CCPS510: Database Systems 1
Instructor - Ersan Cam

Comprehensive Movie and TV Show Database: A Scalable Solution for Entertainment

Data Management
Ofir Sela (501153003)
Kajaram Kanthasamy (501115080)
Kyle Purves (500315422)

Problem Statement

Our project is developing a comprehensive Movie and TV Show Database Management System. The problem lies in the lack of centralized and structured data for managing production companies, directors, actors, movies, TV shows, and related reviews. Current systems often lack the flexibility to handle complex relationships, such as actors appearing in multiple seasons, movies belonging to genres, and reviews tied to movies and TV shows. This database aims to efficiently manage this data, ensuring accuracy, consistency, and scalability for businesses such as streaming platforms, production houses, or entertainment analytics services.

Providing the Solution

The database will provide a structured and centralized solution for managing the intricate relationships between movies, TV shows, actors, genres, production companies, and reviews. The system will ensure data integrity and eliminate redundancy by using relational database principles. Features such as foreign keys, cascading updates, and constraints will maintain referential integrity. Queries can be performed to analyze trends (e.g., average ratings, box office returns, actor contributions), enabling decision-makers to derive actionable insights for improving content strategies and managing resources efficiently.

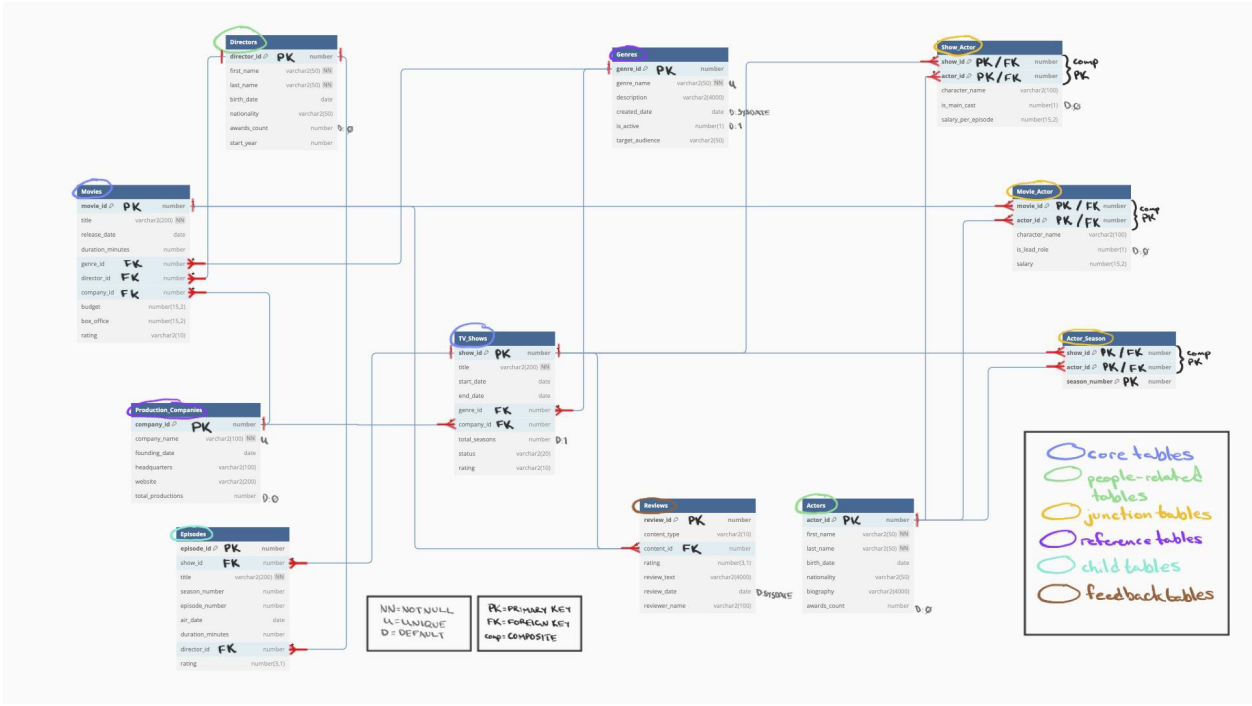
How the Database Fits

The database will act as the core data layer in a complete system architecture. It will be accessed through an API that enables secure and efficient data retrieval and manipulation. The API will provide data to a front-end web or mobile application for use by users such as administrators, content analysts, or general audiences. The front-end application will allow users to search for movies and TV shows, view actor profiles, and submit reviews, which will then be stored in the database. This integration ensures the database becomes part of a fully functional software process, facilitating seamless interactions between the user interface and the underlying data.

