

# HOMEWORK 3: DATA ANALYSIS

Presented by DAT204M Group 2



# INTRO AND ARCHITECTURE

# SOURCE AND DATASET

IMDb

Since the goal of the project is to create a dashboard about movie stats, we sourced our data from IMDb, the world's most popular and authoritative source for movie, TV and celebrity content.

The screenshot shows the IMDb Charts page for the "Most Popular Movies" chart. The page has a dark theme with a yellow header bar. At the top, there is a navigation bar with the IMDb logo, a "Menu" icon, a dropdown for "All", a search bar, a magnifying glass icon, "IMDbPro", a "Watchlist" button, a "Sign In" button, and a language switch for "EN".

The main content area is titled "IMDb Charts" and "Most Popular Movies", with a subtitle "As determined by IMDb users". It displays a list of 100 titles, with "The Shawshank Redemption" at the top and "The Godfather" as the second item shown.

On the right side, there is a sidebar titled "You have rated" which shows "0/100 (0%)" and a checkbox for "Hide titles you've rated". Below this, there is a section titled "More to explore" with links to "Charts", "Top Box Office (US) >", "From the past weekend", and "IMDb Top 250 Movies >".

The main list of movies includes:

- The Shawshank Redemption**: 67 (+ 3) - 1994, 2h 22m, R, 9.3 (2.9M), Rate
- The Godfather**: 73 (+ 11) - 1972, 2h 55m, R, 9.2 (2M), Rate

# SOURCE AND DATASET

Since the goal of the project is to create a dashboard about movie stats, we sourced our data from IMDb, the world's most popular and authoritative source for movie, TV and celebrity content.

```
movies_df = pd.DataFrame(load_progress()[0])
reviews_df = pd.DataFrame(load_progress()[1])

print("Movies DataFrame:")
movies_df.head(3)

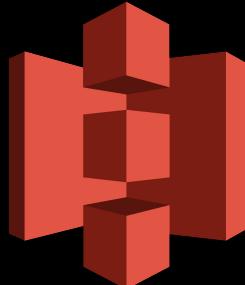
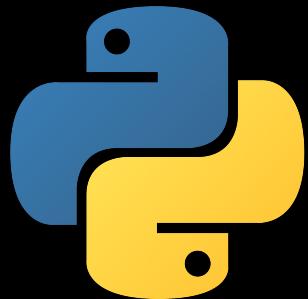
Movies DataFrame:
   movie_code  Title  Year  Revenue  Budget  Runtime  Opening_weekend_US  Rating  Production_company  Genre  IMDb_code  release_date
0         278  The Shawshank  1994    28341469  25000000       142          28341469     R  Castle Rock Entertainment  Drama,  tt0111161  1994-09-23
1         238  The Godfather  1972   245066411   6000000       175          245066411     R  Paramount Pictures, Alfran  Drama,  tt0068646  1972-03-14
2         240  The Godfather  1974   102600000  13000000       202          102600000     R  Paramount Pictures, The  Drama,  tt0071562  1974-12-20

print("Reviews DataFrame:")
reviews_df.head(10)

Reviews DataFrame:
   movie_code  IMDb_code  review_date  rating_of_movie  actual_review
0         278  tt0111161  2016-04-29T18:08:41.892Z        9.0  very good movie 9.5/10
1         278  tt0111161  2016-07-10T00:16:50.561Z       10.0  Some birds aren't meant to be caged.\n\nTh...
2         278  tt0111161  2017-11-11T15:09:34.114Z       6.0  Make way for the best film ever made people. ...
3         278  tt0111161  2018-05-01T05:51:13.756Z       10.0  There is a reason why this movie is at the top...
4         278  tt0111161  2018-10-18T15:08:48.777Z       NaN  It's still puzzling to me why this movie exact...
5         278  tt0111161  2019-07-30T08:25:48.402Z       NaN  I will not say that the film is predictable, b...
6         278  tt0111161  2021-09-18T19:56:48.348Z       10.0  First time seeing this in probably close to 20...
7         278  tt0111161  2023-01-14T18:44:02.525Z       NaN  No 1 movie for all the time
8         278  tt0111161  2023-04-03T15:38:39.540Z       3.0  This is much more predictable and Hollywood th...
9         278  tt0111161  2023-06-29T17:48:41.064Z       NaN  This movie is great
```

# TECHNOLOGIES USED

The following platforms were used for data collection, processing, loading, and visualization:



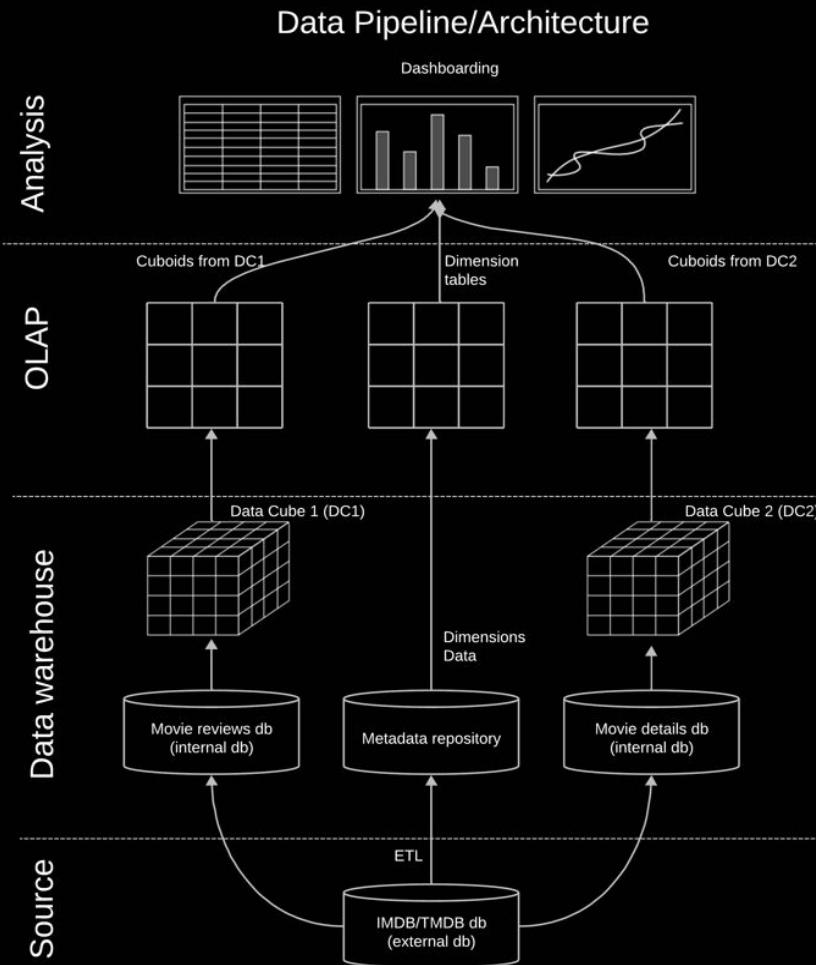
# DATA PIPELINE

IMDb

We followed a 3-tier approach of data warehousing. From source, we have:

1. Data warehouse
2. OLAP
3. Analysis

(to be continued in OLAP section)



# DATA PIPELINE

```
def get_top_movies(api_key, page=1):
    url = f"{BASE_TMDB_URL}/movie/top_rated?api_key={api_key}&language=en-US&page={page}"
    response = requests.get(url)
    movies = response.json().get('results', [])
    top_movies = [(movie['title'], movie['id'], movie['release_date']) for movie in movies]
    return top_movies
```

```
def fetch_movie_details(tmdb_id, api_key):
    url = f"{BASE_TMDB_URL}/movie/{tmdb_id}?api_key={api_key}&language=en-US"
    response = requests.get(url)
    movie = response.json()

    imdb_id = movie.get('imdb_id')
    rating = movie.get('release_dates', {}).get('results', [{}])[0].get('certification', 'Not Rated')

    simplified_rating = 'Not Rated'
    if 'R' in rating:
        simplified_rating = 'R'
    elif 'PG' in rating:
        simplified_rating = 'PG'
    elif 'G' in rating:
        simplified_rating = 'G'

    return {
        'movie_code': movie.get('id'),
        'Title': movie.get('title'),
        'Year': movie.get('release_date', '').split('-')[0],
        'Revenue': movie.get('revenue'),
        'Budget': movie.get('budget'),
        'Runtime': movie.get('runtime'),
        'Opening_weekend_US': movie.get('revenue'),
        'Rating': simplified_rating,
        'Production_company': ', '.join([comp['name'] for comp in movie.get('production_companies', [])]),
        'Genre': ', '.join([genre['name'] for genre in movie.get('genres', [])]),
        'IMDb_code': imdb_id
    }
```

