````
---
title: "Mini Project #02"
author: "Aachal Ghimire"
date: "2024-10-13"
output:
    html_document:
        code-fold: hide
cache: true
---

Hi,

In this project, I am stepping into the role of a Hollywood development executive, where my job is to create new and exciting movie ideas. My goal is to develop fresh, data-driven concepts that bring something unique to the screen. To make that happen, I'll be using IMDb data [the IMDb non- commercial release] (https://datasets.imdbws.com/) to dig deep into Hollywood's history, exploring patterns of success and failure in the industry.

With this data, we'll analyze the track records of directors, actors, and genres to see what combinations have resonated with audiences. We'll also look at metrics like box office earnings, critic and audience ratings, and even compare trends across decades to understand the factors that drive hits and avoid flops. By interpreting this information, I'll gain valuable insights to guide the creation of movie concepts that have the potential to stand out in today's market.

## **Libraries used**

```{r message=FALSE, warning=FALSE}
library(tidyverse)
library(stringr)
library(readr)
library(knitr)
library(ggplot2)
```

## Data

```{r message=FALSE, warning=FALSE, cache=TRUE}

#IMDb_name_basics

get_imdb_file <- function(fname){
    BASE_URL <- "https://datasets.imdbws.com/"
    fname_ext <- paste0(fname, ".tsv.gz")
    if(!file.exists(fname_ext)){
        FILE_URL <- paste0(BASE_URL, fname_ext)
        download.file(FILE_URL,
                      destfile = fname_ext)
    }
    as.data.frame(readr::read_tsv(fname_ext, lazy=FALSE))
}

NAME_BASICS <- read.csv("/Users/aachalghimire/Documents/GitHub/STA9750-2024-FALL/name_basics_small.csv", header = TRUE, sep = ",")


```

```{r  message=FALSE, warning=FALSE, cache=TRUE}
#IMDb_titile_basics

TITLE_BASICS     <- read.csv("/Users/aachalghimire/Documents/GitHub/STA9750-2024-FALL/title_basics_small.csv", header = TRUE, sep = ",")
```

```{r message=FALSE, warning=FALSE, cache=TRUE}
#IMDb_title_Episodes
````
````

```
TITLE_EPISODES   <- read.csv("/Users/aachalghimire/Documents/GitHub/STA9750-2024-
FALL/title_episodes_small.csv", header = TRUE, sep = ",")
```

```{r message=FALSE, warning=FALSE, cache=TRUE}
#IMDb_title_ratings

TITLE_RATINGS    <- read.csv("/Users/aachalghimire/Documents/GitHub/STA9750-2024-
FALL/title_ratings_small.csv", header = TRUE, sep = ",")
```

```{r message=FALSE, warning=FALSE, cache=TRUE}
#IMDb_title_crew

TITLE_CREW       <- read.csv("/Users/aachalghimire/Documents/GitHub/STA9750-2024-
FALL/title_crew_small.csv", header = TRUE, sep = ",")
```

```{r message=FALSE, warning=FALSE, cache=TRUE}
#IMDb_title_principals

TITLE_PRINCIPALS <- read.csv("/Users/aachalghimire/Documents/GitHub/STA9750-2024-
FALL/title_principals_small.csv", header = TRUE, sep = ",")

```

## Data Sub-Sampling

```{r}
library(dplyr)
library(stringr)
NAME_BASICS <- NAME_BASICS |>
    filter(str_count(knownForTitles, ",") > 1)
```

```{r}
TITLE_RATINGS |>
    ggplot(aes(x=numVotes)) +
    geom_histogram(bins=30) +
    xlab("Number of IMDB Ratings") +
    ylab("Number of Titles") +
    ggtitle("Majority of IMDB Titles Have Less than 100 Ratings") +
    theme_bw() +
    scale_x_log10(label=scales::comma) +
    scale_y_continuous(label=scales::comma)
```

```{r}
TITLE_RATINGS |>
    pull(numVotes) |>
    quantile()
```

```{r}
TITLE_RATINGS <- TITLE_RATINGS |>
    filter(numVotes >= 100)
```

