

CSE 560 - PROJECT MILESTONE 1

❖ **Project Name:** Cinematic Insight Odyssey (Movie DB)

❖ **Team Name:** Insight Illuminati

❖ **Team Members (UBID):**

- Pranav Polavarapu (pranavpo)
- Sai Keerthi Chelluri (schellur)
- Nikhil Raj Yammani (nyammani)

❖ **Problem Statement:**

- Developing a movie database that allows users to browse and retrieve movie information efficiently. The system would provide detailed info about all-things-movies - from cast, crew, genre to ratings and unheard cinematic gems.
- By analyzing this database, users can gain insights into popular genres, directorial success, actor influence, and other trends in the movie industry.
- The goal is to create a comprehensive platform that enhances the movie-watching experience by providing access to a vast repository of movie-related data.
- The magnitude of the near-to-real-time Data that's available from sources like IMDB - calls for a well-defined, relational Database, which can open up doors for meaningful analytical insights through advanced querying.

❖ **Target Users & Business Insights:**

➤ **Demographics and Movie Preferences:**

- **Target Users:** Marketing teams, content developers, and researchers.
- **Business Insights:** Tailor marketing campaigns and content development to specific demographics based on their preferences. Understand which demographics are more engaged with movies.

- **Movie Performance Analysis:**
 - **Target Users:** Production houses, distributors, and marketing teams.
 - **Business Insights:** Inform movie production and distribution decisions by understanding factors influencing audience reception. Identify genres and talent with a higher potential for success.
- **Content Exploration and Recommendation Systems:**
 - **Target Users:** Streaming platforms, movie websites, and marketing teams.
 - **Business Insights:** Increase user engagement by providing personalized movie recommendations and improving content discoverability. Understand audience preferences for movie series and franchises.
- **More Insights:**
 - **Target Users:** Researchers, analysts, and industry professionals.
 - **Business Insights:** Analyze crew collaboration patterns and their impact on movie success. Explore how cultural factors influence movie preferences. Analyze award-winning movies to identify common characteristics contributing to their success.

❖ **DATASET:**

- **Primary Source:** [IMDB Non-commercial Datasets](#). These are a set of 7 gzipped, TSV files (.tsv.gz) which are updated on a daily-basis.
- [Name.basics.tsv.gz](#) | [title.akas.tsv.gz](#) | [title.basics.tsv.gz](#) | [title.crew.tsv.gz](#) | [title.episode.tsv.gz](#) | [title.principals.tsv.gz](#) | [title.ratings.tsv.gz](#)
- Contain wide-domain information ranging from genres, ratings, crew, cast, people involved, movie titles/alternate titles etc;

❖ **Proposed Schema & ER Diagram:**

- The following is our proposed project schema & relations for the project. We designed & derived **11 tables in Total** - that can be transformed, restructured, derived & loaded **from** the **original 7 IMDB files**.

Table Name	Column Names	Primary Key(s)	Foreign Key(s)
Movies	movie_id, title, original_title, start_year, end_year, runtime_minutes, is_adult, average_rating, num_votes	movie_id	
Genre	genre_id, genre_name	genre_id	
MovieGenres	movie_id, genre_id	(movie_id, genre_id)	movie_id (FK) references Movies(movie_id), genre_id (FK) references Genres(genre_id)
People	person_id, name, birth_year, death_year	person_id	
Occupations	occupation_id, occupation_title	occupation_id	
CastAndCrew	movie_id, person_id, occupation_id, start_date, end_date	(movie_id, person_id, occupation_id)	movie_id (FK) references Movies(movie_id), person_id (FK) references People(person_id), occupation_id (FK) references Occupations(occupation_id)
AlternateTitles	alternate_title_id, movie_id, ordering, region, language, types, attributes, is_original_title	(movie_id, ordering)	movie_id (FK) references Movies(movie_id)
Title_Crew	title_crew_id, movie_id, director_id, writer_id	(movie_id, director_id, writer_id)	movie_id (FK) references Movies(movie_id), director_id (FK) references People(person_id), writer_id (FK) references People(person_id)
Episodes	episode_id, movie_id, parent_movie_id, season_number, episode_number	(movie_id, season_number, episode_number)	movie_id (FK) references Movies(movie_id), parent_movie_id (FK) references Movies(movie_id)
FilmRoles	film_role_id, movie_id, person_id, ordering, character_name	(movie_id, person_id, ordering)	movie_id (FK) references Movies(movie_id), person_id (FK) references People(person_id)
Rating	rating_id, movie_id, rating_timestamp	(movie_id, rating_timestamp)	movie_id (FK) references Movies(movie_id)

➤ **Entities:**

- **Movies:** Represents individual movies or TV series.
- **Genre:** Represents different genres available.
- **People:** Represents info of actors, directors, writers, and other personnel involved in movies.
- **Occupations:** Represents different job titles in the movie industry.
- **MovieGenres:** Relates movies to genres (many-to-many relationship).
- **CastAndCrew:** Relates movies to people with their specific occupations (many-to-many relationship).
- **AlternateTitles:** Stores alternative titles associated with movies.
- **Title_Crew:** Links movies to specific directors and writers.
- **Episodes:** Represents individual episodes within TV series.
- **FilmRoles:** Relates movies to people associated with specific roles or characters.
- **Rating:** Captures ratings given to movies.

