ETL Report

Increasing Audio Streaming User Retention via Music Recommendation System

Land’o’datalakes: Vanessa Gleason, Alistair Marsden, Olivier Rochaix, Eduard Stalmakov

ETL Process 09/22/22

Introduction

Audio streaming services benefit financially through the engagement of users on their platform. Our plan to promote retention and engagement of users is by recommending specific songs based on the songs a Spotify user has in their playlists. Our data sources contain playlist information for Spotify users and also the characteristics of over six hundred thousand tracks on Spotify. We also used an API call of computer and internet use from the US Census Bureau to identify the demographics of individuals who stream music. These three datasets were all extracted and transformed before finally being loaded into a SQL database.

Data Sources

1. Boland, D. (n.d.). *Streamable Playlists with User Data*. Streamable playlists with User Data.

   Retrieved September 19, 2022, from http://www.dcs.gla.ac.uk/~daniel/spud/

2. Ay, Y. E. (2021, April). *Spotify dataset 1921-2020, 600K+ tracks*. Kaggle. Retrieved September 26,

2022, from https://www.kaggle.com/datasets/yamaerenay/spotify-dataset-19212020-600k-tracks

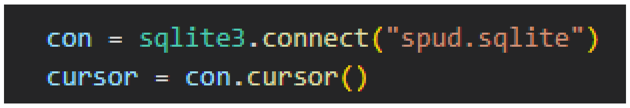
3. Bureau, U. S. C. (2019, November). *Current Population Survey: Computer and Internet Use Supplement*. Retrieved September 23, 2022, from

   https://api.census.gov/data/2019/cps/internet/nov

Extraction

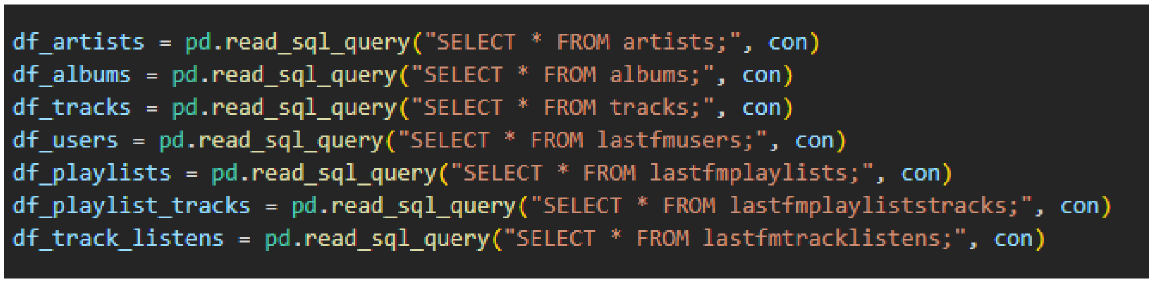
Source 1: Streamable Playlists with User Data

1. Go to this link: http://www.dcs.gla.ac.uk/~daniel/spud/ and click "Download dataset"
2. Open spud.sqlite file with the following command:

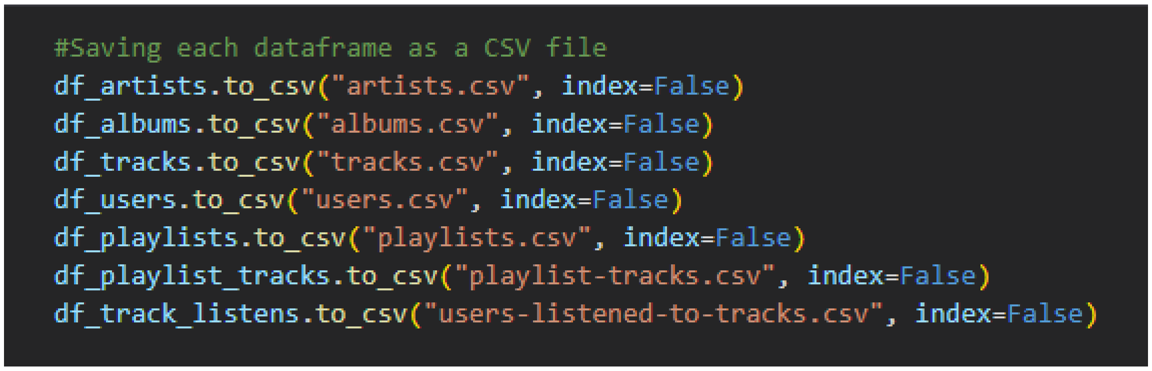


```
con = sqlite3.connect("spud.sqlite")
cursor = con.cursor()
```