## Semi_join

```{r}
TITLE_BASICS <- TITLE_BASICS |>
    semi_join(TITLE_RATINGS,
              join_by(tconst == tconst))

TITLE_CREW <- TITLE_CREW |>
    semi_join(TITLE_RATINGS,
              join_by(tconst == tconst))
```

```
TITLE_EPISODES_1 <- TITLE_EPISODES |>
    semi_join(TITLE_RATINGS,
              join_by(tconst == tconst))
TITLE_EPISODES_2 <- TITLE_EPISODES |>
    semi_join(TITLE_RATINGS,
              join_by(parentTconst == tconst))

TITLE_EPISODES <- bind_rows(TITLE_EPISODES_1,
                            TITLE_EPISODES_2) |>
    distinct()

TITLE_PRINCIPALS <- TITLE_PRINCIPALS |>
    semi_join(TITLE_RATINGS, join_by(tconst == tconst))


rm(TITLE_EPISODES_1)
rm(TITLE_EPISODES_2)
```

## Initial Exploration

```{r}
# Replace `\\N' with NA and then convert into numeric
# glimpse() to view the characters
NAME_BASICS|> glimpse()

NAME_BASICS_NA <- NAME_BASICS |>
    mutate(birthYear = as.numeric(replace(birthYear, birthYear == "\\N", NA)),
           deathYear = as.numeric(replace(deathYear, deathYear == "\\N", NA)))
```

## Initial Exploration

### Task 1

```{r warning=FALSE}
# replace "\\N" with NA
# changed the charcters for the TITLE table using as.numeric and as.logical
library(dplyr)


TITLE_BASICS_CCOL <- TITLE_BASICS|>
    mutate(
        isAdult = as.logical(isAdult),
        startYear = as.numeric(replace(startYear, startYear == "\\N", NA)),
        endYear = as.numeric(replace(endYear, endYear == "\\N", NA)),
        runtimeMinutes = as.numeric(runtimeMinutes))

TITLE_BASICS_CCOL |> glimpse()


#glimpse, offers a concise overview of a dataframe

```

### Task 2

1.How many movies are in our data set? How many TV series? How many TV episodes?

```{r}
# filter
library(dplyr)
total_count <- TITLE_BASICS_CCOL |>
    filter(titleType %in% c("movie", "tvSeries", "tvEpisode")) |>
    group_by(titleType) |>
    summarize(count= n())

#Display the result
```

```
print(total_count)
```
> There are **131662** movies, **155722** tvEpisode, and **29789** tvSeries.

2. Who is the oldest living person in our data set?

```{r}
#birthYear is not NA and deathYear is NA

oldest_living_person<- NAME_BASICS_NA |>
    filter(!is.na(birthYear) & is.na(deathYear)) |>
    slice_min(birthYear)

#birthYear>1900
oldest_living_person_1900 <- NAME_BASICS_NA |>
    filter(!is.na(birthYear) & is.na(deathYear), birthYear >=1900) |>
    slice_min(birthYear)

#age thresold is 110 years
#assign current year
#Filter and find the oldest person
current_year <- as.numeric(format(Sys.Date(), "%Y"))
year <- current_year - 110

oldest_living_person_110 <- NAME_BASICS_NA |>
    filter(!is.na(birthYear) & is.na(deathYear),birthYear >= year) |>
    arrange(birthYear) |>
    slice(1)


#display the result

print(oldest_living_person)
# uncomment to view the table
# print(oldest_living_person_1900)
print(oldest_living_person_110)

```

> We approached the question in three stages. First, analyzing the unfiltered data identified **Robert De Visée (born 1655)** as the oldest living person, which is unlikely due to missing death years in the database. Next, we filtered for people born after 1900, which narrowed the list but still included entries with missing death years. Finally, we calculated a reasonable threshold by subtracting 110 years from the current year, identifying **Antonio Anelli (born 1914)** as the likely oldest living person, with a higher chance of being accurate.

3. There is one TV Episode in this data set with a perfect 10/10 rating and at least 200,000 IMDb ratings. What is it? What series does it belong to?

```{r}
#inner_join to join tables from TITLE_RATING, TITLE_EPISODES, TITLE_BASICS_CCOL
#select(-col) to drop unwanted columns

perfect_rating <- TITLE_RATINGS |>
    filter(averageRating == 10, numVotes >= 200000) |>
    inner_join(TITLE_EPISODES, by = 'tconst') |>
    inner_join(TITLE_BASICS_CCOL, by = c("parentTconst" = 'tconst')) |>
    select(-isAdult, -startYear,-endYear, -runtimeMinutes, -genres )

#print result
print(perfect_rating)
```

