_size = 100000  # Adjust the chunk size as needed
txns_chunks = pd.read_csv("../txns.csv", header=0,
chunksize=chunk_size)

# Initialize an empty list to store filtered chunks
filtered_chunks = []

# Loop through the chunks
for chunk in txns_chunks:
    # Filter for 2023
    txns_2023 = chunk.loc[pd.to_datetime(chunk['purchdate']).dt.year
== 2023]

    # Convert purchtime to datetime
    txns_2023['purchdate'] = pd.to_datetime(txns_2023['purchdate'])

    # Break up purchtime into date and time columns
    txns_2023['Date'] = txns_2023['purchdate'].dt.date
    txns_2023['Time'] = txns_2023['purchdate'].dt.time

    # Replace the created time pst column with a simpler version
    txns_2023['date_created'] = pd.to_datetime(txns_2023['Date'])
    txns_2023 = txns_2023.drop(['Date', 'purchdate'], axis=1)

    # Created the month, year, day, and quarter columns
    txns_2023['year'] = txns_2023['date_created'].dt.year
    txns_2023['month'] = txns_2023['date_created'].dt.month
    txns_2023['day'] = txns_2023['date_created'].dt.day
    txns_2023['quarter'] = txns_2023['date_created'].dt.quarter

    # Append the filtered chunk to the list
    filtered_chunks.append(txns_2023)

# Concatenate all filtered chunks into one DataFrame
txns_2023 = pd.concat(filtered_chunks, ignore_index=True)
```

```
# Save the DataFrame as a CSV file
txns_2023.to_csv("txns_2023.csv", index=False)
```

Now we want to focus on AVOD streams in particular so we want to take the entire file and separate it into different csvs for different years for trend analysis

```
# code to iterate through the entire avod file and print out each year
so
# we can make CSVs for each year

chunk_size = 100000  # Adjust the chunk size as needed
avod_chunks = pd.read_csv("../avod.csv", header=0,
chunksize=chunk_size)

unique_years = set()  # To store unique years

for chunk in avod_chunks:

unique_years.update(pd.to_datetime(chunk['starttime']).dt.year.unique(
))

print("Unique years available in the file:", sorted(unique_years))

import pandas as pd

# Define the range of years
years = range(2016, 2023 + 1)  # 2016 to 2023

# Define the chunk size
chunk_size = 100000  # Adjust the chunk size as needed

# Read the chunks of the file
avod_chunks = pd.read_csv("../avod.csv", header=0,
chunksize=chunk_size)

# Loop through each year
for year in years:
    # Initialize an empty list to store filtered chunks for the
current year
    filtered_chunks = []

    # Loop through the chunks
    for chunk in avod_chunks:
        # Convert starttimepst to datetime
        chunk['starttime'] = pd.to_datetime(chunk['starttime'])
        # Filter for the current year
        year_chunk = chunk[chunk['starttime'].dt.year == year]
        # Append the filtered chunk to the list
        filtered_chunks.append(year_chunk)
```

```
    # Concatenate all filtered chunks into one DataFrame for the
current year
    avod_year = pd.concat(filtered_chunks, ignore_index=True)

    # Save the DataFrame as a CSV file for the current year
    avod_year.to_csv(f"avod_{year}.csv", index=False)

    # Reset the iterator to read the CSV file again for the next year
    avod_chunks = pd.read_csv("../avod.csv", header=0,
chunksize=chunk_size)
```

Now we can begin feature engineering for each file, grouping by extid and by quarter

```
import pandas as pd

# Read CSV file
df = pd.read_csv("avod_2023.csv")

# Pivot the DataFrame to create different columns for each category
under 'platform' and 'quarter'
pivot_df = df.pivot_table(index=['extid', 'quarter'],
columns='platform', aggfunc='size', fill_value=0)

# Grouping by 'extid' and 'quarter' and counting the number of unique
'streams'
stream_counts = df.groupby(['extid', 'quarter'])
['streamsess'].nunique()

# Grouping by 'extid' and 'quarter' and counting the number of unique
'contents'
content_counts = df.groupby(['extid', 'quarter'])['conts'].nunique()

# Grouping by 'extid' and 'quarter' and counting the number of unique
'platforms'
platform_counts = df.groupby(['extid', 'quarter'])['plat'].nunique()

# Concatenate the pivot_df with other aggregated results
result_df = pd.concat([stream_counts, content_counts, platform_counts,
pivot_df], axis=1)
result_df.reset_index(inplace=True)

# Renaming the columns
result_df.columns = ['extid', 'quarter', 'stream_count',
'content_count','platform_counts'] + ['platform_' + str(col) for col
in pivot_df.columns]

print(result_df)

result_df.to_csv("result.csv", index=False)
```

```
print("Saved result_df as result.csv")

df1= pd.read_csv("tvod_2023.csv")
df1['starttime'] = pd.to_datetime(df1['starttime'])
df1['quarter'] = df1['starttime'].dt.quarter

