Truls_assignment1

February 18, 2024

1 CA1: Dataframe Manipulation with Spotify Data

1.1 Introduction

Pandas is an extremely powerful tool to handle large amounts of tabular data. In this compulsory assignment, you will use Pandas to explore one of the TA's personal spotify data in depth.

Additional information: - Feel free to create additional code cells if you feel that one cell per subtask is not sufficient. - Remember, Pandas uses very efficient code to handle large amounts of data. For-loops are not efficient. If you ever have to use a for-loop to loop over the rows in the DataFrame, you have *probably* done something wrong. - Label all graphs and charts if applicable.

1.2 Task

I typically enjoy indie and rock music. I am a big fan of everything from old-fashioned rock and roll like Led Zeppelin and Jimi Hendrix, to newer indie artists like Joji and Lana Del Rey. This is why my spotify wrapped for 2023 came as quite a surprise:

Now, I'm no hater of pop music, but this was unexpected. For this assignment, you will investigate my listening habits, including a deep dive into my Ariana Grande listening habits, and try to find an answer to why she was my top artist; was there a fault in the spotify algorithm? Am I actually secretly an *Arianator*? (yes, I did have to look that up). Or am I just lying to myself about how often I listen to guilty pleasure music?

1.3 Part 1: Initial loading and exploration

1.0 Import necessary libraries: pandas, numpy, matplotlib.pyplot (other libraries such as seaborn or plotly are also allowed if you want prettier plots). It might also be a good idea to use **os** for task 2.0

[526]:
```
import matplotlib.pyplot as plt
import pandas as pd
```

1.1 Loading the data Load the dataset in the file `streaming_history_0.csv` into a Pandas DataFrame called `df_spotify_0`.

[527]:
```
history = pd.read_csv("streaminghistory0.csv", header=0)
df_spotify_0 = pd.DataFrame(history)
```

1.2 Help function Use the Python command `help` to help you understand how to use the `pd.DataFrame.head` and `pd.DataFrame.tail` methods.

[ ]:

1.3 Getting an overview Print the first `five` and last `ten` rows of the dataframe. Have a quick look at which columns are in the dataset.

[528]:
```
num_entries = df_spotify_0.shape[0]
new_indexes = [f'Track {x + 1}' for x in range(num_entries)]
df_spotify_0.index = new_indexes
new_headers = {'endTime': 'End Time', 'artistName': 'Artist', 'trackName':␣
 ↪'Track', 'msPlayed': 'Playtime (Ms)'}
df_spotify_0 = df_spotify_0.rename(columns=new_headers)

df0_sorted = df_spotify_0.sort_values(by=['Playtime (Ms)'], ascending=False)
```

[529]:
```
df0_sorted.head(5)
```

[529]:

	End Time	Artist	Track	Playtime (Ms)
Track 5718	2022-12-20 19:30	The Smiths	How Soon Is Now? - 2011 Remaster	408173.0
Track 11702	2023-01-02 17:05	Lorde	Solar Power	392859.0
Track 6402	2022-12-21 17:59	10cc	I'm Not In Love	366640.0
Track 9990	2022-12-30 11:44	Tame Impala	New Person, Same Old Mistakes	363240.0
Track 10262	2022-12-30 15:51	Peach Pit	Tommy's Party	361760.0

[530]:
```
df0_sorted.tail(10)
```

[530]:

	End Time	Artist	Track	Playtime (Ms)
Track 11241	2023-01-01 21:53	Ariana Grande		
Track 11228	2023-01-01 21:47	Ariana Grande		
Track 10221	2022-12-30 13:59	Lorde		
Track 820	2022-12-08 18:52	Lorde		
Track 11215	2023-01-01 21:44	Fana Hues		
Track 11212	2023-01-01 21:44	Dominic Fike		
Track 11205	2023-01-01 21:44	Tame Impala		
Track 2192	2022-12-12 11:08	Sir Chloe		
Track 11184	2023-01-01 21:41	Ariana Grande		
Track 8583	2022-12-27 13:08	Bring Me The Horizon		

```
Track 11241                 Knew Better / Forever Boy           0.0
Track 11228  break up with your girlfriend, i'm bored           0.0
Track 10221                          Writer In The Dark         0.0
Track 820                                     Supercut          0.0
Track 11215                                     Icarus          0.0
Track 11212                                      Wurli          0.0
Track 11205                            The Boat I Row           0.0
Track 2192                                  Too Close           0.0
Track 11184                                successful           0.0
Track 8583                                  What You Need        0.0
```

[531]:
```
df_spotify_0.shape[0]
```

[531]:
```
11959
```

1.4 Formatting correctly When working with Pandas, it's very useful to have columns which contains dates in a specific format called *datetime*. This allows for efficient manipulation and analysis of time-series data, such as sorting, filtering by date or time, and resampling for different time periods. Figure out which column(s) would be appropriate to convert to datetime, if any, and if so, perform the conversion to the correct format.