> **Breaking Bad season 5, episode 14** has the perfect rationg of 10/10 with **229313** rating

4. What four projects is the actor Mark Hamill most known for?

```{r}
#filter & identify the primary name = Mark Hamill
# filter and connect the data with tconst

NAME_BASICS_NA <- NAME_BASICS_NA|>
    mutate(primaryName = str_trim(primaryName))

Mark_Hamill_id <- NAME_BASICS_NA |>
    filter(primaryName == "Mark Hamill") |>
    select(nconst) |>
    pull()

#Filter his works in TITLE_PRINCIPAL

Mark_Hamill_works <- TITLE_PRINCIPALS |>
    filter(nconst == Mark_Hamill_id ) |>
    select(tconst)

# Select and join his top projects with ratings

Mark_Hamill_Projects <- Mark_Hamill_works |>
    inner_join(TITLE_BASICS_CCOL, by = "tconst") |>
    select(primaryTitle, tconst) |>
    inner_join(TITLE_RATINGS, by = "tconst") |>
    select(primaryTitle, averageRating, numVotes) |>
    arrange(desc(numVotes)) |>
    head(4)

print(Mark_Hamill_Projects)

```

> Mark Hamill is widely known for his iconic role as Luke Skywalker in Star Wars. His top four projects, based on IMDb’s highest number of user votes, are Star Wars: Episode IV – A New Hope, Episode V – The Empire Strikes Back, Episode VI – Return of the Jedi, and Episode VII – The Force Awakens. These movies highlight his most celebrated contributions to the series and cinema.

5. What TV series, with more than 12 episodes, has the highest average rating?

```{r}

#calculating episodes for each series and filter >12

total_episodes <- TITLE_EPISODES |>
    group_by(parentTconst) |>
    summarise(total_episodes = n()) |>
    filter(total_episodes > 12)

#making sure there are series with more than 12 eps
#print(total_episodes)

# Ensuring column types match and remove missing values if necessary
TITLE_BASICS_CCOL <- TITLE_BASICS_CCOL |>
    mutate(tconst = as.character(tconst))

total_episodes <- total_episodes |>
    mutate(parentTconst = as.character(parentTconst))

# filter tvSeries and join with total episodes

series_episodes <- TITLE_BASICS_CCOL |>
    filter(titleType == "tvSeries") |>
    inner_join(total_episodes, by = c("tconst" = "parentTconst"))

#print(series_episodes)

#Joining with rating and finding the highest rated series
```

```
TVSeries_12eps_rating <- series_episodes |>
    inner_join(TITLE_RATINGS, by = "tconst") |>
    arrange(desc(averageRating)) |>
    select(primaryTitle, total_episodes, averageRating) |>
    slice(1)

print(TVSeries_12eps_rating)

```

> **Craft Games** among tvShows with total of **318 episodes** has the highest rating of **9.7**

6. The TV series Happy Days (1974-1984) gives us the common idiom “jump the shark”. The phrase comes from a controversial fifth season episode (aired in 1977) in which a lead character literally jumped over a shark on water skis. Idiomatically, it is used to refer to the moment when a once-great show becomes ridiculous and rapidly looses quality.

Is it true that episodes from later seasons of Happy Days have lower average ratings than the early seasons?

```{r}

#calculating all the episodes of Happy Days

Happy_days_season <- TITLE_BASICS_CCOL |>
    filter(primaryTitle == "Happy Days", titleType == "tvSeries") |>
    select(tconst) |>
    pull()

#episodes by season number

HappyDays_episodes <- TITLE_EPISODES |>
    filter(parentTconst %in% Happy_days_season) |>
    mutate(seasonCategory = ifelse(seasonNumber <= 4, "early", "later"))

HappyDays_rating <- HappyDays_episodes |>
    inner_join(TITLE_RATINGS, by= 'tconst') |>
    group_by(seasonCategory) |>
    summarize(averageRating = mean(averageRating, na.rm = TRUE))

print(HappyDays_rating)

```

> Based on the analysis, we observe a decline in average ratings from early to later seasons. The early seasons had an average rating of 7.58, whereas the later seasons dropped to 6.91. This indicates a noticeable decrease in viewer ratings as the series progressed.

