Assignment 2 – Spotify Dataset cleaning and Transformation

Sneha Jayachandran (23979684)
Undram Chinges (24498014)
Pavithra Devi Balashanmugam (24251484)
Muhammad Aariful Islam (24584192)
Saloni Saloni (24557371)

2025-04-29

Contents

1. Load Libraries and Read Data

```r
library(readr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(naniar)
```

```
## Warning: package 'naniar' was built under R version 4.4.3
```

```
spotify_df <- read_csv("Spotify dataset.csv")
```

```
## New names:
## * `` -> `...1`
```

```
## Rows: 114000 Columns: 21
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (5): track_id, artists, album_name, track_name, track_genre
## dbl (15): ...1, popularity, duration_ms, danceability, energy, key, loudness...
## lgl  (1): explicit
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
original_rows <- nrow(spotify_df)
```

2. Initial Data Exploration

```
glimpse(spotify_df)
```

```
## Rows: 114,000
## Columns: 21
## $ ...1             <dbl> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,~
## $ track_id         <chr> "5SuOikwiRyPMVoIQDJUgSV", "4qPNDBW1i3p13qLCt0Ki3A", "~
## $ artists          <chr> "Gen Hoshino", "Ben Woodward", "Ingrid Michaelson;ZAY~
## $ album_name       <chr> "Comedy", "Ghost (Acoustic)", "To Begin Again", "Craz~
## $ track_name       <chr> "Comedy", "Ghost - Acoustic", "To Begin Again", "Can'~
## $ popularity       <dbl> 73, 55, 57, 71, 82, 58, 74, 80, 74, 56, 74, 69, 52, 6~
## $ duration_ms      <dbl> 230666, 149610, 210826, 201933, 198853, 214240, 22940~
## $ explicit         <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALS~
## $ danceability     <dbl> 0.676, 0.420, 0.438, 0.266, 0.618, 0.688, 0.407, 0.70~
## $ energy           <dbl> 0.4610, 0.1660, 0.3590, 0.0596, 0.4430, 0.4810, 0.147~
## $ key              <dbl> 1, 1, 0, 0, 2, 6, 2, 11, 0, 1, 8, 4, 7, 3, 2, 4, 2, 1~
## $ loudness         <dbl> -6.746, -17.235, -9.734, -18.515, -9.681, -8.807, -8.~
## $ mode             <dbl> 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,~
## $ speechiness      <dbl> 0.1430, 0.0763, 0.0557, 0.0363, 0.0526, 0.1050, 0.035~
## $ acousticness     <dbl> 0.0322, 0.9240, 0.2100, 0.9050, 0.4690, 0.2890, 0.857~
## $ instrumentalness <dbl> 1.01e-06, 5.56e-06, 0.00e+00, 7.07e-05, 0.00e+00, 0.0~
## $ liveness         <dbl> 0.3580, 0.1010, 0.1170, 0.1320, 0.0829, 0.1890, 0.091~
## $ valence          <dbl> 0.7150, 0.2670, 0.1200, 0.1430, 0.1670, 0.6660, 0.076~
## $ tempo            <dbl> 87.917, 77.489, 76.332, 181.740, 119.949, 98.017, 141~
## $ time_signature   <dbl> 4, 4, 4, 3, 4, 4, 3, 4, 4, 4, 4, 3, 4, 4, 4, 3, 4, 4,~
## $ track_genre      <chr> "acoustic", "acoustic", "acoustic", "acoustic", "acou~
```

```
str(spotify_df)
```

```
## spc_tbl_ [114,000 x 21] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ...1             : num [1:114000] 0 1 2 3 4 5 6 7 8 9 ...
##  $ track_id         : chr [1:114000] "5SuOikwiRyPMVoIQDJUgSV" "4qPNDBW1i3p13qLCt0Ki3A" "1iJBSr7s7jYXzl
##  $ artists          : chr [1:114000] "Gen Hoshino" "Ben Woodward" "Ingrid Michaelson;ZAYN" "Kina Grann
```

```
##  $ album_name      : chr [1:114000] "Comedy" "Ghost (Acoustic)" "To Begin Again" "Crazy Rich Asians
##  $ track_name      : chr [1:114000] "Comedy" "Ghost - Acoustic" "To Begin Again" "Can't Help Falling
##  $ popularity      : num [1:114000] 73 55 57 71 82 58 74 80 74 56 ...
##  $ duration_ms     : num [1:114000] 230666 149610 210826 201933 198853 ...
##  $ explicit        : logi [1:114000] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ danceability    : num [1:114000] 0.676 0.42 0.438 0.266 0.618 0.688 0.407 0.703 0.625 0.442 ...
##  $ energy          : num [1:114000] 0.461 0.166 0.359 0.0596 0.443 0.481 0.147 0.444 0.414 0.632 ...
##  $ key             : num [1:114000] 1 1 0 0 2 6 2 11 0 1 ...
##  $ loudness        : num [1:114000] -6.75 -17.23 -9.73 -18.52 -9.68 ...
##  $ mode            : num [1:114000] 0 1 1 1 1 1 1 1 1 1 ...
##  $ speechiness     : num [1:114000] 0.143 0.0763 0.0557 0.0363 0.0526 0.105 0.0355 0.0417 0.0369 0.0
##  $ acousticness    : num [1:114000] 0.0322 0.924 0.21 0.905 0.469 0.289 0.857 0.559 0.294 0.426 ...
##  $ instrumentalness: num [1:114000] 1.01e-06 5.56e-06 0.00 7.07e-05 0.00 0.00 2.89e-06 0.00 0.00 4.1
##  $ liveness        : num [1:114000] 0.358 0.101 0.117 0.132 0.0829 0.189 0.0913 0.0973 0.151 0.0735
##  $ valence         : num [1:114000] 0.715 0.267 0.12 0.143 0.167 0.666 0.0765 0.712 0.669 0.196 ...
##  $ tempo           : num [1:114000] 87.9 77.5 76.3 181.7 119.9 ...
##  $ time_signature  : num [1:114000] 4 4 4 3 4 4 3 4 4 4 ...