[532]:
```
df_spotify_0.dtypes
```

[532]:
```
End Time          object
Artist            object
Track             object
Playtime (Ms)    float64
dtype: object
```

[533]:
```
df_spotify_0['End Time'] = pd.to_datetime(df_spotify_0['End Time']).dt.
 ↪strftime('%Y-%m-%d %H:%M')
df_spotify_0.head(10)
```

[533]:
```
                  End Time                Artist  \
Track 1   2022-12-03 02:02  Cigarettes After Sex
Track 2   2022-12-03 02:02         Leonard Cohen
Track 3   2022-12-06 21:05          Vlad Holiday
Track 4   2022-12-06 21:05                 Lorde
Track 5   2022-12-06 21:05         Ariana Grande
Track 6   2022-12-07 00:21     Caroline Polachek
Track 7   2022-12-07 00:21     Kaizers Orchestra
Track 8   2022-12-07 00:21          Vlad Holiday
Track 9   2022-12-07 00:21          Vlad Holiday
Track 10  2022-12-07 00:21         Ariana Grande

                                  Track  Playtime (Ms)
Track 1                           Truly        30000.0
```

```
Track 2   Take This Waltz - Paris Version         8210.0
Track 3                  So Damn Into You        37895.0
Track 4                              Team         8984.0
Track 5                         Into You         1221.0
Track 6             Hit Me Where It Hurts         1648.0
Track 7                        Prosessen          348.0
Track 8                  So Damn Into You         1555.0
Track 9                  So Damn Into You         1486.0
Track 10                     Best Mistake         8824.0
```

[534]:
```
df_spotify_0['End Time'].min()
```

[534]: '2022-12-03 02:02'

[535]:
```
df_spotify_0['End Time'].max()
```

[535]: '2023-01-02 20:59'

1.5 Unique artists Find how many unique artists are in the dataset.

[536]:
```
len(df_spotify_0['Artist'].unique())
```

[536]: 495

1.6 Unique songs Find how many unique songs are in the dataset.

[537]:
```
len(df_spotify_0['Track'].unique())
```

[537]: 1308

1.3.1 Part 1: Questions

Q1: Which columns are in the dataset?

A1: End Time, Artist, Track and Playtime

Q2: What timeframe does the dataset span?

A2: From 3rd December 2022 until 2nd January 2023

Q3: How many unique artists are in the dataset?

A3: 495

Q4: How many unique songs are in the dataset?

A4: 1308

1.4 Part 2: Working with all the data

2.0 Importing all the dataframes In Task 1, you only worked with about a month worth of data. Now, you will work with over a year worth.

In the *spotify_data* folder, there is more than just one listening record. Load each of the 14 listening records into a dataframe (1 dataframe per listening record), and concatenate them together into one large dataframe named `df`.

[538]:
```
df_list = []

for i in range(14):
    dataframe = pd.DataFrame(pd.read_csv(f"spotify_data/streaminghistory{i}.
 ↪csv"))
    df_list.append(dataframe)

df = pd.concat(df_list, axis=0)
df
```

[538]:

	endTime	artistName	trackName	msPlayed
0	2022-12-03 02:02	Cigarettes After Sex	Truly	30000.0
1	2022-12-03 02:02	Leonard Cohen	Take This Waltz - Paris Version	8210.0
2	2022-12-06 21:05	Vlad Holiday	So Damn Into You	37895.0
3	2022-12-06 21:05	Lorde	Team	8984.0
4	2022-12-06 21:05	Ariana Grande	Into You	1221.0
…	…	…	…	…
11967	2023-12-07 21:13	Lana Del Rey	Art Deco	38298.0
11968	2023-12-07 21:13	Ariana Grande	off the table (with The Weeknd)	13448.0
11969	2023-12-07 21:14	Ariana Grande	my hair	23757.0
11970	2023-12-07 21:14	Leonard Cohen	Thanks for the Dance	9317.0
11971	2023-12-07 21:17	The Vaccines	Your Love Is My Favourite Band	14661.0

[167439 rows x 4 columns]

[539]:
```
df.shape[0]
```

[539]:
```
167439
```

2.1 Sorting by time Datasets often aren't perfect. One example of an issue that could occur is that the time-based data might not be in chronological order. If this were to happen, the rows in

your dataframe could be in the wrong order. To ensure this isn't an issue in your dataframe, you should sort the dataframe in chronological order, from oldest to newest.