Get API response and movie attributes

Extract movie attributes

# DATA PIPELINE

```
def fetch_tmdb_reviews(tmdb_id, api_key, imdb_code):
    url = f'{BASE_TMDB_URL}/movie/{tmdb_id}/reviews?api_key={api_key}&language=en-US'
    response = requests.get(url)
    reviews = response.json().get('results', [])
    review_data = []
    for review in reviews:
        review_data.append({
            'movie_code': tmdb_id,
            'IMDb_code': imdb_code,
            'review_date': review.get('created_at'),
            'rating_of_movie': review.get('author_details', {}).get('rating'),
            'actual_review': review.get('content')
        })
    return review_data
```

Get movie reviews based from `imdb_code`, and `tmdb_id`

Utilized pickle module, to save and load progress

```
def save_progress(movie_data, reviews_data, filename='movies_data.pkl', reviews_filename='reviews_data.pkl'):
    try:
        with open(filename, 'wb') as f:
            pickle.dump(movie_data, f)
        print(f"Saved movie data to {filename}")
        with open(reviews_filename, 'wb') as f:
            pickle.dump(reviews_data, f)
        print(f"Saved reviews data to {reviews_filename}")
    except (pickle.PickleError, IOError) as e:
        print(f"Error saving data: {e}")

def load_progress(filename='movies_data.pkl', reviews_filename='reviews_data.pkl'):
    movie_data = []
    reviews_data = []

    try:
        if os.path.exists(filename):
            with open(filename, 'rb') as f:
                movie_data = pickle.load(f)
    except Exception as e:
        print(f"Error loading {filename}: {e}")

    try:
        if os.path.exists(reviews_filename):
            with open(reviews_filename, 'rb') as f:
                reviews_data = pickle.load(f)
    except Exception as e:
        print(f"Error loading {reviews_filename}: {e}")

    return movie_data, reviews_data
```

# DATA PIPELINE

```

def get_movies_and_reviews(movie_filename='movies_data.pkl', reviews_filename='reviews_data.pkl'):
    movie_data, reviews_data = load_progress(movie_filename, reviews_filename)
    approximate_entry_size_kb = 5
    data_size_limit_kb = 1 * 1024 * 1024 # 1GB in KB
    accumulated_size_kb = sum([len(str(movie)) / 1024 for movie in movie_data])
    page = (len(movie_data) // 20) + 1

    while accumulated_size_kb < data_size_limit_kb:
        top_movies = get_top_movies(TMDB_API_KEY, page)
        if not top_movies:
            break

        for title, tmdb_id, release_date in top_movies:
            if any(movie['movie_code'] == tmdb_id for movie in movie_data):
                continue

            movie_info = fetch_movie_details(tmdb_id, TMDB_API_KEY)
            imdb_code = movie_info.get('IMDb_code')
            reviews = fetch_tmdb_reviews(tmdb_id, TMDB_API_KEY, imdb_code)

            if reviews:
                movie_info['release_date'] = release_date
                movie_data.append(movie_info)
                reviews_data.extend(reviews)
                accumulated_size_kb += approximate_entry_size_kb
                print(f"Fetched data for: {title}")
                save_progress(movie_data, reviews_data)
                time.sleep(1)

            if accumulated_size_kb >= data_size_limit_kb:
                print(f'Reached data size limit of approximately {data_size_limit_kb / (1024 * 1024)} GB')
                break
        page += 1

    save_progress(movie_data, reviews_data)

    movie_df = pd.DataFrame(movie_data)
    reviews_df = pd.DataFrame(reviews_data)

    movie_df.to_csv(movie_filename, index=False)
    reviews_df.to_csv(reviews_filename, index=False)

    print(f"Data saved to {movie_filename} and {reviews_filename}")

    return movie_df, reviews_df

```

## Main function:

1. Check if a pickle file is available
2. Set data size limit to 1gb
3. Extract attributes
4. Save progress
5. Convert to pandas to csv

# SAMPLE RESULTS



```
movie_data, reviews_data = get_movies_and_reviews()  
save_progress(movie_data, reviews_data)
```

```
Fetched data for: The Shawshank Redemption  
Saved movie data to movies_data.pkl  
Saved reviews data to reviews_data.pkl
```

```
Fetched data for: The Godfather  
Saved movie data to movies_data.pkl  
Saved reviews data to reviews_data.pkl  
Fetched data for: The Godfather Part II  
Saved movie data to movies_data.pkl  
Saved reviews data to reviews_data.pkl  
Fetched data for: Schindler's List  
Saved movie data to movies_data.pkl  
Saved reviews data to reviews_data.pkl
```



# PICKLE CONTINUATION



```
movie_data, reviews_data = get_movies_and_reviews()  
save_progress(movie_data, reviews_data)
```

