Assignment 3

CptS 575 Data Science

Athul Jose P 11867566

1.a.

The number of players with Free Throws per game greater than 0.5 and Assists per game greater than 0.7

```r
# loading packages
library(dplyr)

# loading dataset
nba_stats <- read.csv("NBA_Stats_23_24.csv")

# players with FT > 0.5 and AST > 0.7
n_players <- nba_stats %>%
  filter(FT > 0.5, AST > 0.7) %>%
  nrow()

# print result
print(n_players)
```

```
[1] 405
```

1.b.

```r
# rearrangin in descending order
top_10_players <- nba_stats %>%
  select(Player, Tm, FG, TOV, PTS) %>%
  arrange(desc(PTS)) %>%
  head(10)

# print result
print(top_10_players)
```

```
                    Player  Tm   FG TOV  PTS
1              Joel Embiid PHI 11.5 3.8 34.7
2              Luka Don?i? DAL 11.5 4.0 33.9
3    Giannis Antetokounmpo MIL 11.5 3.4 30.4
4  Shai Gilgeous-Alexander OKC 10.6 2.2 30.1
5            Jalen Brunson NYK 10.3 2.4 28.7
6             Devin Booker PHO  9.4 2.6 27.1
7             Kevin Durant PHO 10.0 3.3 27.1
8             Jayson Tatum BOS  9.1 2.5 26.9
9              De'Aaron Fox SAC  9.7 2.6 26.6
10        Donovan Mitchell CLE  9.1 2.8 26.6
```

```
# player with seventh highest points
seventh_highest_player <- top_10_players[7, "Player"]
cat("The player with the seventh highest points is:", seventh_highest_player)
```

```
The player with the seventh highest points is: Kevin Durant
```

1.c.

```
# adding columns
nba_stats <- nba_stats %>%
  mutate(FGP = round((FG / FGA) * 100, 2),
         FTP = round((FT / FTA) * 100, 2))

# updated dataframe
head(nba_stats)
```

```
  Rk             Player  Pos Age  Tm  G GS   MP  FG  FGA   FG. X3P X3PA  X3P. X2P
1  1 Precious Achiuwa PF-C  24 TOT 74 18 21.9 3.2  6.3 0.501 0.4  1.3 0.268 2.8
2  1 Precious Achiuwa    C  24 TOR 25  0 17.5 3.1  6.8 0.459 0.5  1.9 0.277 2.6
3  1 Precious Achiuwa   PF  24 NYK 49 18 24.2 3.2  6.1 0.525 0.3  1.0 0.260 2.9
4  2      Bam Adebayo    C  26 MIA 71 71 34.0 7.5 14.3 0.521 0.2  0.6 0.357 7.3
5  3     Ochai Agbaji   SG  23 TOT 78 28 21.0 2.3  5.6 0.411 0.8  2.7 0.294 1.5
6  3     Ochai Agbaji   SG  23 UTA 51 10 19.7 2.1  4.9 0.426 0.9  2.8 0.331 1.2
  X2PA  X2P.  eFG.  FT FTA   FT. ORB DRB  TRB AST STL BLK TOV  PF  PTS   FGP
1  5.0 0.562 0.529 0.9 1.5 0.616 2.6 4.0  6.6 1.3 0.6 0.9 1.1 1.9  7.6 50.79
2  4.9 0.528 0.497 1.0 1.7 0.571 2.0 3.4  5.4 1.8 0.6 0.5 1.2 1.6  7.7 45.59
3  5.1 0.578 0.547 0.9 1.4 0.643 2.9 4.3  7.2 1.1 0.6 1.1 1.1 2.1  7.6 52.46
4 13.7 0.528 0.529 4.1 5.5 0.755 2.2 8.1 10.4 3.9 1.1 0.9 2.3 2.2 19.3 52.45
5  2.8 0.523 0.483 0.5 0.7 0.661 0.9 1.8  2.8 1.1 0.6 0.6 0.8 1.5  5.8 41.07
6  2.1 0.551 0.520 0.3 0.4 0.750 0.7 1.8  2.5 0.9 0.5 0.6 0.7 1.3  5.4 42.86
    FTP
1 60.00
2 58.82
3 64.29
4 74.55
5 71.43
6 75.00
```

```
# FGP and FTP for Josh Giddey
josh_giddey_stats <- nba_stats %>%
  filter(Player == "Josh Giddey") %>%
  select(Player, FGP, FTP)

# print result
josh_giddey_stats
```

```
       Player   FGP   FTP
1 Josh Giddey 47.17 81.25
```