##  $ track_genre     : chr [1:114000] "acoustic" "acoustic" "acoustic" "acoustic" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   ...1 = col_double(),
##   ..   track_id = col_character(),
##   ..   artists = col_character(),
##   ..   album_name = col_character(),
##   ..   track_name = col_character(),
##   ..   popularity = col_double(),
##   ..   duration_ms = col_double(),
##   ..   explicit = col_logical(),
##   ..   danceability = col_double(),
##   ..   energy = col_double(),
##   ..   key = col_double(),
##   ..   loudness = col_double(),
##   ..   mode = col_double(),
##   ..   speechiness = col_double(),
##   ..   acousticness = col_double(),
##   ..   instrumentalness = col_double(),
##   ..   liveness = col_double(),
##   ..   valence = col_double(),
##   ..   tempo = col_double(),
##   ..   time_signature = col_double(),
##   ..   track_genre = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```r
head(spotify_df)
```

```
## # A tibble: 6 x 21
##    ...1 track_id   artists album_name track_name popularity duration_ms explicit
##   <dbl> <chr>      <chr>   <chr>      <chr>           <dbl>       <dbl> <lgl>
## 1     0 5SuOikwiR~ Gen Ho~ Comedy     Comedy             73      230666 FALSE
## 2     1 4qPNDBW1i~ Ben Wo~ Ghost (Ac~ Ghost - A~         55      149610 FALSE
## 3     2 1iJBSr7s7~ Ingrid~ To Begin ~ To Begin ~         57      210826 FALSE
## 4     3 6lfxq3CG4~ Kina G~ Crazy Ric~ Can't Hel~         71      201933 FALSE
## 5     4 5vjLSffim~ Chord ~ Hold On    Hold On            82      198853 FALSE
## 6     5 01MVOl9Kt~ Tyrone~ Days I Wi~ Days I Wi~         58      214240 FALSE
```

```
## # i 13 more variables: danceability <dbl>, energy <dbl>, key <dbl>,
## #   loudness <dbl>, mode <dbl>, speechiness <dbl>, acousticness <dbl>,
## #   instrumentalness <dbl>, liveness <dbl>, valence <dbl>, tempo <dbl>,
## #   time_signature <dbl>, track_genre <chr>
```

```r
summary(spotify_df)
```

```
##       ...1          track_id           artists           album_name       
##  Min.   :     0   Length:114000      Length:114000      Length:114000     
##  1st Qu.: 28500   Class :character   Class :character   Class :character  
##  Median : 57000   Mode  :character   Mode  :character   Mode  :character  
##  Mean   : 57000                                                           
##  3rd Qu.: 85499                                                           
##  Max.   :113999                                                           
##   track_name          popularity      duration_ms       explicit      
##  Length:114000      Min.   :  0.00   Min.   :      0   Mode :logical  
##  Class :character   1st Qu.: 17.00   1st Qu.: 174066   FALSE:104253   
##  Mode  :character   Median : 35.00   Median : 212906   TRUE :9747     
##                     Mean   : 33.24   Mean   : 228029                  
##                     3rd Qu.: 50.00   3rd Qu.: 261506                  
##                     Max.   :100.00   Max.   :5237295                  
##   danceability        energy             key            loudness      
##  Min.   :0.0000   Min.   :0.0000   Min.   : 0.000   Min.   :-49.531  
##  1st Qu.:0.4560   1st Qu.:0.4720   1st Qu.: 2.000   1st Qu.:-10.013  
##  Median :0.5800   Median :0.6850   Median : 5.000   Median : -7.004  
##  Mean   :0.5668   Mean   :0.6414   Mean   : 5.309   Mean   : -8.259  
##  3rd Qu.:0.6950   3rd Qu.:0.8540   3rd Qu.: 8.000   3rd Qu.: -5.003  
##  Max.   :0.9850   Max.   :1.0000   Max.   :11.000   Max.   :  4.532  
##       mode         speechiness       acousticness    instrumentalness
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.0000   Min.   :0.00e+00  
##  1st Qu.:0.0000   1st Qu.:0.03590   1st Qu.:0.0169   1st Qu.:0.00e+00  
##  Median :1.0000   Median :0.04890   Median :0.1690   Median :4.16e-05  
##  Mean   :0.6376   Mean   :0.08465   Mean   :0.3149   Mean   :1.56e-01  
##  3rd Qu.:1.0000   3rd Qu.:0.08450   3rd Qu.:0.5980   3rd Qu.:4.90e-02  
##  Max.   :1.0000   Max.   :0.96500   Max.   :0.9960   Max.   :1.00e+00  
##     liveness         valence           tempo        time_signature 
##  Min.   :0.0000   Min.   :0.0000   Min.   :  0.00   Min.   :0.000  
##  1st Qu.:0.0980   1st Qu.:0.2600   1st Qu.: 99.22   1st Qu.:4.000  
##  Median :0.1320   Median :0.4640   Median :122.02   Median :4.000  
##  Mean   :0.2136   Mean   :0.4741   Mean   :122.15   Mean   :3.904  
##  3rd Qu.:0.2730   3rd Qu.:0.6830   3rd Qu.:140.07   3rd Qu.:4.000  
##  Max.   :1.0000   Max.   :0.9950   Max.   :243.37   Max.   :5.000  
##  track_genre       
##  Length:114000     
##  Class :character  
##  Mode  :character  
##                    
##                    
## 
```

