Bollywood Theatre Release Analysis

Steps Taken

Data Collection

↓

Dataset Cleaning

↓

Data Processing

↓

Data Analysis

↓

Data Visualisation

Tools Used

- Spreadsheets
- BeautifulSoup
- Pandas
- Numpy
- Matplotlib
- Seaborn
- ChatGPT

Abstract

The project "Bollywood Theatre Release Analysis" presents a comprehensive study of Bollywood movie releases, focusing on various aspects that influence their performance and success in the theatre. This analysis was initiated by collecting data from renowned sources such as IMDb and Wikipedia using advanced web scraping techniques facilitated by Python libraries like BeautifulSoup. This approach ensured a robust data collection process encompassing critical information on movie releases, box office statistics, genres, directors, and actors.

The scraped data underwent a rigorous cleaning process to rectify inconsistencies and inaccuracies, leading to a well-structured and reliable dataset. Subsequently, data processing and manipulation were carried out using tools such as Pandas and NumPy to derive meaningful insights from the dataset. The analysis aimed to uncover patterns and trends in Bollywood movie releases, with a particular focus on box office performance, genre preferences, director and actor impact on movie success, and the factors contributing to the recent downturn in the Bollywood industry.

A significant part of the analysis involved identifying the reasons behind the decline in Bollywood's appeal and box office performance. Factors such as changing audience preferences, the impact of digital streaming platforms, and content quality were explored to provide a comprehensive understanding of the challenges faced by the industry.

To effectively communicate the findings, the project utilised data visualisation tools like Matplotlib and Seaborn, creating visually appealing and informative charts and graphs. These visualisations highlighted key insights and trends, making the analysis accessible and understandable to a broad audience.

The project employed a diverse set of tools and technologies, including Spreadsheets for initial data handling, BeautifulSoup for web scraping, Pandas and NumPy for data processing, and Matplotlib and Seaborn for data visualisation. The use of ChatGPT also facilitated advanced analytical capabilities and the generation of insights.

This project not only provides a detailed analysis of Bollywood movie releases but also offers valuable recommendations for stakeholders in the industry to address the challenges and adapt to the evolving landscape of the entertainment sector.

Data Collection

In this project, finding the right data was quite challenging due to the lack of suitable open-source datasets. Although various datasets were available, they were insufficient for the detailed analysis I intended to perform. Therefore, I decided to extract data directly from reliable sources such as IMDb and Wikipedia.

This extraction was initiated using advanced web scraping techniques facilitated by Python libraries like BeautifulSoup. I developed two different functions tailored for each website: one for IMDb and another for Wikipedia. The scraped data was then stored in two separate CSV files for further processing.

For the extraction of data from IMDb, I saved the source HTML code, which included detailed information on over 2,000 movies, in a text file. For data extraction from Wikipedia, I utilised ChatGPT to generate the Wikipedia URLs for all those 2,000 movies.

From Wikipedia, I extracted the following fields: Movie Name, URL, Director, Producer, Writer, Casting, Budget, and Box Office. From IMDb, the extracted fields included Movie Name, Duration, Year, Rating, and Rating Count. These datasets were then organised into two separate tables, setting the stage for further cleaning and processing of the data.

TOOLS USED: Pandas, BeautifulSoup, Spreadsheets

In next page the source code of functions for Web Scraping is given.

Source Code For Web Scraping from IMDb

```python
from bs4 import BeautifulSoup
import pandas as pd
import os

def extract_page_details(input_folder_path, output_excel_path):
    results = []

    for filename in os.listdir(input_folder_path):
        if filename.endswith(".txt"):  # Process only text files
            file_path = os.path.join(input_folder_path, filename)
            with open(file_path, 'r', encoding='utf-8') as file:
                html_snippet = file.read()

            soup = BeautifulSoup(html_snippet, 'html.parser')

            movie = ''
            year = ''
            duration = ''
            rating = ''
            rating_count = ''

            elements = soup.find_all(class_="sc-74bf520e-3 klvfeN dli-parent")
            for element in elements:
                movie_title = element.find(class_="ipc-title__text")
                if movie_title:
                    movie = movie_title.text.strip()

                metadata_items = element.find_all(class_="sc-b189961a-8 kLaxqf dli-title-metadata-item")
                for metadata_item in metadata_items:
                    text = metadata_item.text.strip()
                    if 'h' in text:
                        duration = text
                    elif text.isdigit() and len(text) == 4:
                        year = text

                rating_element = element.find(class_="ipc-rating-star--rating")
                if rating_element:
                    rating = rating_element.text.strip()

                rating_count_element = element.find(class_="ipc-rating-star--voteCount")
                if rating_count_element:
                    rating_count = rating_count_element.text.strip()

                results.append({'Movie_Name': movie, 'Duration': duration, 'Year': year, 'Rating': rating, 'Rating_count': rating_count})

    df = pd.DataFrame(results)
    df.to_excel(output_excel_path, index=False)
    print(f"Excel file saved successfully: {output_excel_path}")

input_folder_path = '/Users/apple/Desktop/movie'
output_excel_path = '/Users/apple/Desktop/Activity_8.xlsx'
extract_page_details(input_folder_path, output_excel_path)
```

Function For Web Scraping from WIKIPEDIA

```python
def extract_page_details_from_excel(input_excel_path, output_excel_path):
    # Read the input Excel file
    df = pd.read_excel(input_excel_path)
    # Read the input Excel file
    print("Column names in the input file:", df.columns.tolist())
    results = []
    for index, row in df.iterrows():
        movie_name = row['Movies']
        wikipedia_url = row['Link']
        movie = movie_name
        writter = ''
        director = ''
        producer = ''
        cast = ''
        bud = ''
        earn = ''
        musician = ''
        try:
            response = requests.get(wikipedia_url)
            response.raise_for_status()  # Raise an HTTPError for bad responses (4xx and 5xx)
            soup = BeautifulSoup(response.content, 'html.parser')
            infobox = soup.find(class_="infobox vevent")
            if infobox:
                # Extract data from infobox
                metadata_items = infobox.find_all('tr')
                for item in metadata_items:
                    th = item.find('th')
                    td = item.find('td')
                    if th and td:
                        header = th.text.strip()
                        value = td.text.strip()
                        if 'directed by' in header.lower():
                            director = value
                        elif 'produced by' in header.lower():
                            producer = value
                        elif 'starring' in header.lower():
                            cast = value
                        elif 'budget' in header.lower():
                            bud = value
                        elif 'box office' in header.lower():
                            earn = value
                        elif 'music by' in header.lower():
                            musician = value
                        elif 'written by' in header.lower():
                            writter = value
        except requests.RequestException as e:
            print(f"Failed to retrieve page {wikipedia_url}: {e}")
        except Exception as e
            print(f"An error occurred while processing {wikipedia_url}: {e}")
        # Append the extracted data, including the URL
        results.append({
            'Movie_Name': movie,'URL': wikipedia_url,'Director': director,'Producer': producer,'Writer': writter,'Casting': cast,
            'Music By': musician,'Budget': bud,'Box Office': earn
        })
```

Data Cleaning

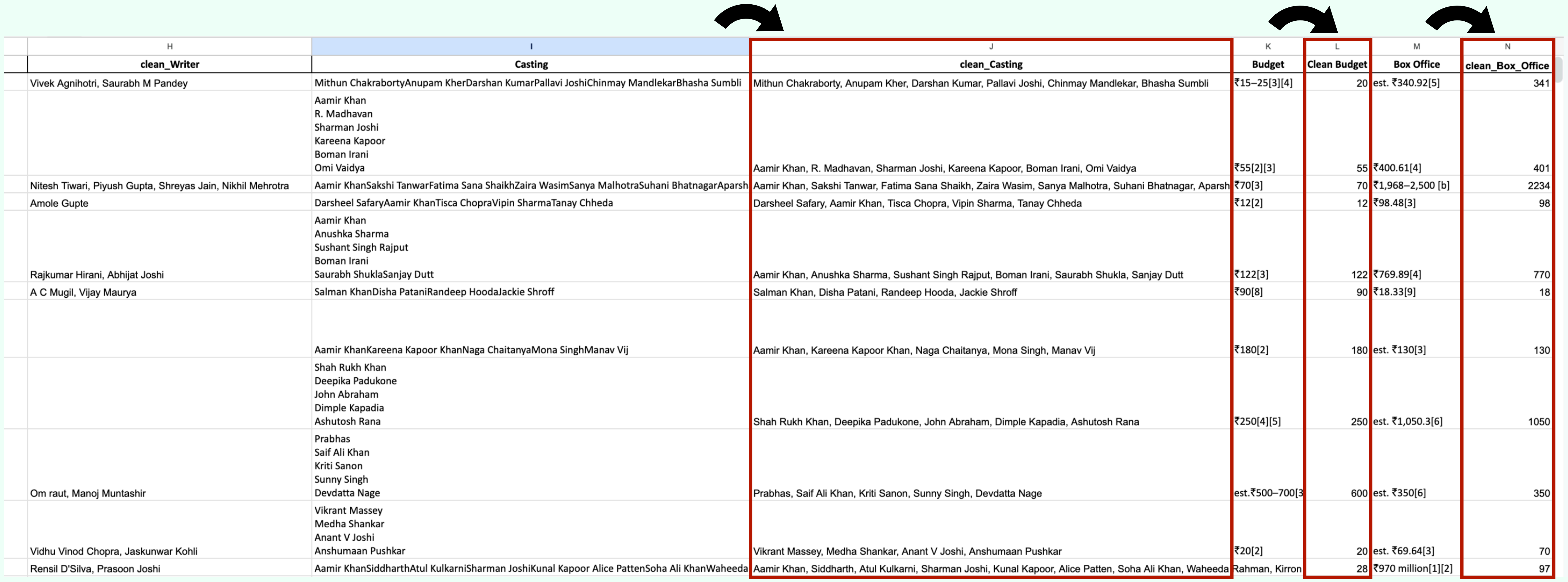
Since Now we have two tables with different fields, we need to clean them and make them consistent.

Cleaning each column individually is crucial due to the unique characteristics of each dataset. As seen in the data below, I created a clean column for each relevant data field. This process involved cleaning each column according to its specific data type and context.