1.d.

```r
# creating new metrics
team_orb_stats <- nba_stats %>%
  group_by(Tm) %>%
  summarise(
    Avg_ORB = mean(ORB, na.rm = TRUE),
    Min_ORB = min(ORB, na.rm = TRUE),
    Max_ORB = max(ORB, na.rm = TRUE)
  ) %>%
  arrange(desc(Avg_ORB))

# print result
team_orb_stats
```

```
# A tibble: 31 x 4
   Tm    Avg_ORB Min_ORB Max_ORB
   <chr>   <dbl>   <dbl>   <dbl>
 1 UTA     1.1       0.2     2.6
 2 POR     1.09      0       3.2
 3 MEM     1.07      0       2.8
 4 ATL     1.02      0.1     4.6
 5 CHI     1         0       3.4
 6 HOU     1         0.1     2.9
 7 TOR     0.98      0       2.9
 8 BRK     0.933     0       2.7
 9 GSW     0.906     0       2
10 BOS     0.889     0       1.9
# i 21 more rows
```

```r
# team with the max Offensive rebounds per game
max_orb_team <- team_orb_stats %>%
  filter(Max_ORB == max(Max_ORB)) %>%
  select(Tm, Max_ORB)

# print result
max_orb_team
```

```
# A tibble: 2 x 2
  Tm    Max_ORB
  <chr>   <dbl>
1 ATL       4.6
2 NYK       4.6
```

1.e.

```r
# creating copies
nba_stats_copy1 <- nba_stats
nba_stats_copy2 <- nba_stats

# Method 1: Impute missing FTP as FGP * average FTP for that team
nba_stats_copy1 <- nba_stats_copy1 %>%
  group_by(Tm) %>%
  mutate(Avg_FTP = mean(FTP, na.rm = TRUE),
         FTP = ifelse(is.na(FTP), FGP * Avg_FTP / 100, FTP)) %>%
  ungroup() %>%
  select(-Avg_FTP)

# print result
head(nba_stats_copy1)
```

```
# A tibble: 6 x 32
     Rk Player Pos     Age Tm        G    GS    MP    FG   FGA   FG.   X3P  X3PA
  <int> <chr>  <chr> <int> <chr> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1     1 Preci~ PF-C     24 TOT      74    18  21.9   3.2   6.3 0.501   0.4   1.3
2     1 Preci~ C        24 TOR      25     0  17.5   3.1   6.8 0.459   0.5   1.9
3     1 Preci~ PF       24 NYK      49    18  24.2   3.2   6.1 0.525   0.3   1
4     2 Bam A~ C        26 MIA      71    71  34     7.5  14.3 0.521   0.2   0.6
5     3 Ochai~ SG       23 TOT      78    28  21     2.3   5.6 0.411   0.8   2.7
6     3 Ochai~ SG       23 UTA      51    10  19.7   2.1   4.9 0.426   0.9   2.8
# i 19 more variables: X3P. <dbl>, X2P <dbl>, X2PA <dbl>, X2P. <dbl>,
#   eFG. <dbl>, FT <dbl>, FTA <dbl>, FT. <dbl>, ORB <dbl>, DRB <dbl>,
#   TRB <dbl>, AST <dbl>, STL <dbl>, BLK <dbl>, TOV <dbl>, PF <dbl>, PTS <dbl>,
#   FGP <dbl>, FTP <dbl>
```

```r
# Method 2: Impute missing FTP with just the average FTP for that team
nba_stats_copy2 <- nba_stats_copy2 %>%
  group_by(Tm) %>%
  mutate(Avg_FTP = mean(FTP, na.rm = TRUE),
         FTP = ifelse(is.na(FTP), Avg_FTP, FTP)) %>%
  ungroup() %>%
  select(-Avg_FTP)

# print result
head(nba_stats_copy2)
```

```
# A tibble: 6 x 32
     Rk Player Pos     Age Tm        G    GS    MP    FG   FGA   FG.   X3P  X3PA
  <int> <chr>  <chr> <int> <chr> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
```

```
1     1 Preci~ PF-C     24 TOT      74    18  21.9   3.2   6.3 0.501   0.4   1.3
2     1 Preci~ C        24 TOR      25     0  17.5   3.1   6.8 0.459   0.5   1.9
3     1 Preci~ PF       24 NYK      49    18  24.2   3.2   6.1 0.525   0.3   1
4     2 Bam A~ C        26 MIA      71    71  34     7.5  14.3 0.521   0.2   0.6
5     3 Ochai~ SG       23 TOT      78    28  21     2.3   5.6 0.411   0.8   2.7
6     3 Ochai~ SG       23 UTA      51    10  19.7   2.1   4.9 0.426   0.9   2.8
# i 19 more variables: X3P. <dbl>, X2P <dbl>, X2PA <dbl>, X2P. <dbl>,
#   eFG. <dbl>, FT <dbl>, FTA <dbl>, FT. <dbl>, ORB <dbl>, DRB <dbl>,
#   TRB <dbl>, AST <dbl>, STL <dbl>, BLK <dbl>, TOV <dbl>, PF <dbl>, PTS <dbl>,
#   FGP <dbl>, FTP <dbl>
```