3. Audio Feature Descriptions

Below is a table summarizing key audio features used in the Spotify dataset:

Feature	Description
`speechiness`	Detects spoken words; higher = more speech-like.
`acousticness`	Likelihood that a track is acoustic.
`instrumentalness`	Predicts absence of vocals.
`liveness`	Likelihood the track is live.
`valence`	Musical positivity from 0 (sad) to 1 (happy).
`tempo`	Beats per minute.
`time_signature`	Beats per bar (e.g., 3, 4).

4. Data Cleaning

```r
# Remove the redundant first column (named '...1')
spotify_df <- spotify_df %>% select(-`...1`)

# Check number of duplicate rows
sum(duplicated(spotify_df))
```

```
## [1] 450
```

```r
# View the duplicate rows (if any)
spotify_df[duplicated(spotify_df), ]
```

```
## # A tibble: 450 x 20
##    track_id        artists album_name track_name popularity duration_ms explicit
##    <chr>           <chr>   <chr>      <chr>           <dbl>       <dbl> <lgl>
##  1 0CDucx9lKxuCZp~ Buena ~ Disco 2    Song for ~         16      219346 FALSE
##  2 2aibwv5hGXSgw7~ Red Ho~ Stadium A~ Snow (Hey~         80      334666 FALSE
##  3 7mULVp0DJrI2Nd~ Joy Di~ Timeless ~ Love Will~          0      204621 FALSE
##  4 6d3RIvHfVkoOtW~ Little~ Serenity   Margot             27       45714 FALSE
##  5 481beimUiUnMUz~ SUPER ~    / ~                 54      255080 FALSE
##  6 57QCT8L3kFQiN4~ Dimmu ~ Spiritual~ The Blazi~         22      277093 FALSE
##  7 3RIeNvIwO6ZRML~ Whored~ Whoredom ~ Gitt Til ~         17      344466 FALSE
##  8 6FmQMYlxRl56FI~ Whored~ Pakt       En Lenke ~         16      396210 FALSE
##  9 6laLzRAweIOHEO~ Alison~ Live       Tiny Brok~         25      188400 FALSE
## 10 5Aq43Qi1j27J5K~ Old & ~ Breakdown  Working O~         21      155733 FALSE
## # i 440 more rows
## # i 13 more variables: danceability <dbl>, energy <dbl>, key <dbl>,
## #   loudness <dbl>, mode <dbl>, speechiness <dbl>, acousticness <dbl>,
## #   instrumentalness <dbl>, liveness <dbl>, valence <dbl>, tempo <dbl>,
## #   time_signature <dbl>, track_genre <chr>
```

```r
# Remove exact duplicates
spotify_df <- spotify_df %>% distinct()

# Find duplicated track_ids
duplicate_track_ids <- spotify_df %>%
  group_by(track_id) %>%
  filter(n() > 1) %>%
  arrange(track_id)

print(duplicate_track_ids)
```

```
## # A tibble: 40,108 x 20
## # Groups:   track_id [16,299]
```

```
##    track_id        artists album_name track_name popularity duration_ms explicit
##    <chr>           <chr>   <chr>      <chr>           <dbl>       <dbl> <lgl>
##  1 001APMDOl3qtx1~ Pink S~ New RnB    Better              0      176320 FALSE
##  2 001APMDOl3qtx1~ Pink S~ New RnB    Better              0      176320 FALSE
##  3 001YQlnDSduXd5~ Soda S~ Soda Ster~ El Tiempo~         38      177266 FALSE
##  4 001YQlnDSduXd5~ Soda S~ Soda Ster~ El Tiempo~         38      177266 FALSE
##  5 003vvx7Niy0yvh~ The Ki~ Hot Fuss   Mr. Brigh~         86      222973 FALSE
##  6 003vvx7Niy0yvh~ The Ki~ Hot Fuss   Mr. Brigh~         86      222973 FALSE
##  7 003vvx7Niy0yvh~ The Ki~ Hot Fuss   Mr. Brigh~         86      222973 FALSE
##  8 004h8smbIoAkUN~ Ouse;P~ Loners Di~ Lovemark           58      219482 TRUE
##  9 004h8smbIoAkUN~ Ouse;P~ Loners Di~ Lovemark           58      219482 TRUE
## 10 006rHBBNLJMpQs~ Calcin~ CP 25 Ano~ Agora Est~         47      260510 FALSE
## # i 40,098 more rows
## # i 13 more variables: danceability <dbl>, energy <dbl>, key <dbl>,
## #   loudness <dbl>, mode <dbl>, speechiness <dbl>, acousticness <dbl>,
## #   instrumentalness <dbl>, liveness <dbl>, valence <dbl>, tempo <dbl>,
## #   time_signature <dbl>, track_genre <chr>
```

```r
# Count how many track_ids are duplicated
spotify_df %>% count(track_id) %>% filter(n > 1) %>% nrow()
```

```
## [1] 16299
```

```r
# See list of duplicates by frequency
spotify_df %>% count(track_id, sort = TRUE) %>% filter(n > 1)
```

```
## # A tibble: 16,299 x 2
##    track_id                   n
##    <chr>                  <int>
##  1 6S3JlDAGk3uu3NtZbPnuhS     9
##  2 2Ey6v4Sekh3Z0RUSISRosD     8
##  3 2kkvB3RNRzwjFdGhaUA0tz     8
##  4 08kTa3SL9sV6Iy8KLKtGql     7
##  5 0RSGPiykniIg8m7JhiAVv7     7
##  6 0YLSjVxSb5FT1Bo8Tnxr8j     7
##  7 0e5LcankE0UyJUuCoq1uH2     7
##  8 1Gqpa08T7eBAvPQj9O9L2Q     7
##  9 2aaClnypAakdAmLw74JXxB     7
## 10 2qgXrzJsry4KgYoJCpuaul     7
## # i 16,289 more rows
```

```r
# Keep only most popular version of each track_id
spotify_df <- spotify_df %>%
  group_by(track_id) %>%
  slice_max(order_by = popularity, n = 1, with_ties = FALSE) %>%
  ungroup()

# Confirm no more duplicated track_ids
spotify_df %>% count(track_id) %>% filter(n > 1) %>% nrow()
```

```
## [1] 0
```

```r
#converting all Latino and Latin to Latin
spotify_df$track_genre <- ifelse(spotify_df$track_genre %in% c("latino", "latin"), "latin", spotify_df$t
```