For instance, the casting column contained entries in an unsuitable format for analysis. For example, a cell with the value "Mithun ChakrabortyAnupam Kher" was transformed into "Mithun Chakraborty, Anupam Kher." This format is more organised and makes the data easier to interpret and analyse. Similarly, the Box Office column exhibited inconsistencies in its format and contained non-numeric values. For instance, one of the cells had a value like "970 million[1][2]," which posed difficulties in analysis. I converted this value into a standardised format, "97," where the unit for all entries in this column is "crore".

Tools Used: Spreadsheets, ChatGPT

Similarly I have cleaned all my columns and made the data consistent.



H	I	J	K	L	M	N
clean_Writer	Casting	clean_Casting	Budget	Clean Budget	Box Office	clean_Box_Office
Vivek Agnihotri, Saurabh M Pandey	Mithun ChakrabortyAnupam KherDarshan KumarPallavi JoshiChinmay MandlekarBhasha Sumbli	Mithun Chakraborty, Anupam Kher, Darshan Kumar, Pallavi Joshi, Chinmay Mandlekar, Bhasha Sumbli	₹15–25[3][4]	20	est. ₹340.92[5]	341
	Aamir Khan R. Madhavan Sharman Joshi Kareena Kapoor Boman Irani Omi Vaidya	Aamir Khan, R. Madhavan, Sharman Joshi, Kareena Kapoor, Boman Irani, Omi Vaidya	₹55[2][3]	55	₹400.61[4]	401
Nitesh Tiwari, Piyush Gupta, Shreyas Jain, Nikhil Mehrotra	Aamir KhanSakshi TanwarFatima Sana ShaikhZaira WasimSanya MalhotraSuhani BhatnagarAparsh	Aamir Khan, Sakshi Tanwar, Fatima Sana Shaikh, Zaira Wasim, Sanya Malhotra, Suhani Bhatnagar, Aparsh	₹70[3]	70	₹1,968–2,500 [b]	2234
Amole Gupte	Darsheel SafaryAamir KhanTisca ChopraVipin SharmaTanay Chheda	Darsheel Safary, Aamir Khan, Tisca Chopra, Vipin Sharma, Tanay Chheda	₹12[2]	12	₹98.48[3]	98
Rajkumar Hirani, Abhijat Joshi	Aamir Khan Anushka Sharma Sushant Singh Rajput Boman Irani Saurabh ShuklaSanjay Dutt	Aamir Khan, Anushka Sharma, Sushant Singh Rajput, Boman Irani, Saurabh Shukla, Sanjay Dutt	₹122[3]	122	₹769.89[4]	770
A C Mugil, Vijay Maurya	Salman KhanDisha PataniRandeep HoodaJackie Shroff	Salman Khan, Disha Patani, Randeep Hooda, Jackie Shroff	₹90[8]	90	₹18.33[9]	18
	Aamir KhanKareena Kapoor KhanNaga ChaitanyaMona SinghManav Vij	Aamir Khan, Kareena Kapoor Khan, Naga Chaitanya, Mona Singh, Manav Vij	₹180[2]	180	est. ₹130[3]	130
	Shah Rukh Khan Deepika Padukone John Abraham Dimple Kapadia Ashutosh Rana	Shah Rukh Khan, Deepika Padukone, John Abraham, Dimple Kapadia, Ashutosh Rana	₹250[4][5]	250	est. ₹1,050.3[6]	1050
Om raut, Manoj Muntashir	Prabhas Saif Ali Khan Kriti Sanon Sunny Singh Devdatta Nage	Prabhas, Saif Ali Khan, Kriti Sanon, Sunny Singh, Devdatta Nage	est.₹500–700[3	600	est. ₹350[6]	350
Vidhu Vinod Chopra, Jaskunwar Kohli	Vikrant Massey Medha Shankar Anant V Joshi Anshumaan Pushkar	Vikrant Massey, Medha Shankar, Anant V Joshi, Anshumaan Pushkar	₹20[2]	20	est. ₹69.64[3]	70
Rensil D'Silva, Prasoon Joshi	Aamir KhanSiddharthAtul KulkarniSharman JoshiKunal Kapoor Alice PattenSoha Ali KhanWaheeda	Aamir Khan, Siddharth, Atul Kulkarni, Sharman Joshi, Kunal Kapoor, Alice Patten, Soha Ali Khan, Waheeda	Rahman, Kirron	28	₹970 million[1][2]	97

Data Processing

Since we now have two cleaned tables, the next step is to join them. To achieve this, a suitable join key is necessary. Using ChatGPT, I created an additional column in the IMDb table named "URL," which contains the corresponding Wikipedia URL for each movie.

While both tables already contain a "Movie Name" column, it cannot be used as a join key due to the possibility of multiple movies sharing the same name. However, the recently created "URL" column in the IMDb table, which mirrors the "URL" column in the Wikipedia table, serves as a unique identifier for each movie. An example of such a URL is: `https://en.wikipedia.org/wiki/Dilwale_Dulhania_Le_Jayenge`.

To establish a common key for merging the tables, I created a new column called "ID" in both tables. This column has values after the last "/" in the URL, effectively providing a unique identifier for each movie entry. With this appropriate Joining Key in both tables, we performed an inner join using Pandas to merge the tables.

Additionally, I created three more fields in the merged table, "Inflated Budget" , "Inflated Box Office," and "Performance". First two fields allow for an analysis of financial aspects independent of the currency value during the release year whereas "Performance" columns gives tags to every films based on there Box Office Performance.

Tools Used: Pandas, ChatGPT, Numpy, Spreadsheets

Codes For Data Processing Operations

Code for merging IMDb and WIKI table

```
main_df=pd.merge(IMDb_df, WIKI_df, on='ID', how='inner')
df.head()
```

Codes For Data Processing Operations

Code for creating ‘Inflated Budget’ and ‘Inflated Box Office’ fields

```
inflation_rate = 0.06
current_year = datetime.now().year
main_df['years_since_price'] = current_year - main_df['Year']
main_df['Inflated_Box_Office'] = main_df['Box_Office'] * ((1 + inflation_rate) **
main_df['years_since_price'])
main_df['Inflated_Budget'] = main_df['Budget'] * ((1 + inflation_rate) **
main_df['years_since_price'])
main_df.drop(columns='years_since_price', inplace=True)
main_df['Inflated_Box_Office']=main_df['Inflated_Box_Office'].round(2)
main_df['Inflated_Budget']=main_df['Inflated_Budget'].round(2)
main_df.head()
```

Code for creating ‘Performance’ field

```
main_df['Difference'] = main_df['Inflated_Box_Office'] - main_df['Inflated_Budget']
conditions = [
    (main_df['Difference'] < 0), #case1
    (main_df['Difference'] >= 0) & (main_df['Difference'] < 0.25 * main_df['Inflated_Budget']), #case2
    (main_df['Difference'] >= 0.25 * main_df['Inflated_Budget']) & (main_df['Difference'] < 0.75 *
    main_df[‘Inflated_Budget']), #case3
    (main_df['Difference'] >= 0.75 * main_df['Inflated_Budget']) & (main_df['Difference'] <
    main_df[‘Inflated_Budget']), #case4
    (main_df['Difference'] >= main_df['Inflated_Budget']) & (main_df['Difference'] < 3 *
    main_df[‘Inflated_Budget']), #case5
    ((main_df['Difference'] >= 6 * main_df['Inflated_Budget']) & (main_df['Inflated_Box_Office'] < 300)) |
    ((main_df['Difference'] >= 3 * main_df['Inflated_Budget']) & (main_df['Difference'] < 6 *
    main_df[‘Inflated_Budget'])), #case6
    (main_df['Difference'] >= 6 * main_df[‘Inflated_Budget']) #case7
]
performance_labels = ['Flop', 'Average', 'Semihit', 'Hit', 'Superhit', 'Blockbuster', 'All Time
BlockBuster']
main_df['Performance'] = np.select(conditions, performance_labels, default='Unknown')
main_df.drop(columns='Difference', inplace=True)
```

Data Analysis

Since we now have two cleaned tables, the next step is to join them. To achieve this, a suitable join key is necessary. Using ChatGPT, I created an additional column in the IMDb table named "URL," which contains the corresponding Wikipedia URL for each movie.

While both tables already contain a "Movie Name" column, it cannot be used as a join key due to the possibility of multiple movies sharing the same name. However, the recently created "URL" column in the IMDb table, which mirrors the "URL" column in the Wikipedia table, serves as a unique identifier for each movie. An example of such a URL is: `https://en.wikipedia.org/wiki/Dilwale_Dulhania_Le_Jayenge`.

To establish a common key for merging the tables, I created a new column called "ID" in both tables. This column has values after the last "/" in the URL, effectively providing a unique identifier for each movie entry. With this appropriate Joining Key in both tables, we performed an inner join using Pandas to merge the tables.

Additionally, I created three more fields in the merged table, "Inflated Budget" , "Inflated Box Office," and "Performance". First two fields allow for an analysis of financial aspects independent of the currency value during the release year whereas "Performance" columns gives tags to every films based on there Box Office Performance.