3. Extract each table to a pandas dataframe and save each dataframe as a csv file to your desired directory



```
df_artists = pd.read_sql_query("SELECT * FROM artists;", con)
df_albums = pd.read_sql_query("SELECT * FROM albums;", con)
df_tracks = pd.read_sql_query("SELECT * FROM tracks;", con)
df_users = pd.read_sql_query("SELECT * FROM lastfmusers;", con)
df_playlists = pd.read_sql_query("SELECT * FROM lastfmplaylists;", con)
df_playlist_tracks = pd.read_sql_query("SELECT * FROM lastfmplayliststracks;", con)
df_track_listens = pd.read_sql_query("SELECT * FROM lastfmtracklistens;", con)
```



```
#Saving each dataframe as a CSV file
df_artists.to_csv("artists.csv", index=False)
df_albums.to_csv("albums.csv", index=False)
df_tracks.to_csv("tracks.csv", index=False)
df_users.to_csv("users.csv", index=False)
df_playlists.to_csv("playlists.csv", index=False)
df_playlist_tracks.to_csv("playlist-tracks.csv", index=False)
df_track_listens.to_csv("users-listened-to-tracks.csv", index=False)
```

Source 2: Spotify dataset 1921-2020, 600K+ tracks

1. Go to this link: https://www.kaggle.com/datasets/yamaerenay/spotify-dataset-19212020-600k-tracks

   and click "Download"

2. Save the CSV file to your desired directory

Source 3: Census API

1. Create a request using the url http://api.census.gov/data/2019/cps/internet/nov
2. Called in variables HRNUMHOU, HEINHOME, PESEX, PREMPNOT, PRTAGE, PTDTRACE, HEFAMINC, and PEAUDIO by State.

```
url =
f'https://api.census.gov/data/2019/cps/internet/nov?get=HRNUMHOU,HEINHOME,
PESEX,PREMPNOT,PRTAGE,PTDTRACE,HEFAMINC,PEAUDIO&for=state:*&PERRP=2&key=fc
ac44908845afd9ddbb9e2aa118919772edbddb'

res = requests.get(url).json()

df = pd.DataFrame(res)

df.columns = df.iloc[0]

df.drop(df.index[0], inplace=True)

df
```

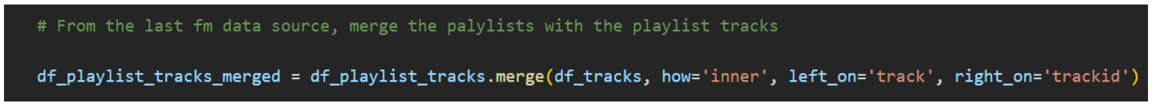
3. Save 'df' as a CSV into desired directory

Transformation

Source 1: Streamable playlists with User Data

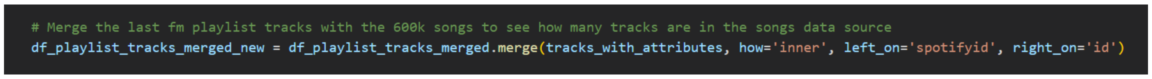
Part 1: Filtering the desired playlists

1. Open "tracks.csv", "playlist-tracks.csv", "users.csv",and "playlists.csv" into a pandas dataframes called:
   - `df_tracks`
   - `df_playlist_tracks`
   - `df_users`
   - `df_playlists`
2. Merge the `df_tracks` and `df_playlist_tracks` dataframes



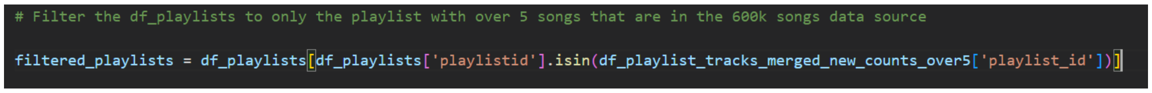
```
# From the last fm data source, merge the palylists with the playlist tracks

df_playlist_tracks_merged = df_playlist_tracks.merge(df_tracks, how='inner', left_on='track', right_on='trackid')
```

3. Open source 2 csv file into a pandas dataframe called `tracks_with_attributes`
4. Merge the previous dataframe with the `tracks_with_attributes`