5. Data Loss Calculation

```
cleaned_rows <- nrow(spotify_df)
rows_removed <- original_rows - cleaned_rows
percent_lost <- (rows_removed / original_rows) * 100

cat("Rows removed:", rows_removed, "\n")
```

```
## Rows removed: 24259
```

```
cat("Percentage of data lost:", round(percent_lost, 2), "%\n")
```

```
## Percentage of data lost: 21.28 %
```

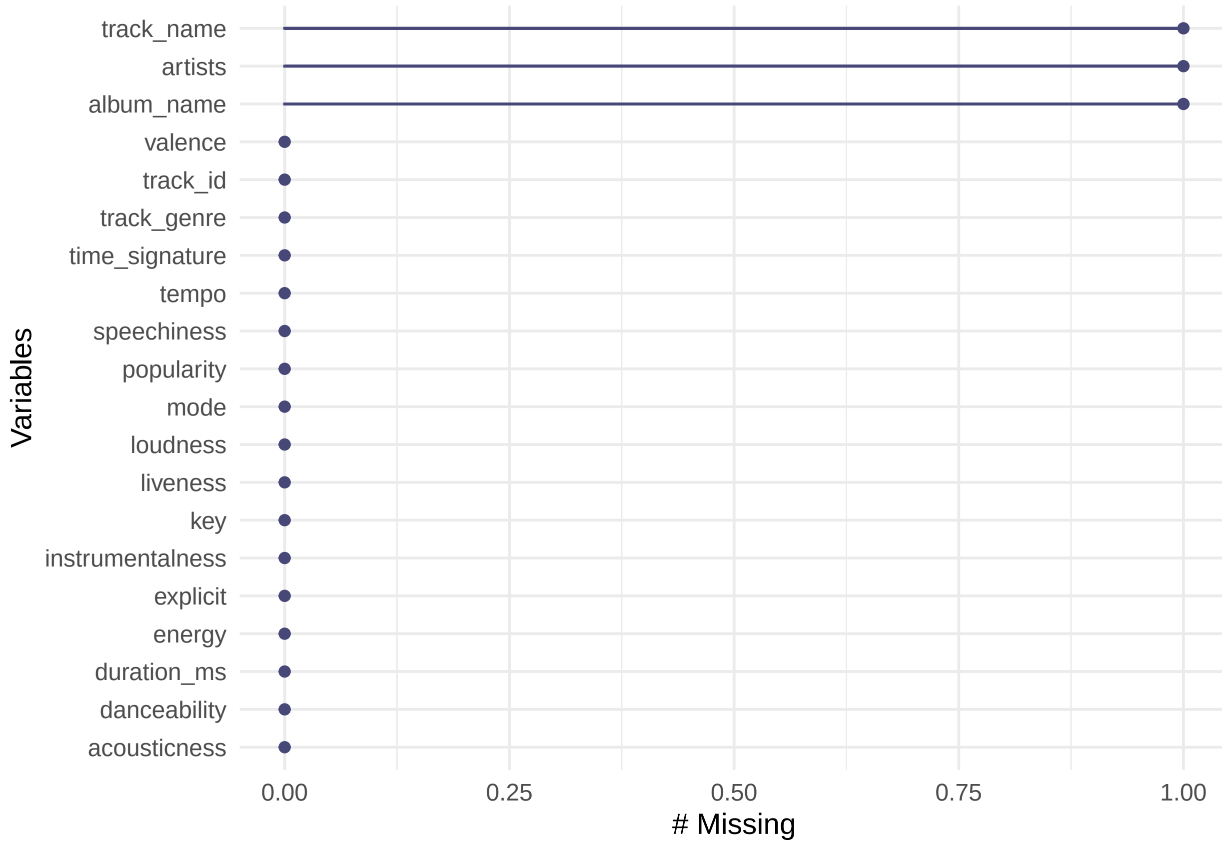
Although 21.28% of the dataset was removed, the remaining 89,000+ rows still represent a rich dataset free of redundancy.