[540]:
```
df.sort_values(by=['endTime'])
```

[540]:
```
                endTime            artistName  \
0      2022-12-03 02:02  Cigarettes After Sex
1      2022-12-03 02:02         Leonard Cohen
2      2022-12-06 21:05          Vlad Holiday
3      2022-12-06 21:05                 Lorde
4      2022-12-06 21:05         Ariana Grande
…                 …                     …
11968  2023-12-07 21:13         Ariana Grande
11970  2023-12-07 21:14         Leonard Cohen
11969  2023-12-07 21:14         Ariana Grande
11971  2023-12-07 21:17          The Vaccines
9062                NaN         The Lumineers

                             trackName  msPlayed
0                                Truly   30000.0
1      Take This Waltz - Paris Version    8210.0
2                    So Damn Into You   37895.0
3                                Team    8984.0
4                            Into You    1221.0
…                                  …       …
11968  off the table (with The Weeknd)   13448.0
11970             Thanks for the Dance    9317.0
11969                           my hair   23757.0
11971   Your Love Is My Favourite Band   14661.0
9062                           Ophelia     371.0

[167439 rows x 4 columns]
```

2.2 Setting a timeframe For this investigation, we are only interested in investigating listening patterns from **2023**. Remove any data not from **2023** from the DataFrame.

[541]:
```
date_limit = pd.to_datetime('2023-01-01')
df['endTime'] = pd.to_datetime(df['endTime'])
mask = df['endTime'] < date_limit
df = df[~mask]
df
```

[541]:
```
                  endTime      artistName                        trackName  \
10881 2023-01-01 01:17:00   Ariana Grande                          7 rings
10882 2023-01-01 01:17:00   Ariana Grande                          7 rings
10883 2023-01-01 01:17:00   Ariana Grande                        positions
10884 2023-01-01 01:17:00       Peach Pit                Being so Normal
10885 2023-01-01 01:17:00  Kelly Clarkson      Santa, Can't You Hear Me
```

```
…                    …             …                                  …
11967 2023-12-07 21:13:00    Lana Del Rey                          Art Deco
11968 2023-12-07 21:13:00   Ariana Grande  off the table (with The Weeknd)
11969 2023-12-07 21:14:00   Ariana Grande                          my hair
11970 2023-12-07 21:14:00   Leonard Cohen             Thanks for the Dance
11971 2023-12-07 21:17:00    The Vaccines   Your Love Is My Favourite Band

       msPlayed
10881     139.0
10882     487.0
10883     417.0
10884    2205.0
10885     278.0
…           …
11967   38298.0
11968   13448.0
11969   23757.0
11970    9317.0
11971   14661.0

[156558 rows x 4 columns]
```

2.3 Deleting rows Often in Data Science, you will encounter when a row entry has the value *NaN*, indicating missing data. These entries can skew your analysis, leading to inaccurate conclusions. For this task, identify and remove any rows in your DataFrame that contain NaN values. Later in the course, you might encounter other techniques of dealing with missing data, typically reffered to as *data imputation*. Here, though, you are just supposed to delete the entire rows with missing data.

[542]:
```
df = df.dropna()
df
```

[542]:
```
                   endTime      artistName                          trackName  \
10881 2023-01-01 01:17:00   Ariana Grande                            7 rings
10882 2023-01-01 01:17:00   Ariana Grande                            7 rings
10883 2023-01-01 01:17:00   Ariana Grande                          positions
10884 2023-01-01 01:17:00       Peach Pit                  Being so Normal
10885 2023-01-01 01:17:00  Kelly Clarkson         Santa, Can't You Hear Me
…                    …             …                                  …
11967 2023-12-07 21:13:00    Lana Del Rey                          Art Deco
11968 2023-12-07 21:13:00   Ariana Grande  off the table (with The Weeknd)
11969 2023-12-07 21:14:00   Ariana Grande                          my hair
11970 2023-12-07 21:14:00   Leonard Cohen             Thanks for the Dance
11971 2023-12-07 21:17:00    The Vaccines   Your Love Is My Favourite Band

       msPlayed
10881     139.0
```

```
10882     487.0
10883     417.0
10884    2205.0
10885     278.0
…           …
11967   38298.0
11968   13448.0
11969   23757.0
11970    9317.0
11971   14661.0

[156539 rows x 4 columns]
```

2.4 Convert from milliseconds to seconds From `msPlayed`, create a new column `secPlayed` with the data converted from milliseconds to seconds. Then delete the column `msPlayed`.

[543]:
```
df.loc[:, 'secPlayed'] = df['msPlayed'] / 1000
df = df.drop(columns=['msPlayed'])
df
```

```
C:\Users\kroel\AppData\Local\Temp\ipykernel_9020\3655342099.py:1:
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/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.loc[:, 'secPlayed'] = df['msPlayed'] / 1000
```