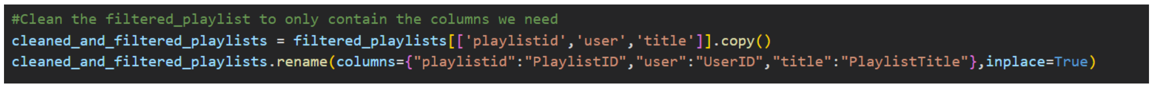
```
# Merge the last fm playlist tracks with the 600k songs to see how many tracks are in the songs data source
df_playlist_tracks_merged_new = df_playlist_tracks_merged.merge(tracks_with_attributes, how='inner', left_on='spotifyid', right_on='id')
```

5. Save only the playlists which have more than 5 songs on them (playlist value counts > 5) into a dataframe called `df_playlist_tracks_merged_new_counts_over5`
6. Filter the `df_playlists` dataframe to only those tracks



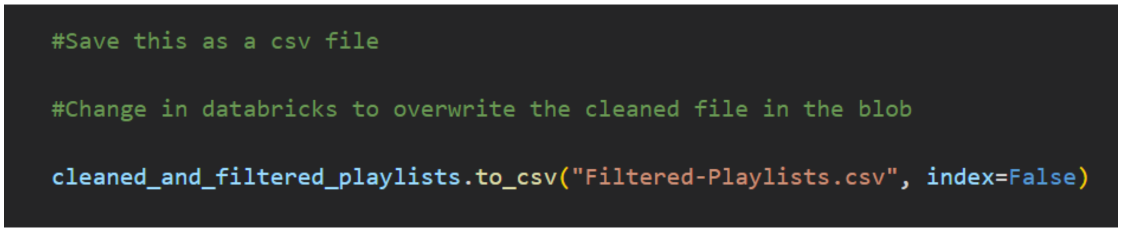
```
# Filter the df_playlists to only the playlist with over 5 songs that are in the 600k songs data source

filtered_playlists = df_playlists[df_playlists['playlistid'].isin(df_playlist_tracks_merged_new_counts_over5['playlist_id'])]
```

7. Rename the columns



```
#Clean the filtered_playlist to only contain the columns we need
cleaned_and_filtered_playlists = filtered_playlists[['playlistid','user','title']].copy()
cleaned_and_filtered_playlists.rename(columns={"playlistid":"PlaylistID","user":"UserID","title":"PlaylistTitle"},inplace=True)
```

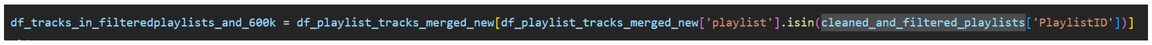
8. Save to a csv file



```
#Save this as a csv file

#Change in databricks to overwrite the cleaned file in the blob

cleaned_and_filtered_playlists.to_csv("Filtered-Playlists.csv", index=False)
```

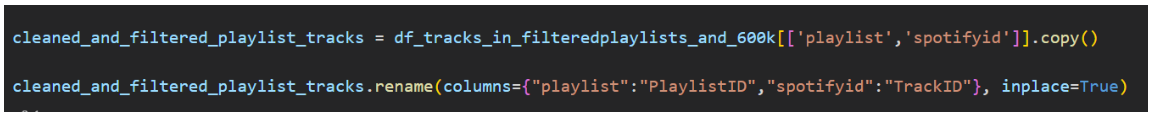
Part 2: Filtering the desired playlist tracks

9. Filter the `df_playlist_tracks_merged_new to only the tracks that show up in the cleaned_and_filtered_playlists dataframe`



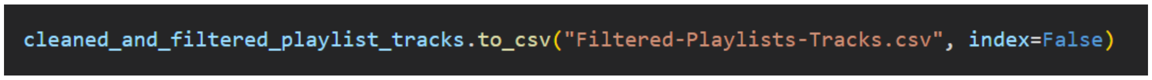
```
df_tracks_in_filteredplaylists_and_600k = df_playlist_tracks_merged_new[df_playlist_tracks_merged_new['playlist'].isin(cleaned_and_filtered_playlists['PlaylistID'])]
```

10. `Filter to only the playlist and spotifyid columns and rename them`



```
cleaned_and_filtered_playlist_tracks = df_tracks_in_filteredplaylists_and_600k[['playlist','spotifyid']].copy()

cleaned_and_filtered_playlist_tracks.rename(columns={"playlist":"PlaylistID","spotifyid":"TrackID"}, inplace=True)
```