# Pivot the DataFrame to create different columns for each category
under 'platform' and 'quarter'
pivot_df1 = df1.pivot_table(index=['extid', 'quarter'],
columns='plat', aggfunc='size', fill_value=0)

# Pivot the DataFrame to create different columns for each category
under 'videoquality' and 'quarter'
pivot_df2 = df1.pivot_table(index=['extid', 'quarter'],
columns='vide_qual', aggfunc='size', fill_value=0)

# Pivot the DataFrame to create different columns for each category
under 'offer' and 'quarter'
pivot_df3 = df1.pivot_table(index=['extid', 'quarter'], columns='off',
aggfunc='size', fill_value=0)

# Grouping by 'extid' and 'quarter' and counting the number of unique
'streamingsessionid'
stream_counts = df1.groupby(['extid', 'quarter'])
['streamsess'].nunique()

# Grouping by 'extid' and 'quarter' and counting the number of unique
'contentid'
content_counts = df1.groupby(['extid', 'quarter'])['cont'].nunique()

# Grouping by 'extid' and 'quarter' and counting the number of unique
'platform'
platform_counts = df1.groupby(['extid', 'quarter'])['plat'].nunique()

# Grouping by 'extid' and 'quarter' and counting the occurrences of
each 'quality'
quality_counts = df1.groupby(['extid', 'quarter'])
['videqual'].nunique()

# Grouping by 'extid' and 'quarter' and counting the occurrences of
each 'offer'
offer_counts = df1.groupby(['extid', 'quarter'])['off'].nunique()

# Concatenate the pivot_df with other aggregated results
result_df1 = pd.concat([stream_counts, content_counts,
platform_counts, quality_counts, offer_counts, pivot_df1, pivot_df2,
pivot_df3], axis=1)
result_df1.reset_index(inplace=True)

# Renaming the columns
```

```
new_columns = ['extid', 'quarter', 'stream_count', 'content_count',
'platform_count', 'quality_count', 'offer_count'] \
              + ['platform_' + str(col) for col in pivot_df1.columns]
\
              + ['quality_' + str(col) for col in pivot_df2.columns] \
              + ['offer_' + str(col) for col in pivot_df3.columns]

result_df1.columns = new_columns

print(result_df1)


result_df1.to_csv("tvod_agg.csv", index=False)

print("Saved result_df as tvod_agg.csv")

# Grouping by 'extid' and 'quarter' and counting the number of unique
'contentid'
content_counts = df3.groupby(['extid', 'quarter'])['cont'].nunique()

# Grouping by 'extid' and 'quarter' and calculating the average star
rating
avg_rating = df3.groupby(['extid', 'quarter'])['stars'].mean()

# Concatenate the pivot_df with other aggregated results
result_df = pd.concat([content_counts, avg_rating], axis=1)
result_df.reset_index(inplace=True)

# Renaming the columns
result_df.columns = ['extid', 'quarter', 'content_count','avg_rating']

print(result_df)


result_df.to_csv("rating_agg.csv", index=False)

print("Saved result_df as rating_agg.csv")


df2['date_mod'] = pd.to_datetime(df2['date_mod'])
df2['quarter'] = df2['date_mod'].dt.quarter

# Grouping by 'extid' and 'quarter' and counting the number of unique
'modifications'
mod = df2.groupby(['extid', 'quarter'])['Mod'].count()

# Grouping by 'extid' and 'quarter' and number of deletes
delete = df2.groupby(['extid', 'quarter'])['Del'].count()

# Concatenate the pivot_df with other aggregated results
result_df = pd.concat([mod, delete], axis=1)
```

```
result_df.reset_index(inplace=True)

# Renaming the columns
result_df.columns = ['extid', 'quarter', 'mod','delete']

print(result_df)


result_df.to_csv("mod_agg.csv", index=False)

print("Saved result_df as mod_agg.csv")

# Pivot the DataFrame to create different columns for each category
under offer
pivot_df2 = df4.pivot_table(index=['extid', 'quarter'], columns='off',
aggfunc='size', fill_value=0)

# Pivot the DataFrame to create different columns for each category
under library
pivot_df3 = df4.pivot_table(index=['extid', 'quarter'], columns='lib',
aggfunc='size', fill_value=0)

# Pivot the DataFrame to create different columns for each category
under purchase
pivot_df4 = df4.pivot_table(index=['extid', 'quarter'],
columns='purch', aggfunc='size', fill_value=0)

# Pivot the DataFrame to create different columns for each category
under content
pivot_df5 = df4.pivot_table(index=['extid', 'quarter'],
columns='cont', aggfunc='size', fill_value=0)

# Pivot the DataFrame to create different columns for each category
under videoquality
pivot_df6 = df4.pivot_table(index=['extid', 'quarter'],
columns='vidqual', aggfunc='size', fill_value=0)



# Grouping by 'extid' and 'quarter' and counting the number of unique
'contentid'
content_counts = df4.groupby(['extid', 'quarter'])['cont'].nunique()