[543]:
```
                   endTime      artistName                          trackName  \
10881 2023-01-01 01:17:00   Ariana Grande                            7 rings
10882 2023-01-01 01:17:00   Ariana Grande                            7 rings
10883 2023-01-01 01:17:00   Ariana Grande                          positions
10884 2023-01-01 01:17:00       Peach Pit                    Being so Normal
10885 2023-01-01 01:17:00  Kelly Clarkson          Santa, Can't You Hear Me
…                     …             …                                  …
11967 2023-12-07 21:13:00    Lana Del Rey                           Art Deco
11968 2023-12-07 21:13:00   Ariana Grande  off the table (with The Weeknd)
11969 2023-12-07 21:14:00   Ariana Grande                           my hair
11970 2023-12-07 21:14:00   Leonard Cohen              Thanks for the Dance
11971 2023-12-07 21:17:00    The Vaccines   Your Love Is My Favourite Band

       secPlayed
10881      0.139
10882      0.487
10883      0.417
10884      2.205
10885      0.278
```

```
…          …
11967     38.298
11968     13.448
11969     23.757
11970      9.317
11971     14.661

[156539 rows x 4 columns]
```

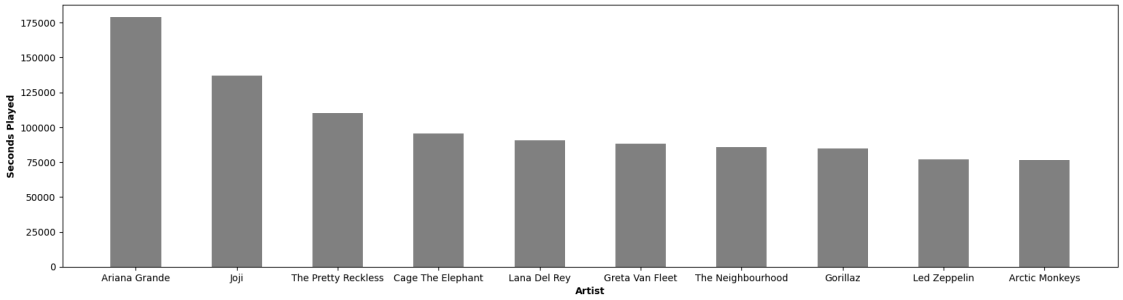
2.5 Finding top 10 favorite artists Find the top `ten` artists with the highest total play time (in seconds). Plot your findings in a bar graph.
(hint: start by creating a new DataFrame with only `artistName` and your time column. To proceed, you will also likely need the `groupby` command from Pandas.)

[544]:
```
artist_df = df[['artistName', 'secPlayed']]
artist_df = pd.DataFrame(artist_df.groupby('artistName')['secPlayed'].sum().
 ↪reset_index())
artist_df = artist_df.sort_values(by=['secPlayed'], ascending=False).head(10)
artist_df
```

[544]:
```
              artistName   secPlayed
50         Ariana Grande  178996.003
390                 Joji  137229.562
803  The Pretty Reckless  110293.430
135    Cage The Elephant   95587.575
443         Lana Del Rey   90543.113
316      Greta Van Fleet   88026.405
795    The Neighbourhood   85673.375
308             Gorillaz   84858.371
449         Led Zeppelin   77030.802
47         Arctic Monkeys   76444.236
```

1.4.1 Barplot:

[545]:
```
x = artist_df['artistName']
y = artist_df['secPlayed']

plt.figure(figsize=(20, 5))
plt.bar(x, y, width=0.5, color='grey')
plt.xlabel('Artist', fontweight='bold')
plt.ylabel('Seconds Played', fontweight='bold')
plt.show()
```



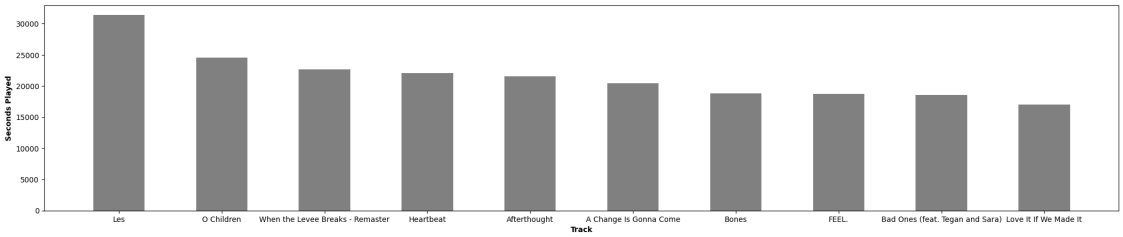
2.6 Finding top 10 favorite songs Find the top `ten` songs with the highest play time. Create a graph visualizing the results.

[546]:
```
song_df = df[['trackName', 'secPlayed']]
song_df = pd.DataFrame(song_df.groupby('trackName')['secPlayed'].sum().
 ↪reset_index())
song_df = song_df.sort_values(by=['secPlayed'], ascending=False).head(10)
song_df
```

[546]:

	trackName	secPlayed
2122	Les	31403.364
2671	0 Children	24558.414
4173	When the Levee Breaks - Remaster	22631.721
1566	Heartbeat	22056.629
149	Afterthought	21599.564
81	A Change Is Gonna Come	20414.317
487	Bones	18860.916
1161	FEEL.	18696.637
339	Bad Ones (feat. Tegan and Sara)	18558.247
2264	Love It If We Made It	17018.248

[547]:
```
x = song_df['trackName']
y = song_df['secPlayed']

plt.figure(figsize=(26, 5))
plt.bar(x, y, width=0.5, color='grey')
plt.xlabel('Track', fontweight='bold')
plt.ylabel('Seconds Played', fontweight='bold')
plt.show()
```



1.5 Part 3: Further analysis

3.0 Average listening time by hour Generate a plot that displays the average amount of time that music is played for each hour of the day.