Tools Used: Pandas, ChatGPT, Numpy, Spreadsheets, Matplotlib, Seaborn

Top 10 Movies based on Box Office

```
top_10_movies = main_df.nlargest(10, 'Box_Office')
top_10_movies.loc[:,['Movie_Name', 'Box_Office']]
```

Movie_Name	Box_Office
Dangal	2234.0
Jawan	1148.0
Pathaan	1050.0
Bajrangi Bhaijaan	944.0
Animal	918.0
Secret Superstar	905.0
PK	770.0
Gadar 2	691.0
Sultan	623.0
Sanju	587.0

Top 10 Movies after adjusting inflation

```
top_10_movies = main_df.nlargest(10, ‘Inflated_Box_Office')
top_10_movies.loc[:,['Movie_Name', ‘Inflated_Box_Office']]
```

Movie_Name	Inflated_Box_Office
Dangal	3560.66
Bajrangi Bhaijaan	1594.87
Hum Aapke Hain Koun...!	1435.87
PK	1378.95
Secret Superstar	1360.79
Jawan	1216.88
Disco Dancer	1167.26
Awara	1125.77
Pathaan	1113.00
Dilwale Dulhania Le Jayenge	1083.68

Top 10 Movies based on IMDb rating

```python
movie_list = main_df.loc[:,['Movie_Name', 'Rating']]
highly_rated = movie_list.sort_values(by='Rating', ascending=False).head(10).round(1)
print(highly_rated)
```

Movie_Name	Rating
12th Fail	8.9
Rocketry: The Nambi Effect	8.7
The Kashmir Files	8.6
Dosti	8.5
Laapataa Ladies	8.5
Black Friday	8.4
3 Idiots	8.4
Chandu Champion	8.3
Satya	8.3
Chhichhore	8.3

Top 10 least rated movies based on IMDb rating

```python
movie_list = main_df.loc[:,['Movie_Name', 'Rating']]
highly_rated = movie_list.sort_values(by='Rating', ascending=True).head(10).round(1)
print(highly_rated)
```

Movie_Name	Rating
Ram Gopal Varma Ki Aag	1.4
Prem Aggan	1.7
Himmatwala	1.7
Humshakals	1.7
Kyaa Kool Hain Hum 3	1.9
Race 3	1.9
Radhe	1.9
Jackpot	2.0
Drona	2.0
Namaste England	2.1

Top 10 Actors of 90's based on Movie Rating

```
filtered_main = main_df[(main_df['Year'] > 1940) & (main_df['Year'] < 2000)]

filtered_main_exploded = filtered_main.assign(Casting=filtered_main['Casting'].str.split(', 
')).explode('Casting')
actor_list = filtered_main_exploded.groupby('Casting').agg(
    average_rating=('Rating', 'mean'),
    movie_count1=('Inflated_Box_Office', 'count')
)
actor_list=actor_list[actor_list['movie_count1']>=10]
top_artists = actor_list.sort_values(by='average_rating', ascending=False).head(10).round(1)
print(top_artists)
```

Actors	Average_Movie_Rating
Shashi Kapoor	7.2
Vinod Khanna	6.8
Pran	6.8
Jackie Shroff	6.8
Rishi Kapoor	6.7
Sunny Deol	6.7
Paresh Rawal	6.6
Amitabh Bachchan	6.6
Aamir Khan	6.6
Kajol	6.5

Top 10 Actors of 90's based on Box Office

```
filtered_main = main_df[(main_df['Year'] > 1940) & (main_df['Year'] < 2000)]

filtered_main_exploded = filtered_main.assign(Casting=filtered_main['Casting'].str.split(', 
')).explode('Casting')
actor_list = filtered_main_exploded.groupby('Casting').agg(
    average_box_office=('Inflated_Box_Office', 'mean'),
    movie_count1=('Inflated_Box_Office', 'count')
)
actor_list=actor_list[actor_list['movie_count1']>=10]

top_artists = actor_list.sort_values(by='average_box_office', ascending=False).head(10).round(1)
print(top_artists)
```

Note: We Have Adjusted The Box Office with Inflation and the value are in crore.

Actors	Average_BOX_Office
Kajol	248.8
Shashi Kapoor	213.9
Karisma Kapoor	196.4
Salman Khan	189.8
Madhuri Dixit	181.0
Shah Rukh Khan	175.4
Rishi Kapoor	160.3
Vinod Khanna	144.9
Jackie Shroff	141.1
Sunny Deol	133.4

Top 10 Actors with poorly rated films

```
filtered_main_exploded = main_df.assign(Casting=main_df['Casting'].str.split(', ')).explode('Casting')
actor_list = filtered_main_exploded.groupby('Casting').agg(
    average_rating=('Rating', 'mean'),
    movie_count1=('Inflated_Box_Office', 'count')
)
actor_list=actor_list[actor_list['movie_count1']>=6]

top_artists = actor_list.sort_values(by='average_rating', ascending=True).head(10).round(1)
print(top_artists)
```

Actors	Rating
Sunny Leone	2.8
Tiger Shroff	3.6
Rahul Dev	4.5
Amrita Arora	4.5
Ayesha Takia	4.8
Zayed Khan	4.8
Pulkit Samrat	4.9
Sohail Khan	4.9
Arjun Kapoor	4.9
Jacqueline Fernandez	4.9

Top 10 Actors with highly rated films

```python
filtered_main_exploded = main_df.assign(Casting=main_df['Casting'].str.split(', ')).explode('Casting')
actor_list = filtered_main_exploded.groupby('Casting').agg(
    average_rating=('Rating', 'mean'),
    movie_count1=('Inflated_Box_Office', 'count')
)
actor_list=actor_list[actor_list['movie_count1']>=6]

top_artists = actor_list.sort_values(by='average_rating', ascending=False).head(10).round(1)
print(top_artists)
```

Actors	Rating
Nargis	7.7
Raj Kapoor	7.6
Sushant Singh Rajput	7.5
Dilip Kumar	7.5
Vinay Pathak	7.2
Shashi Kapoor	7.2
Neetu Singh	7.2
Swara Bhaskar	7.1
Radhika Apte	7.1
Rajat Kapoor	7.1

Actors associated with genre: Rating

```
filtered_main_exploded = main_df.assign(Casting=main_df['Casting'].str.split(', ')).explode('Casting')
actor_list= filtered_main_exploded.groupby(['Casting','Genre']).agg(
    average_rating=('Rating', 'mean'),
    movie_count=('Inflated_Box_Office', 'count')
)
actor_list=actor_list[actor_list['movie_count']>=10]

top_actors = actor_list.sort_values(by='average_rating', ascending=False).head(20).round(1)
print(top_actors)
```

Genre	Genre	Rating
Shah Rukh Khan	Romantic	7.1
Amitabh Bachchan	Drama	6.9
Rishi Kapoor	Drama	6.8
Boman Irani	Comedy	6.7
Amitabh Bachchan	Thriller	6.6
Paresh Rawal	Comedy	6.6
Shah Rukh Khan	Drama	6.4
Shah Rukh Khan	Action	6.3
Rajpal Yadav	Comedy	6.3
Preity Zinta	Romantic	6.1
Ajay Devgn	Drama	6.0
Sanjay Dutt	Comedy	6.0
Amrish Puri	Action	6.0
Anupam Kher	Comedy	6.0
Akshay Kumar	Comedy	5.8
Arshad Warsi	Comedy	5.8
Johnny Lever	Comedy	5.8
John Abraham	Action	5.8
Sunny Deol	Action	5.8
Govinda	Comedy	5.8

Explanation: This table provides insights into the genres for which each actor is most recognised. For instance, in the first record, Shah Rukh Khan is predominantly known for his highly-rated romantic movies. The "Rating" column reflects the average rating of all the movies within the dataset that an actor has performed in, specific to the respective genre. For example, the average rating for comedy movies featuring Govinda stands at 5.8.

Shah Rukh Khan - Romantic
Amitabh Bachchan - Drama
Boman Irani - Comedy
Amitabh Bachchan - Thriller
Shah Rukh Khan - Action

Note: Movie Rating's are based on IMDb rating.

Actors associated with genre: Profit Returns

```
filtered_main_exploded['Profit'] = filtered_main_exploded['Inflated_Box_Office'] /
filtered_main_exploded['Inflated_Budget']

actor_list = filtered_main_exploded.groupby(['Casting', 'Genre']).agg(
    Times_Return=('Profit', 'mean'),
    movie_count=('Inflated_Box_Office', 'count')
)

actor_list = actor_list[actor_list['movie_count'] >= 10]

top_actors = actor_list.sort_values(by='Times_Return', ascending=False).head(20).round(1)

print(top_actors)
```

Genre	Genre	Profit _Returns
Shah Rukh Khan	Romance	7.5
Salman Khan	Romance	4.9
Amitabh Bachchan	Action	4.1
Amrish Puri	Action	4.1
Shakti Kapoor	Action	4.0
Sunny Deol	Action	3.8
Amitabh Bachchan	Thriller	3.8
Amitabh Bachchan	Drama	3.8
Jackie Shroff	Action	3.7
Sonakshi Sinha	Action	3.6
Anil Kapoor	Action	3.4
Govinda	Comedy	3.3
Boman Irani	Comedy	3.3
Salman Khan	Action	3.2
Shah Rukh Khan	Action	3.3
Salman Khan	Drama	2.8

Explanation: This table offers valuable insights into the genres where each actor shines the most in terms of profitability. For example, in the first record, Shah Rukh Khan demonstrates exceptional profitability in romantic movies. The 'Profit_Returns' column indicates how many times the budget a movie earns within its specific genre. For instance, comedy films featuring Govinda yield a Profit_Returns of 3.3 times the budget.

Shah Rukh Khan - Romantic
Amitabh Bachchan -Action
Govinda - Comedy
Amitabh Bachchan - Thriller
Salman Khan - Drama

Note: Last Record I.e for Drama genre is at 20th position, the records between 15th and 20th row has not been included due to space issues.

Genre Analysis

```
genre_list = main_df.groupby('Genre').agg(
    average_rating=('Rating', 'mean'),
    movie_count1=('Inflated_Box_Office', 'count')
)
genre_list=genre_list[genre_list['movie_count1']>=40]
top_genre = genre_list.sort_values(by='average_rating', ascending=False).head(10).round(1)
print(top_genre)
```

Genre	Rating	Movie_Count
Drama	6.5	236
Thriller	6.3	147
Romance	5.8	172
Comedy	5.8	214
Action	5.5	209

Analysis: This table stands as a testament to Bollywood's deep roots in family and drama films, which have played a pivotal role in its establishment. While Bollywood is home to numerous action films, they don't have left a mark as profound as the dramas. Thrillers, romances, and comedies also hold a special place, embodying the diverse essence of Bollywood.