6. Missing Values

```
colSums(is.na(spotify_df))
```

```
##         track_id          artists       album_name       track_name
##                0                1                1                1
##       popularity      duration_ms         explicit     danceability
##                0                0                0                0
##           energy              key         loudness             mode
##                0                0                0                0
##      speechiness     acousticness instrumentalness         liveness
##                0                0                0                0
##          valence            tempo   time_signature      track_genre
##                0                0                0                0
```

```
gg_miss_var(spotify_df)
```




```
missing_rows <- spotify_df %>% filter(if_any(everything(), is.na))
print(missing_rows)
```

```
## # A tibble: 1 x 20
##   track_id         artists album_name track_name popularity duration_ms explicit
##   <chr>            <chr>   <chr>      <chr>           <dbl>       <dbl> <lgl>
## 1 1kR4gIb7nGxHPI3~ <NA>    <NA>       <NA>                0           0 FALSE
## # i 13 more variables: danceability <dbl>, energy <dbl>, key <dbl>,
## #   loudness <dbl>, mode <dbl>, speechiness <dbl>, acousticness <dbl>,
## #   instrumentalness <dbl>, liveness <dbl>, valence <dbl>, tempo <dbl>,
## #   time_signature <dbl>, track_genre <chr>
```

```
spotify_df <- spotify_df %>%
  filter(!(is.na(track_name) & is.na(artists) & is.na(album_name)))
```

To clean the Spotify dataset, we first checked which columns had missing information. We then used a visual chart to quickly see where the most gaps were. Next, we looked at rows that were partly empty and removed any that had no track name, artist, or album—since these didn't provide any useful details. We also found some rows where the word "NULL" or "null" was typed instead of real values. These were treated as missing and cleaned accordingly to keep the dataset accurate and usable.

7. Data Transformation

```
unique(spotify_df$explicit)
```

```
## [1]  TRUE FALSE
```

```r
summary(spotify_df$valence)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.2490  0.4570  0.4695  0.6820  0.9950
```

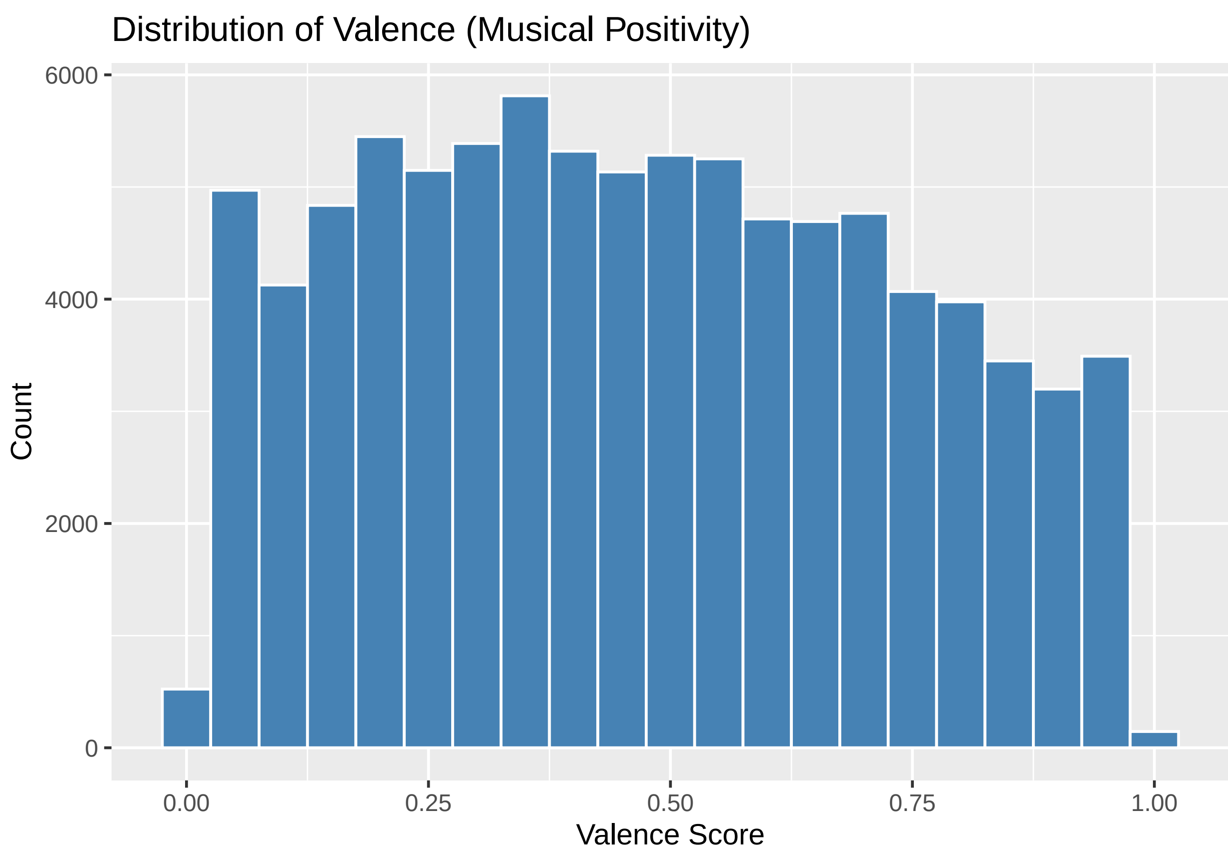
```r
any(is.na(spotify_df$valence))
```

```
## [1] FALSE
```

```r
range(spotify_df$valence)
```

```
## [1] 0.000 0.995
```

```r
# Plot valence distribution
ggplot(spotify_df, aes(x = valence)) +
  geom_histogram(binwidth = 0.05, fill = "steelblue", color = "white") +
  labs(title = "Distribution of Valence (Musical Positivity)",
       x = "Valence Score",
       y = "Count")
```