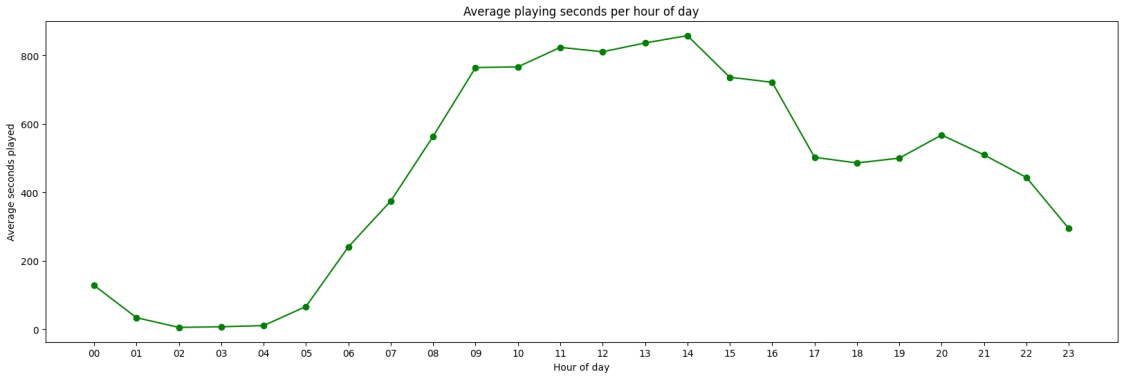
[548]:
```
no_of_days_listened = df['endTime'].dt.floor('D').nunique()
no_of_days_listened
```

[548]: 340

[549]:
```
df['endTime'].dt.hour.nunique()
```

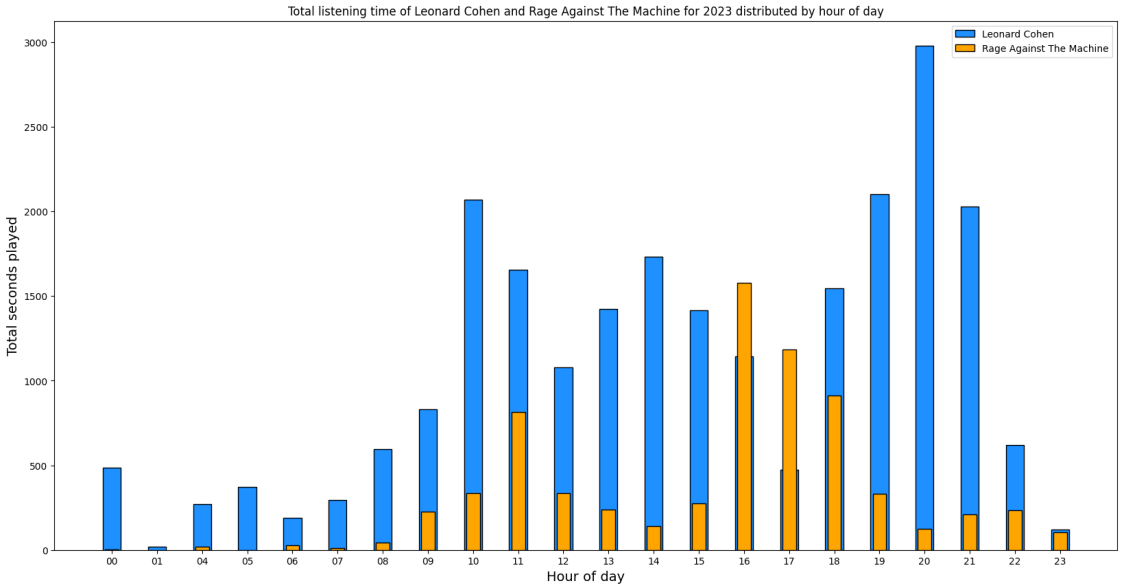
[549]: 24

[550]:
```
average_hour_df = pd.DataFrame(df.groupby(df['endTime'].dt.hour.apply(lambda x:␣
 ↪f'{x:02d}'))['secPlayed'].sum().reset_index())
average_hour_df['secPlayed'] = average_hour_df['secPlayed'] /␣
 ↪no_of_days_listened
average_hour_df.rename(columns={'endTime': 'Hour'}, inplace=True)
average_hour_df

plt.figure(figsize=(20, 6))
plt.plot(average_hour_df['Hour'], average_hour_df['secPlayed'], marker='o',␣
 ↪color='g')
plt.title('Average playing seconds per hour of day')
plt.xlabel('Hour of day')
plt.ylabel('Average seconds played')
plt.xticks(average_hour_df['Hour'])
plt.show()
```



3.1 Morning music and evening music I think many people find that some types of music are more suitable for morning listening and some music is more suitable for evening listening. Create a plot that compares the play time of the artists *Leonard Cohen* and *Rage Against the Machine* on an hour-by-hour basis. See if there are any differences.