Top 10 Director with highly rated films

```
filtered_main_exploded = main_df.assign(Director=main_df['Director'].str.split(', ')).explode('Director')

director_list = filtered_main_exploded.groupby('Director').agg(
    rating_avg=('Rating', 'mean'),
    movie_count=('Inflated_Box_Office', 'count')
)

director_list = director_list[director_list['movie_count'] >= 6]
director_list = director_list.loc[:, ['rating_avg', 'movie_count']]

top_director = director_list.sort_values(by='rating_avg', ascending=False).head(10).round(1)

print(top_director)
```

Directors	Rating
Rajkumar Hirani	7.8
Raj Kapoor	7.4
Yash Chopra	7.4
Dibakar Banerjee	7.2
Anurag Kashyap	7.2
Sanjay Leela Bhansali	7.1
Kabir Khan	7.0
Hansal Mehta	6.9
Rajkumar Santoshi	6.8
Imtiaz Ali	6.8

Top 10 Director: Box Office

```
filtered_main_exploded = main_df.assign(Director=main_df['Director'].str.split(', ')).explode('Director')

director_list = filtered_main_exploded.groupby('Director').agg(
    avg_Box_Office=('Inflated_Box_Office', 'mean'),
    movie_count=('Inflated_Box_Office', 'count')
)

director_list = director_list[director_list['movie_count'] >= 6]
director_list = director_list.loc[:, ['avg_Box_Office']]

top_director = director_list.sort_values(by='avg_Box_Office', ascending=False).head(10).round(1)

print(top_director)
```

Director	Average_Box_Office
Rajkumar Hirani	701.1
Kabir Khan	485.6
Siddharth Anand	427.3
Raj Kapoor	421.7
Rohit Shetty	360.9
Sanjay Leela Bhansali	359.3
Karan Johar	345.9
Rakesh Roshan	273.3
Prabhu Deva	270.6
Anil Sharma	247.0

Top 10 writer with highly rated films

```python
filtered_main_exploded = main_df.assign(Writer=main_df['Writer'].str.split(', ')).explode('Writer')

director_list = filtered_main_exploded.groupby('Writer').agg(
    rating_avg=('Rating', 'mean'),
    movie_count=('Inflated_Box_Office', 'count')
)

writer_list = writer_list[writer_list['movie_count'] >= 6]
writer_list = writer_list.loc[:, ['rating_avg', 'movie_count']]

top_writer = writer_list.sort_values(by='rating_avg', ascending=False).head(10).round(1)

print(top_writer)
```

Writer	Rating
Salim Javed	7.5
Juhi Chaturvedi	7.4
Abhijat Joshi	7.4
Rajkumar Hirani	7.3
Vidhu Vinod Chopra	7.3
Neeraj Pandey	7.2
Gulzar	7.2
Nitesh Tiwari	7.2
Anurag Kashyap	7.2
Sriram Raghavan	7.1

Top 10 Writer: Box Office

```python
filtered_main_exploded = main_df.assign(Writer=main_df['Writer'].str.split(', ')).explode('Writer')

Writer_list = filtered_main_exploded.groupby('Writer').agg(
    avg_Box_Office=('Inflated_Box_Office', 'mean'),
    movie_count=('Inflated_Box_Office', 'count')
)

Writer_list = Writer_list[Writer_list['movie_count'] >= 5]
Writer_list = Writer_list.loc[:, ['avg_Box_Office']]

top_writer = Writer_list.sort_values(by='avg_Box_Office', ascending=False).head(10).round(1)

print(top_writer)
```

Writer	Average_Box_Office
Nitesh Tiwari	719.3
Nikhil Mehrotra	714.5
Rajkumar Hirani	629.6
Abhijat Joshi	622.9
Ali Abbas Zafar	539
Aditya Chopra	420.7
Farhad Sajid	393.8
Yunus Sajawal	356.8
Vijay Krishna Acharya	313.9
Shiraz Ahmed	294.3

Bollywood: Struggling since pandemic?

To determine whether Bollywood is experiencing a decline, I will conduct an analysis of the Performance table. This analysis will involve comparing the percentage of movies that achieved different performance categories—such as Flop, Hit, and Blockbuster—during the last decade (2010-2019) with their counterparts in the post-pandemic period (2020-2024).

By examining the shifts in these percentages, I aim to identify trends that indicate whether Bollywood is facing challenges in maintaining its historical success rates or if it continues to produce a significant number of hits and blockbusters. This comparative analysis will provide insights into the current state of the industry and help to evaluate whether Bollywood is struggling or thriving in the post-pandemic era.

```
filtered_main = main_df[(main_df['Year'] > 2010) & (main_df['Year'] < 2020)]
total_movie = filtered_main['Movie_Name'].count()

performance_stats = filtered_main.groupby('Performance').agg(
    movie_count=('Inflated_Box_Office', 'count'),
).reset_index()

performance_stats['movie_count'] = ((performance_stats['movie_count'] / total_movie) * 100).round(2)

top = performance_stats.sort_values(by='movie_count', ascending=False)

print(top)
```

Performance	Last_Decade
Flop	29.54
Average	9.49
Semihit	14.98
Hit	3.16
Superhit	23.84
Blockbuster	16.67
All Time Blockbuster	2.32

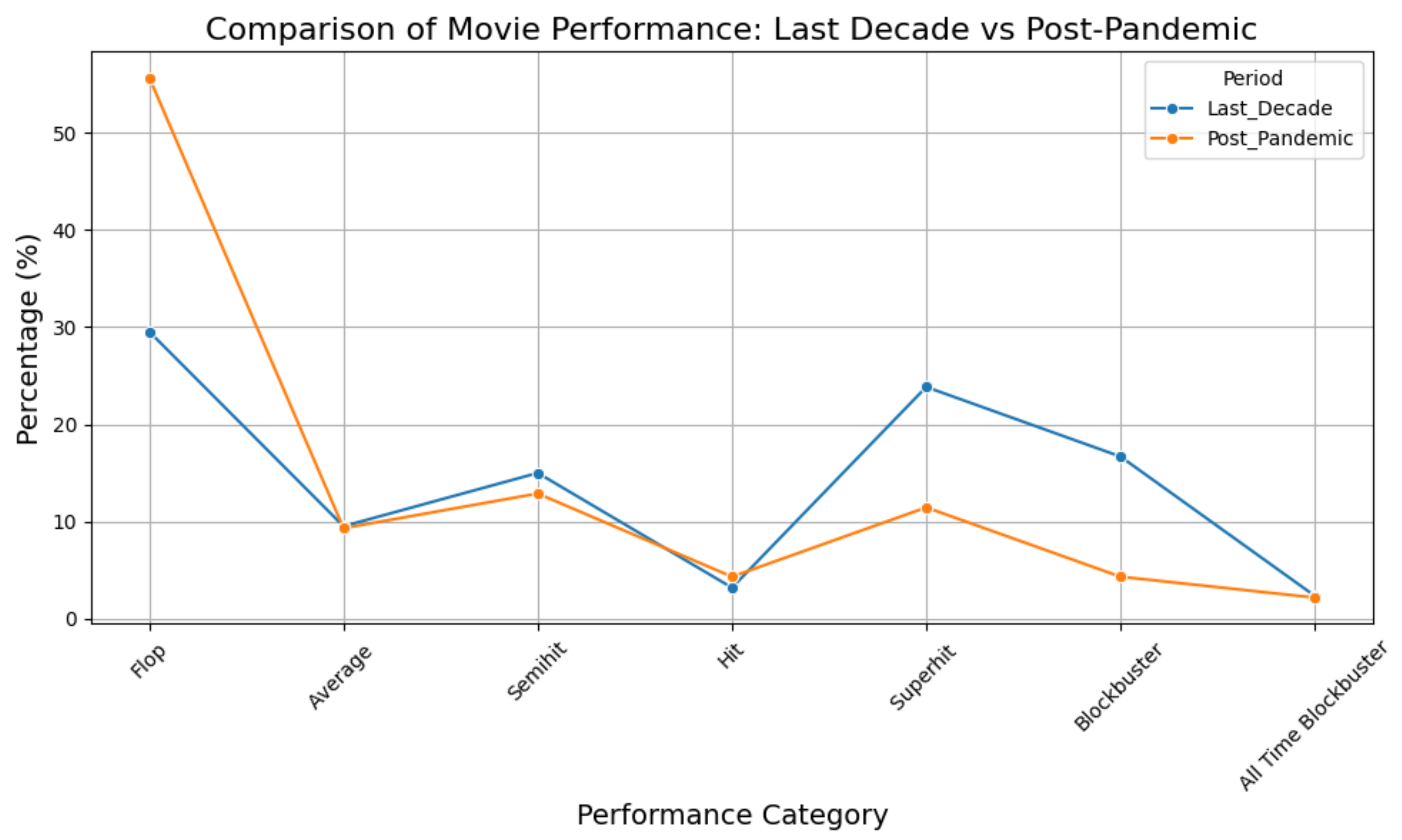
Similarly we find table for movies since 2020 and merged them. Now we are going to use matplotlib and seaborn to visualise the analysis.

Analysis 1: Significant increase in flop movies after 2019

```python
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

df=pd.read_csv(‘Performance_Camparison.csv')

df_melted = df.melt(id_vars='Performance', value_vars=['Last_Decade', 'Post_Pandemic'],
                    var_name='Period', value_name='Percentage')

plt.figure(figsize=(10, 6))
sns.lineplot(data=df_melted, x='Performance', y='Percentage', hue='Period', marker='o')

plt.title('Comparison of Movie Performance: Last Decade vs Post-Pandemic', fontsize=16)

plt.xlabel('Performance Category', fontsize=14)
plt.ylabel('Percentage (%)', fontsize=14)

plt.xticks(rotation=45)
plt.grid(True)

plt.legend(title='Period', loc='upper right')

plt.tight_layout()
plt.show()
```

Note: Visual Analysis in next page

Analysis 1: Significant increase in flop movies after 2019



In the graph, a significant increase in the percentage of flop movies is evident, rising from approximately 30% in the last decade to 56% post-pandemic. This marked rise clearly indicates a decline in the overall performance of Bollywood movies in recent years. Additionally, there is a noticeable decrease in the percentage of Superhit and Blockbuster films, further supporting the notion that the industry's ability to produce high-performing movies has diminished.

Based on this analysis, it can be inferred that Bollywood is indeed struggling in the post-pandemic era. However, to justify this we need to analyse a bit more.

