Spotify Analysis

Emma Johnson

Last Update: Nov. 11, 2025

Spotify Listening Analysis

This project will use my Extended Streaming History, which is available by request for any user of Spotify. See support.spotify.com for more detailed information on how to request, fetch, and understand your data.

PART 1: Load Libraries and Data

First, I'll start by loading the necessary packages I'll be using in this analysis. I'll note that a lot of the work will be done using `dplyr`, which is part of the `tidyverse` package.

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.2     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.1.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(jsonlite)
```

```
##
## Attaching package: 'jsonlite'
##
## The following object is masked from 'package:purrr':
##
##     flatten
```

```
full_data <- list.files(spotify_path,
                        pattern = "Streaming_History_Audio",
                        full.names = TRUE) %>%
  map_dfr(fromJSON) %>%
  suppressMessages()

glimpse(full_data)
```

```
## Rows: 176,214
## Columns: 23
## $ ts                                <chr> "2015-11-02T00:36:29Z", "2015-11-02T~
## $ platform                          <chr> "iOS 9.1 (iPhone6,1)", "iOS 9.1 (iPh~
## $ ms_played                         <int> 9798, 192381, 190595, 121533, 1904, ~
## $ conn_country                      <chr> "US", "US", "US", "US", "US", "US", ~
## $ ip_addr                           <chr> "174.25.247.192", "174.25.247.192", ~
## $ master_metadata_track_name        <chr> "Kanye - Steve Aoki & twoloud Remix"~
## $ master_metadata_album_artist_name <chr> "The Chainsmokers", "Daya", "Drake",~
## $ master_metadata_album_album_name  <chr> "Kanye", "Daya", "Right Hand", "Guts~
## $ spotify_track_uri                 <chr> "spotify:track:07wExbSybe0F6BtQjkjGS~
## $ episode_name                      <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ episode_show_name                 <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ spotify_episode_uri               <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ audiobook_title                   <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ audiobook_uri                     <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ audiobook_chapter_uri             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ audiobook_chapter_title           <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ reason_start                      <chr> "clickrow", "clickrow", "trackdone",~
## $ reason_end                        <chr> "endplay", "trackdone", "trackdone",~
## $ shuffle                           <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, F~
## $ skipped                           <lgl> TRUE, FALSE, FALSE, TRUE, TRUE, TRUE~
## $ offline                           <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, F~
## $ offline_timestamp                 <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ incognito_mode                    <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, F~
```

For the most part, the data types are compatible with the following analysis, except for the `ts` attribute. I want to coerce this to a dat format, so that'll be the first tiny step.

```
full_data$ts <- lubridate::ymd_hms(full_data$ts, tz = "UTC")
```

There appears to be a lot of `NA` values in my data, mostly because I am a frequent *music* listener, a less-frequent *podcast* listener, and basically a nonexistent *audiobook* listener.

So, the next step is to separate the data based on listening type. I'll also go through some additional cleaning to remove any super-incomplete data, as it serves meaningless in this analysis.

```
## Starting with audiobooks (smallest dataset):
# note that `audiobook_uri` is the unique identifier for audiobooks

# in base R:
audiobooks <- full_data[which(!is.na(full_data$audiobook_uri)), ]

# or

# using dplyr:
audiobooks <- full_data %>%
  filter(!is.na(audiobook_uri))

## Next, podcasts (larger than audiobooks, smaller than music):
# `spotify_episode_uri` is the unique identifier for podcasts

podcasts <- full_data[which(!is.na(full_data$spotify_episode_uri)), ]
```

```r
# or

podcasts <- full_data %>%
  filter(!is.na(spotify_episode_uri))

## Finally, music (largest dataset):
# simplest to assume music makes up the original set minus audiobooks and podcasts

music <- full_data[which(is.na(full_data$audiobook_uri) & is.na(full_data$spotify_episode_uri)),]

# or

music <- full_data %>%
  filter(is.na(audiobook_uri) & is.na(spotify_episode_uri))
```

Now we have datasets dedicated to the different listening types: `music`, `podcasts`, and `audiobooks`, but each holds unnecessary information about the others due to the preservation of *all* of the original columns.

For example, we do not need the information from columns 6 through 12 in the `audiobook` dataset. These columns (i.e. `master_metadata_track_name` through `spotify_episode_uri`) give no information on `audiobooks` and are full of `NA`s.