[551]:
```
cohen_data = df[df['artistName'] == 'Leonard Cohen']
ratm_data = df[df['artistName'] == 'Rage Against The Machine']

cohen_hourly = cohen_data.groupby(cohen_data['endTime'].dt.hour)['secPlayed'].
 ↪sum().reset_index()
cohen_hourly['endTime'] = cohen_hourly['endTime'].apply(lambda x: f'{x:02d}')
ratm_hourly = ratm_data.groupby(ratm_data['endTime'].dt.hour)['secPlayed'].
 ↪sum().reset_index()
ratm_hourly['endTime'] = ratm_hourly['endTime'].apply(lambda x: f'{x:02d}')

plt.figure(figsize=(20, 10))
plt.bar(cohen_hourly['endTime'], cohen_hourly['secPlayed'], width=0.4,␣
 ↪color='dodgerblue', edgecolor='black', label='Leonard Cohen')
plt.bar(ratm_hourly['endTime'], ratm_hourly['secPlayed'], width=0.3,␣
 ↪color='orange', edgecolor='black', label='Rage Against The Machine')
plt.title('Total listening time of Leonard Cohen and Rage Against The Machine␣
 ↪for 2023 distributed by hour of day')
plt.xlabel('Hour of day', size=14)
plt.ylabel('Total seconds played', size=14)
plt.legend()
plt.show()
```



3.2 Analysing skipped songs Determining whether a song was skipped or listened to can be challenging. For this analysis, we'll simplify by defining a skipped song as any track played for less than 30 seconds. Conversely, a song played for 30 seconds or more is considered listened to.
Add a column to your DataFrame to reflect this criteria: set the value to 1 if the song was played for less than 30 seconds (indicating a skipped song), and 0 if it was played for 30 seconds or longer.

[552]:
```
df['skipped'] = (df['secPlayed'] < 30).astype(int)
df
```

[552]:

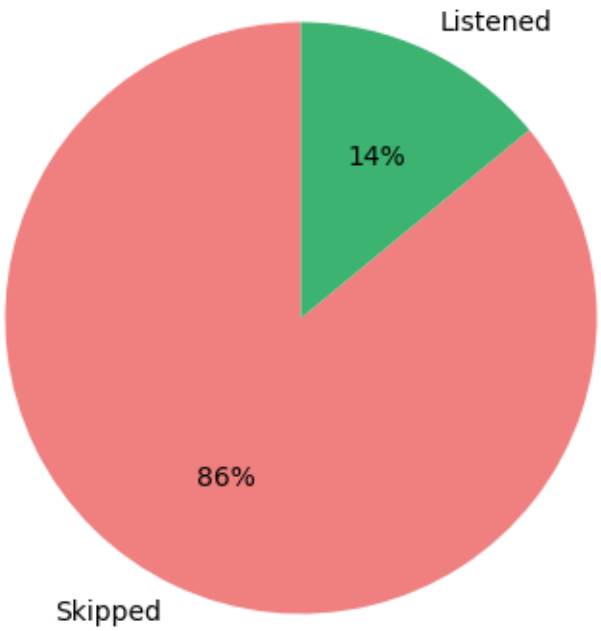
	endTime	artistName	trackName	\
10881	2023-01-01 01:17:00	Ariana Grande	7 rings	
10882	2023-01-01 01:17:00	Ariana Grande	7 rings	
10883	2023-01-01 01:17:00	Ariana Grande	positions	
10884	2023-01-01 01:17:00	Peach Pit	Being so Normal	
10885	2023-01-01 01:17:00	Kelly Clarkson	Santa, Can't You Hear Me	
…	…	…	…	
11967	2023-12-07 21:13:00	Lana Del Rey	Art Deco	
11968	2023-12-07 21:13:00	Ariana Grande	off the table (with The Weeknd)	
11969	2023-12-07 21:14:00	Ariana Grande	my hair	
11970	2023-12-07 21:14:00	Leonard Cohen	Thanks for the Dance	
11971	2023-12-07 21:17:00	The Vaccines	Your Love Is My Favourite Band	

	secPlayed	skipped
10881	0.139	1
10882	0.487	1
10883	0.417	1
10884	2.205	1

```
10885      0.278        1
…            …      …
11967     38.298        0
11968     13.448        1
11969     23.757        1
11970      9.317        1
11971     14.661        1

[156539 rows x 5 columns]
```