Method 1: Imputing missing FTP as FGP multiplied by the team's average FTP. This method assumes that a player's FTP should be correlated with their general shooting accuracy (FGP). Thus, adjust the missing values based on both their personal performance (FGP) and the team's overall performance (Avg_FTP). This method assumes that a player's FGP is indicative of their FTP. This would be reasonable if a player's field-goal shooting skill translates into free-throw ability, but this is not always the case (as free-throw shooting can be a distinct skill). This method is may be better because it personalizes the imputation by using both the player's FGP and the team's average FTP. It balances the player's shooting performance with the team context, making the imputation more tailored.

Method 2: Imputing missing FTP with the team's average FTP. This method assumes that missing values for FTP are simply best estimated by the team's overall average FTP, with no correlation to the player's FGP. All players on the same team have similar free-throw abilities, which might be a broader and less accurate assumption compared to Method 1. While this method being simpler, could be less accurate since it doesn't consider the player's shooting ability and assumes uniform free-throw performance within the team, which can lead to unrealistic imputations.

2.a.

```r
# loading packages
library(tidyr)
library(dplyr)
library(readr)

# loading dataset
data("billboard", package = "tidyr")

# tidying the dataset
billboard_tidy <- billboard %>%
  pivot_longer(
    cols = starts_with("wk"),
    names_to = "week",
    values_to = "rank",
    values_drop_na = TRUE
  ) %>%
  mutate(week = parse_number(week))

# print result
head(billboard_tidy)
```

```
# A tibble: 6 x 5
  artist track                   date.entered  week  rank
  <chr>  <chr>                   <date>       <dbl> <dbl>
1 2 Pac  Baby Don't Cry (Keep... 2000-02-26       1    87
2 2 Pac  Baby Don't Cry (Keep... 2000-02-26       2    82
3 2 Pac  Baby Don't Cry (Keep... 2000-02-26       3    72
4 2 Pac  Baby Don't Cry (Keep... 2000-02-26       4    77
5 2 Pac  Baby Don't Cry (Keep... 2000-02-26       5    87
6 2 Pac  Baby Don't Cry (Keep... 2000-02-26       6    94
```

The line `mutate(week = parse_number(week))` is essential for properly tidying the data because it extracts the numeric portion from the column names that represent weeks, such as "wk1", "wk2", etc. These original column names are strings that contain both text ("wk") and numbers, which can hinder analysis. By using `parse_number(week)`, we ensure that only the numeric values, such as 1, 2, 3, are retained in the week column. This is important because we need the week values to be numeric for proper sorting, comparison, and analysis. Without this transformation, the week column would remain a string, which could cause issues when ordering the data or performing calculations. For example, "wk10" might be placed before "wk2" if treated as a string, and the data would not be usable for numerical operations or visualizations based on week numbers. Therefore, this line is critical for ensuring that the data is both tidy and functional for analysis.