Fetched data for: 12 Angry Men

Saved movie data to movies\_data.pkl

Saved reviews data to reviews\_data.pkl

Fetched data for: Spirited Away

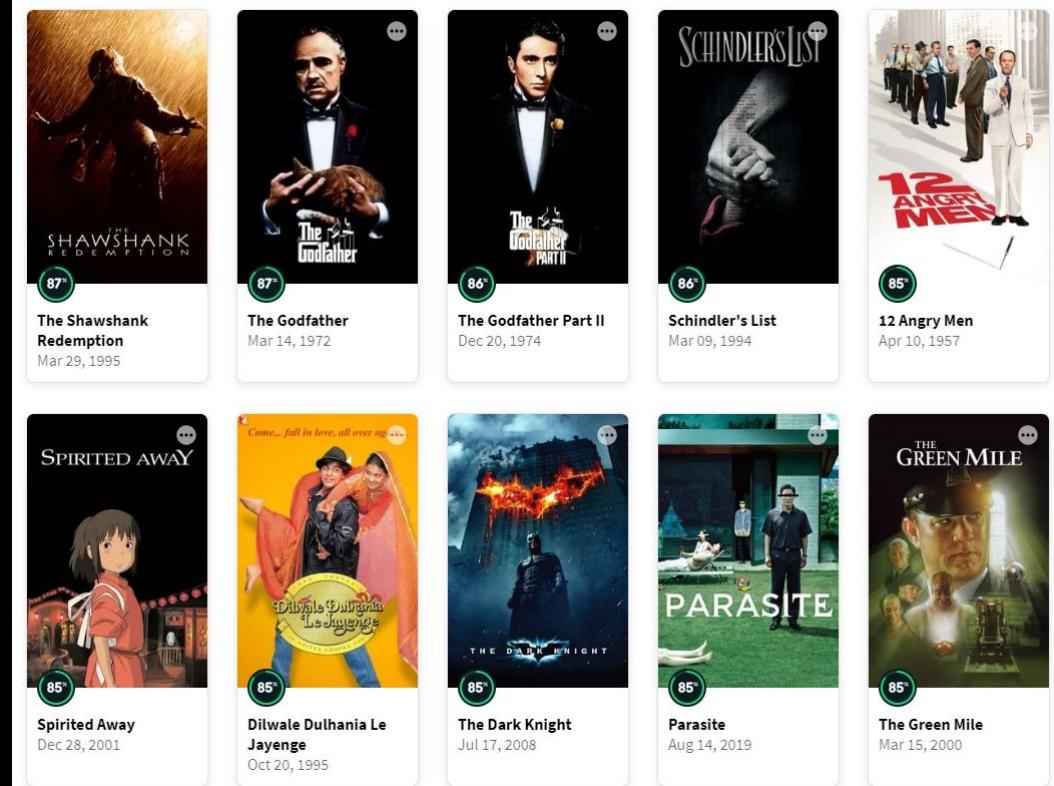
Saved movie data to movies\_data.pkl

Saved reviews data to reviews\_data.pkl

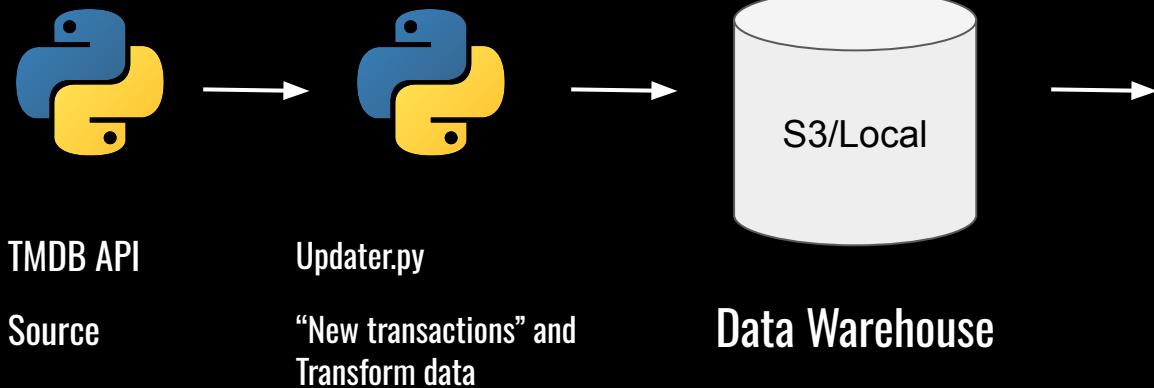
Fetched data for: Dilwale Dulhania Le Jayenge

Saved movie data to movies\_data.pkl

Saved reviews data to reviews\_data.pkl



# ONLINE TRANSACTION PROCESSING (OLTP)



# ONLINE TRANSACTION PROCESSING (OLTP)

```
def update_reviews(movies_df, reviews_df):
    updated_reviews = []
    for index, row in movies_df.iterrows():
        tmdb_id = row['movie_code']
        imdb_id = row['IMDb_code']
        new_reviews = fetch_tmdb_reviews(tmdb_id, TMDB_API_KEY)

        for new_review in new_reviews:
            review_date = new_review['review_date']
            rating_of_movie = new_review['rating_of_movie']
            actual_review = new_review['actual_review']

            existing_reviews = reviews_df[(reviews_df['IMDb_code'] == imdb_id) &
                                           (reviews_df['review_date'] == review_date)]

            if existing_reviews.empty:
                # If no existing review, add a new row
                reviews_df = reviews_df.append({
                    'IMDb_code': imdb_id,
                    'review_date': review_date,
                    'rating_of_movie': rating_of_movie,
                    'actual_review': actual_review
                }, ignore_index=True)
                updated_reviews.append((row['Title'], actual_review))
                print(f"Added new review for: {row['Title']}")

            else:
                # Updates the data if needed
                existing_review = existing_reviews.iloc[0]
                if (existing_review['actual_review'] != actual_review):
                    reviews_df.loc[existing_reviews.index, 'actual_review'] = actual_review
                    updated_reviews.append((row['Title'], actual_review))
                    print(f"Updated review for: {row['Title']}")

            time.sleep(1)

    # keep log of changes
    if updated_reviews:
        with open('update_log.txt', 'w') as log_file:
            for title, review in updated_reviews:
                log_file.write(f"Updated review for {title}: {review}\n")
        print("Update log saved to update_log.txt")
    else:
        print("No new reviews found.")

    return reviews_df, updated_reviews
```

1. Initializes variables and iterates through the movies DataFrame.
2. Fetches new reviews for each movie using the movie\_code and IMDb\_code columns.
3. Checks if a review already exists and updates, otherwise, it adds a new review.
4. Logs the changes to a file.
5. Returns the updated DataFrame and a list of updated reviews.

## SAMPLE EXECUTION

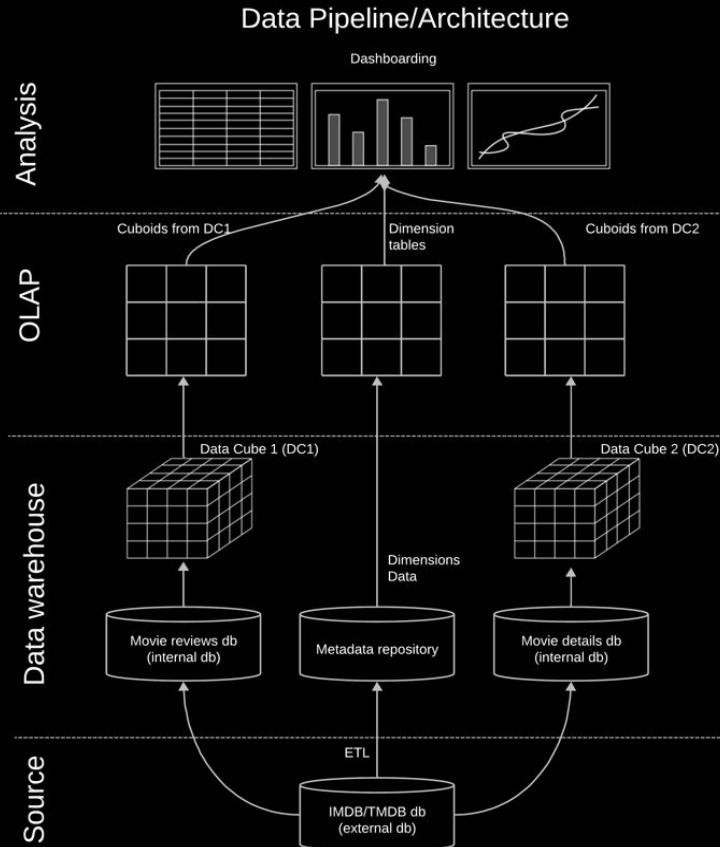
```
if __name__ == "__main__":
    movies_df = pd.read_pickle('movies_data.pkl')
    reviews_df = pd.read_pickle('reviews_data.pkl')

    reviews_df, updated_reviews = update_reviews(movies_df, reviews_df)

    reviews_df.to_pickle('updated_reviews_data.pkl')
```

No new reviews found.

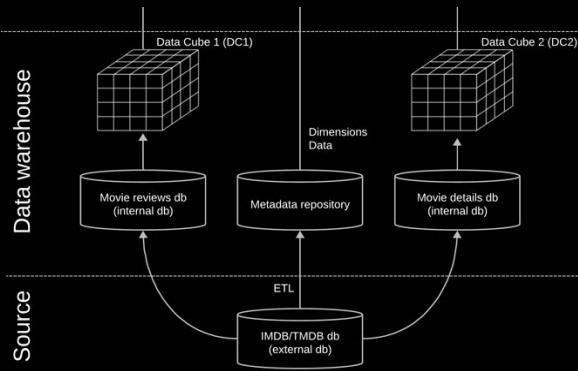
# DATA ARCHITECTURE



## Three-tier Architecture

1. Data warehouse - ETL the data from outside server (OLTP) then dump the pkl files to local drive.
2. OLAP
3. Analysis

# DATA ARCHITECTURE - WAREHOUSE



Movie details

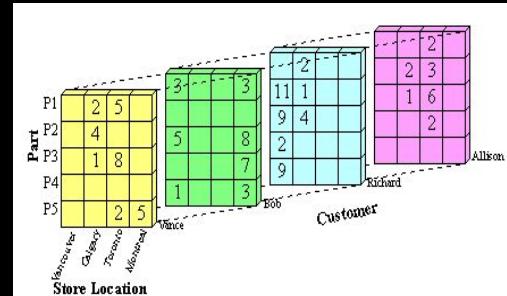
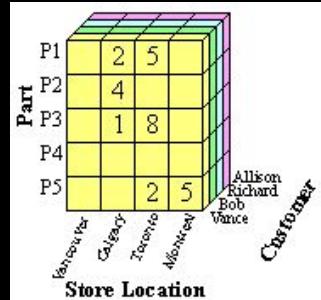
```
return {
    'movie_code': movie.get(
        'Title': movie.get('title'),
        'Year': movie.get('releas
        'Revenue': movie.get('re
        'Budget': movie.get('bu
        'Runtime': movie.get('ru
        'Opening_weekend_US': m
        'Rating': simplified_rat
        'Production_company': ,
        'Genre': ', '.join([genr
        'IMDb_code': imdb_id
    )
}
```