```r
glimpse(audiobooks)
```

```
## Rows: 7
## Columns: 23
## $ ts                                <dttm> 2025-03-02 15:31:16, 2025-03-02 15:~
## $ platform                          <chr> "ios", "ios", "ios", "ios", "ios", "~
## $ ms_played                         <int> 5210, 23070, 920281, 960, 1368564, 9~
## $ conn_country                      <chr> "US", "US", "US", "US", "US", "US", ~
## $ ip_addr                           <chr> "2600:387:f:5e12::b", "2600:387:f:5e~
## $ master_metadata_track_name        <chr> NA, NA, NA, NA, NA, NA, NA
## $ master_metadata_album_artist_name <chr> NA, NA, NA, NA, NA, NA, NA
## $ master_metadata_album_album_name  <chr> NA, NA, NA, NA, NA, NA, NA
## $ spotify_track_uri                 <chr> NA, NA, NA, NA, NA, NA, NA
## $ episode_name                      <chr> NA, NA, NA, NA, NA, NA, NA
## $ episode_show_name                 <chr> NA, NA, NA, NA, NA, NA, NA
## $ spotify_episode_uri               <chr> NA, NA, NA, NA, NA, NA, NA
## $ audiobook_title                   <chr> "Outliers", "Outliers", "Outliers", ~
## $ audiobook_uri                     <chr> "spotify:show:2ApiAL2eA0xUMpOw5SmZy3~
## $ audiobook_chapter_uri             <chr> "spotify:episode:7MURv1ukpmuUTdTvE4A~
## $ audiobook_chapter_title           <chr> "Track 10", "Track 7", "Track 8", "T~
## $ reason_start                      <chr> "clickrow", "clickrow", "clickrow", ~
## $ reason_end                        <chr> "endplay", "endplay", "endplay", "en~
## $ shuffle                           <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, F~
## $ skipped                           <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, ~
## $ offline                           <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, F~
## $ offline_timestamp                 <dbl> 1740929462, 1740929476, 1740929499, ~
## $ incognito_mode                    <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, F~
```

The `glimpse` function helps us identify which columns are useful for each dataset. Using `glimpse`, I found that columns 6 through 9 are specific to `music`, columns 10 through 12 are specific to `podcasts`, and columns 13 through 16 are specific to `audiobooks`.

The next step is to decide which columns to keep, and which to discard. We will do this uniquely for each dataset, according to our previous findings listed above.

```
## Removing unneccessary columns from audiobooks dataset:
audiobooks <- audiobooks[, -(6:12)]

## On to the podcasts data:
podcasts <- podcasts[, -c(6:9, 13:16)]

## Now the music data:
music <- music[, -(10:16)]
```

As a quick check, we now see that the `audiobooks` data is left with columns that describe this listening habit, removing the ones that offered no information.

```
glimpse(audiobooks)
```

```
## Rows: 7
## Columns: 16
## $ ts                       <dttm> 2025-03-02 15:31:16, 2025-03-02 15:31:39, 202~
## $ platform                 <chr> "ios", "ios", "ios", "ios", "ios", "ios", "ios"
## $ ms_played                <int> 5210, 23070, 920281, 960, 1368564, 9898, 105330
## $ conn_country             <chr> "US", "US", "US", "US", "US", "US", "US"
## $ ip_addr                  <chr> "2600:387:f:5e12::b", "2600:387:f:5e12::b", "1~
## $ audiobook_title          <chr> "Outliers", "Outliers", "Outliers", "Outliers"~
## $ audiobook_uri            <chr> "spotify:show:2ApiAL2eA0xUMpOw5SmZy3", "spotif~
## $ audiobook_chapter_uri    <chr> "spotify:episode:7MURv1ukpmuUTdTvE4AfUR", "spo~
## $ audiobook_chapter_title <chr> "Track 10", "Track 7", "Track 8", "Track 9", "~
## $ reason_start             <chr> "clickrow", "clickrow", "clickrow", "clickrow"~
## $ reason_end               <chr> "endplay", "endplay", "endplay", "endplay", "e~
## $ shuffle                  <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE
## $ skipped                  <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE
## $ offline                  <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE
## $ offline_timestamp        <dbl> 1740929462, 1740929476, 1740929499, 1740930475~
## $ incognito_mode           <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE
```

1.1: Music Data

I'm not overly interested in all of the features in the `music` dataset so far, so I'll clean it up a bit more to focus on the features I'm interested in analyzing.

To be specific, I will discard the following columns: `platform`, `conn_country`, `ip_addr`, `offline`, `offline_timestamp`, and `incognito_mode`.