3.3 Plotting skipped songs Create a pie-chart that compares amount of skipped songs to amount of non-skipped songs.

[588]:
```
skipped_count = df['skipped'].sum()
listened_count = len(df) - skipped_count

labels = ['Skipped', 'Listened']
colors = ['lightcoral', 'mediumseagreen']
sizes = [skipped_count, listened_count]

plt.figure(figsize=(15, 5))
plt.pie(sizes, labels=labels, colors=colors, startangle=90, autopct=f'%1.0f%%')
plt.show()
```



3.4 Artists by percentage of songs skipped For each artist in the dataset, calculate which percentage of their songs was skipped. Store this information in a new DataFrame called `df_skipped`. Store the percentage of skipped songs in a new column named `SkipRate`

Example: If an artist has **100** songs in your dataset and **25** of these were skipped, the percentage of skipped songs for this artist would be $\frac{25}{100} = 25\%$

[554]:
```
df_songs_per_artist = df.groupby('artistName')['trackName'].count().
↪reset_index()
df_skipped_per_artist = df.groupby('artistName')['skipped'].sum().reset_index()
df_skipped = pd.merge(df_songs_per_artist, df_skipped_per_artist,␣
↪on='artistName')
df_skipped['skipRate'] = df_skipped['skipped'] / df_skipped['trackName'] * 100
df_skipped
```

[554]:

	artistName	trackName	skipped	skipRate
0	10cc	28	19	67.857143
1	2Pac	513	442	86.159844
2	3 Doors Down	2	1	50.000000
3	4 Non Blondes	122	88	72.131148
4	50 Cent	28	19	67.857143
..	…	…	…	…
951	squeeda	3	2	66.666667
952	tenkousei.	37	37	100.000000
953	trxxshed	2	1	50.000000
954	xander.	8	3	37.500000
955	Édith Piaf	155	146	94.193548

[956 rows x 4 columns]

[587]:
```
df_skipped.sort_values(by='trackName', ascending=False).head(15)
```

[587]:

	artistName	trackName	skipped	skipRate
50	Ariana Grande	19337	19246	99.529400
135	Cage The Elephant	4627	4082	88.221310
390	Joji	3611	2694	74.605372
803	The Pretty Reckless	3278	2570	78.401464
443	Lana Del Rey	3028	2480	81.902246
795	The Neighbourhood	3011	2472	82.098970
449	Led Zeppelin	2966	2527	85.198921
47	Arctic Monkeys	2623	2147	81.852840
308	Gorillaz	2535	1973	77.830375
316	Greta Van Fleet	2255	1700	75.388027
132	BØRNS	2239	1909	85.261277
211	Des Rocs	1960	1816	92.653061

```
157     Childish Gambino       1732     1431  82.621247
932             grandson       1536     1437  93.554688
473                Lorde       1531     1411  92.161986
```

3.5 Comparing artists by skip-rate Find the `three` top artists with the lowest skip-rate and the `three` with the highest. Print their names, along with their skip-rate.

[555]:
```
df_skipped.sort_values(by='skipRate', ascending=False).head(3)
```

[555]:
```
            artistName  trackName  skipped  skipRate
328     Hannah Montana         54       54     100.0
28   Alexander Stewart         47       47     100.0
560        No Vacation          2        2     100.0
```

[556]:
```
df_skipped.sort_values(by='skipRate').tail(3)
```

[556]:
```
         artistName  trackName  skipped  skipRate
290         G Mills         36       36     100.0
628           Ramón         10       10     100.0
417  Kelly Clarkson          3        3     100.0
```