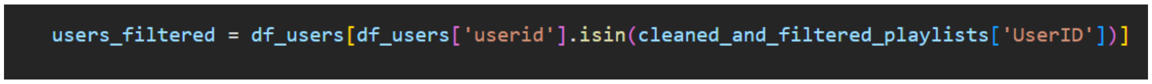
11. Save to a csv file



```
cleaned_and_filtered_playlist_tracks.to_csv("Filtered-Playlists-Tracks.csv", index=False)
```

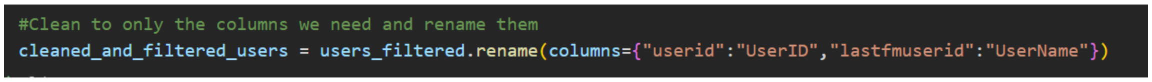
Part 2: Filtering the desired users

12. Filter the `df_users` dataframe to only the users whose playlists are in the `cleaned_and_filtered_playlists` dataframe

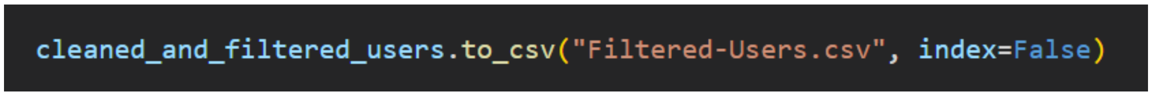


```
  users_filtered = df_users[df_users['userid'].isin(cleaned_and_filtered_playlists['UserID'])]
```

13. Rename columns and save to a csv file



```
#Clean to only the columns we need and rename them
cleaned_and_filtered_users = users_filtered.rename(columns={"userid":"UserID","lastfmuserid":"UserName"})
```

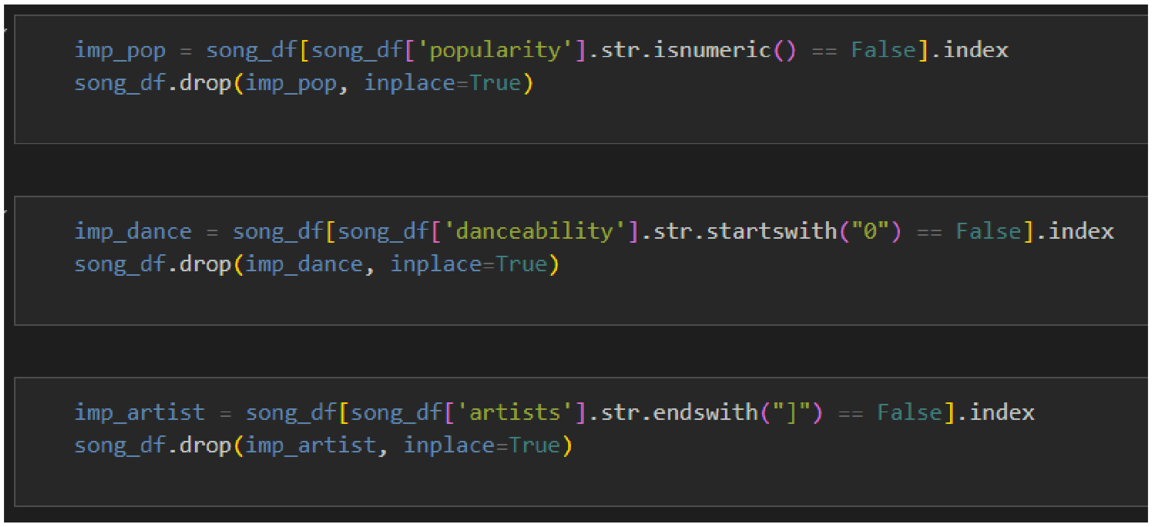


```
cleaned_and_filtered_users.to_csv("Filtered-Users.csv", index=False)
```

<u>Source 2: Spotify dataset 1921-2020, 600K+ tracks</u>

Part 1: Tracks

1. Read the "Tracks-600k-songs.csv" into a pandas dataframe
2. Check for improperly entered values in the "popularity", "danceability", and "artists" columns, then drop the rows in question.