Movie reviews

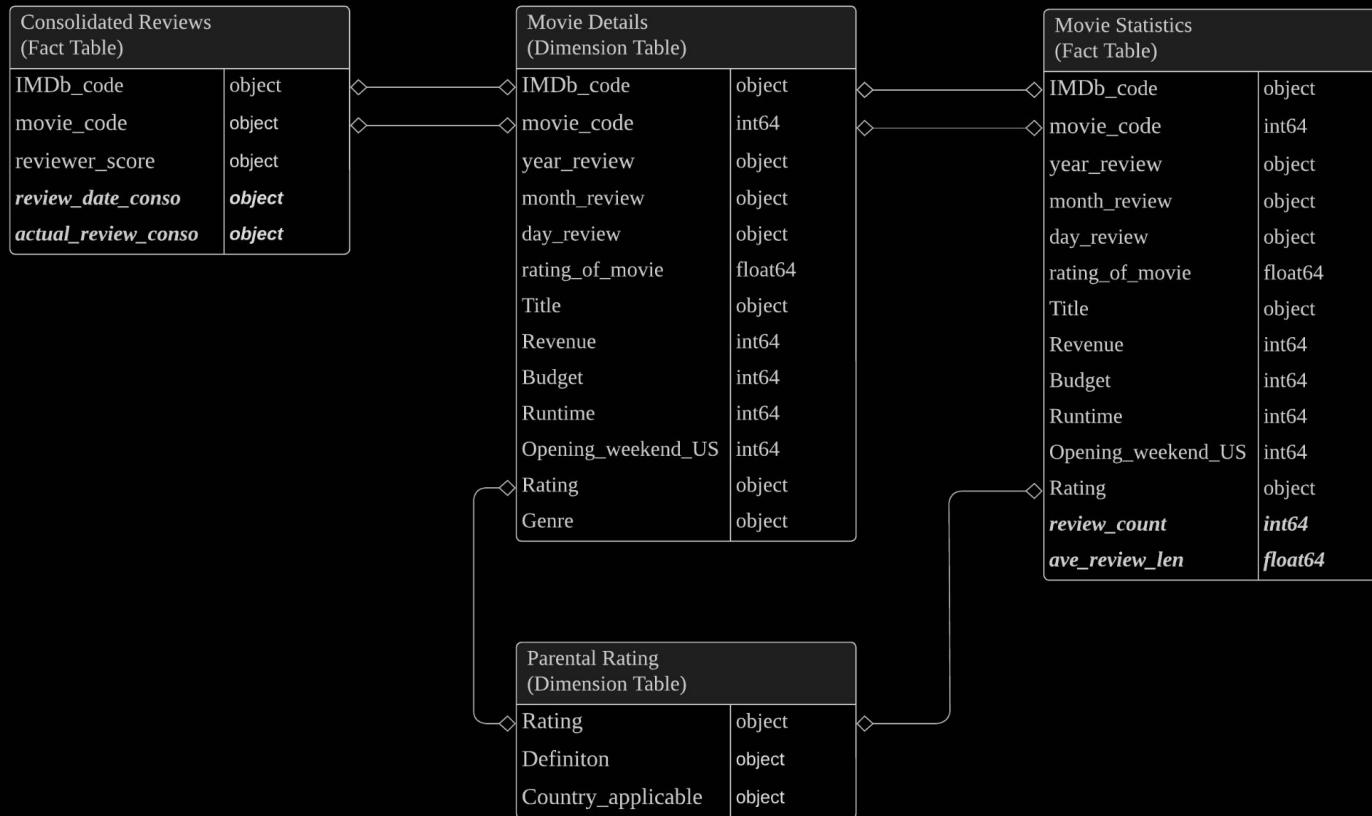
```
review_data = []
for review in reviews:
    review_data.append({
        'movie_code': tmc,
        'IMDb_code': imd
        'review_date': re
        'rating_of_movie'
        'actual_review':
    })
}
```

## Data Warehouse

1. Convert pkl file to data frame
2. Data cleaning
3. Construct fact tables (data cubes) and dimension tables from raw data and metadata details



# DATA ARCHITECTURE - CONSTELLATION SCHEMA (STAR)

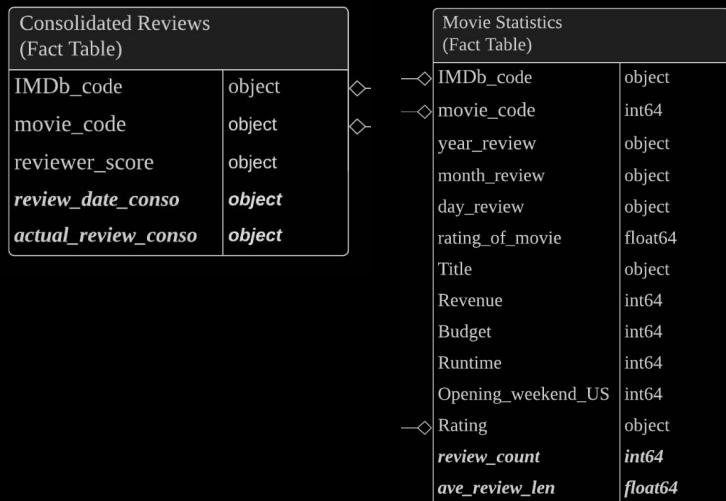


# DATA ARCHITECTURE - CONSTELLATION SCHEMA (STAR)

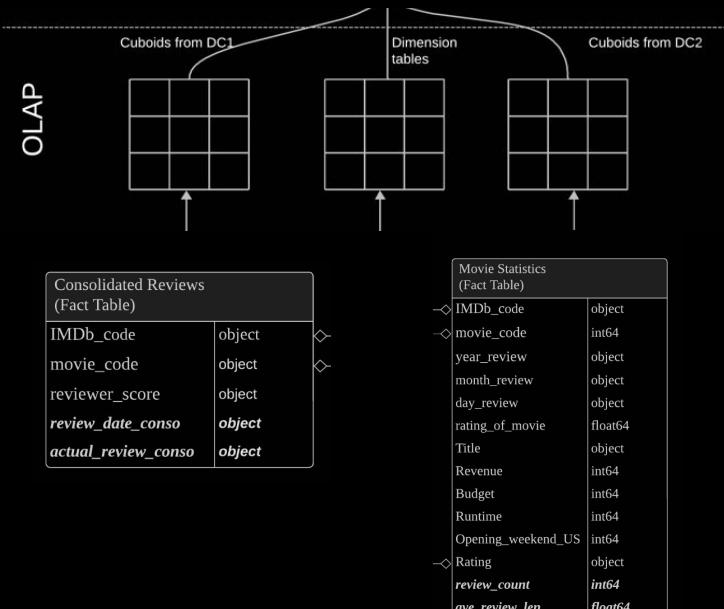


## Advantages using constellation schema

1. The user will have to access lesser data volume depending on the task (NLP vs basic movie analytics)
2. Data are broken down into smaller data cubes (e.g. movie statistics and movie reviews)
3. Still has the flexibility of connecting and processing both fact tables



# DATA ARCHITECTURE - OLAP



## Initial OLAP operations

- Movie reviews table (bottom left) can give all the reviews per movie and reviewer score. All the stop words are removed.
- Movie statistics table (bottom right) can readily give the average review length and count of reviews per movie, release date, rating of movie.
- Revenue, budget, genre, runtime and parental rating are added for easy access.