Analysis 2: Decrease in Average Box Office post 2019

```
filtered_main = main_df[(main_df['Year'] > 2019) & (main_df['Year'] < 2025)]

average_budget= filtered_main['Inflated_Budget'].mean().round(1)
average_box_office= filtered_main['Inflated_Box_Office'].mean().round(1)

print(average_budget)
print(average_box_office)
```

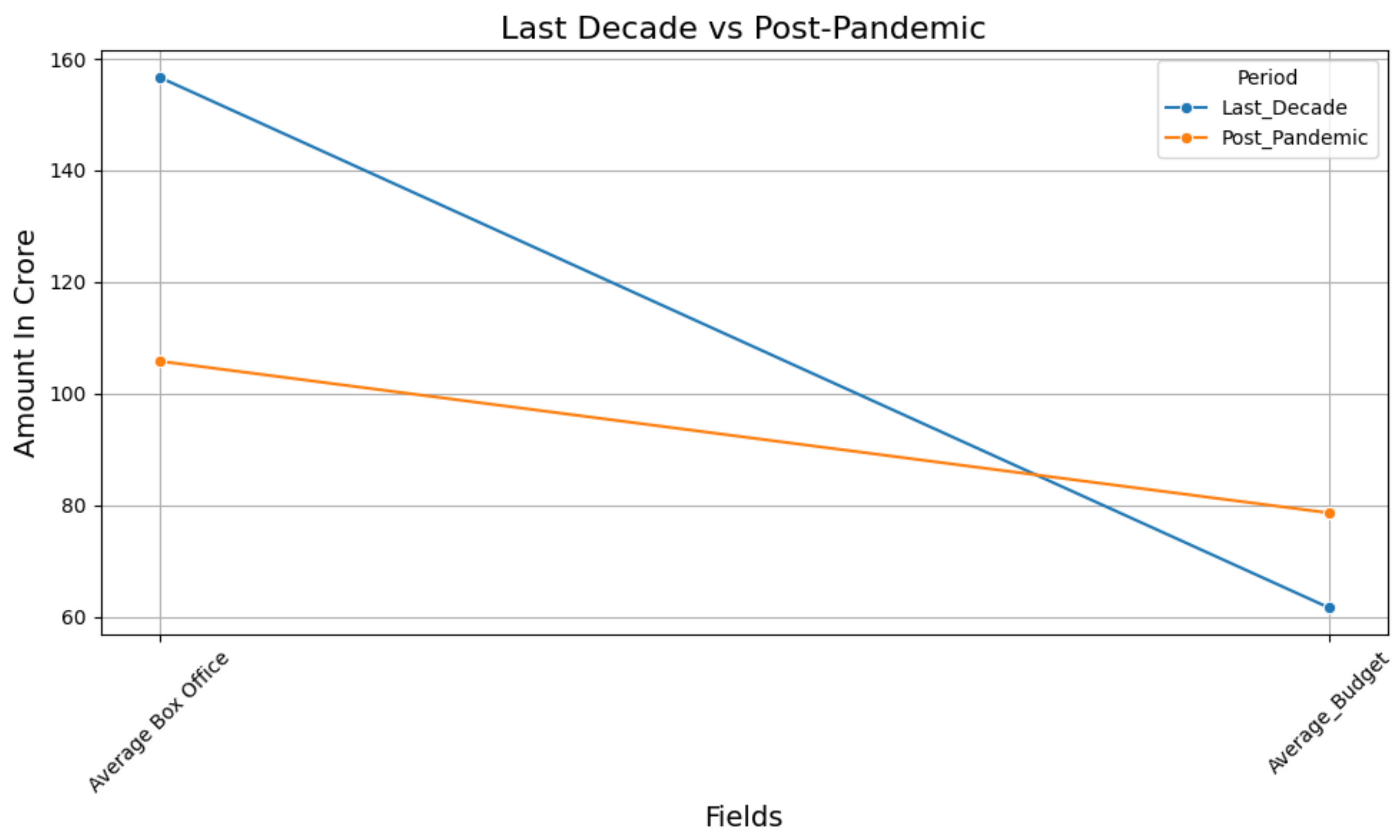
```
filtered_main = main_df[(main_df['Year'] > 2009) & (main_df['Year'] < 2020)]

average_budget= filtered_main['Inflated_Budget'].mean().round(1)
average_box_office= filtered_main['Inflated_Box_Office'].mean().round(1)

print(average_budget)
print(average_box_office)
```

From these two codes and it's output, created a table which compares the Average Box Office of last decade with Average Box Office since 2020. We have also compared Average Rating

Fields	Last_Decade	Post_Pandemic
Average Box Office	156.7	105.8
Average Budget	61.6	78.6

Analysis 2: Decrease in Average Box Office post 2019

```
df2_melted = df2.melt(id_vars='Fields', value_vars=['Last_Decade', 'Post_Pandemic'],
                    var_name='Period', value_name='Amount')

plt.figure(figsize=(10, 6))
sns.lineplot(data=df2_melted, x='Fields', y='Amount', hue='Period', marker='o')

plt.title('Last Decade vs Post-Pandemic', fontsize=16)

plt.xlabel('Fields', fontsize=14)
plt.ylabel('Amount In Crore', fontsize=14)

plt.xticks(rotation=45)
plt.grid(True)

plt.legend(title='Period', loc='upper right')

plt.tight_layout()
plt.show()
```

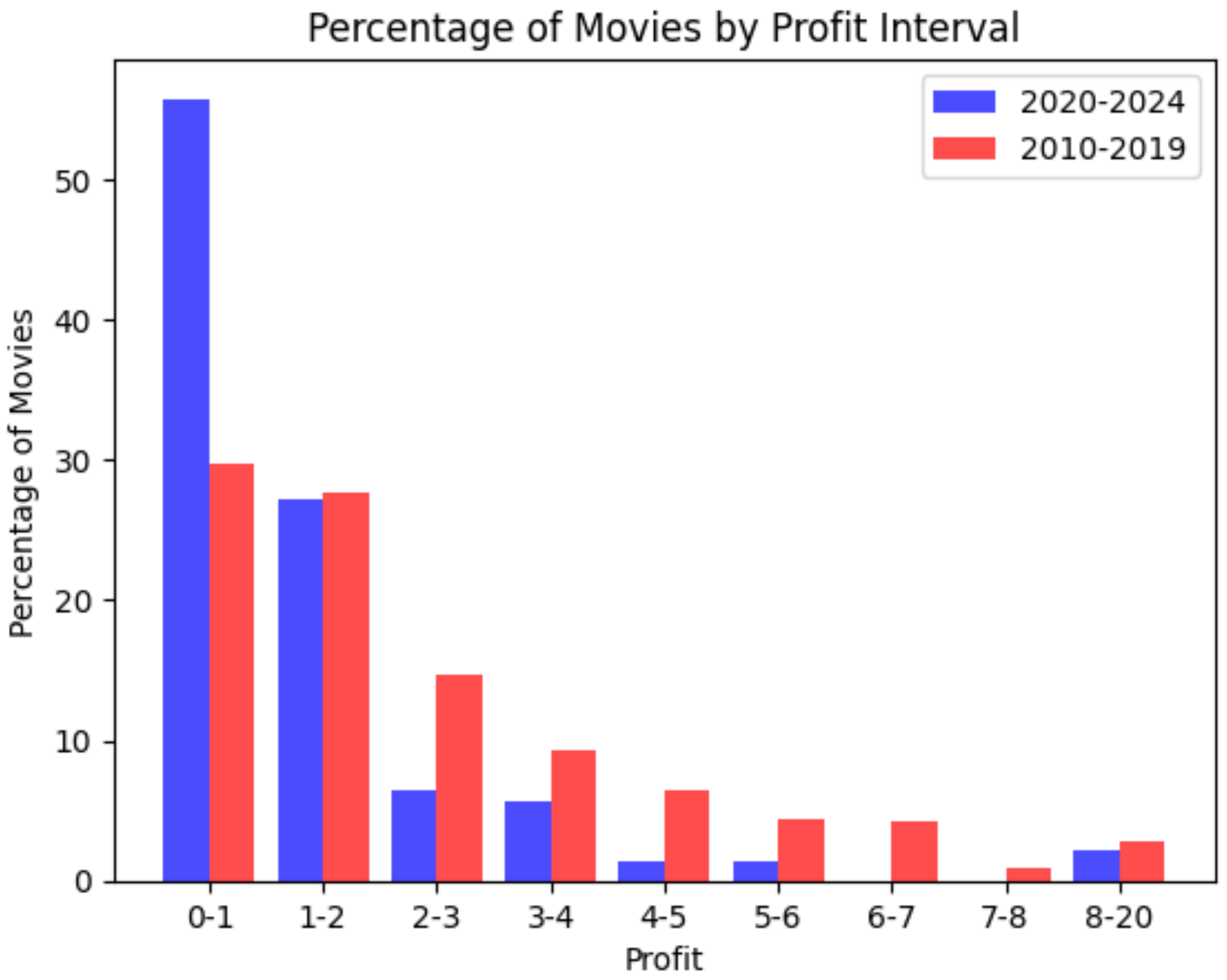


We can observe a significant drop in the average box office collections despite an increase in average film budgets.
This trend clearly indicates that there are serious underlying issues that need to be addressed. Moving forward, we will explore the reasons behind these concerning numbers

Analysis 3: Few Successful movies post pandemic

```python
import matplotlib.pyplot as plt

main_df['Profit'] = main_df['Box_Office'] / main_df[‘Budget']

bins = [0, 1, 2, 3, 4, 5, 6, 7, 8, 20]

counts_current, _ = np.histogram(current_decade['Profit'], bins=bins)
proportions_current = counts_current / counts_current.sum() * 100

counts_previous, _ = np.histogram(previous_decade['Profit'], bins=bins)
proportions_previous = counts_previous / counts_previous.sum() * 100


width = 0.4
x = np.arange(len(bins) - 1)

plt.bar(x - width/2, proportions_current, width=width, alpha=0.7, color='blue', label='2020-2024')
plt.bar(x + width/2, proportions_previous, width=width, alpha=0.7, color='red', label='2010-2019')

plt.title('Percentage of Movies by Profit Interval')
plt.xlabel('Profit')
plt.ylabel('Proportion of Movies')

plt.xticks(x, [f"{bins[i]}-{bins[i+1]}" for i in range(len(bins)-1)])

plt.legend()

plt.show()
```

Our analysis reveals a notable increase in the percentage of successful films during the decade from 2010 to 2019. However, following the pandemic, the percentage of movies that failed to generate profit has risen significantly to 60%. In contrast, this figure was approximately 30% in the previous decade. This trend indicates a marked decline in the rate of successful movies in the post-pandemic landscape.