ER DIAGRAM:



Movies Table:

movie_id (SERIAL): Primary key for the movie, automatically generated.
title (TEXT): Title of the movie.
original_title (TEXT): Original title of the movie.
start_year (INTEGER): Year when the movie was released.
end_year (INTEGER): Year when the movie ended (if applicable, otherwise null).
runtime_minutes (INTEGER): Duration of the movie in minutes.
is_adult (BOOLEAN): Indicates if the movie is intended for adults.
average_rating (NUMERIC): Average rating of the movie.
num_votes (INTEGER): Number of votes the movie has received.

Genre Table:

genre_id (SERIAL): Primary key for the genre, automatically generated.
genre_name (TEXT): Name of the genre.

MovieGenres Table:

movie_id (INTEGER): Foreign key referencing movie_id in Movies table.
genre_id (INTEGER): Foreign key referencing genre_id in Genre table.
People Table:

person_id (SERIAL): Primary key for the person, automatically generated.
name (TEXT): Name of the person.
birth_year (INTEGER): Year of birth of the person.
death_year (INTEGER): Year of death of the person (if applicable, otherwise null).

Occupations Table:

occupation_id (SERIAL): Primary key for the occupation, automatically generated.
occupation_title (TEXT): Title of the occupation.
CastAndCrew Table:

movie_id (INTEGER): Foreign key referencing movie_id in Movies table.
person_id (INTEGER): Foreign key referencing person_id in People table.
occupation_id (INTEGER): Foreign key referencing occupation_id in Occupations table.
start_date (DATE): Start date of the person's involvement in the movie.
end_date (DATE): End date of the person's involvement in the movie (if applicable, otherwise null).

AlternateTitles Table:

alternate_title_id (SERIAL): Primary key for the alternate title, automatically generated.
movie_id (INTEGER): Foreign key referencing movie_id in Movies table.
ordering (INTEGER): Order of the alternate title.
region (TEXT): Region for which the alternate title is applicable.
language (TEXT): Language of the alternate title.
types (TEXT[]): Array of types for the alternate title.
attributes (TEXT[]): Array of attributes for the alternate title.
is_original_title (BOOLEAN): Indicates if the alternate title is the original title of the movie.

TitleCrew Table:

title_crew_id (SERIAL): Primary key for the title crew, automatically generated.
movie_id (INTEGER): Foreign key referencing movie_id in Movies table.
director_id (INTEGER): Foreign key referencing person_id in People table for the director.
writer_id (INTEGER): Foreign key referencing person_id in People table for the writer.

Episodes Table:

episode_id (SERIAL): Primary key for the episode, automatically generated.
movie_id (INTEGER): Foreign key referencing movie_id in Movies table.
parent_movie_id (INTEGER): Foreign key referencing movie_id in Movies table for the parent movie.
season_number (INTEGER): Season number of the episode.
episode_number (INTEGER): Episode number within the season.

FilmRoles Table:

film_role_id (SERIAL): Primary key for the film role, automatically generated.
movie_id (INTEGER): Foreign key referencing movie_id in Movies table.
person_id (INTEGER): Foreign key referencing person_id in People table.
ordering (INTEGER): Order of the film role.
character_name (TEXT): Name of the character played by the person in the movie.

Rating Table:

movie_id (INTEGER): Primary key referencing movie_id in Movies table.
rating_timestamp (TIMESTAMP): Timestamp of the rating.
rating_source (TEXT): Source of the rating.
rating_value (NUMERIC): Value of the rating.

❖ **Mapping Transformations for LOADING Data:**

- The IMDB Data (.tsv.gz files) is not in a ready-to-load format. This is because of multiple columnar feature combinations being represented in just 7 primary files.
- To LOAD or Copy Data into our CREATED Table Relations - we must first transform the given data files using specific mappings for each table, and create new TSV files for each of the 11 tables,, such that the schema definition columns match with the final mapping transformations or filtered tsv files.
- For this purpose, we implemented multiple Python scripts - covering all 11 tables as desired.