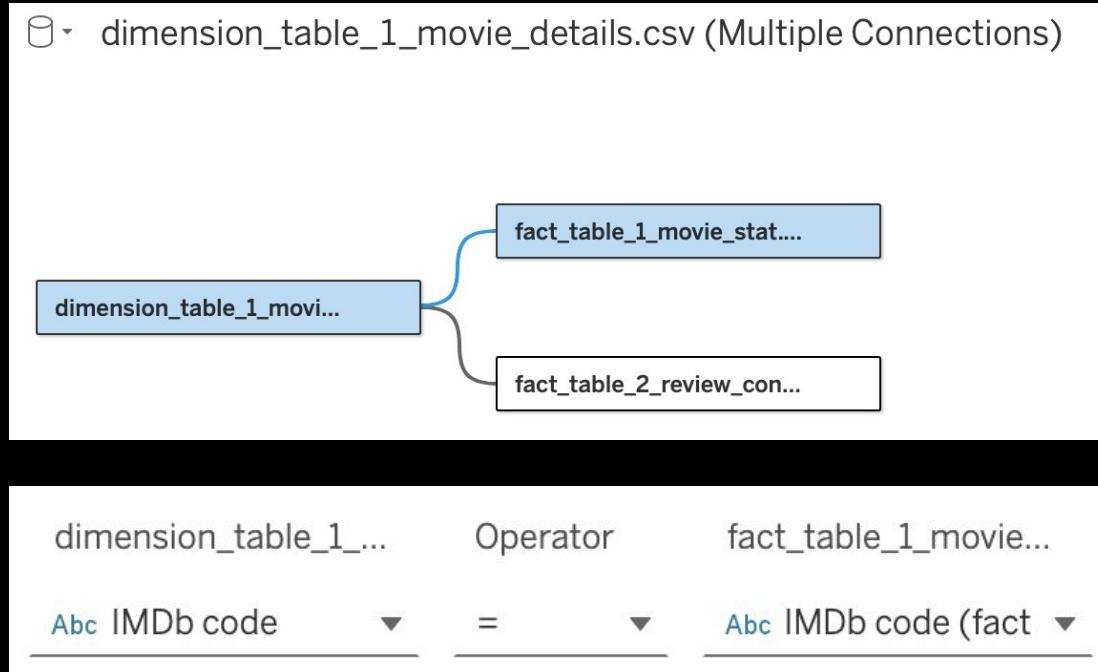


# DASHBOARD DEMO

[https://public.tableau.com/app/profile/lizelle.cruz/viz/IMDBDashboard\\_17223681730490/IMDBDashboard?publish=yes](https://public.tableau.com/app/profile/lizelle.cruz/viz/IMDBDashboard_17223681730490/IMDBDashboard?publish=yes)

# DATA SOURCE CONNECTION

We linked the dimension table and fact tables using the IMDb code in order to build dashboards taking variables from across the three data sources.



# TABLEAU WORKBOOK

IMDb

Tableau - IMDb Dashboard

Data Analytics

dimension\_table\_1\_movie

fact\_table\_1\_movie\_stat

fact\_table\_2\_review\_cons...

Search

Tables

dimension\_table\_1\_movie...

- # Day Released
- Abc Genre
- Abc IMDb code
- # Month Released
- # Movie Code
- Abc Production company
- Abc Rating
- Abc Title
- # Year Released
- # Budget
- # Opening weekend US
- # Revenue
- # Runtime
- # dimension\_table\_1\_mo...

fact\_table\_1\_movie\_stat...

fact\_table\_2\_review\_cons...

- Abc Actual Review Conso
- =Abc Actual Review Conso - ...
- Abc IMDb code (fact table 2...)
- # Movie Code (Fact Table...)
- Abc Review Date Conso
- # Reviewer Score
- # fact\_table\_2\_review\_c...

Pages

Columns AVG(Reviewer Score)

Rows Title

HIGHEST-RATED FILMS

CLICK HERE TO FILTER...

2024

MILLER'S GIRL 7.667

ABIGAIL 7.500

DEADPOOL & WOLVERINE 7.500

CIVIL WAR 7.500

THE BEEKEEPER 7.200

DUNE: PART TWO 7.200

THE BIKERIDERS 7.000

LATE NIGHT WITH THE DEVIL 7.000

IRISH WISH 7.000

BADLAND HUNTERS 7.000

CHALLENGERS 6.750

REBEL MOON - PART TWO: THE SCARLEV 6.667

MONKEY MAN 6.600

HIT MAN 6.600

KINGDOM OF THE PLANET OF THE APES 6.500

INSIDE OUT 2 6.500

FURIOSA: A MAD MAX SAGA 6.500

THE FALL GUY 6.333

LOVE LIES BLEEDING 6.250

ROAD HOUSE 6.000

LAND OF BAD 6.000

KINDS OF KINDNESS 6.000

IMMACULATE 6.000

GHOSTBUSTERS: FROZEN EMPIRE 6.000

Avg. Reviewer Score

Data Source IMDB Dashboard Highest-Rated Films Top 10 Highest-Grossing Films Top 10 Highest-Budgeted Films Number of Movies Released Per ... Avg Revenue Per Year Top IMDB Dramas Top 10 Longest

# DASHBOARD

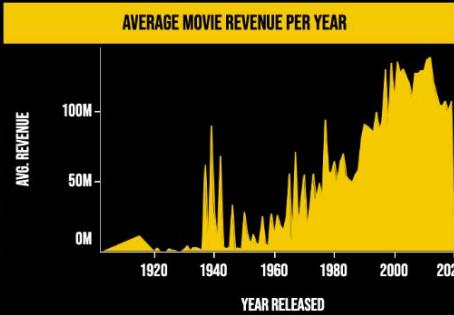
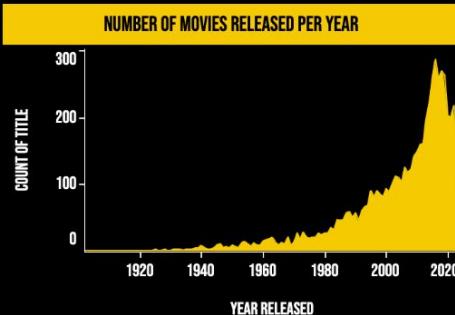
IMDb

IMDb

## MOVIE DASHBOARD

[CLICK HERE TO FILTER TOP FILMS BY YEAR](#)

2024



TOP 10 HIGHEST-GROSSING FILMS

TITLE	REVENUE
INSIDE OUT 2	1,469,875,604
DUNE: PART TWO	711,844,359
DESPICABLE ME 4	596,680,150
GODZILLA X KONG: T.	567,156,493
KUNG FU PANDA 4	543,900,620
DEADPOOL & WOLVE..	438,300,000
Kingdom of the PLA..	396,246,195
BAD BOYS: RIDE OR ..	388,242,778
A QUIET PLACE: DAY..	244,911,350
THE GARFIELD MOVIE	220,506,337

TOP 10 HIGHEST-BUDGETED FILMS

TITLE	BUDGET
INSIDE OUT 2	200,000,000
DEADPOOL & WOLVERINE	200,000,000
ARGYLLE	200,000,000
DUNE: PART TWO	190,000,000
FURIOSA: A MAD MAX SA..	170,000,000
KINGDOM OF THE PLANET..	160,000,000
GODZILLA X KONG: THE N..	150,000,000
BEVERLY HILLS COP: AXEL..	150,000,000
THE FALL GUY	125,000,000
IF	110,000,000