Percentage of Movies by Profit Interval
2020-2024
2010-2019
Percentage of Movies
0
10
20
30
40
50
0-1
1-2
2-3
3-4
4-5
5-6
6-7
7-8
8-20
Profit

Bollywood's Post-Pandemic Underperformance: An In-Depth Analysis

Shift in genre preferences : Box Office Analysis

```python
filtered_main = main_df[(main_df['Year'] > 2009) & (main_df['Year'] < 2020)]

genre_list = filtered_main.groupby('Genre').agg(
    Avg_Box_Office= ('Inflated_Box_Office', ‘mean’).round(1),
    movie_count=('Inflated_Box_Office', 'count')
)
genre_list= genre_list[genre_list['movie_count']>20]

genre_list=genre_list.sort_values(by='Avg_Box_Office', ascending=False)
print(genre_list)
```

```python
filtered_main = main_df[(main_df['Year'] > 2009) & (main_df['Year'] < 2020)]

genre_list = filtered_main.groupby('Genre').agg(
    Avg_Box_Office= ('Inflated_Box_Office', ‘mean’).round(1),
    movie_count=('Inflated_Box_Office', 'count')
)
genre_list= genre_list[genre_list['movie_count']>20]

genre_list=genre_list.sort_values(by='Avg_Box_Office', ascending=False)
print(genre_list)
```

Genre	Avg_Box_Office (<2020)	Avg_Box_Office (>2019)
Drama	150.0	38.8
Thriller	83.4	61.7
Romance	114.4	103.6
Comedy	148.3	58.4
Action	250.7	211.9

Note: This table is generated my merging tables formed from above two codes.

Shift in genre preferences : IMDb Rating Analysis

```
filtered_main = main_df[(main_df['Year'] > 2009) & (main_df['Year'] < 2020)]

genre_list = filtered_main.groupby('Genre').agg(
    avg_rating= ('Rating', ‘mean’).round(1),
    movie_count=('Inflated_Box_Office', 'count')
)
genre_list= genre_list[genre_list[‘movie_count']>=20]

genre_list=genre_list.sort_values(by='avg_rating', ascending=False)
print(genre_list)
```

```
filtered_main = main_df[(main_df['Year'] > 2019) & (main_df['Year'] < 2025)]

genre_list = filtered_main.groupby('Genre').agg(
    avg_rating= ('Rating', ‘mean’)round(1),
    movie_count=('Inflated_Box_Office', 'count')
)
genre_list= genre_list[genre_list['movie_count']>=10]

genre_list=genre_list.sort_values(by='avg_rating', ascending=False)
print(genre_list)
```

Genre	Avg_Rating(<2020)	Avg_Rating(>2019)
Drama	6.6	6.4
Thriller	6.2	6.4
Romance	5.7	5.8
Comedy	5.6	5.8
Action	5.3	4.9

Note: This table is generated by merging tables formed from above two codes.

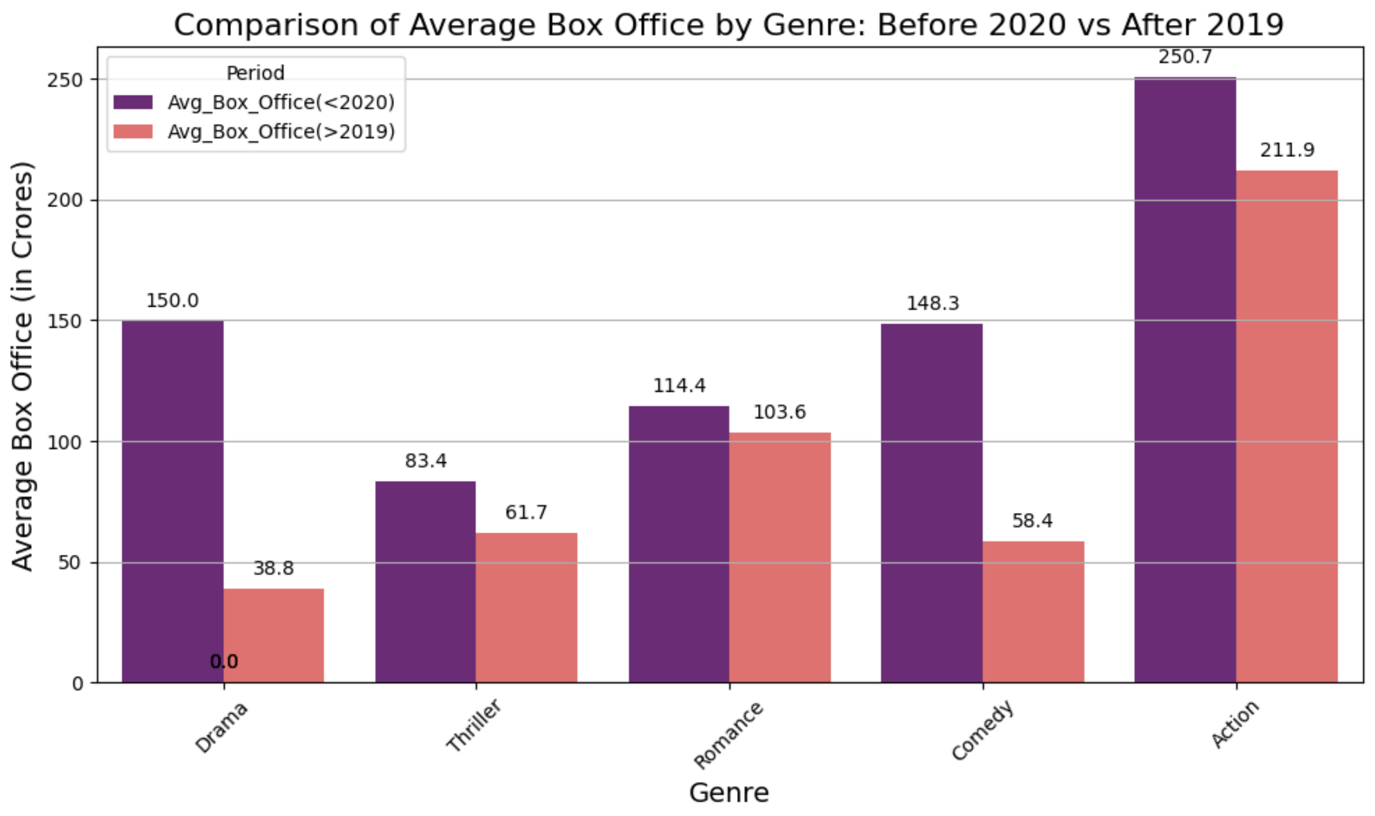
Visualisation

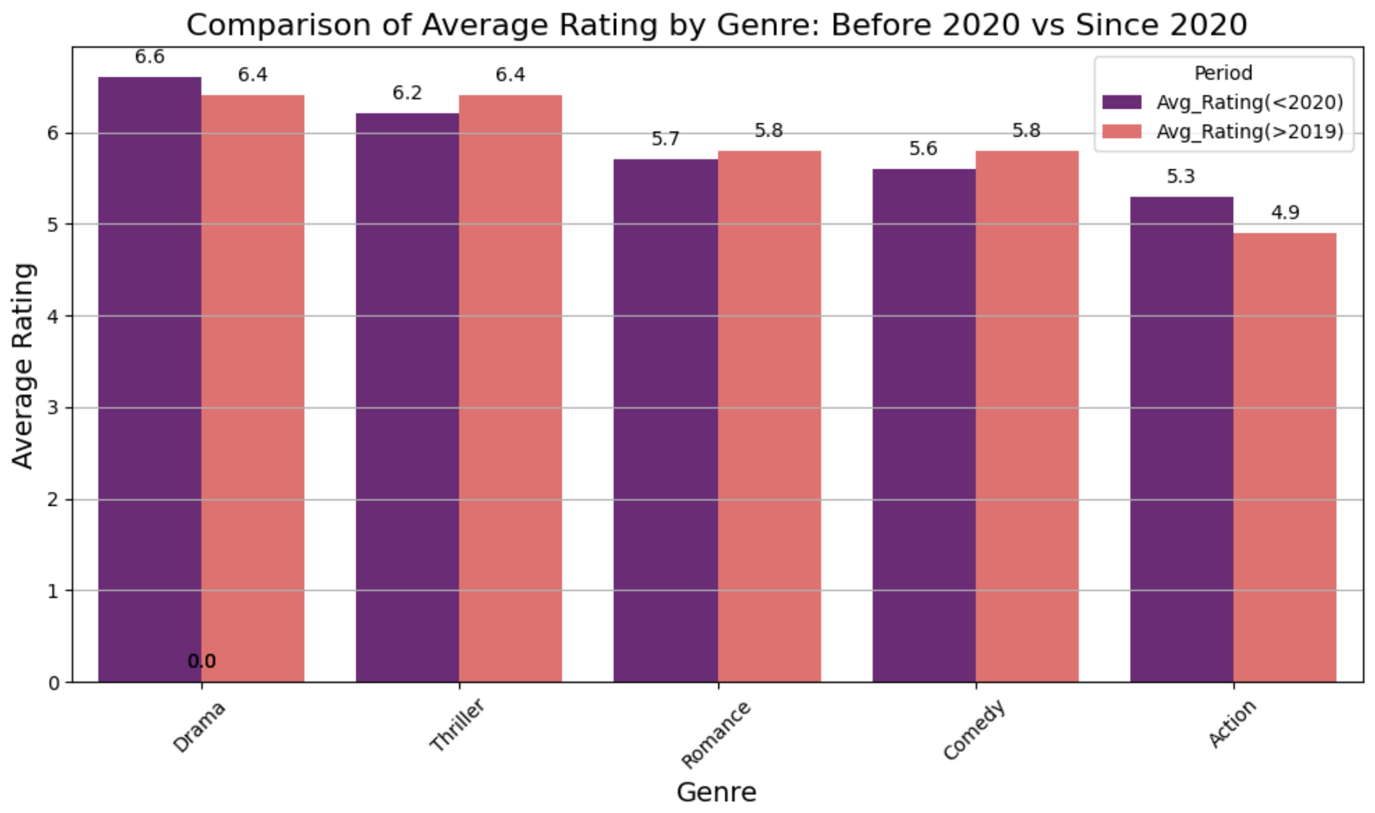
Shift in genre preferences : Box Office Analysis

```
df3=pd.read_csv('Performance_Camparison - Analysis3.csv')

df3_melted = df3.melt(id_vars='Genre',
                    value_vars=['Avg_Box_Office(<2020)', 'Avg_Box_Office(>2019)'],
                    var_name='Period', value_name='Avg_Box_Office')

plt.figure(figsize=(10, 6))
sns.barplot(data=df3_melted, x='Genre', y='Avg_Box_Office', hue='Period', palette='magma')

plt.title('Comparison of Average Box Office by Genre: Before 2020 vs After 2019', fontsize=16)
plt.xlabel('Genre', fontsize=14)
plt.ylabel('Average Box Office (in Crores)', fontsize=14)
plt.xticks(rotation=45)
plt.grid(True, axis='y')

for p in plt.gca().patches:
    plt.gca().annotate(f'{p.get_height():.1f}',
                       (p.get_x() + p.get_width() / 2., p.get_height()),
                       ha='center', va='center',
                       xytext=(0, 10),
                       textcoords='offset points')

plt.tight_layout()
plt.show()
```