## Quantifying Success

## Task 3: custom success matrix

```{r}
#adding success rating to TITLE_RATING column

TITLE_RATING_SUCCESS <- TITLE_RATINGS |>
    mutate(success = averageRating * log(numVotes))

#if want to print TITLE_RATING_SUCCESS
#print(TITLE_RATING_SUCCESS)
```

1. Choose the top 5-10 movies on your metric and confirm that they were indeed box office successes.

```{r}
```

```
top_sucessful_movies <- TITLE_RATING_SUCCESS |>
    inner_join(TITLE_BASICS_CCOL, by = "tconst") |>
    arrange(desc(success)) |>
    slice(1:10) |>
    select(primaryTitle, averageRating, numVotes, success)

print(top_sucessful_movies)
```

> top 10 successful movies using our success matrix.

2. Choose 3-5 movies with large numbers of IMDb votes that score poorly on your success metric and confirm that they are indeed of low quality.

```{r}
low_successful_movies <- TITLE_RATING_SUCCESS |>
    inner_join(TITLE_BASICS_CCOL, by = "tconst" ) |>
    filter(numVotes >= 50000) |>
    arrange(success) |>
    slice(1:5) |>
    select(primaryTitle, averageRating, numVotes, success)
print(low_successful_movies)

```

> Our analysis reveals a list of movies that, despite being popular, have low ratings. Cross-referencing reviews on platforms like Google confirms that these movies are generally considered low quality.

3. Choose a prestige actor or director and confirm that they have many projects with high scores on your success metric.

```{r}
# we have chosen Zendaya
actor_id <- NAME_BASICS_NA |>
    filter(primaryName == "Zendaya") |>
    select(nconst) |>
    pull()

#filter for Zendaya's projects

actor_projects <- TITLE_PRINCIPALS |>
    filter(nconst == actor_id) |>
    select(tconst)

# join the project table with rating to measure the success

actor_success <- actor_projects |>
    inner_join(TITLE_BASICS_CCOL, by = 'tconst') |>
    inner_join(TITLE_RATING_SUCCESS, by = 'tconst') |>
    arrange(desc(success)) |>
    select(primaryTitle, averageRating, numVotes, success)

#print the result
print(head(actor_success,5))

```

> Our analysis confirms that Zendaya has a successful portfolio with multiple highly rated projects that perform strongly on our success metric, reflecting both high quality and broad popularity.

4. Perform at least one other form of ‘spot check’ validation.

```{r}
#top 3 movie bassed on success matrix
#bottom 3 movies based on success matrix

top_3movies <- TITLE_RATING_SUCCESS |>
```

```
    arrange(desc(success)) |>
    slice_head( n = 3)

bottom_3movies <- TITLE_RATING_SUCCESS |>
    arrange(success) |>
    slice_head(n = 3)

#combinig two tabels for a unified check

spot_check_movies <- bind_rows(top_3movies, bottom_3movies)
print(spot_check_movies)
```

> Our spot check confirmed that the success metric aligns well with IMDb ratings, accurately identifying both high- and low-rated movies, which shows that our analysis is reliable.

5. Come up with a numerical threshold for a project to be a ‘success’; that is, determine a value v such that movies above v are all “solid” or better.

```{r, message=FALSE, echo=FALSE}
# Load necessary libraries
library(dplyr)

# calculate based on averageRating and numVotes
TITLE_RATING_SUCCESS2 <- TITLE_RATING_SUCCESS |>
    mutate(success = averageRating * log(numVotes + 1))

# Calculate the threshold (75th percentile) for success, fixing the typo to 'threshold'
threshold <- quantile(TITLE_RATING_SUCCESS2$success, 0.75, na.rm = TRUE)

# Filter for successful movies with success >= threshold and averageRating >= 7
successful_movies <- TITLE_RATING_SUCCESS2 |>
    filter(success >= threshold, averageRating >= 7) |>
    arrange(desc(success)) |>
    slice(1:5)

# Display the threshold and successful movies
print(paste("Success threshold (75th percentile):", threshold))
print(successful_movies)

# For a stricter threshold (e.g., 90th percentile)
threshold <- quantile(TITLE_RATING_SUCCESS2$success, 0.90, na.rm = TRUE)

# For a more lenient threshold (e.g., 50th percentile, the median)
threshold <- quantile(TITLE_RATING_SUCCESS2$success, 0.50, na.rm = TRUE)


```

> Based on our analysis and the calculated threshold, the 75th percentile for the success metric is **49.0371**.