```
music <- music %>% select(-c('platform', 'conn_country', 'ip_addr', 'offline', 'offline_timestamp', 'in

glimpse(music)
```

```
## Rows: 174,664
## Columns: 10
## $ ts                            <dttm> 2015-11-02 00:36:29, 2015-11-02 00:~
## $ ms_played                     <int> 9798, 192381, 190595, 121533, 1904, ~
```

```
## $ master_metadata_track_name        <chr> "Kanye - Steve Aoki & twoloud Remix"~
## $ master_metadata_album_artist_name <chr> "The Chainsmokers", "Daya", "Drake",~
## $ master_metadata_album_album_name  <chr> "Kanye", "Daya", "Right Hand", "Guts~
## $ spotify_track_uri                 <chr> "spotify:track:07wExbSybe0F6BtQjkjGS~
## $ reason_start                      <chr> "clickrow", "clickrow", "trackdone",~
## $ reason_end                        <chr> "endplay", "trackdone", "trackdone",~
## $ shuffle                           <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, F~
## $ skipped                           <lgl> TRUE, FALSE, FALSE, TRUE, TRUE, TRUE~
```

1.2: Podcasts Data

For the `podcasts` data, I'll remove the same features. Namely, `platform`, `conn_country`, `ip_addr`, `offline`, `offline_timestamp`, and `incognito_mode`.

```
podcasts <- podcasts %>% select(-c('platform', 'conn_country', 'ip_addr', 'offline', 'offline_timestamp

glimpse(podcasts)
```

```
## Rows: 1,543
## Columns: 9
## $ ts                  <dttm> 2018-05-27 18:42:36, 2018-05-27 19:00:37, 2018-09~
## $ ms_played           <int> 56960, 1075264, 7244, 538032, 335466, 38139, 38613~
## $ episode_name        <chr> "Decoding Our Emotions", "How Art Changes Us", "Fa~
## $ episode_show_name   <chr> "TED Radio Hour", "TED Radio Hour", "Arthouse Lege~
## $ spotify_episode_uri <chr> "spotify:episode:0uukdwBghtGTypI28wuupS", "spotify~
## $ reason_start        <chr> "clickrow", "clickrow", "appload", "clickrow", "cl~
## $ reason_end          <chr> "endplay", "endplay", "endplay", "endplay", "logou~
## $ shuffle             <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, F~
## $ skipped             <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, F~
```

1.3: Audiobooks Data

Finally, I'll do the same for the `audiobooks` data, removing the same columns.

```
audiobooks <- audiobooks %>% select(-c('platform', 'conn_country', 'ip_addr', 'offline', 'offline_times

glimpse(audiobooks)
```

```
## Rows: 7
## Columns: 10
## $ ts                       <dttm> 2025-03-02 15:31:16, 2025-03-02 15:31:39, 202~
## $ ms_played                <int> 5210, 23070, 920281, 960, 1368564, 9898, 105330
## $ audiobook_title          <chr> "Outliers", "Outliers", "Outliers", "Outliers"~
## $ audiobook_uri            <chr> "spotify:show:2ApiAL2eA0xUMpOw5SmZy3", "spotif~
## $ audiobook_chapter_uri    <chr> "spotify:episode:7MURv1ukpmuUTdTvE4AfUR", "spo~
## $ audiobook_chapter_title <chr> "Track 10", "Track 7", "Track 8", "Track 9", "~
## $ reason_start             <chr> "clickrow", "clickrow", "clickrow", "clickrow"~
## $ reason_end               <chr> "endplay", "endplay", "endplay", "endplay", "e~
## $ shuffle                  <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE
## $ skipped                  <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE
```

PART 2: Analyzing Music Listening Habits

Now that the data is cleaned, I can begin analyzing my listening habits. I'll start with the `music` dataset, replicating findings from Spotify Wrapped and developing my own.

2.1: Total Time Spent Listening

So far, each song is recorded by milliseconds played. As far as I'm concerned, the total amount of time spent listening is not overly interesting when recorded in ms. We'll convert these to minutes, hours, and days for easier interpretation.

I want to draw attention to the nature of this data. It has been recorded over 10 years, so calculating the total time spent listening to music many also be interesting to analyze as a portion of a year.