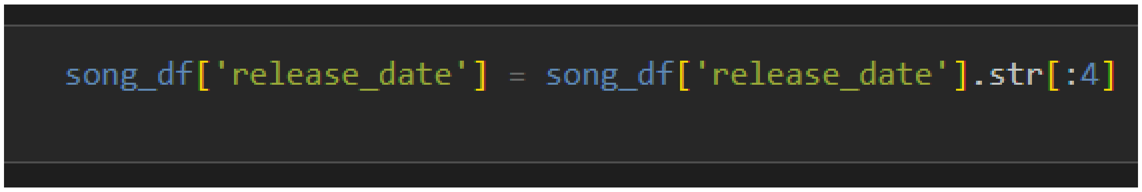


```
imp_pop = song_df[song_df['popularity'].str.isnumeric() == False].index
song_df.drop(imp_pop, inplace=True)
```

```
imp_dance = song_df[song_df['danceability'].str.startswith("0") == False].index
song_df.drop(imp_dance, inplace=True)
```

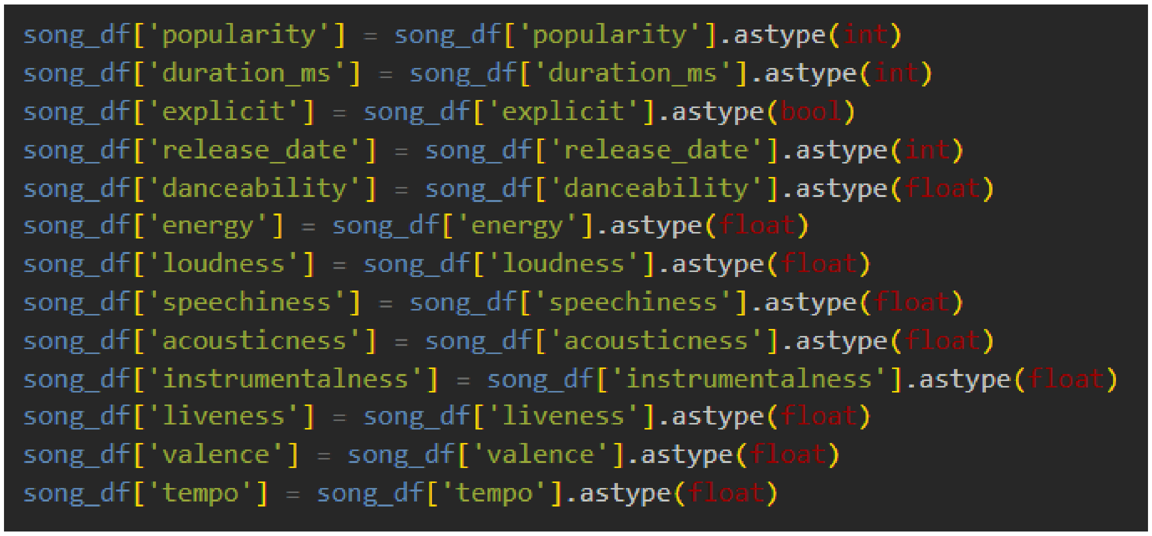
```
imp_artist = song_df[song_df['artists'].str.endswith("]") == False].index
song_df.drop(imp_artist, inplace=True)
```

3. Ensure that all values in the “release_date” column are properly formatted. Our group chose to grab the year the songs were released.



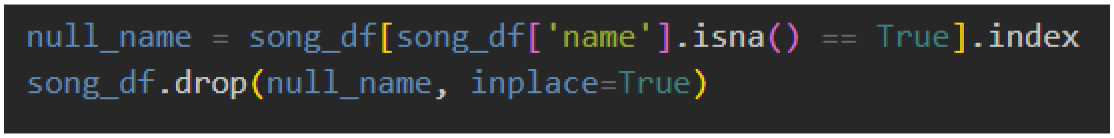
```
song_df['release_date'] = song_df['release_date'].str[:4]
```

4. Ensure that each value has been set to the proper data type. Our group used this code:



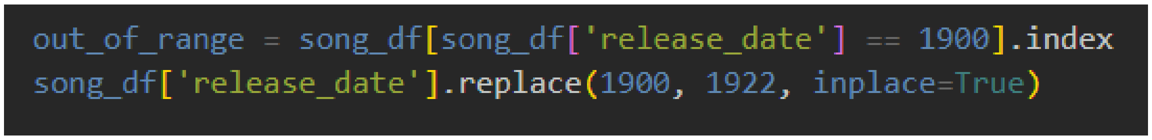
```
song_df['popularity'] = song_df['popularity'].astype(int)
song_df['duration_ms'] = song_df['duration_ms'].astype(int)
song_df['explicit'] = song_df['explicit'].astype(bool)
song_df['release_date'] = song_df['release_date'].astype(int)
song_df['danceability'] = song_df['danceability'].astype(float)
song_df['energy'] = song_df['energy'].astype(float)
song_df['loudness'] = song_df['loudness'].astype(float)
song_df['speechiness'] = song_df['speechiness'].astype(float)
song_df['acousticness'] = song_df['acousticness'].astype(float)
song_df['instrumentalness'] = song_df['instrumentalness'].astype(float)
song_df['liveness'] = song_df['liveness'].astype(float)
song_df['valence'] = song_df['valence'].astype(float)
song_df['tempo'] = song_df['tempo'].astype(float)
```