## Examining Success. by Genre and Decade

## Task 4

1. What was the genre with the most “successes” in each decade?

```{r}
# Filter for movies in title_basics and title_rating

movie_ratings <- TITLE_BASICS_CCOL|>
    inner_join(TITLE_RATING_SUCCESS, by= 'tconst')

#new colum for decade

decade_movie_rating <- movie_ratings |>
```

```
    mutate(decade = floor(startYear/10)*10) |>
    filter(decade >= 1980)

# group by decade and genre

decade_genre <- decade_movie_rating |>
    group_by(decade, genres) |>
    summarize(total_success = sum(success, na.rm = TRUE), .groups = 'drop')

most_successful_by_decade_genre <- decade_genre |>
  group_by(decade) |>
  slice_max(order_by = total_success, n = 1)


#Display the result

print(most_successful_by_decade_genre)
```
> From our analysis, we can say that **Comedy** was the most popular genre from the 1980s onwards. However, since 2020, **Drama** has emerged as the leading genre. It will be interesting to see which genre dominates in the next decade.

2. What genre consistently has the most “successes”? What genre used to reliably produced “successes” and has fallen out of favor?

```{r}
# Step 1: Aggregate success by genre and decade

success_by_genre_and_decade <- decade_movie_rating |>
    separate_rows(genres, sep = ",") |>
    group_by(decade, genres) |>
    summarize(total_success = sum(success, na.rm = TRUE), .groups = 'drop')

# Step 2: Identify the genre with the most consistent success

consistent_success_by_genre <- success_by_genre_and_decade |>
    group_by(genres) |>
    summarize(total_success_across_decades = sum(total_success), .groups = 'drop') |>
    arrange(desc(total_success_across_decades))

# Display the genre with the most consistent success across decades
print("Genres with the most consistent success:")
print(consistent_success_by_genre)

# Step 3: Find the genres that have fallen out of favor
# Summarize total success before and after 2000 for each genre
success_before_2000 <- success_by_genre_and_decade |>
    filter(decade < 2000) |>
    group_by(genres) |>
    summarize(success_before_2000 = sum(total_success, na.rm = TRUE))

success_after_2000 <- success_by_genre_and_decade |>
    filter(decade >= 2000) |>
    group_by(genres) |>
    summarize(success_after_2000 = sum(total_success, na.rm = TRUE))

# Combine success data before and after 2000
genre_success_trends <- success_before_2000 |>
    full_join(success_after_2000, by = "genres") |>
    mutate(success_before_2000 = coalesce(success_before_2000, 0),
           success_after_2000 = coalesce(success_after_2000, 0),
           decline = success_before_2000 - success_after_2000) |>
    arrange(desc(decline))

# Display genres with a significant decline
fallen_genres <- genre_success_trends |>
    filter(decline > 0)  # Genres that have declined after 2000
```

```
print("Genres that have fallen out of favor:")
print(fallen_genres)

```


```{r, echo=FALSE}
library(ggplot2)

# Bar Plot comparing success before and after 2000 for each genre
ggplot(genre_success_trends, aes(x = reorder(genres, -decline), y = decline, fill =
genres)) +
  geom_bar(stat = "identity") +
  labs(title = "Genres with Decline in Success After 2000",
       x = "Genres",
       y = "Decline in Success (Before 2000 vs After 2000)") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        legend.position = "none") +
  scale_fill_viridis_d()  # Automatically generate colors for all genres


```
> As per our analysis **Drama** is the most successful genre. While, for **Adult**
genre it was sucessful before 2000 and we see a decile of 67220.62 after 2000.

3. What genre has produced the most “successes” since 2010? Does it have the highest
success rate or does it only have a large number of successes because there are many
productions in that genre?

```{r}
# Filter data for movies released after 2010
movies_since_2010 <- decade_movie_rating |>
    filter(decade >= 2010)

# Summarize total successes by genre since 2010
success_by_genre_since_2010 <- movies_since_2010 |>
    separate_rows(genres, sep = ",") |>
    group_by(genres) |>
    summarize(total_success = sum(success, na.rm = TRUE),
              total_productions = n(),  # Total number of productions
              success_rate = total_success / total_productions) |>
    arrange(desc(total_success))

# Display the genre with the most successes
most_successful_genre_since_2010 <- success_by_genre_since_2010 |> slice_max(order_by =
total_success, n = 1)
print("Most successful genre since 2010:")
print(most_successful_genre_since_2010)

# Display the success rate of the most successful genre
print("Success rate of the most successful genre:")
print(most_successful_genre_since_2010$success_rate)


```
> Since 2010, **Drama** has been the most successful genre with high volume of content
and sucess rate.