```
music %>%
  summarise(total_ms_played = sum(ms_played)) %>%
  mutate(total_mins_played = total_ms_played / 60000,
         total_hours_played = total_mins_played / 60,
         total_days_played = total_hours_played / 24,
         total_years_played = total_days_played / 365)
```

```
##   total_ms_played total_mins_played total_hours_played total_days_played
## 1     21875483368          364591.4           6076.523          253.1885
##   total_years_played
## 1           0.693667
```

Interpreting these results, it appears that I've spent a total of over 253 days listening to music on Spotify over the past decade. This equates to over 2/3 of a year, meaning if I started a timer right now, it would take over 8 months of continuous listening to match the amount of time I've spent listening to music on Spotify!

2.1.1: Creating a Function for Total Time Calculation Looking ahead, I'll want to recycle some of this code for the `podcasts` and `audiobooks` datasets, so I'll save the general framework as a function to use later. Note that the `total_years_played` will be left out, since I assume the amount of time spent listening to `podcasts` and `audiobooks` will be far less than `music`.

```
total_time <- function(df) {
  return(df %>%
    summarise(total_ms_played = sum(ms_played)) %>%
    mutate(total_mins_played = total_ms_played / 60000,
           total_hours_played = total_mins_played / 60,
           total_days_played = total_hours_played / 24)
  )
}
```

2.2: Top Songs

Since this data is recorded over such a long period of time, there are many different approaches we could take to classify "top" songs. For example, does a top song mean it has the most plays? What about the most minutes spent listening? Should it be broken up by year or month, or simply over the whole course of being a Spotify user? Clearly, there are many different combinations and angles to take when deciding on this metric. Since the results may vary, I'll go over some different ways we can find the "top" songs.

2.2.1: Top 10 Songs by Total Minutes Played Our first decision will be made by considering the songs with the most minutes spent listening over the entire course of being a Spotify user. This means that the results will show how many minutes total I've spent listening to each song over the past decade.

```
music %>%
  group_by(track = master_metadata_track_name,
           artist = master_metadata_album_artist_name) %>%
  summarise(total_mins = sum(ms_played) / 60000, .groups = 'drop') %>%
  arrange(desc(total_mins)) %>%
  head(10)
```

```
## # A tibble: 10 x 3
##    track                artist           total_mins
##    <chr>                <chr>                 <dbl>
##  1 Nights               Frank Ocean           1010.
##  2 GHOST!               Kid Cudi               751.
##  3 Super Rich Kids      Frank Ocean            744.
##  4 Nikes                Frank Ocean            702.
##  5 Pink + White         Frank Ocean            668.
##  6 Me and Your Mama     Childish Gambino       658.
##  7 Ivy                  Frank Ocean            639.
##  8 Devil In A New Dress Kanye West             619.
##  9 With Me              dvsn                   616.
## 10 Flashing Lights      Kanye West             602.
```

2.2.2: Top 10 Songs by Total Plays Consider how some songs are much longer than others. The length of a song may inflate the total amount of time spent listening to it when considered a top song. For example, If one song is 2 minutes long and another is 4 minutes long, the former song can be played twice in the same amount of time it takes to listen to the latter song once.

We could also use the frequency of which a song appears in the data as a metric for top songs. We will call this frequency per song the total amount of `plays`. Again, we will consider this over the full course of the data.

```
music %>%
  group_by(track = master_metadata_track_name,
           artist = master_metadata_album_artist_name) %>%
  summarise(plays = n(), .groups = 'drop') %>%
  arrange(desc(plays)) %>%
  head(10)
```

```
## # A tibble: 10 x 3
##    track                     artist       plays
##    <chr>                     <chr>        <int>
##  1 Sundress                  A$AP Rocky     303
##  2 Nights                    Frank Ocean    281
##  3 Pink + White              Frank Ocean    279
##  4 Love Is Only a Feeling    Joey Bada$$    272
##  5 SLOW DANCING IN THE DARK  Joji           242
##  6 Ivy                       Frank Ocean    241
##  7 Revenge                   XXXTENTACION   241
##  8 Nikes                     Frank Ocean    234
##  9 L$D                       A$AP Rocky     225
## 10 White Ferrari             Frank Ocean    222
```

Note that this interpreting this metric does not mean that a songs with 300 plays is played in full 300 times. It could be played for 10 seconds and then skipped. We could filter our data further to only consider songs that are played all the way through.

Using the `reason_end` feature, we can discover the different reasons why an individual observation for a song ends and starts recording data for another. The different levels of this feature are as follows:

```
levels(as.factor(music$reason_end))
```

```
##  [1] "backbtn"                      "endplay"
##  [3] "fwdbtn"                       "logout"
##  [5] "remote"                       "trackdone"
##  [7] "trackerror"                   "unexpected-exit"
##  [9] "unexpected-exit-while-paused" "unknown"
```

Next, we will only consider that data for songs that play all the way through by filtering the `reason_end` feature to only include `trackdone`.

```
music %>%
  filter(reason_end == "trackdone") %>%
  group_by(track = master_metadata_track_name,
           artist = master_metadata_album_artist_name) %>%
  summarise(plays = n(), .groups = 'drop') %>%
  arrange(desc(plays)) %>%
  head(10)
```

```
## # A tibble: 10 x 3
##    track                                                          artist   plays
##    <chr>                                                          <chr>    <int>
##  1 Pink + White                                                   Frank O~   206
##  2 Sundress                                                       A$AP Ro~   185
##  3 Nights                                                         Frank O~   181
##  4 Love Is Only a Feeling                                         Joey Ba~   173
##  5 Revenge                                                        XXXTENT~   171
##  6 SLOW DANCING IN THE DARK                                       Joji       148
##  7 Avalon                                                         $uicide~   147
##  8 Flashing Lights                                                Kanye W~   142
##  9 Godspeed                                                       Frank O~   142
## 10 If Self-Destruction Was an Olympic Event, I'd Be Tonya Harding $uicide~   139
```

Notice how the results are different this time around. This highlights the importance of a data scientist's knowledge of the data when making decisions.

2.3: Top Artists

Similar to the problem of how to classify a top song, we run into the issue of how to classify a top artist. Again, there are different ways to approach this, but we can use the same reasoning as before.

2.3.1: Top 10 Artists by Total Minutes Played The only thing we need to modify to go from top songs to top artists is our grouping variables. Before, we grouped by both `track` and `artist`, but now we will only group by `artist`.

```r
music %>%
  group_by(artist = master_metadata_album_artist_name) %>%
  summarise(total_mins = sum(ms_played) / 60000, .groups = 'drop') %>%
  arrange(desc(total_mins)) %>%
  head(10)
```

```
## # A tibble: 10 x 2
##    artist             total_mins
##    <chr>                   <dbl>
##  1 Frank Ocean            11565.
##  2 Kanye West             10572.
##  3 Drake                   9196.
##  4 $uicideboy$             8216.
##  5 Travis Scott            7726.
##  6 Kid Cudi                6662.
##  7 Mac Miller              6021.
##  8 Tyler, The Creator      5851.
##  9 The Weeknd              5487.
## 10 Tame Impala             5409.
```

2.3.2: Top 10 Artists by Total Plays This time, I will disregard the `reason_end` feature, as I am not concerned with how it plays a role in counting plays for artists. Note that you could easily incorporate this feature as a filter, similarly to how we did it for songs.

```r
music %>%
  group_by(artist = master_metadata_album_artist_name) %>%
  summarise(plays = n(), .groups = 'drop') %>%
  arrange(desc(plays)) %>%
  head(10)
```

```
## # A tibble: 10 x 2
##    artist             plays
##    <chr>              <int>
##  1 Frank Ocean         4457
##  2 Kanye West          4294
##  3 $uicideboy$         4239
##  4 Drake               4073
##  5 Travis Scott        3342
##  6 XXXTENTACION        2681
##  7 Tyler, The Creator  2620
##  8 Post Malone         2614
##  9 Kid Cudi            2412
## 10 A$AP Rocky          2236
```

At a glance, there is not much of a change in the lineups based on the two different methods. However, the order of the artists does change, which is interesting to note. This again highlights the importance of understanding the data and how different metrics can lead to different interpretations of what is "top".

2.4: Top Albums

Finally, we will reuse our train of thought to classify top albums.

2.4.1: Top 10 Albums by Total Minutes Played This time around, `master_metadata_album_album_name` (aliased as `album`) and `master_metadata_album_artist_name` (aliased as `artist`) will be our grouping variables.