Note: Using Bar Plot we are going to analyse. Similarly we have code for Average Rating Analysis.

Visualisation





Analysis: In the first bar plot, a significant drop in the average box office collections for Drama and Comedy genres is evident.
The second bar plot shows that the average ratings for these genres have remained relatively consistent. This suggests that while the quality of movies hasn't declined, audiences are not as enthusiastic about Drama and Comedy as they once were.

Bollywood should focus on enhancing the quality of Action and mass-appeal movies, as these genres are currently resonating more with audiences.

However, much improvement is still needed in the Action genre.

It's clear that Bollywood is in a transition phase, and it may take the rest of this decade for the industry to establish itself more strongly in the Action genre as well.

Established actors are not performing up to their expected caliber

```
filtered_main_2010s = main_df[(main_df['Year'] > 2010) & (main_df['Year'] < 2020)]
filtered_main_exploded_2010s
filtered_main_2010s.assign(Casting=filtered_main_2010s['Casting'].str.split(', ')).explode('Casting')

actor_list_2010s = filtered_main_exploded_2010s.groupby('Casting').agg(
    movie_count_2010s=('Inflated_Box_Office', 'count')
).reset_index()

actor_list_2010s_filtered = actor_list_2010s[actor_list_2010s['movie_count_2010s'] >= 5]
artist_stats_2010s = filtered_main_exploded_2010s.groupby('Casting').agg(
    average_box_office_2010s=('Inflated_Box_Office', 'mean'),
    movie_count_2010s=('Inflated_Box_Office', 'count')
).reset_index()

artist_stats_filtered_2010s =
artist_stats_2010s[artist_stats_2010s['Casting'].isin(actor_list_2010s_filtered['Casting'])]
top_artists_2010s = artist_stats_filtered_2010s.sort_values(by='average_box_office_2010s',
ascending=False).round(1).head(10)

filtered_main_2020s = main_df[(main_df['Year'] > 2019) & (main_df['Year'] < 2025)]
filtered_main_exploded_2020s =
filtered_main_2020s.assign(Casting=filtered_main_2020s['Casting'].str.split(', ')).explode('Casting')

actor_list_2020s = filtered_main_exploded_2020s.groupby('Casting').agg(
    movie_count_2020s=('Inflated_Box_Office', 'count')
).reset_index()

actor_list_2020s_filtered =
actor_list_2020s[actor_list_2020s['Casting'].isin(top_artists_2010s['Casting'])]

artist_stats_2020s = filtered_main_exploded_2020s.groupby('Casting').agg(
    average_box_office_2020s=('Inflated_Box_Office', 'mean'),
    movie_count_2020s=('Inflated_Box_Office', 'count')
).reset_index()

artist_stats_filtered_2020s =
artist_stats_2020s[artist_stats_2020s['Casting'].isin(actor_list_2020s_filtered['Casting'])]

top_artists_2020s = artist_stats_filtered_2020s.sort_values(by='average_box_office_2020s',
ascending=False).round(1).head(10)
print("Top 10 Artists (2010-2019):\n", top_artists_2010s)
print("\nTop 10 Artists (2020-2024):\n", top_artists_2020s)
```

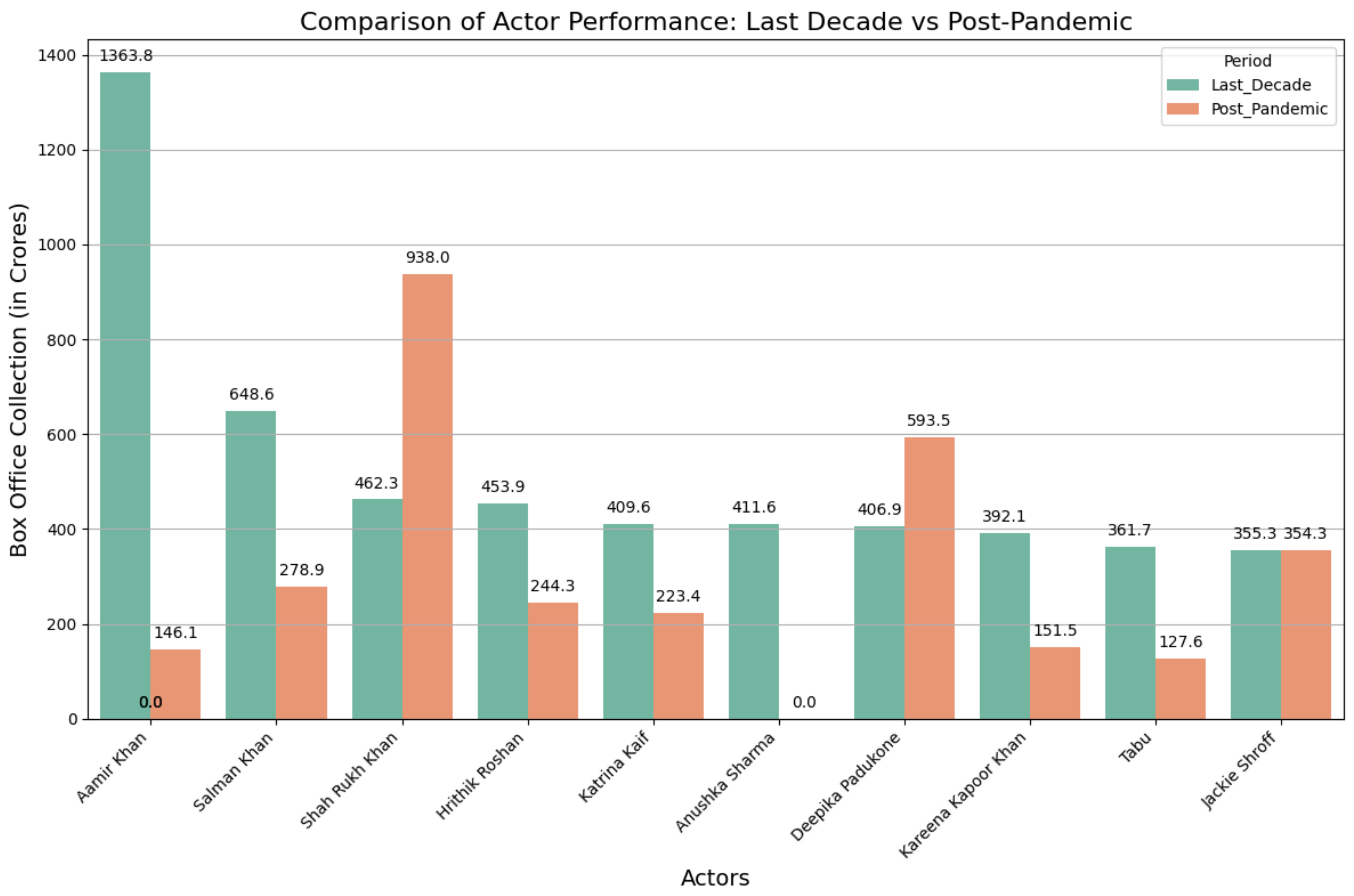
Established actors are not performing up to their expected caliber

Actors	Last_Decade	Post_Pandemic
Aamir Khan	1363.8	146.1
Salman Khan	648.6	278.9
Shah Rukh Khan	462.3	938
Hrithik Roshan	453.9	244.3
Katrina Kaif	409.6	223.4
Anushka Sharma	411.6	0
Deepika Padukone	406.9	593.5
Kareena Kapoor Khan	392.1	151.5
Tabu	361.7	127.6
Jackie Shroff	355.3	354.3

Note: This table is generated by merging tables formed from previous code.
Below code is for visualisation through Bar Plot.

```
df5=pd.read_csv('Performance_Camparison - Analysis5.csv')

df5_melted = df5.melt(id_vars='Actors', value_vars=['Last_Decade', 'Post_Pandemic'],
                    var_name='Period', value_name='Value')

plt.figure(figsize=(12, 8))
sns.barplot(data=df5_melted, x='Actors', y='Value', hue='Period', palette='Set2')

plt.title('Comparison of Actor Performance: Last Decade vs Post-Pandemic', fontsize=16)
plt.xlabel('Actors', fontsize=14)
plt.ylabel('Box Office Collection (in Crores)', fontsize=14)
plt.xticks(rotation=45, ha='right')
plt.grid(True, axis=‘y')

for p in plt.gca().patches:
    plt.gca().annotate(f'{p.get_height():.1f}',
                       (p.get_x() + p.get_width() / 2., p.get_height()),
                       ha='center', va='center',
                       xytext=(0, 10),
                       textcoords='offset points')

plt.tight_layout()
plt.show()
```

Visualisation



Analysis: In the plot above, we can see how established actors have been struggling since 2020. Except for Shah Rukh Khan and Deepika Padukone, most actors who performed well in the last decade are now facing challenges. This is a significant factor contributing to Bollywood's current struggles.