Example script:

```

1: import pandas as pd

# Define the schema for the People table
people_schema = {
    "nconst": "person_id",
    "primaryName": "name",
    "birthYear": "birth_year",
    "deathYear": "death_year"
}

# Load the IMDb name.basics.tsv.gz file
people_data_path = "/Users/pranavpolavarapu/Desktop/SUNY UB /Sem 2/DMQL - CSE 560/Team Project/IMDB Dataset/data.tsv"
people_df = pd.read_csv(people_data_path, sep='\t')

# Filter the columns based on the schema
people_columns = list(people_schema.keys())
people_table = people_df[people_columns]

# Rename columns to match the schema
people_table = people_table.rename(columns=people_schema)

# Save the People table data to a TSV file
people_table_path = "/Users/pranavpolavarapu/Desktop/SUNY UB /Sem 2/DMQL - CSE 560/Team Project/TABLES DATA/People.tsv"
people_table.to_csv(people_table_path, sep='\t', index=False)

print("People table data saved to:", people_table_path)

```

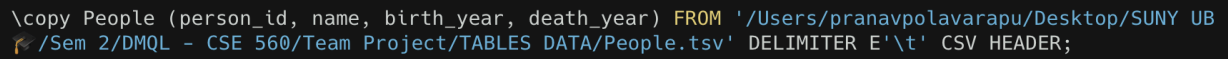
People table data saved to: /Users/pranavpolavarapu/Desktop/SUNY UB /Sem 2/DMQL - CSE 560/Team Project/TABLES DATA/People.tsv

⌕ ⌕ People.tsv — showing 128 MB of 393.7 MB

person_id	name	birth_year	death_year
nm0000001	Fred Astaire	1899	1987
nm0000002	Lauren Bacall	1924	2014
nm0000003	Brigitte Bardot	1934	\N
nm0000004	John Belushi	1949	1982
nm0000005	Ingmar Bergman	1918	2007
nm0000006	Ingrid Bergman	1915	1982
nm0000007	Humphrey Bogart	1899	1957
nm0000008	Marlon Brando	1924	2004
nm0000009	Richard Burton	1925	1984
nm0000010	James Cagney	1899	1986
nm0000011	Gary Cooper	1901	1961
nm0000012	Bette Davis	1908	1989
nm0000013	Doris Day	1922	2019
nm0000014	Olivia de Havilland	1916	2020
nm0000015	James Dean	1931	1955
nm0000016	Georges Delerue	1925	1992
nm0000017	Marlene Dietrich	1901	1992
nm0000018	Kirk Douglas	1916	2020
nm0000019	Federico Fellini	1920	1993
nm0000020	Henry Fonda	1905	1982
nm0000021	Joan Fontaine	1917	2013
nm0000022	Clark Gable	1901	1960
nm0000023	Judy Garland	1922	1969
nm0000024	John Gielgud	1904	2000
nm0000025	Jerry Goldsmith	1929	2004

❖ LOADING DATA:

- This was done using the \copy commands in the psql terminal.
- The source paths were: The transformed new TSV Files.
- Repeated for all 11 CREATED TABLES.



```
\copy People (person_id, name, birth_year, death_year) FROM '/Users/pranavpolavarapu/Desktop/SUNY UB  
/Sem 2/DMQL - CSE 560/Team Project/TABLES DATA/People.tsv' DELIMITER E'\t' CSV HEADER;
```

❖ Running QUERIES & Gathering Insights:

- **Business Insight (Problem) + CODE + Result** --> Shown in each of the 4 pictures:


```

1  -- Actors/Actresses with Highest Apperances on-screen
2  SELECT p.name AS actor_name, COUNT(DISTINCT c.movie_id) AS movie_count
3  FROM people p
4  JOIN castandcrew c ON p.person_id = c.person_id
5  WHERE c.occupation IN ('actor', 'actress')
6  GROUP BY p.name
7  ORDER BY movie_count DESC
8  LIMIT 20;
9

```

Data Output Messages Notifications











	actor_name text	movie_count bigint
1	Kenjiro Ishimaru	10849
2	Vic Sotto	10574
3	Sameera Sherief	10434
4	Tito Sotto	9992
5	Joel de Leon	9947
6	Manuela do Monte	8146
7	Delhi Kumar	8118
8	Arnold Clavio	8026
9	Pia Arcangel	8025
10	Giovanna Grigio	7947
11	Sudha Chandran	7394
12	Audrey Peters	7265
13	Subhalekha Sudhakar	7264
14	Ron Tomme	7239
15	Yuki Nozawa	6995
16	Yuma Sanada	6994
17	Neha Gowda	6596

```

-- Highest Voted+Rated Movies - Directors
SELECT
    m.title AS movie_title,
    p.name AS director_name,
    r.rating_value AS rating,
    r.num_votes AS num_votes
FROM
    movies m
JOIN
    castandcrew cc ON m.movie_id = cc.movie_id
JOIN
    people p ON cc.person_id = p.person_id AND cc.occupation = 'director'
JOIN
    rating r ON m.movie_id = r.movie_id
WHERE
    r.rating_value BETWEEN 8.5 AND 10
ORDER BY
    r.num_votes DESC
LIMIT
    20;