TOP 10 LONGEST FILMS

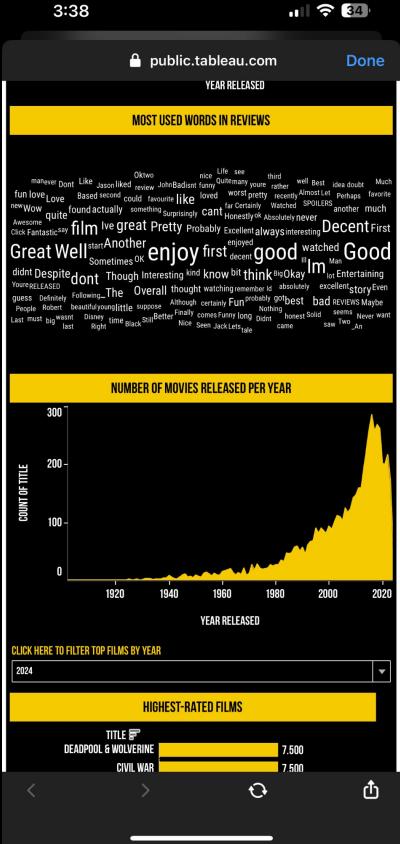
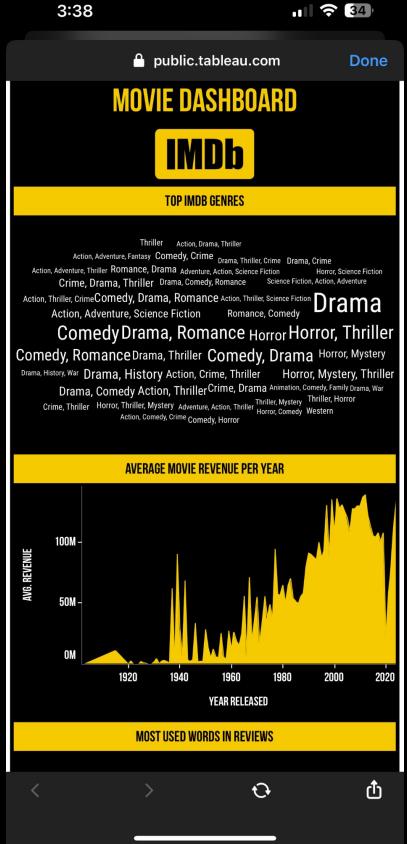
TITLE	AVG. RUNTIME
DUNE: PART TWO	167
KINDS OF KINDNESS	164
FURIOSA: A MAD MAX SA..	149
KINGDOM OF THE PLANET..	145
ARGYLLE	139
CHALLENGERS	132
DEADPOOL & WOLVERINE	128
THE FALL GUY	126
REBEL MOON - PART TWO..	123
ALIENOID: RETURN TO THE..	122

HIGHEST-RATED FILMS

TITLE	AVG. REVIEWER SCORE
MILLER'S GIRL	10.000
ABIGAIL	7.667
DEADPOOL & WOLVERINE	7.500
CIVIL WAR	7.500
THE BEEKEEPER	7.200
DUNE: PART TWO	7.200
THE BIKERIDERS	7.000
LATE NIGHT WITH THE DE..	7.000
IRISH WISH	7.000
BADLAND HUNTERS	7.000
CHALLENGERS	6.750