Actors like Salman Khan, Aamir Khan, and Hrithik Roshan need to be more selective with their scripts, as they still have the ability to draw audiences to theatres. Similarly, female actors like Katrina Kaif and Kareena Kapoor are also finding it difficult to attract audiences, possibly due to age-related factors. Anushka Sharma, on the other hand, has been largely absent from the scene this decade.

Bollywood can potentially revive itself if these established actors start choosing scripts more thoughtfully, as Indian audiences still often go to theatres primarily to see their favourite stars.

Things are likely to change sooner rather than later.

Established directors are not performing up to their expected

```python
filtered_main_2010s = main_df[(main_df['Year'] > 2010) & (main_df['Year'] < 2020)]
filtered_main_exploded_2010s = 
filtered_main_2010s.assign(Director=filtered_main_2010s['Director'].str.split(', ')).explode('Director')

director_stats_2010s = filtered_main_exploded_2010s.groupby('Director').agg(
    movie_count_2010s=('Inflated_Box_Office', 'count'),
    average_box_office_2010s=('Inflated_Box_Office', 'mean')
).reset_index()

director_stats_2010s_filtered = director_stats_2010s[director_stats_2010s['movie_count_2010s'] >= 2]

top_directors_2010s = director_stats_2010s_filtered.sort_values(by='average_box_office_2010s',
ascending=False).head(10)

filtered_main_2020s = main_df[(main_df['Year'] > 2019) & (main_df['Year'] < 2025)]
filtered_main_exploded_2020s = 
filtered_main_2020s.assign(Director=filtered_main_2020s['Director'].str.split(', ')).explode('Director')

director_stats_2020s = filtered_main_exploded_2020s.groupby('Director').agg(
    movie_count_2020s=('Inflated_Box_Office', 'count'),
    average_box_office_2020s=('Inflated_Box_Office', 'mean')
).reset_index()

director_stats_2020s_filtered = 
director_stats_2020s[director_stats_2020s['Director'].isin(top_directors_2010s['Director'])]

top_directors_2020s = director_stats_2020s_filtered.sort_values(by='average_box_office_2020s',
ascending=False).round(1).head(10)

print("Top 10 Directors (2010-2019):\n", top_directors_2010s)
print("\nTop 10 Directors (2020-2024):\n", top_directors_2020s)
```

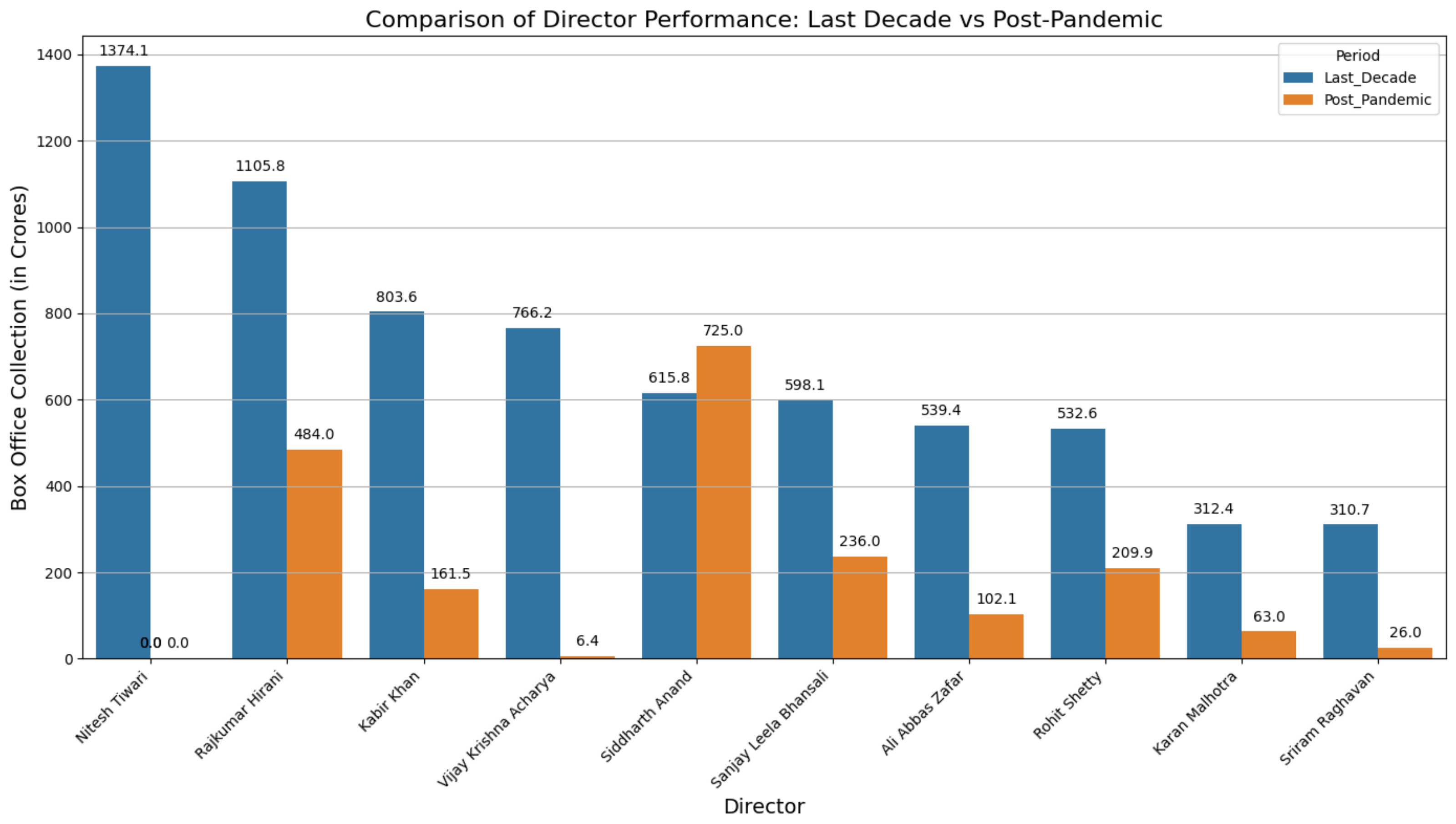
Established director are not performing up to their expected

Director	Last_Decade	Post_Pandemic
Nitesh Tiwari	1374.1	0
Rajkumar Hirani	1105.8	484
Kabir Khan	803.6	161.5
Vijay Krishna Acharya	766.2	6.4
Siddharth Anand	615.8	725
Sanjay Leela Bhansali	598.1	236
Ali Abbas Zafar	539.4	102.1
Rohit Shetty	532.6	209.9
Karan Malhotra	312.4	63
Sriram Raghavan	310.7	26

Note: This table is generated by merging tables formed from previous code.
Below code is for visualisation through Bar Plot.

```
df6=pd.read_csv('Performance_Camparison - Analysis6.csv’)

df6_melted = df6.melt(id_vars='Director', value_vars=['Last_Decade', 'Post_Pandemic'],
                    var_name='Period', value_name='Value')

plt.figure(figsize=(14, 8))
sns.barplot(data=df6_melted, x='Director', y='Value', hue='Period', palette='tab10')

plt.title('Comparison of Director Performance: Last Decade vs Post-Pandemic', fontsize=16)
plt.xlabel('Director', fontsize=14)
plt.ylabel('Box Office Collection (in Crores)', fontsize=14)
plt.xticks(rotation=45, ha='right')
plt.grid(True, axis='y')

for p in plt.gca().patches:
    plt.gca().annotate(f'{p.get_height():.1f}',
                       (p.get_x() + p.get_width() / 2., p.get_height()),
                       ha='center', va='center',
                       xytext=(0, 10),
                       textcoords='offset points')

plt.tight_layout()
plt.show()
```

Visualisation



Analysis: In the plot above, we observe that established directors have been struggling since 2020. Except for Siddharth, most directors who were successful in the last decade are now facing challenges. This situation significantly contributes to Bollywood's current struggles.
Directors like Kabir Khan, Vijay Krishna Acharya, and Ali Abbas Zafar need to be more selective with their scripts. Nitesh Tiwari has been largely absent this decade but is currently working on the highly anticipated project, 'Ramayana'. Meanwhile, Rajkumar Hirani and Sanjay Leela Bhansali have delivered hits with 'Dunki' and 'Gangubai Kathiawadi', but these projects are not as high-profile as their previous work.

Conclusion

Bollywood is Struggling and here is what comes through my analysis.

Transition Phase: The industry is undergoing a transition, with viewers' enthusiasm for Drama and Comedy genres waning unless the scripts are exceptionally well-crafted.

Performance of Established Actors: Established actors have not consistently delivered performances that meet their past standards. It is crucial for them to select scripts more thoughtfully to resonate with contemporary audiences.

Struggles of Established Female Actors: Except for Deepika Padukone, many prominent female actors from the last decade have struggled to make a significant impact. This suggests a need for more compelling roles and narratives for female leads.

Directors and Writers Collaboration: Directors must place a stronger emphasis on script quality and foster closer collaboration with writers. Writers should be given greater importance in the creative process to enhance storytelling.

Impact of OTT Platforms: The rise of OTT platforms has profoundly affected traditional theater cinema, altering viewing habits and expectations.

Shift to New Talent: There is a gradual shift from established actors to emerging talent. It will take time for new actors to build their value and establish a strong presence in the industry.

Adoption of New Technologies: Bollywood should embrace advanced technologies like VFX to meet audiences' growing expectations for high-quality visuals in theatres.

Focus on Quality over Quantity: The industry should prioritise producing fewer, high-quality films rather than a high volume of mediocre ones. This approach can help rebuild trust with audiences and create a stronger impact at the box office.

References

IMDb Website: www.imdb.com/search/title/?title_type=feature&countries=IN&languages=hi&sort=num_votes,asc

Wikipedia Website: www.wikipedia.org

Chatgpt: openai.com/index/chatgpt