2.b.

Number of entries removed when values_drop_na set to true

```r
# initial dataset
billboard_no_drop <- billboard %>%
  pivot_longer(
    cols = starts_with("wk"),
    names_to = "week",
    values_to = "rank"
  )
initial_count <- nrow(billboard_no_drop)

# dataset with drop_na
billboard_with_drop <- billboard %>%
  pivot_longer(
    cols = starts_with("wk"),
    names_to = "week",
    values_to = "rank",
    values_drop_na = TRUE
  )
final_count <- nrow(billboard_with_drop)
entries_removed <- initial_count - final_count

# print result
entries_removed
```

```
[1] 18785
```

2.c.

```r
# implicit_missing data frame
implicit_missing <- billboard_tidy %>%
  group_by(track) %>%
  summarise(min_week = min(week), max_week = max(week)) %>%
  rowwise() %>%
  mutate(missing_weeks = list(setdiff(seq(min_week, max_week), billboard_tidy$week[billboard_ti
  filter(length(missing_weeks) > 0)

# print result
implicit_missing
```

```
# A tibble: 15 x 4
# Rowwise:
   track                      min_week max_week missing_weeks
   <chr>                       <dbl>   <dbl> <list>
 1 Amazed                          1      64 <int [9]>
 2 Bounce With Me                  1      22 <int [2]>
 3 Could I Have This Ki...         1      20 <int [1]>
 4 Feelin' Good                    1      19 <int [2]>
 5 Feels Like Love                 1      21 <int [1]>
 6 Give Me You                     1      10 <int [1]>
 7 Go On                           1      21 <int [1]>
 8 Higher                          1      65 <int [8]>
 9 How Many Licks?                 1      11 <int [2]>
10 I Wanna Be With You             1      27 <int [4]>
11 I'm Outta Love                  1      12 <int [9]>
12 It's Always Somethin...         1      15 <int [1]>
13 No Mercy                        1      10 <int [2]>
14 Pop A Top                       1      18 <int [1]>
15 Sexual                          1      33 <int [4]>
```

An explicit missing value in a dataset is one that is clearly marked as missing, often represented as NA indicating that data was expected but not provided. In contrast, an implicit missing value occurs when certain data points are simply absent from the dataset, but their absence implies missing information. Implicit missing values are not explicitly flagged; instead, they are inferred from missing records. In the dataset, implicit missing values occur when a song falls off the chart and no further data is recorded for subsequent weeks. For example, if a song appears for 10 weeks and then drops off, there will be no records for weeks beyond the 10th. These gaps imply that the song did not chart in those weeks, but this missing information is not explicitly noted as NA in the dataset, making it an implicit missing value.

2.d.

In the tidied dataset, the features generally have appropriate data types, but there are a few considerations for improvement. The artist and track features are character variables, which is suitable since they contain text-based information. The week feature, representing the week number on the chart, is numeric and should remain as such for correct ordering and calculations. The rank feature, representing the song's ranking position, is also numeric, which is appropriate for performing numerical operations. However, the date.entered feature, which records the date a song entered the chart, might initially be stored as a string. For time-based analysis, such as calculating the duration a song stayed on the chart, it would be better suited as a Date type. Converting date.entered to a Date type allows for easier manipulation and calculations involving dates, such as time spans or chronological sorting. Thus, converting date.entered to the correct format ensures the data is ready for thorough analysis.

```r
# data types of features
str(billboard_tidy)
```

```
tibble [5,307 x 5] (S3: tbl_df/tbl/data.frame)
 $ artist      : chr [1:5307] "2 Pac" "2 Pac" "2 Pac" "2 Pac" ...
 $ track       : chr [1:5307] "Baby Don't Cry (Keep..." "Baby Don't Cry (Keep..." "Baby Don't C
 $ date.entered: Date[1:5307], format: "2000-02-26" "2000-02-26" ...
 $ week        : num [1:5307] 1 2 3 4 5 6 7 1 2 3 ...
 $ rank        : num [1:5307] 87 82 72 77 87 94 99 91 87 92 ...
```

```
# converting 'date.entered' to Date type
billboard_tidy <- billboard_tidy %>%
  mutate(date.entered = as.Date(date.entered, format = "%Y-%m-%d"))
str(billboard_tidy)
```