5. Drop tracks that have no name, as our recommender will not work nicely with null values.



```
null_name = song_df[song_df['name'].isna() == True].index
song_df.drop(null_name, inplace=True)
```

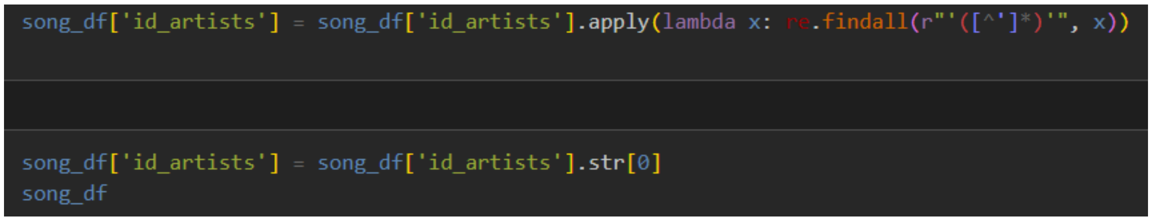
6. Ensure that the “release_date” column’s values are properly set within the bounds of 1921-2020.



```
out_of_range = song_df[song_df['release_date'] == 1900].index
song_df['release_date'].replace(1900, 1922, inplace=True)
```

7. Transform values in “artists” from a string to a list, then grab the first artist to attribute the track to them. Our group grabbed the first artist because some tracks had upwards of 60 artists, which would have created many unnecessary columns in our dataframe if

they were expanded.



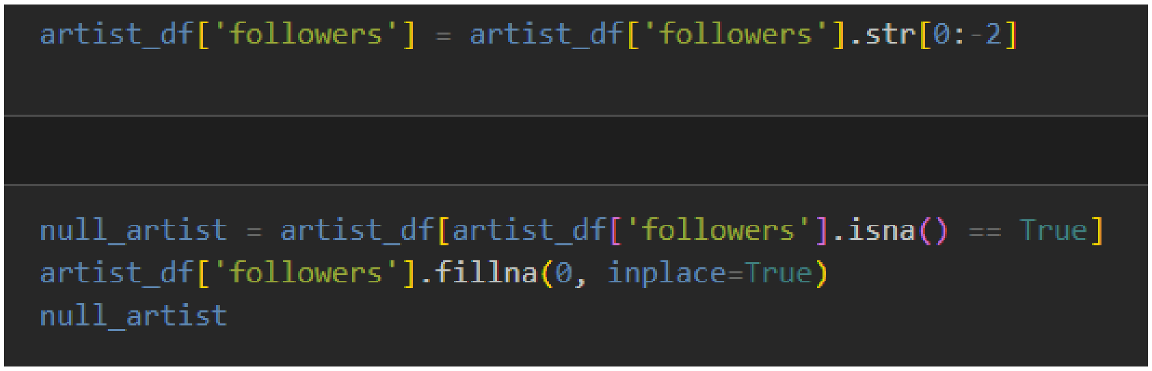
```
song_df['id_artists'] = song_df['id_artists'].apply(lambda x: re.findall(r"'([^']*)'", x))
```

```
song_df['id_artists'] = song_df['id_artists'].str[0]
song_df
```

8. Save the dataframe to a CSV

Part 2: Artists

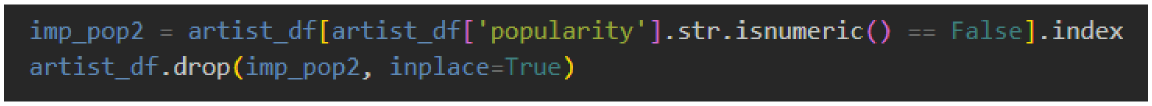
1. Read the “artists.csv” file into a pandas dataframe.
2. In the “followers” column, remove the null values as well as the decimal points.



```
artist_df['followers'] = artist_df['followers'].str[0:-2]
```