4. What genre has become more popular in recent years?

```{r, echo=FALSE}
# Decade_movie_rating_clean

decade_movie_rating_clean <- decade_movie_rating |>
    filter(!is.na(genres), genres != "\\N", genres != "")

# Summarizing successes by genre in the 2010s and 2020s

success_by_genre_2010s <- decade_movie_rating_clean |>
```

```
    filter(decade == 2010) |>
    separate_rows(genres, sep = ",") |>
    group_by(genres) |>
    summarize(total_success_2010s = sum(success, na.rm = TRUE))

success_by_genre_2020s <- decade_movie_rating_clean |>
    filter(decade == 2020) |>
    separate_rows(genres, sep = ",") |>
    group_by(genres) |>
    summarize(total_success_2020s = sum(success, na.rm = TRUE))

# Combine the success data for the 2010s and 2020s
success_comparison <- success_by_genre_2010s |>
    full_join(success_by_genre_2020s, by = "genres") |>
    mutate(total_success_2010s = coalesce(total_success_2010s, 0),
           total_success_2020s = coalesce(total_success_2020s, 0),
           growth_rate = ifelse(total_success_2010s > 0,
                                (total_success_2020s - total_success_2010s) /
total_success_2010s * 100,
                                NA)) |>
    arrange(desc(growth_rate))

# Display the genre with the highest growth rate
most_popular_recent_genre <- success_comparison |> slice_max(order_by = growth_rate, n
= 1)
print("Genre with the highest growth rate from the 2010s to the 2020s:")
print(most_popular_recent_genre)

```

> **The Game- Show** has shown the highest growth rate of 20.74% from 2010 to 2020

## Successful Personnel in the Genre

## Task 5

Identify (at least) two actors and one director who you will target as the key talent for your movie. Write a short “pitch” as to why they are likely to be successful. You should support your pitch with at least one graphic and one table.

For my upcoming psychological thriller, we are excited to announce a powerhouse team that blends established talent with rising stars, led by an acclaimed director known for his mastery of tension and atmosphere. This project is anchored by **Zendaya**, **Blake Lively**, and **David Fincher**—a dynamic trio that promises both commercial appeal and critical success.

Meet the Team:
Zendaya: She’s not just a star, she’s a supernova. With an average IMDb rating of 8.2, Zendaya has taken Hollywood by storm, from Euphoria to Dune. Her ability to effortlessly switch between emotional vulnerability and fierce intensity is exactly what our lead role demands. Plus, let’s be real—audiences can’t get enough of her, and neither can we!

Blake Lively: Class, charm, and a sprinkle of danger. Blake’s IMDb rating of 7.9 proves she knows how to bring depth to suspense-driven films (The Shallows, anyone?). She’s got that magnetic screen presence, where you can’t help but wonder, “What’s she going to do next?” Pair her with Zendaya, and we’ve got the ultimate power duo to keep audiences on the edge of their seats.

David Fincher: If we’re making a psychological thriller, we need the king of suspense—Fincher! With an 8.5 IMDb average, this guy practically breathes tension. From the eerie brilliance of Gone Girl to the chilling mind-bend of Fight Club, Fincher is the master at making you question everything. He’ll take our story and turn it into an unforgettable, mind-twisting journey.

Why This Trio Works:
Zendaya and Blake bring the fire on screen, and Fincher knows exactly how to fan the flames. Their track record in the thriller, drama, and horror genres is undeniable. Check out the numbers in the table given.

Picture this: Zendaya as the determined but haunted protagonist, Blake as her enigmatic and maybe untrustworthy friend, all under Fincher's masterful lens. The suspense, the twists, the psychological games—this team has everything it takes to create a film that will leave the audience talking (and rewatching) for years.