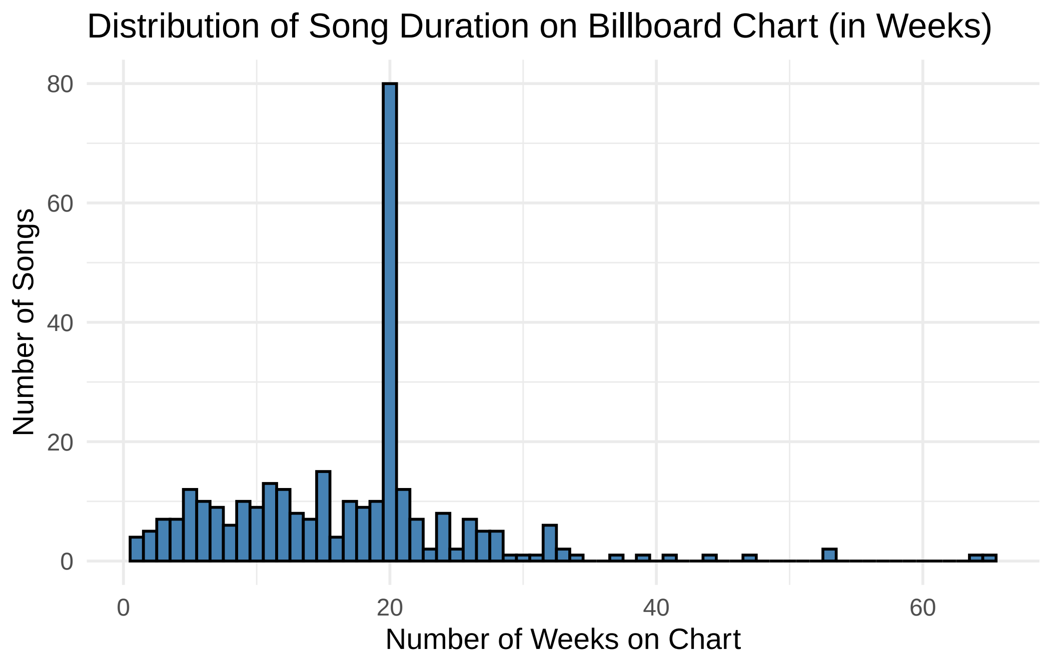
```
tibble [5,307 x 5] (S3: tbl_df/tbl/data.frame)
 $ artist      : chr [1:5307] "2 Pac" "2 Pac" "2 Pac" "2 Pac" ...
 $ track       : chr [1:5307] "Baby Don't Cry (Keep..." "Baby Don't Cry (Keep..." "Baby Don't C
 $ date.entered: Date[1:5307], format: "2000-02-26" "2000-02-26" ...
 $ week        : num [1:5307] 1 2 3 4 5 6 7 1 2 3 ...
 $ rank        : num [1:5307] 87 82 72 77 87 94 99 91 87 92 ...
```

2.e.

```
# loading ggplot2
library(ggplot2)

# number of weeks each song stayed on the chart
weeks_on_chart <- billboard_tidy %>%
  group_by(track) %>%
  summarise(weeks_on_chart = max(week) - min(week) + 1)

# plotting the results
ggplot(weeks_on_chart, aes(x = weeks_on_chart)) +
  geom_histogram(binwidth = 1, fill = "steelblue", color = "black") +
  labs(
    title = "Distribution of Song Duration on Billboard Chart (in Weeks)",
    x = "Number of Weeks on Chart",
    y = "Number of Songs"
  ) +
  theme_minimal()
```