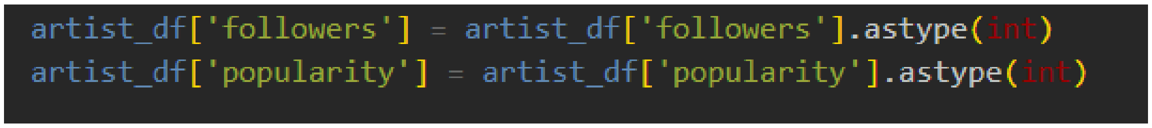
```
null_artist = artist_df[artist_df['followers'].isna() == True]
artist_df['followers'].fillna(0, inplace=True)
null_artist
```

3. Ensure that values in the “popularity” column are properly entered, and drop any rows that have incorrectly entered information.



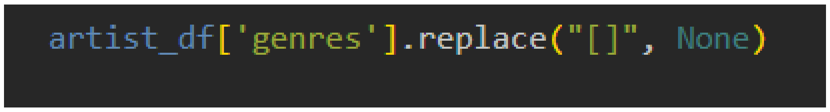
```
imp_pop2 = artist_df[artist_df['popularity'].str.isnumeric() == False].index
artist_df.drop(imp_pop2, inplace=True)
```

4. Ensure that the “followers” and “popularity” columns’ values are cast as the right datatypes.



```
artist_df['followers'] = artist_df['followers'].astype(int)
artist_df['popularity'] = artist_df['popularity'].astype(int)
```

5. Replace empty square brackets in the “genres” column with null values.



```
artist_df['genres'].replace("[]", None)
```

6. Save the dataframe to a CSV.

Source 3: Census API

1. Open the “Dataset-Census-Raw.csv” into a pandas dataframe.

2. Rename the columns in the df to be easily understandable.

```
df.rename(columns={'HEINHOME':'Does anyone use the Internet at home?',

'HRNUMHOU':'Number of household members',

'PESEX':'Sex',

'state':'State',

'PREMPNOT':'Employment Status',

'PRTAGE':'Age',

'PTDTRACE':'Race',

'HEFAMINC':'Household-total income last 12 months',

'PEAUDIO':'Stream or Download Music?'}, inplace = True)
```

3. Change the numerical values in the dataframe to their actual values as a copied dataframe.

```
df2 = df.copy()

df2['Does anyone use the Internet at home?'] = df2['Does anyone use the
Internet at home?'].replace(['1','2','-1'],['Yes','No','No'])

df2['Sex'] = df2['Sex'].replace(['1','2'],['Male','Female'])

df2['Employmnet Status'] = df2['Employment
Status'].replace(['1','2','3','4','-1'],['Employed','Unemployed','Discoura
ged-Not in labor force','Other-Not in labor force','Other'])

df2['Race'] = df2['Race'].replace(['2','5','4','1','3'],['Black
Only','Hawaiian/Pacific Islander Only','Asian Only','White Only','American
Indian, Alaskan Native Only'])

df2['Race'] =
df2['Race'].replace(['7','20','8','17','16','6','12','18','21','22','9','1
4','10','23','11','13','15','25','26','19','24'],'Other')

df2['Household-total income last 12 months'] = df2['Household-total income
last 12
months'].replace(['1','2','3','4','5','6','7','8','9','10','11','12','13',
'14','15','16'],

['Less than $5000','$5000 To $7499','$7500 To $9999','$10000 To
$12499','$12500 To $14999','$15000 To $19999','$20000 To $24999','$25000
To $29999',

'$30000 To $34999','$35000 To $39999','$40000 To $49999','$50000 To
$59999','$60000 To $74999','$75000 To $99999','$100000 To
$149999','$150000 or More'])

df2['Stream or Download Music?'] = df2['Stream or Download
Music?'].replace(['1','2','-1'],['Yes','No','No'])

df2['State'] =
df2['State'].replace(['1','2','4','5','6','8','9','10','11','12','13','15'
,'16','17','18','19','20','22','23','24','25',

'26','27','28','29','30','31','32','33','34','35','36','37','38','39','40'
,'41','42','44','45','46','47','48','49','50','51','53','54','55','56','21
'],

['AL','AK','AZ','AR','CA','CO','CT','DE','DC','FL','GA','HI','ID','IL','IN
','IA','KS','LA','ME','MD','MA','MI','MN','MS','MO','MT','NE',

'NV','NH','NJ','NM','NY','NC','ND','OH','OK','OR','PA','RI','SC','SD','TN'
,'TX','UT','VT','VA','WA','WV','WI','WY','KY'])
```