Visualize the Success:
Here's a sneak peek at how these talents are killing it on IMDb:

```{r, echo=FALSE, message=FALSE, warning=FALSE}

library(dplyr)
library(ggplot2)
library(knitr)

# Step 1: Filter for movies in Thriller, Drama, or Horror genres
movies <- TITLE_BASICS_CCOL |>
  filter(titleType == "movie" & grepl("Thriller|Drama|Horror", genres))

# Step 2: Join the filtered movies with their ratings
movies_ratings <- movies |>
  inner_join(TITLE_RATING_SUCCESS, by = 'tconst')

# Step 3: Filter for specific actors and director (Zendaya, Blake Lively, David
Fincher)
actors <- NAME_BASICS_NA |>
  filter(grepl("Zendaya|Blake Lively", primaryName)) |>
  select(nconst, primaryName, primaryProfession)

director <- NAME_BASICS_NA |>
  filter(grepl("David Fincher", primaryName)) |>
  select(nconst, primaryName, primaryProfession)

# Step 4: Join actors and director with relevant movies
actors_movies <- TITLE_PRINCIPALS |>
  inner_join(actors, by = "nconst") |>
  inner_join(movies_ratings, by = "tconst")

director_movies <- TITLE_PRINCIPALS |>
  inner_join(director, by = "nconst") |>
  inner_join(movies_ratings, by = "tconst")

# Step 5: Calculate the average IMDb rating for each actor and the director
avg_ratings_actors <- actors_movies |>
  group_by(primaryName, primaryProfession) |>
  summarize(average_rating = mean(averageRating, na.rm = TRUE), total_movies = n()) |>
  arrange(desc(average_rating))

avg_ratings_director <- director_movies |>
  group_by(primaryName, primaryProfession) |>
  summarize(average_rating = mean(averageRating, na.rm = TRUE), total_movies = n()) |>
  arrange(desc(average_rating))

# Step 6: Combine actors and director data
combined_data <- bind_rows(avg_ratings_actors, avg_ratings_director)

# Step 7: Display the combined data as a table
kable(combined_data, caption = "Average IMDb Ratings for Zendaya, Blake Lively, and
David Fincher")

#step 8: Data Visualization:
#Bar plot to visualize the IMDb ratings

ggplot(combined_data, aes(x = reorder(primaryName, -average_rating), y =
average_rating, fill = primaryProfession)) +
  geom_bar(stat = "identity") +
  labs(title = "Average IMDb Ratings for Key Personnel",
       x = "Name", y = "Average IMDb Rating") +
```

```
  theme_minimal() +
  coord_flip()

# Different Genres contributed
combined_genre_data <- bind_rows(actors_movies, director_movies) |>
  separate_rows(genres, sep = ",")  # Split multiple genres into separate rows

# Group by person and genre to get the total IMDb ratings contribution by genre
contribution_data <- combined_genre_data |>
  group_by(primaryName, genres) |>
  summarize(total_success = sum(averageRating, na.rm = TRUE), total_movies = n()) |>
  ungroup()

# Create the stacked bar chart to show genre contribution to IMDb ratings
ggplot(contribution_data, aes(x = primaryName, y = total_success, fill = genres)) +
  geom_bar(stat = "identity", position = "stack") +  # Stacked bars
  labs(title = "Contribution of Genres to IMDb Ratings for Key Personnel",
       x = "Name", y = "Total IMDb Rating",
       fill = "Genres") +
  theme_minimal()
```

Now, that we have our data, Let’s dive into why Zendaya, Blake Lively, and David Fincher are the dream team for this psychological thriller!

Meet the Team:

**Zendaya**: She’s not just a star, she’s a supernova. With an average IMDb rating of 8.2, Zendaya has taken Hollywood by storm, from Euphoria to Dune. Her ability to effortlessly switch between emotional vulnerability and fierce intensity is exactly what our lead role demands. Plus, let’s be real—audiences can’t get enough of her, and neither can we!

**Blake Lively**: Class, charm, and a sprinkle of danger. Blake’s IMDb rating of 7.9 proves she knows how to bring depth to suspense-driven films (The Shallows, anyone?). She’s got that magnetic screen presence, where you can’t help but wonder, “What’s she going to do next?” Pair her with Zendaya, and we’ve got the ultimate power duo to keep audiences on the edge of their seats.

**David Fincher**: If we’re making a psychological thriller, we need the king of suspense—Fincher! With an 8.5 IMDb average, this guy practically breathes tension. From the eerie brilliance of Gone Girl to the chilling mind-bend of Fight Club, Fincher is the master at making you question everything. He’ll take our story and turn it into an unforgettable, mind-twisting journey.