```

[a Output](#)
[Messages](#)
[Notifications](#)

							
movie_title	director_name	rating	num_votes				
text	text	numeric	numeric				
The Shawshank Redemption	Frank Darabont	9.3	2861216				
The Dark Knight	Christopher Nolan	9.0	2842501				
Inception	Christopher Nolan	8.8	2524044				
Fight Club	David Fincher	8.8	2296545				
Forrest Gump	Robert Zemeckis	8.8	2233427				
Pulp Fiction	Quentin Tarantino	8.9	2197100				
Interstellar	Christopher Nolan	8.7	2062535				
The Matrix	Lana Wachowski	8.7	2032857				
The Matrix	Lilly Wachowski	8.7	2032857				
The Godfather	Francis Ford Coppola	9.2	1993109				
The Lord of the Rings: The Fellowship of the Ring	Peter Jackson	8.9	1987538				
The Lord of the Rings: The Return of the King	Peter Jackson	9.0	1959764				
Se7en	David Fincher	8.6	1779643				

```

1  -- Finding Top Actor Pairs Who Worked Together Frequently on Highest Voted/Rated Films
2  SELECT
3      p1.name AS actor1_name,
4      p2.name AS actor2_name,
5      COUNT(*) AS num_collaborations
6  FROM
7      (
8          SELECT
9              m.movie_id
10         FROM
11             movies m
12         JOIN
13             rating r ON m.movie_id = r.movie_id
14         WHERE
15             r.rating_value BETWEEN 8.5 AND 9
16         ORDER BY
17             r.num_votes DESC
18         LIMIT
19             20
20     ) top_movies
21 JOIN
22     CastAndCrew cc1 ON top_movies.movie_id = cc1.movie_id
23 JOIN
24     CastAndCrew cc2 ON cc1.movie_id = cc2.movie_id
25 JOIN

```

Data Output Messages Notifications



	actor1_name text	actor2_name text	num_collaborations bigint
1	Elijah Wood	Orlando Bloom	3
2	Ian McKellen	Orlando Bloom	3
3	Elijah Wood	Ian McKellen	3
4	Elijah Wood	Viggo Mortensen	2
5	Viggo Mortensen	Orlando Bloom	2
6	Christian Bale	Michael Caine	2
7	Viggo Mortensen	Ian McKellen	2
8	Brad Pitt	Edward Norton	1
9	Brad Pitt	Andrew Kevin Walker	1
10	Alec Guinness	Mark Hamill	1
11	Christian Bale	Aaron Eckhart	1
12	Christian Bale	Heath Ledger	1
13	Christian Bale	Hugh Jackman	1
14	Brad Pitt	Zach Grenier	1
15	Brad Pitt	Morgan Freeman	1

```

1  -- Highest voted films by Director S.S Rajamouli
2  SELECT m.title AS movie_title, p.name AS person_name, c.occupation, r.num_votes
3  FROM movies m
4  JOIN castandcrew c ON m.movie_id = c.movie_id
5  JOIN people p ON c.person_id = p.person_id
6  JOIN rating r ON m.movie_id = r.movie_id
7  WHERE p.name = 'S.S. Rajamouli'
8  ORDER BY r.num_votes DESC;
9

```

Data Output Messages Notifications



	movie_title text	person_name text	occupation text	num_votes numeric
1	RRR	S.S. Rajamouli	director	165288
2	Baahubali: The Beginning	S.S. Rajamouli	director	136043
3	Baahubali 2: The Conclusion	S.S. Rajamouli	director	113406
4	Eega	S.S. Rajamouli	director	27015
5	Magadheera	S.S. Rajamouli	director	24824
6	Chatrapathi	S.S. Rajamouli	director	5566
7	Vikramarkudu	S.S. Rajamouli	director	4654
8	Thief of Yama	S.S. Rajamouli	director	4536
9	Simhadri	S.S. Rajamouli	director	3489
10	Titu Talks	S.S. Rajamouli	self	3266
11	Maryada Ramanna	S.S. Rajamouli	director	3148
12	Challenge	S.S. Rajamouli	director	2594
13	Student No. 1	S.S. Rajamouli	director	2055
14	Andala Rakshasi	S.S. Rajamouli	producer	724

➤ **Further Changes & Progress of the project shall be shown by the Next Milestone (2).**