4. Convert strings to integers as needed.

```
df2['Number of household members'] = df2['Number of household
members'].astype(int)

df2['Age'] = df2['Age'].astype(int)
```

5. Drop the columns that we decided were no longer needed.

```
df2.drop(columns=['PERRP','Employment Status'],inplace=True)
```

6. Fix a spelling error

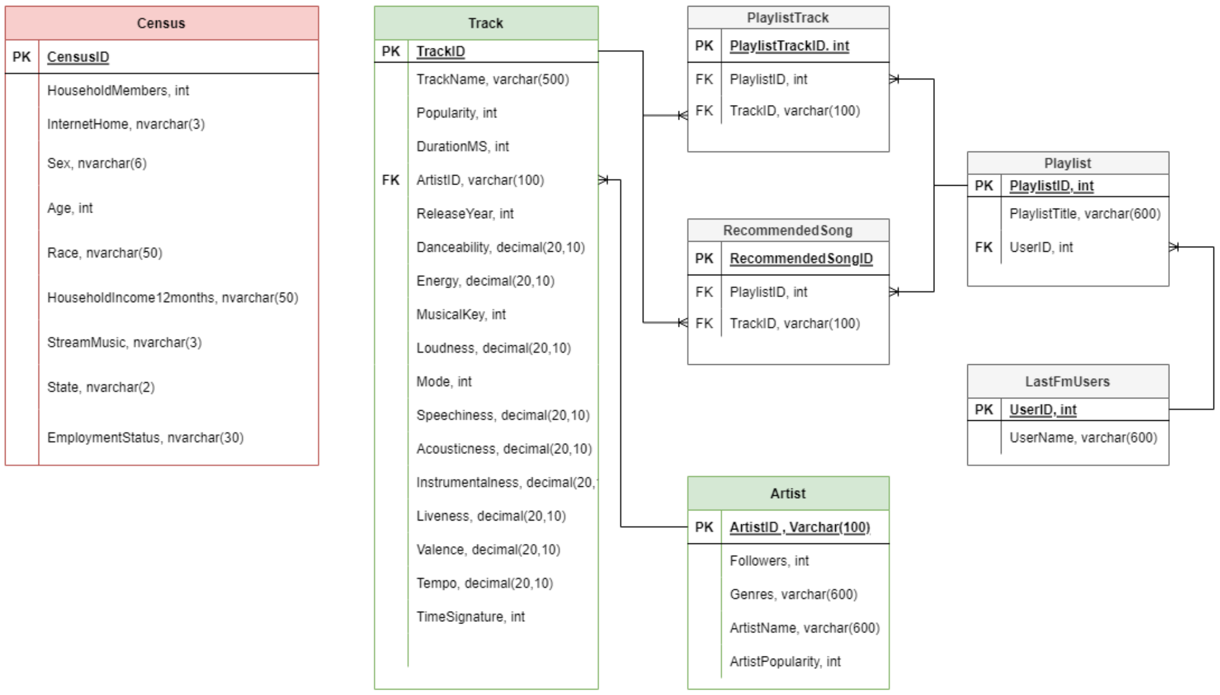
```
df2.rename(columns={'Employmnet Status':'Employment Status'},
inplace=True)
```

7. Save the dataframe as a CSV.

```
df2.to_csv('Dataset-Census-Audio.csv')
```

Load

To load the data into the SQL database, use this ERD diagram as a reference to the column names and datatypes. The data was loaded using Apache Spark within Azure Databricks.



Source 1: Streamable Playlists with User Data

1. In the Azure Databrick, open “Filtered-Playlists.csv”
2. Change the PlaylistID and UserID datatypes to int
3. Load this spark dataframe into the Playlist SQL table

4. Open the “Filtered-Users.csv”
5. Change the UserID datatype to int
6. Load this spark dataframe into the LastFmUsers SQL table

Source 3: Census API

1. In the Azure Databrick, open “Dataset-Census-Audio.csv”
2. Change the column names to match the formatting of the SQL table.
3. Load this spark dataframe into the Census SQL table

Conclusion

Our data sources will all be sent into a final SQL database. This database was then used to create our visualizations and our recommender models in order to answer our exploratory questions.