## Nostalgia and Remakes
## Task 6: Finding a Classic Movie to Remake

```{r}

# Define X as the number of top movies you want to select
X <- 10  # Random number chosen

# Filter classic movies (before 1995) with high IMDb rating and number of votes
classic_movies <- TITLE_BASICS_CCOL |>
  filter(titleType == "movie" & startYear < 1995) |>
  inner_join(TITLE_RATING_SUCCESS, by = "tconst") |>
  filter(numVotes > 10000, averageRating >= 7.5) |>
  arrange(desc(averageRating)) |>
  head(X)

# Select necessary columns for review
classic_movies_selected <- classic_movies |>
  select(tconst, primaryTitle, startYear, averageRating, numVotes)
```

```
# Display the top X classic movies
print(classic_movies_selected)
# Filter for movies released after 1995 (potential remakes)
recent_movies <- TITLE_BASICS_CCOL |>
  filter(titleType == "movie" & startYear >= 1995) |>
  select(tconst, primaryTitle, startYear)

# Find remakes by comparing titles with the same name
remakes <- classic_movies_selected |>
  inner_join(recent_movies, by = "primaryTitle")
print(remakes)
# This will find movies with the same title

# Exclude remakes to get the movies that haven't been remade
no_remake_candidates <- classic_movies_selected |>
  anti_join(remakes, by = "primaryTitle")

# Display the list of movies that have not been remade
print(no_remake_candidates)


```
> To remake the movie that has not been made for past 25 years we choose **The Shawshank Redemption** with my key talent: Zendaya, Blake Lively, and David Fincher.

```{r}
# First, find the tconst for The Shawshank Redemption (if you already have this, skip this step)
shawshank_tconst <- TITLE_BASICS_CCOL |>
  filter(primaryTitle == "The Shawshank Redemption") |>
  select(tconst) |>
  pull()

# Retrieve key personnel (actors, directors, writers) from The Shawshank Redemption
shawshank_personnel <- TITLE_PRINCIPALS |>
  filter(tconst == shawshank_tconst) |>
  inner_join(NAME_BASICS_NA, by = "nconst") |>
  filter(grepl("actor|director|writer", primaryProfession))

# Check if key personnel are still alive (i.e., no deathYear or NA deathYear)
alive_personnel <- shawshank_personnel |>
  filter(is.na(deathYear) | deathYear == "\\N") |>
  select(primaryName, primaryProfession, birthYear, deathYear)

if (nrow(alive_personnel) > 0) {
  print("The following key personnel are still alive:")
  print(alive_personnel)

  #unstatement if want to opload a csv file
  #write.csv(alive_personnel, "shawshank_alive_personnel.csv", row.names = FALSE)

  # Example message to the legal department
  message_to_legal <- paste("Key personnel from The Shawshank Redemption are still alive. Please review the attached CSV for potential legal actions regarding securing rights for a remake.")
  print(message_to_legal)

} else {
  print("No key personnel from The Shawshank Redemption are still alive.")
}
```


## Putting It Together
## Task 7

> ## Elevator Pitch:
>

> Imagine... a psychological thriller that combines the electrifying talents of Zendaya, Blake Lively, and the directorial genius of David Fincher. With Zendaya’s star power and proven box office draw (IMDb rating: 8.2) paired with Blake Lively’s knack for high-tension drama (IMDb rating: 7.9), we have a duo that will keep audiences glued to their seats. Under the masterful guidance of David Fincher (IMDb rating: 8.5), a director who has made thrillers like Gone Girl and Fight Club modern classics, this film will deliver an intense, mind-bending story that guarantees repeat viewership.
>
>In a genre where suspense and psychological depth reign supreme, Fincher’s precision storytelling will be elevated by Zendaya and Blake’s dynamic performances. We’ve seen the market for psychological thrillers explode in recent years, and with this all-star team, our film is poised to become the next big hit.
>
> **Teaser Trailer Formula**:
From director David Fincher, the visionary mind behind Gone Girl and Fight Club,
And from Zendaya, the beloved star of Euphoria and Dune,
And from Blake Lively, Hollywood icon of high-stakes drama in The Shallows,
Comes the timeless tale: Mind Games
A story of trust, betrayal, and survival,
Coming soon to a theater near you.
>
> With this winning combination of talent and a gripping story, our psychological thriller is set to captivate audiences and dominate the box office.
>
>
> - Aachal Ghimire