# Grouping by 'extid' and 'quarter' and total money spent
totalmoney = df4.groupby(['extid', 'quarter'])['mon'].sum()

# Grouping by 'extid' and 'quarter' and counting the occurrences of
each 'purchase'
purch_type = df4.groupby(['extid', 'quarter'])['purch'].nunique()

# Concatenate the pivot_df with other aggregated results
```

```
result_df1 = pd.concat([content_counts, totalmoney, purch_type,
                        pivot_df2, pivot_df3, pivot_df4, pivot_df5,
pivot_df6], axis=1)
result_df1.reset_index(inplace=True)

# Renaming the columns
new_columns = ['extid', 'quarter',
'content_count','totalmoney','purchasetype'] \
              + ['txnoffer_' + str(col) for col in pivot_df2.columns]
\
              + ['library_' + str(col) for col in pivot_df3.columns] \
              + ['purchase_' + str(col) for col in pivot_df4.columns]
\
              + ['content_' + str(col) for col in pivot_df5.columns] \
              + ['video_' + str(col) for col in pivot_df6.columns]


result_df1.columns = new_columns

print(result_df1)


result_df1.to_csv("half_txns.csv", index=False)

print("Saved result_df1 as half_txns.csv")
```

Now we can combine our cleaned datasets to make a full/final dataset for our machine learning model

```
# import all files
avod = pd.read_csv("avod_agg.csv")
tvod = pd.read_csv("tvod_agg.csv")
mod = pd.read_csv("mod_agg.csv")
rating = pd.read_csv("rating_agg.csv")
txns = pd.read_csv("half_txns.csv")
ref_cam = pd.read_csv("ref_cam.csv")
```

feature engineering/binning occured at every merge section. However could not include the code due to NDA

```
# combine avod and tvod
merged_df = pd.merge(avod, tvod, on=['extid', 'quarter'],
suffixes=('_avod', '_tvod'), how= 'outer')
merged_df = merged_df.fillna(0)
merged_df.rename(columns={'stream_count_avod':
'avod','stream_count_tvod':'tvod'}, inplace=True)
merged_df.to_csv("finalatvod.csv", index=False)

print("Saved as finalatvod.csv")
```

```
# Merge the modifications and ratings on 'extid' and 'quarter'
lil_merge = pd.merge(mod, rating, on=['extid', 'quarter'],
suffixes=('_mod', '_rat'), how= 'outer')
lil_merge = lil_merge.fillna(0)

lil_merge['edits'] = lil_merge['mod'] + lil_merge['delete']

lil_merge.drop(columns=['mod','delete'], inplace=True)

lil_merge.rename(columns={'content_count': '#contents_rated'},
inplace=True)

# merge avod/tvod file and modifications/ratings file
fourth = pd.read_csv("finalatvod.csv")
_merge = pd.merge(fourth, lil_merge, on=['extid', 'quarter'],
suffixes=('_atvod', '_lil'), how = 'outer')
_merge = _merge.fillna(0)
_merge.to_csv("final_four_merge.csv", index=False)

print("Saved as final_four_merge.csv")

# merge transactions and referrer/campaigns
full_txns = pd.merge(txns, ref_cam, on=['extid', 'quarter'],
suffixes=('_txns', '_rc'), how='outer')
full_txns = full_txns.fillna(0)
full_txns.to_csv("final_txns.csv", index=False)

print("Saved as final_txns.csv")

four = pd.read_csv("final_four_merge.csv")
txns = pd.read_csv("final_txns.csv")
final = pd.merge(four, txns, on=['extid', 'quarter'],
suffixes=('_four', '_txns'), how='outer')
final = final.fillna(0)

final = pd.merge(four, txns, on=['extid', 'quarter'],
suffixes=('_four', '_txns'), how='outer')
final = final.fillna(0)
final.to_csv("final.csv", index=False)

print("Saved as final.csv")
```

Now we want to find the account duration of each customer from the time of their account creation to the beginning of that quarter

```
acc = pd.read_csv("../acc.csv")

all = pd.merge(final, acc, on='extid', how = 'left')

# Create the 'acc_stand' column based on the 'quarter' column
quarter_to_acc_stand = {
```

```
    1: '01-01-2023',
    2: '04-01-2023',
    3: '07-01-2023',
    4: '10-01-2023'
}

all['acc_stand'] = all['quarter'].map(quarter_to_acc_stand)


all['acc_stand'] = pd.to_datetime(all['acc_stand'])

all['acc_stand'] = pd.to_datetime(all['acc_stand'])
all['create_date'] = pd.to_datetime(all['createdtimepst'])

# Subtract the two datetime columns to get the timedelta in days
all['acc_dur'] = abs((all['acc_stand'] - all['create_date']).dt.days)

all.drop(columns=['createdtimepst', 'acc_stand', 'create_date'],
inplace=True)

all.to_csv("ALL.csv", index=False)
```