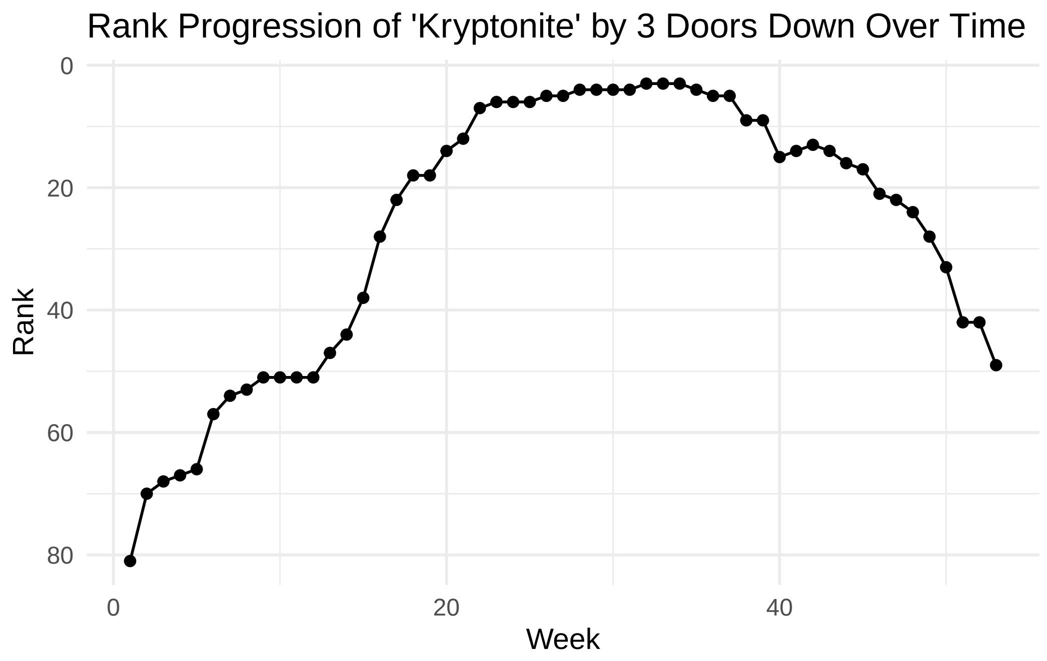


This plot is interesting because it reveals the **longevity of songs** on the Billboard chart, providing insights into the patterns of song popularity. By visualizing the distribution of how many weeks different tracks remain on the chart, we can observe whether the music industry is dominated by short-term hits or if there are many songs that enjoy long-lasting popularity. This distribution helps us understand the dynamics of the music market, whether it is driven by quick turnovers or sustained interest in certain songs. The histogram can also highlight outlier songs that remain on the chart for significantly longer periods indicating their exceptional popularity or cultural impact. By exploring this, we can gain a deeper understanding of music trends, audience behavior, and the factors that contribute to a song's enduring success.

2.f.

```r
# "Kryptonite" by 3 Doors Down
kryptonite_data <- billboard_tidy %>%
  filter(track == "Kryptonite")

# rank progression over time
ggplot(kryptonite_data, aes(x = week, y = rank)) +
  geom_line() +
  geom_point() +
  scale_y_reverse() +
  labs(
    title = "Rank Progression of 'Kryptonite' by 3 Doors Down Over Time",
    x = "Week",
    y = "Rank"
  ) +
  theme_minimal()
```