# DASHBOARD - MOBILE VIEW

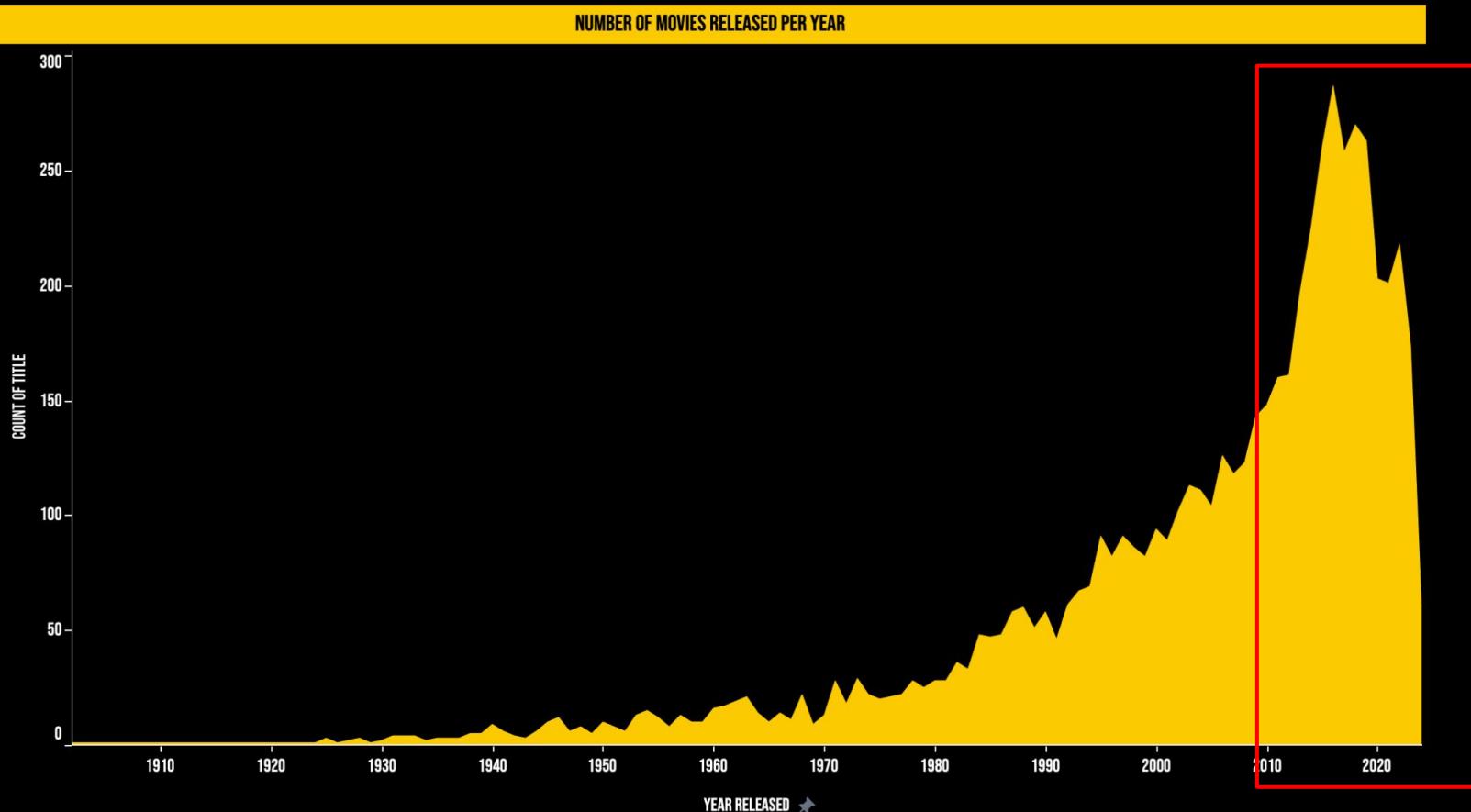
IMDb



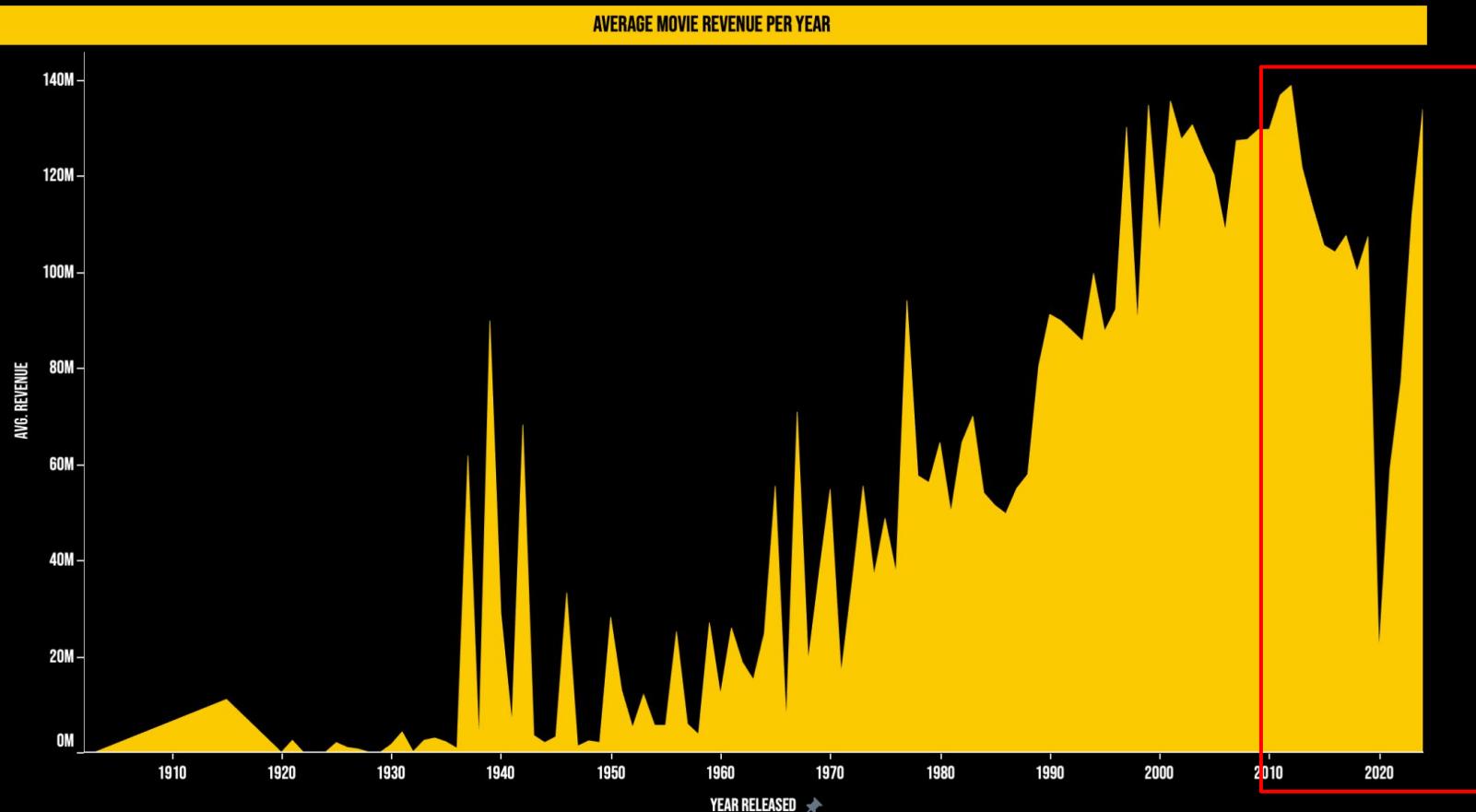


# DATA ANALYSIS AND INSIGHTS

# THE PANDEMIC AFFECTED THE FILM INDUSTRY

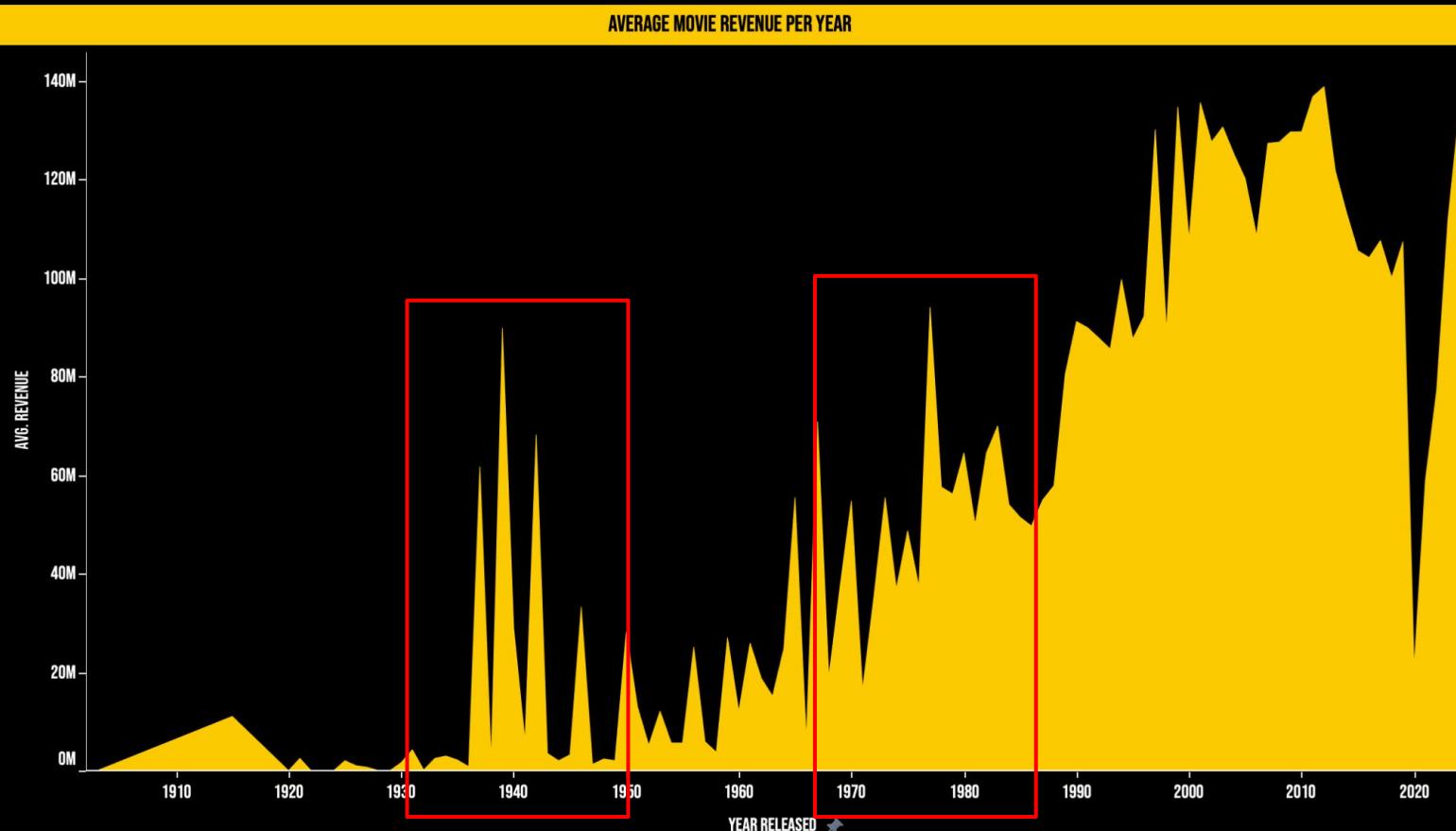


# THE PANDEMIC AFFECTED THE FILM INDUSTRY



# PEAKS IN 1939 AND 1977

IMDb



# DRAMA IS THE PREVAILING GENRE IN THE DATABASE

TOP IMDB GENRES



# THERE IS GENERALLY A POSITIVE SENTIMENT TOWARDS MOVIES

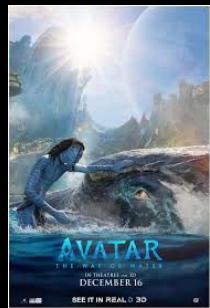
IMDb

MOST USED WORDS IN REVIEWS

A word cloud visualization showing the most used words in movie reviews. The words are arranged in a cluster, with larger words indicating higher frequency. Positive words like 'good', 'great', and 'enjoy' are prominent, while negative words like 'bad', 'terrible', and 'hate' are smaller.

The words include: OK, Quite, Best, must, You're, young, Big, you're, say, Right, tale, see, Jack, Last, came, Based, long, favorite, Still, Better, seems, doubt, man, kind, Funny, big, John, loved, Much, honest, Click, favourite, Almost, recently, something, could, review, Didn't, little, second, Nothing, Certainly, best, nice, Definitely, probably, never, Surprisingly, I'm, funny, Probably, quite, beautiful, Love, story, absolutely, RELEASED, Although, Maybe, excellent, film, Sometimes, Pretty, bit, actually, like, III, The, love, found, Let, think, dont, saw, interesting, ok, always, guess, Entertaining, Like, Despite, Lets, Overall, first, thought, enjoyed, last, remember, far, another, Fantastic, Excellent, know, watched, watching, Honestly, Great, cant, Watched, Never, Following, certainly, Awesome, decent, Perhaps, Fun, Even, bad, Absolutely, much, Seen, Disney, Dont, start, worst, fun, Wow, liked, decent, time, People, Finally, Jason, many, suppose, got, SPOILERS, didnt, Bad, new, want, two, comes, ever, rather, lot, Black, \_An, third, REVIEWs, live, pretty, Bad, idea, Solid, Robert, Nice, wasnt.