8. Feature Engineering

```r
# Mood feature
spotify_df$mood <- case_when(
  spotify_df$valence < 0.3 ~ "Sad",
  spotify_df$valence < 0.6 ~ "Neutral",
  TRUE ~ "Happy"
```

```r
)

# Vibe feature
spotify_df <- spotify_df %>%
  mutate(vibe = case_when(
    valence > 0.6 & energy > 0.6 ~ "Energetic & Happy",
    valence > 0.6 & energy <= 0.6 ~ "Chill & Happy",
    valence <= 0.6 & energy > 0.6 ~ "Energetic & Intense",
    TRUE ~ "Chill & Sad"
  ))
```

9. Genre Exploration

```r
unique_genres <- unique(spotify_df$track_genre)
print(unique_genres)
```

```
##   [1] "german"            "club"              "minimal-techno"
##   [4] "hip-hop"           "comedy"            "chill"
##   [7] "punk-rock"         "bluegrass"         "happy"
##  [10] "drum-and-bass"     "idm"               "alt-rock"
##  [13] "emo"               "honky-tonk"        "industrial"
##  [16] "j-dance"           "grindcore"         "french"
##  [19] "world-music"       "hard-rock"         "forro"
##  [22] "j-pop"             "indian"            "children"
##  [25] "j-rock"            "power-pop"         "pagode"
##  [28] "blues"             "romance"           "study"
##  [31] "afrobeat"          "black-metal"       "grunge"
##  [34] "opera"             "show-tunes"        "heavy-metal"
##  [37] "k-pop"             "progressive-house" "acoustic"
##  [40] "anime"             "ambient"           "dubstep"
##  [43] "iranian"           "singer-songwriter" "synth-pop"
##  [46] "chicago-house"     "kids"              "disco"
##  [49] "pop-film"          "alternative"       "gospel"
##  [52] "mandopop"          "jazz"              "swedish"
##  [55] "tango"             "funk"              "latin"
##  [58] "piano"             "spanish"           "turkish"
##  [61] "salsa"             "electronic"        "goth"
##  [64] "dance"             "malay"             "death-metal"
##  [67] "trance"            "rock"              "country"
##  [70] "hardstyle"         "folk"              "mpb"
##  [73] "electro"           "disney"            "j-idol"
##  [76] "hardcore"          "british"           "punk"
##  [79] "guitar"            "dub"               "deep-house"
##  [82] "r-n-b"             "psych-rock"        "rockabilly"
##  [85] "reggaeton"         "brazil"            "metalcore"
##  [88] "indie-pop"         "dancehall"         "trip-hop"
##  [91] "metal"             "house"             "party"
##  [94] "breakbeat"         "sleep"             "detroit-techno"
##  [97] "samba"             "garage"            "groove"
## [100] "sertanejo"         "reggae"            "sad"
## [103] "ska"               "techno"            "classical"
## [106] "rock-n-roll"       "soul"              "cantopop"
## [109] "new-age"           "edm"               "indie"
```

```
## [112] "pop"
```

```r
length(unique_genres)
```

```
## [1] 112
```

10. Separating the Primary Artist and Secondary Artists

Some tracks list multiple artists separated by semicolons. For clarity and better visualization, we separate the first-listed artist into a new column called `artist_primary`.

```r
library(dplyr)
library(tidyr)

spotify_df<- spotify_df%>%
  separate(artists, into = c("artist_primary", "artist_others"), sep = ";", extra = "merge", fill = "ri




# Define all potential fake-missing values
fake_nulls <- c("NULL", "null", "", "NA", "N/A", "na", "n/a", "undefined", "missing", "-")

# Filter rows that contain any of these in any column (case-insensitive)
possible_missing_rows <- spotify_df[apply(spotify_df, 1, function(row) {
  any(tolower(trimws(row)) %in% tolower(fake_nulls))
}), ]

# View or inspect them
print(possible_missing_rows)
```

```
## # A tibble: 7 x 23
##   track_id          artist_primary artist_others album_name track_name popularity
##   <chr>             <chr>          <chr>         <chr>      <chr>           <dbl>
## 1 13D3WWX3nsF3kCv~ Angels of Lib~ N/A           Telepathi~ Leda               21
## 2 4OxvaRRodfF83n9~ White Noise f~ <NA>          Rising Sun Missing             0
## 3 4QPONdBX2hybq3v~ Angels of Lib~ N/A           Telepathi~ Sophia             22
## 4 4RzY9ZLhrqDwsbT~ Twinkz         <NA>          undefined  undefined          46
## 5 55XrFge2dEZwYzN~ itssvd         Shiloh Dynas~ Missing    Losing In~         76
## 6 5COfkrLpT6t2fHl~ Ado            <NA>          missing    missing            55
## 7 6ncsVXsY8FaxpK5~ itssvd         Shiloh Dynas~ Missing    Love Again         62
## # i 17 more variables: duration_ms <dbl>, explicit <lgl>, danceability <dbl>,
## #   energy <dbl>, key <dbl>, loudness <dbl>, mode <dbl>, speechiness <dbl>,
## #   acousticness <dbl>, instrumentalness <dbl>, liveness <dbl>, valence <dbl>,
## #   tempo <dbl>, time_signature <dbl>, track_genre <chr>, mood <chr>,
## #   vibe <chr>
```