```
music %>%
  group_by(album = master_metadata_album_album_name,
           artist = master_metadata_album_artist_name) %>%
  summarise(total_mins = sum(ms_played) / 60000, .groups = 'drop') %>%
  arrange(desc(total_mins)) %>%
  head(10)
```

```
## # A tibble: 10 x 3
##    album                                artist             total_mins
##    <chr>                                <chr>                   <dbl>
##  1 Blonde                               Frank Ocean             5666.
##  2 channel ORANGE                       Frank Ocean             3664.
##  3 ASTROWORLD                           Travis Scott            3550.
##  4 Currents                             Tame Impala             2484.
##  5 My Beautiful Dark Twisted Fantasy    Kanye West              2286.
##  6 In Utero                             Nirvana                 1977.
##  7 Heaven Or Hell                       Don Toliver             1942.
##  8 Graduation                           Kanye West              1937.
##  9 BALLADS 1                            Joji                    1931.
## 10 IGOR                                 Tyler, The Creator      1921.
```

2.4.2: Top 10 Albums by Total Plays Note that for total plays in this case, plays will be recorded at the level of songs from an album versus a total play of the entire album as a whole.

When it comes to top albums, it is difficult to filter the data to ensure that it is in sequential order for listening to an album from start to finish. This is an important note, as the total plays can be inflated if songs from an album are played out of order, or if only a few songs from an album are played repeatedly. Similarly, I could listen to one song from an album frequently over a long period of time, but never listen to other songs from that album. therefore, this analysis takes careful consideration.

We will first recycle some code and look at the frequency of plays through a familiar lens. I will only consider when `reason_end` is recorded as `trackdone` to assume that songs are played in full when listening to an album cover to cover.

```
music %>%
  filter(reason_end == "trackdone") %>%
  group_by(album = master_metadata_album_album_name,
           artist = master_metadata_album_artist_name) %>%
  summarise(plays = n(), .groups = 'drop') %>%
  arrange(desc(plays)) %>%
  head(10)
```

```
## # A tibble: 10 x 3
##    album                          artist            plays
##    <chr>                          <chr>             <int>
##  1 Blonde                         Frank Ocean        1280
##  2 ASTROWORLD                     Travis Scott        980
##  3 channel ORANGE                 Frank Ocean         830
##  4 Long Term Effects of SUFFERING $uicideboy$         694
```

```
##  5 Heaven Or Hell                Don Toliver          594
##  6 BALLADS 1                     Joji                 593
##  7 Currents                      Tame Impala          580
##  8 ?                             XXXTENTACION         526
##  9 In Utero                      Nirvana              496
## 10 IGOR                          Tyler, The Creator   472
```

There are a couple of underlying problems with choosing top albums this way. I'll begin by presenting a problematic example. Say I have 2 albums, one with 10 songs and another with 20 songs. If the albums are around an hour each, I will listen to twice the amount of plays in the latter situation than the former, because it has more songs. Then, the final tally will be inflated, even though I listened to each album only once within a given hour.

These issues are important to be aware of, and should be considered when interpreting the results. If I wanted to find the true frequency of which I listened to an album from the first to the last song, I'd need to reference extra data on each album that I do not have here. For this reason, we will halt the album analysis here.

2.5: Listening Over the Years

As mentioned earlier, we could also split the data up by year to narrow our vision from a decade-long view to a year-by-year view. First, let's see which years are included in the `full_data`.

```
unique(year(music$ts))
```

```
##  [1] 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025
```

We can see that the data spans from 2015 to 2025. Since I requested this data in the middle of 2025, the data for the year 2025 is incomplete. Similarly, the data for 2015 begins later in that year, since I started listening later than 01/01/2015. Therefore, I'm not going to consider these years for analysis, since they cannot be analyzed with complete information.

I will choose the year 2024 to conduct a single years analysis on. Note that you could change this to accommodate for any other year in the data, if desired.