1.6 Part 4: God Is a Data Scientist - The Ariana Deep-Dive

4.0 Ariana-DataFrame: Create a new DataFrame called *df_ariana*, containing only rows with music by Ariana Grande.

[557]:
```
df_ariana = df[df['artistName'] == 'Ariana Grande']
df_ariana
```

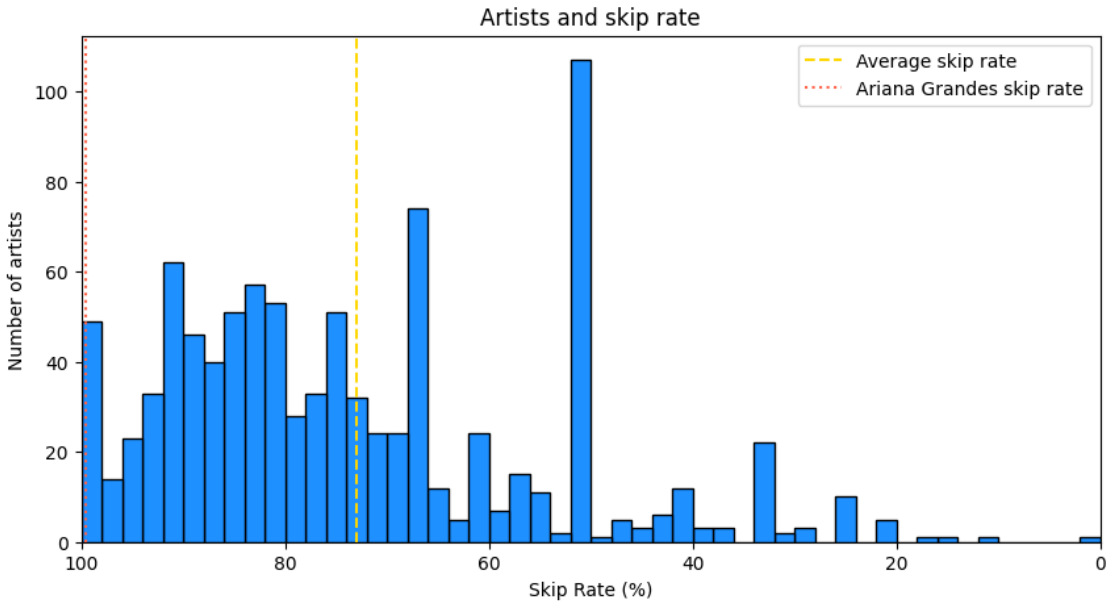
[557]:
```
                   endTime     artistName                          trackName  \
10881 2023-01-01 01:17:00  Ariana Grande                            7 rings
10882 2023-01-01 01:17:00  Ariana Grande                            7 rings
10883 2023-01-01 01:17:00  Ariana Grande                          positions
10887 2023-01-01 01:17:00  Ariana Grande                         Santa Baby
10888 2023-01-01 01:17:00  Ariana Grande     Right There (feat. Big Sean)
…                     …              …                                  …
11948 2023-12-07 17:46:00  Ariana Grande           Almost Is Never Enough
11955 2023-12-07 20:51:00  Ariana Grande                            needy
11961 2023-12-07 21:13:00  Ariana Grande                      pete davidson
11968 2023-12-07 21:13:00  Ariana Grande  off the table (with The Weeknd)
11969 2023-12-07 21:14:00  Ariana Grande                          my hair

       secPlayed  skipped
10881      0.139        1
10882      0.487        1
10883      0.417        1
10887     12.293        1
10888     22.929        1
```

```
…          …      …
11948     28.483        1
11955     26.220        1
11961      0.603        1
11968     13.448        1
11969     23.757        1

[19337 rows x 5 columns]
```

4.1 Average skip rate Create a histogram of the distribution of the skip-rate values of the different artists in your DataFrame `df_skipped`, with skip rates on one axis and number of artists on the other.

Then, retrieve the skip rate for Ariana Grande from your DataFrame `df_skipped`. Run the code in the cell below. Where on this distribution does Ariana Grande fall? Do I skip her songs more than average, or less?

[585]:
```
ariana_skip_rate = df_skipped[df_skipped['artistName'] == 'Ariana Grande']
mean_skip_rate = df_skipped['skipRate'].mean()

plt.figure(figsize=(10, 5))
plt.hist(df_skipped['skipRate'], bins=50, edgecolor='black', color='dodgerblue')
plt.axvline(mean_skip_rate, color='gold', linestyle='dashed')
plt.axvline(ariana_skip_rate.loc[50, 'skipRate'], color='tomato',␣
 ↪linestyle='dotted')
plt.xlim(100, 0)
plt.title('Artists and skip rate')
plt.xlabel('Skip Rate (%)', size=10)
plt.ylabel('Number of artists', size=10)
plt.legend(['Average skip rate', 'Ariana Grandes skip rate'])
plt.show()
```



[569]:
```
ariana_skip_rate
```

[569]:

	artistName	trackName	skipped	skipRate
50	Ariana Grande	19337	19246	99.5294

[570]:
```
mean_skip_rate
```

[570]: 73.04822293282288

1.6.1 Part 4: Questions

Q1: Did I skip a lot of Ariana Grande's songs, or did I not, compared to the rest of the dataset?

A1: Not only did you skip a lot of her songs; as shown in block 587, Ariana Grande was by far the artist you clicked play on the most times - only that you skipped 99.9% of these songs too. As the top 15 of the df_skipped dataframe sorted by 'trackName' shows, you clicked play on 19337 Ariana songs and skipped 19246 of them. Number 2 in the df_skipped dataframe is Cage the Elephant, by whom you played only 4627 songs and skipped 4082

Q2: What might be some possible reasons for Ariana Grande to be my nr.1 artist?

A2: As listed above, you clicked on Arianas songs 14710 more times than 2nd place on your list over most played artists. Probably there's som kind of formula / algorithm calculating top artists that considers the ratio skipped/played/total amount of songs in a way that favours the total amount of plays before skip rate when the numbers are as high as with Ariana here.