11. Retaining Only the Primary Artist

We drop `artist_others` as it is mostly `NA` and rename `artist_primary` back to `artists`.

```r
spotify_df <- spotify_df %>%
  select(-artist_others) %>%
  rename(artists = artist_primary)
```

12. Detecting Non-English Languages in Text Columns

We apply language detection to `artists`, `track_name`, and `album_name` to quantify non-English entries.

```
# Install and load cld3 (only install once)
if (!require(cld3)) install.packages("cld3")
```

```
## Loading required package: cld3

## Warning: package 'cld3' was built under R version 4.4.3
```

```
library(cld3)

df<-spotify_df

# Detect languages in each text column
df$artist_lang <- cld3::detect_language(df$artists)
df$track_lang <- cld3::detect_language(df$track_name)
df$album_lang <- cld3::detect_language(df$album_name)

# Count how many are NOT English
non_en_artists <- sum(df$artist_lang != "en", na.rm = TRUE)
non_en_tracks <- sum(df$track_lang != "en", na.rm = TRUE)
non_en_albums <- sum(df$album_lang != "en", na.rm = TRUE)

# Print counts
cat("Artists with non-English language:", non_en_artists, "\n")
```

```
## Artists with non-English language: 47840
```

```
cat("Track names with non-English language:", non_en_tracks, "\n")
```

```
## Track names with non-English language: 43547
```

```
cat("Albums with non-English language:", non_en_albums, "\n")
```

```
## Albums with non-English language: 42086
```

```
# Optional: show most common detected languages
cat("\nTop artist languages:\n")
```

```
##
## Top artist languages:
```

```
print(sort(table(df$artist_lang), decreasing = TRUE))
```

```
##
##      en      la      es      de      no      pt      it      nl hi-Latn      af
##    6011    2632    2299    1838    1654    1560    1527    1467    1296    1238
##      fy      gl      lb      sr      ms      jv      cy      sn      fi      sv
##    1098    1074    1060    1018    1011     958     923     826     794     794
##      et      eo      ga el-Latn      fr      ha      mg      ca      su ru-Latn
##     725     723     721     705     694     693     683     677     658     646
##      hu      ht      da      tr      xh      id      sl      lt      pl      az
##     641     618     572     572     568     564     519     508     496     484
##      bs      zu      gd     fil      so      mt      eu      zh      ny      sw
##     477     465     446     445     431     428     379     374     360     351
##      lv ja-Latn      ku      uz     ceb      hr      yo      cs      ig      ja
##     350     345     327     292     290     271     271     250     250     246
```

```
##      ro bg-Latn      uk      co      bg      is      st     hmn      sm      be
##     234     233     233     215     207     204     201     192     166     154
##      ru      kk      ky      vi      mi      sq      sk      mk zh-Latn      mn
##     148     135     133     120     105     101      95      93      85      72
##     haw      tg      ko      ar      fa      sd      el      ur
##      65      41      18       2       2       2       1       1
```

```r
cat("\nTop track name languages:\n")
```

```
##
## Top track name languages:
```

```r
print(sort(table(df$track_lang), decreasing = TRUE))
```

```
##
##      en      pt      es      de      zh      gl      ja      it      la      no
##   18026    4055    3329    2080    1882    1772    1757    1376    1275    1143
##      fr      nl      sv      fy      sr      lb      ca      af      tr      fi
##    1050    1031     958     923     904     851     807     715     664     567
##      mg      da      sn hi-Latn      eo      et      jv      ht      so      cy
##     566     564     560     559     550     543     537     498     469     433
##      ga      ha      ru      id     fil     ceb      su      xh      gd el-Latn
##     433     415     398     361     344     342     339     336     333     330
##      pl      mt ru-Latn      lt      ms      zu      sl      hu     haw      ig
##     326     321     319     318     315     279     267     254     253     245
##      ny      bg      bs      ky      co      eu      mi ja-Latn      uk      az
##     245     242     217     215     207     199     187     182     182     181
##      yo      hr      sw      uz      sk      cs      sm      st bg-Latn      ro
##     177     169     163     157     156     152     146     139     126     125
##      lv      ku      be      mk      is      kk zh-Latn      vi      tg      sq
##     124     123     111     107      98      84      82      64      63      54
##      mn     hmn      ko      ne      fa      hi      el      mr      sd      ur
##      42      36      14       7       5       5       4       2       2       2
##      ar      hy      iw      ps      th
##       1       1       1       1       1
```

```r
cat("\nTop album name languages:\n")
```

```
##
## Top album name languages:
```

```r
print(sort(table(df$album_lang), decreasing = TRUE))
```

```
##
##      en      es      pt      de      gl      zh      la      it      ja      nl
##   18121    4082    3763    2126    1835    1529    1479    1471    1419    1330
##      fr      no      af      ca      sr      sv      fy      lb      tr      ru
##    1287    1065     976     959     916     886     761     715     609     560
##      sn      da     ceb      eo      ht      fi      jv      cy hi-Latn      et
##     552     546     531     509     501     488     476     442     400     384
##      xh      mg      id      mt      ha      so      ga      bg      pl      lt
##     369     364     324     324     318     316     296     291     291     290
## ru-Latn el-Latn      zu      su      sl      ms      hu      gd     fil      az
##     288     285     285     268     257     249     245     241     229     213
##      ky      ny      ig      uk      yo      sk     haw      co      bs      hr
##     202     199     180     178     170     166     163     161     157     149
##      eu      sm      uz      be bg-Latn ja-Latn      lv      cs      sw      ro
```

```
##      144      143      143      125      117      116      113      108      101      100
##       st       kk       mk       ku       mi      hmn       is  zh-Latn       tg       sq
##       96       84       82       81       78       72       65       63       46       43
##       mn       vi       ko       ne       el       fa       mr       ml       sd       hi
##       27       25       13       13        7        6        3        2        2        1
##       th       ur
##        1        1
```