```
music_2024 <- music %>%
  filter(lubridate::year(ts) == 2024)
```

Now, we can repeat the same steps as before to find the total time spent listening to music, top songs, and top artists in 2024. Note that I will include only one method per category for brevity, but you could easily modify the earlier code to match other results. The key difference in the code is the swap of the `music` data for the `music_2024` data.

```


total_time(music_2024)
```

2.5.1: Total Time Spent Listening in 2024

```
##   total_ms_played total_mins_played total_hours_played total_days_played
## 1      2729868712          45497.81           758.2969           31.5957
```

```r
music_2024 %>%
  group_by(track = master_metadata_track_name,
           artist = master_metadata_album_artist_name) %>%
  summarise(total_mins = sum(ms_played) / 60000, .groups = 'drop') %>%
  arrange(desc(total_mins)) %>%
  head(10)
```

2.5.2: Top 10 Songs in 2024 by Total Minutes Played

```
## # A tibble: 10 x 3
##    track                          artist           total_mins
##    <chr>                          <chr>                 <dbl>
##  1 EXCALIBUR                      DUCKBOY                192.
##  2 Whatsername                    Green Day              151.
##  3 GHOST!                         Kid Cudi               141.
##  4 Crumbled                       ThxSoMch               136.
##  5 Heart Racing                   Kanii                  130.
##  6 Not Even Ghosts Are This Empty $uicideboy$            122.
##  7 You and I                      d4vd                   120.
##  8 Alive                          Pearl Jam              113.
##  9 stain                          Max Fry                112.
## 10 Me and Your Mama               Childish Gambino       112.
```

```r
music_2024 %>%
  group_by(artist = master_metadata_album_artist_name) %>%
  summarise(plays = n(), .groups = 'drop') %>%
  arrange(desc(plays)) %>%
  head(10)
```

2.5.3: Top 10 Artists in 2024 by Total Plays

```
## # A tibble: 10 x 2
##    artist             plays
##    <chr>              <int>
##  1 $uicideboy$          897
##  2 Tame Impala          334
##  3 Tory Lanez           330
##  4 Kanye West           320
##  5 Mac Miller           295
##  6 Nirvana              283
##  7 Riovaz               275
##  8 Tyler, The Creator   268
##  9 ThxSoMch             248
## 10 d4vd                 235
```

```
music_2024 %>%
  filter(reason_end == "trackdone") %>%
  group_by(album = master_metadata_album_album_name,
           artist = master_metadata_album_artist_name) %>%
  summarise(plays = n(), .groups = 'drop') %>%
  arrange(desc(plays)) %>%
  head(10)
```

2.5.4: Top 10 Albums in 2024 by Total Minutes Played

```
## # A tibble: 10 x 3
##    album                                              artist       plays
##    <chr>                                              <chr>        <int>
##  1 Currents                                           Tame Impala    132
##  2 New World Depression                               $uicideboy$     91
##  3 Alone At Prom                                      Tory Lanez      90
##  4 existential hymns for the average sigma [vol. 9] DUCKBOY         90
##  5 Petals to Thorns                                   d4vd            79
##  6 In Utero                                           Nirvana         65
##  7 Circles                                            Mac Miller      61
##  8 Larger Than Life                                   Brent Faiyaz    60
##  9 Long Term Effects of SUFFERING                     $uicideboy$     60
## 10 CHASE                                              Aaron May       58
```

Exploring these results, I see one in particular that stands out. the album, CHASE by Aaron May is one where I have a favorite song I listen to frequently, but I have never listened to the entire album cover to cover. This highlights the earlier point about the difficulty of analyzing albums without more information.

This suggests a couple ways around this for further discussion. I could've created a minimum count for the amount of unique songs that must appear in my history in order to classify an album as a top album.

For example, say most albums have around 10 songs. I could set a minimum of 7 unique songs that must appear in my listening history to classify an album as a top album. This would help filter out albums where I only listen to one or two songs repeatedly.

PART 3: Analyzing Podcast Listening Habits

3.1: Total Time Spent Listening

```
total_time(podcasts)
```

```
##   total_ms_played total_mins_played total_hours_played total_days_played
## 1      1658997246          27649.95           460.8326          19.20136
```

PART 4: Analyzing Audiobook Listening Habits

4.1: Total Time Spent Listening

```r
total_time(audiobooks)
```

```
##   total_ms_played total_mins_played total_hours_played total_days_played
## 1         2433313           40.55522          0.6759203        0.02816334
```