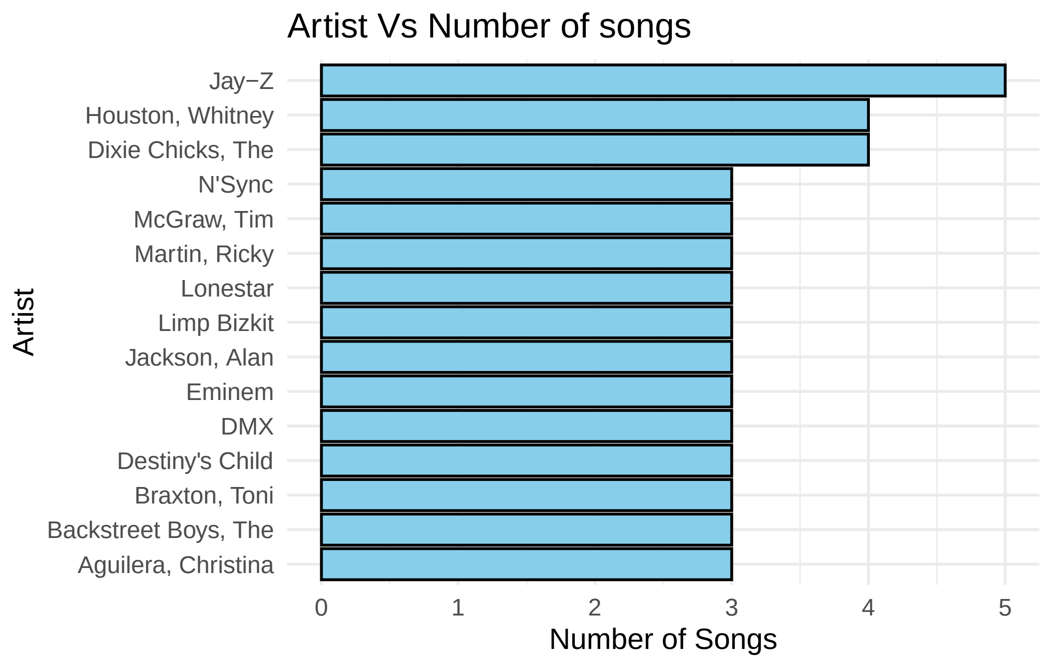


The figure shows the rank progression of the song "Kryptonite" by 3 Doors Down over time on the Billboard chart. The x-axis represents the weeks, while the y-axis (inverted) shows the song's rank, with lower values indicating a better ranking position. The line plot illustrates how the song initially climbs up the rankings, improving its position as the weeks pass. At its peak, the song achieves a strong rank before gradually declining in popularity as time progresses. This visual effectively captures the typical lifecycle of a hit song on the charts, showing both its rise to prominence and its eventual drop-off.

2.g.

```r
# count of songs per artists(top 15)
top_artists <- billboard_tidy %>%
  group_by(artist) %>%
  summarise(song_count = n_distinct(track)) %>%
  arrange(desc(song_count)) %>%
  slice(1:15)

# plotting the data
ggplot(top_artists, aes(x = reorder(artist, song_count), y = song_count)) +
  geom_bar(stat = "identity", fill = "skyblue", color = "black") +
  coord_flip() +
  labs(
    title = "Artist Vs Number of songs",
    x = "Artist",
    y = "Number of Songs"
  ) +
  theme_minimal()
```



The top 15 list of artists by the number of songs on the Billboard chart reflects the dominance of certain artists during the period covered by the dataset. It highlights popular artists who had significant chart success, with some well-known names likely to be expected. Artists like 3 Doors Down and Destiny's Child are not surprising, given their major success and influence during the 2000s. However, the presence of certain artists are unexpected depending on my personal knowledge of the music scene from that time. For example, an artist with a large number of songs but less mainstream recognition might stand out, suggesting that they had consistent chart success but weren't necessarily as visible in broader pop culture. Additionally, artists with shorter-lived fame or those known for just one big hit may appear surprisingly high on the list, which could reflect the strength of their musical catalog during that specific period. Overall, this list can be a fascinating look into trends and popularity in the music industry during the dataset's time range, and it might also prompt some curiosity about artists who were highly successful in their genre or era but aren't as well remembered today.

2.h.

```
# loading RevQtr
RevQtr <- read.csv("RevQtr.csv")

# Tidy the dataset using pivot_longer
RevQtr_tidy <- RevQtr %>%
  pivot_longer(
    cols = starts_with("Qtr"),
    names_to = "Interval_ID",
    values_to = "Revenue"
  ) %>%
  mutate(Interval_Type = "Qtr",
         Interval_ID = parse_number(Interval_ID)) %>%
  select(Group, Year, Interval_Type, Interval_ID, Revenue)

# Display the first few rows
head(RevQtr_tidy)
```

```
# A tibble: 6 x 5
  Group  Year Interval_Type Interval_ID Revenue
  <int> <int> <chr>               <dbl>   <int>
1     1  2022 Qtr                   0.1      61
2     1  2022 Qtr                   0.2      24
3     1  2022 Qtr                   3        81
4     1  2022 Qtr                   0.4      70
5     1  2023 Qtr                   0.1      30
6     1  2023 Qtr                   0.2      92
```

```
# number of rows in the new dataset
nrow(RevQtr_tidy)
```

```
[1] 48
```