# ALL-TIME STATISTICS: MOVIE FRANCHISES PREVAIL



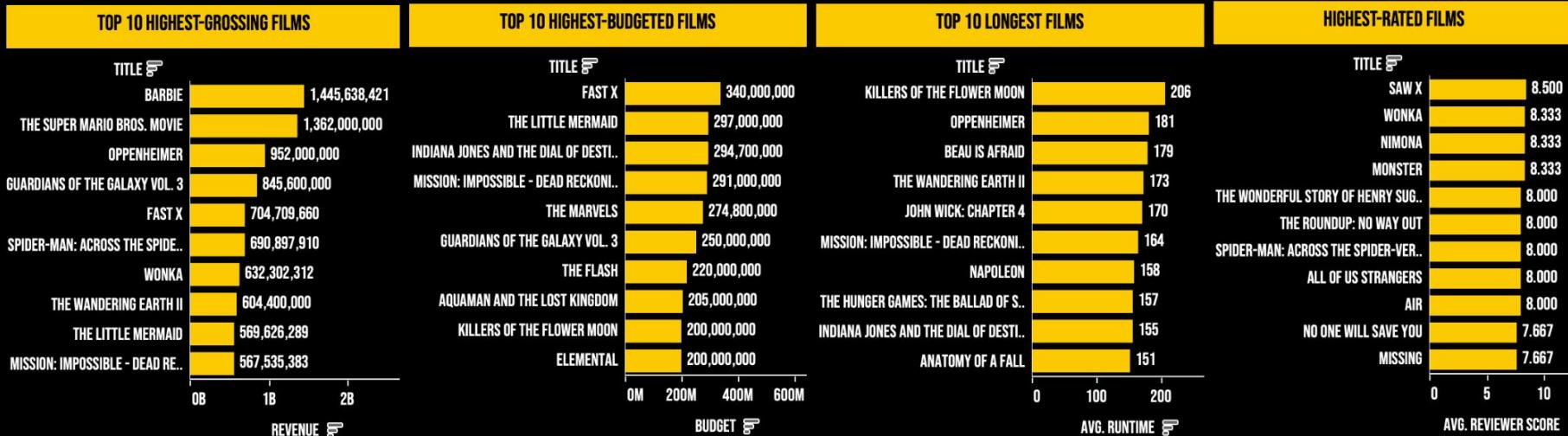
# 2024 STATISTICS

IMDb



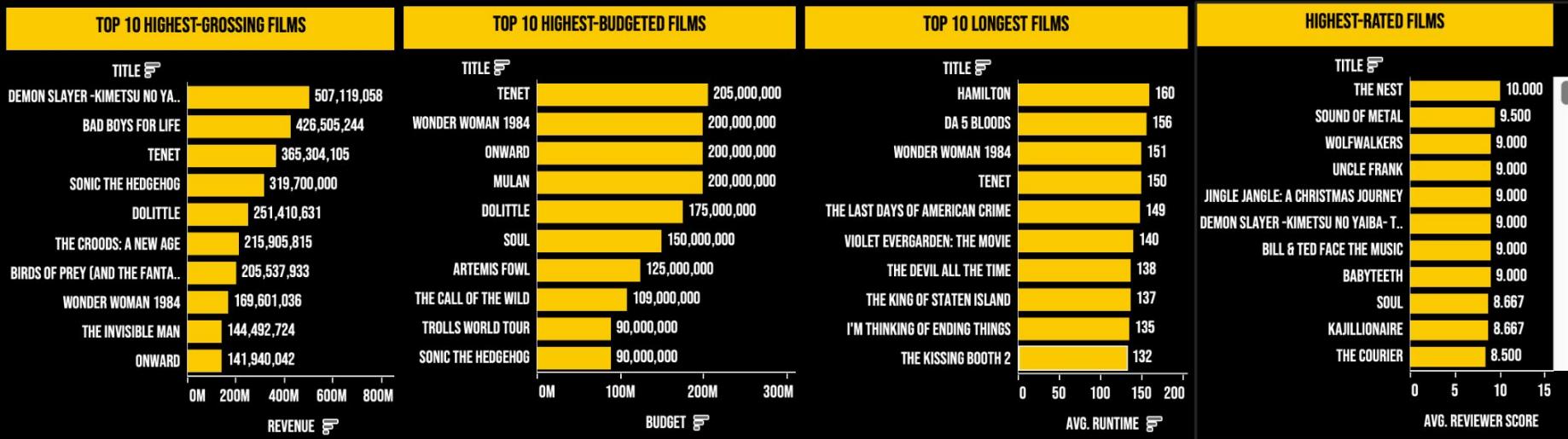
# 2023 STATISTICS

IMDb



# 2020: NO MOVIE REACHED 1 BILLION IN REVENUE

IMDb



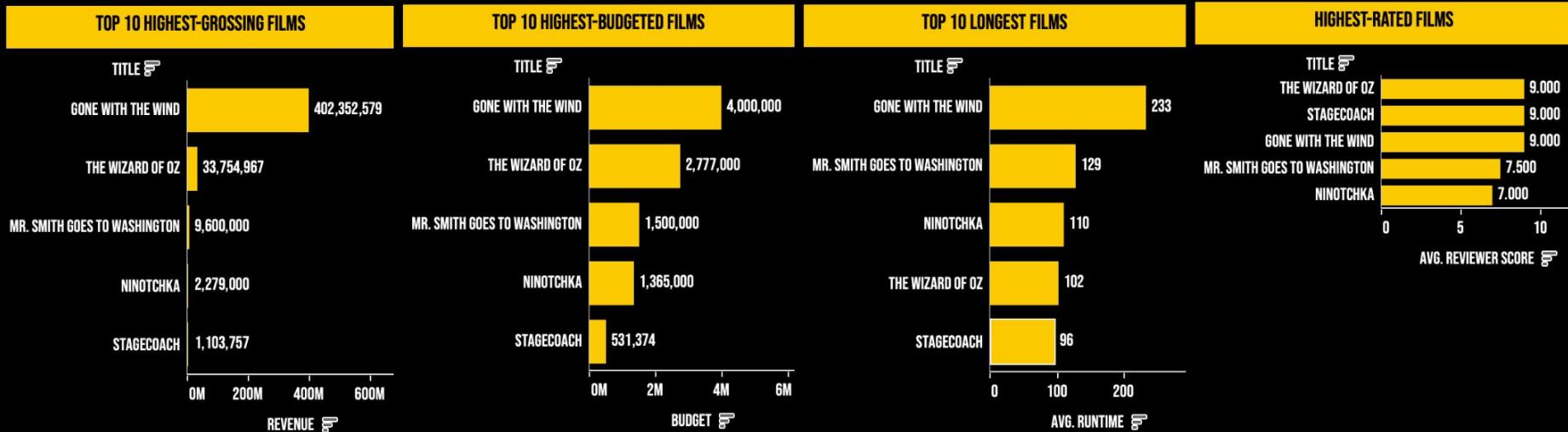
# 1977: THE RISE OF STAR WARS

IMDb



# 1939: GONE WITH THE WIND

IMDb



**Gone with the Wind**, American epic [film](#), released in 1939, that was one of the best known and most successful films of all time. It enjoyed a more-than-30-year reign as the all-time [Hollywood](#) box office champion, and it won eight [Academy Awards](#) (in addition to two honorary awards). Based on the runaway best-selling 1936 novel *Gone with the Wind* by [Margaret Mitchell](#), the movie is almost four hours long and includes an intermission.



# LEARNINGS AND BLOCKERS

# LEARNINGS AND BLOCKERS



LEARNINGS	BLOCKERS
<ul style="list-style-type: none"><li>• Utilize pickle module</li><li>• Understanding Movie as Business</li><li>• Understanding Schema and design</li></ul>	<ul style="list-style-type: none"><li>• Tableau Licensing</li><li>• Stop Words for Review Word Cloud</li><li>• Data warehouse designing</li><li>• Utilize various python modules</li></ul>



IMDb

THANK YOU!