13. Handling Non-English Content in the Dataset

A substantial portion of the dataset includes non-English content:

- Artists with non-English language: **47,841**
- Track names with non-English language: **43,548**
- Albums with non-English language: **42,087**

This means that **nearly half of the dataset** includes multilingual content. To standardize and make it easier to read and visualize, we use **transliteration** to convert names from other scripts into the Latin alphabet (e.g., Chinese → "Chen Yixun").

14. Transliteration

We apply transliteration to all relevant columns for a consistent phonetic representation.

```
# Install and load cld3 (for language detection)
if (!require(cld3)) install.packages("cld3")
library(cld3)

# Step 1: Detect language from track_name (best signal for language)
spotify_df$language <- detect_language(spotify_df$track_name)

# Step 2: Transliterate artists, track names, and album names
library(stringi)
library(stringr)

spotify_df$artists      <- str_to_title(str_squish(stri_trans_general(spotify_df$artists, "Latin-ASCII"
spotify_df$track_name   <- str_to_title(str_squish(stri_trans_general(spotify_df$track_name, "Latin-ASC
spotify_df$album_name   <- str_to_title(str_squish(stri_trans_general(spotify_df$album_name, "Latin-ASC


# Convert all artist names to Latin ASCII-friendly characters
spotify_df$artists_transliterated <- stringi::stri_trans_general(spotify_df$artists, "Latin-ASCII")
spotify_df$artists <- stringi::stri_trans_general(spotify_df$artists, "Latin-ASCII")
```

15. Extracting Top Tracks from the Most Popular Genres

We select five trending genres and extract their top five most popular tracks based on popularity score.

```
# Define your top genres
top_genres <- c("k-pop", "pop-film", "metal", "chill", "latino")

# Filter top tracks per genre
top_tracks <- spotify_df %>%
  filter(track_genre %in% top_genres) %>%
  arrange(desc(popularity)) %>%
  group_by(track_genre) %>%
```

```
  slice_max(order_by = popularity, n = 5) %>%
  ungroup()
```

16. Adding Spotify URLs to the Top Tracks

We create valid Spotify links using each track's unique `track_id`.

```
# Apply the function to your top tracks
top_tracks$spotify_url <- paste0("https://open.spotify.com/track/", top_tracks$track_id)


# Apply the function to your top tracks
spotify_df$spotify_url <- paste0("https://open.spotify.com/track/", spotify_df$track_id)
```

17. Studying the top 50 popular tracks on Spotify

```
# Step 1: Get Top 50 tracks by popularity
top_50_tracks <- spotify_df %>%
  arrange(desc(popularity)) %>%
  distinct(track_id, .keep_all = TRUE) %>%  # ensures unique tracks
  slice_head(n = 50)

# Step 2: View the genres among those top 50
top_50_genres <- top_50_tracks %>%
  count(track_genre, sort = TRUE)

# Display the top 50 genre breakdown
print(top_50_genres)
```

```
## # A tibble: 13 x 2
##    track_genre     n
##    <chr>       <int>
##  1 latin          13
##  2 pop            12
##  3 dance           8
##  4 hip-hop         5
##  5 piano           2
##  6 reggae          2
##  7 rock            2
##  8 alt-rock        1
##  9 chill           1
## 10 country         1
## 11 folk            1
## 12 funk            1
## 13 garage          1
```

18. Reclassifying the Genres for simplicity

```
top_50_tracks <- top_50_tracks %>%
  mutate(parent_genre = case_when(
    track_genre %in% c("pop") ~ "Pop",
    track_genre %in% c("latin", "latino") ~ "Latin",
    track_genre %in% c("dance", "garage", "chill") ~ "Electronic/Dance",
```

```
    track_genre %in% c("hip-hop") ~ "Hip-Hop",
    track_genre %in% c("rock", "alt-rock") ~ "Rock",
    track_genre %in% c("country", "folk") ~ "Country/Folk",
    track_genre %in% c("piano") ~ "Instrumental",
    track_genre %in% c("reggae") ~ "World",
    track_genre %in% c("funk") ~ "Funk/Soul",
    TRUE ~ "Other"
  ))
```

19. Writing into cleaned df into csv and xlsx files.

```
write.csv(spotify_df, "spotify_cleaned.csv", row.names = FALSE)
write.csv(top_50_tracks, "top_50_tracks.csv", row.names = FALSE)
```

```
#install.packages("writexl")   # Run only once
library(writexl)
```

```
## Warning: package 'writexl' was built under R version 4.4.3
```

```
# Save full dataset
write_xlsx(spotify_df, "spotify_cleaned.xlsx")
```

Conclusion

This analysis provided a structured approach to cleaning, transforming, and enriching a large Spotify track dataset. Key tasks included removing duplicates, handling missing values, transliterating multilingual content, and creating meaningful features like mood and vibe. Despite losing over 21% of the data during cleaning, the remaining dataset is robust and ready for visualization. Language detection and transliteration proved essential, as nearly half the entries contained non-English text. By filtering top tracks from trending genres and appending Spotify URLs, the dataset is now also dashboard-ready for user-friendly exploration.