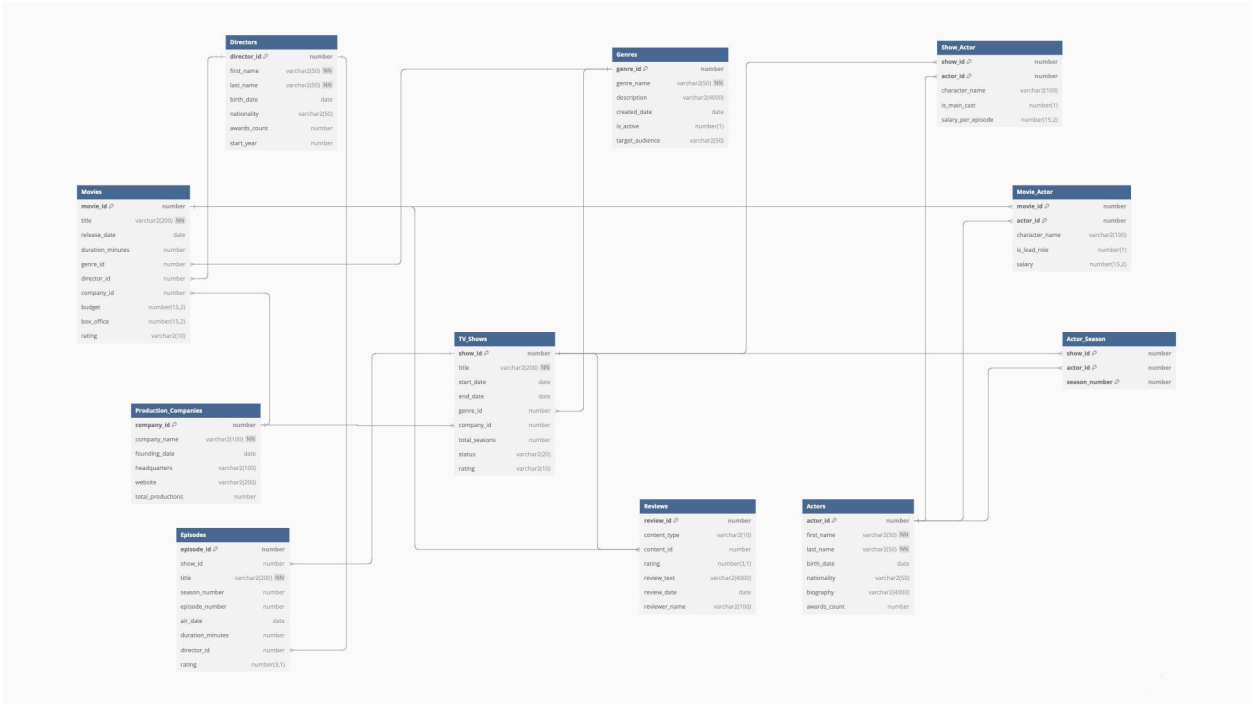
Business Requirements/Rules

- Each movie must belong to one and only one genre.
- A TV show can have multiple episodes, and each episode must belong to exactly one season.
- An actor can appear in multiple movies or TV shows, and for TV shows, their appearances can vary by season.
- A director can direct multiple movies or episodes, but each movie or episode must have exactly one director.
- Each review must correspond to either a movie or a TV show.
- A production company can produce multiple movies or TV shows.
- Genres are pre-defined and cannot be deleted if referenced by any movie or TV show.
- Each genre must have a unique name and description.
- A production company must have a unique name and keep track of the total number of productions they have completed.
- A director can only work on productions starting in or after their specified start year.
- An actor must have a unique combination of first name, last name, and birth date.
- Movies and TV shows must have appropriate age ratings (e.g., G, PG, 14A, 18A for movies; C, C8, PG, 14+, 18+ for TV shows).
- TV shows must have a status (Running, Ended, Cancelled, or On Hiatus).
- Episode numbers must be unique within each season of a TV show.
- Actor salaries are tracked differently for movies (total salary) versus TV shows (salary per episode)

ERD (with detailed notes):



ERD (clean version):



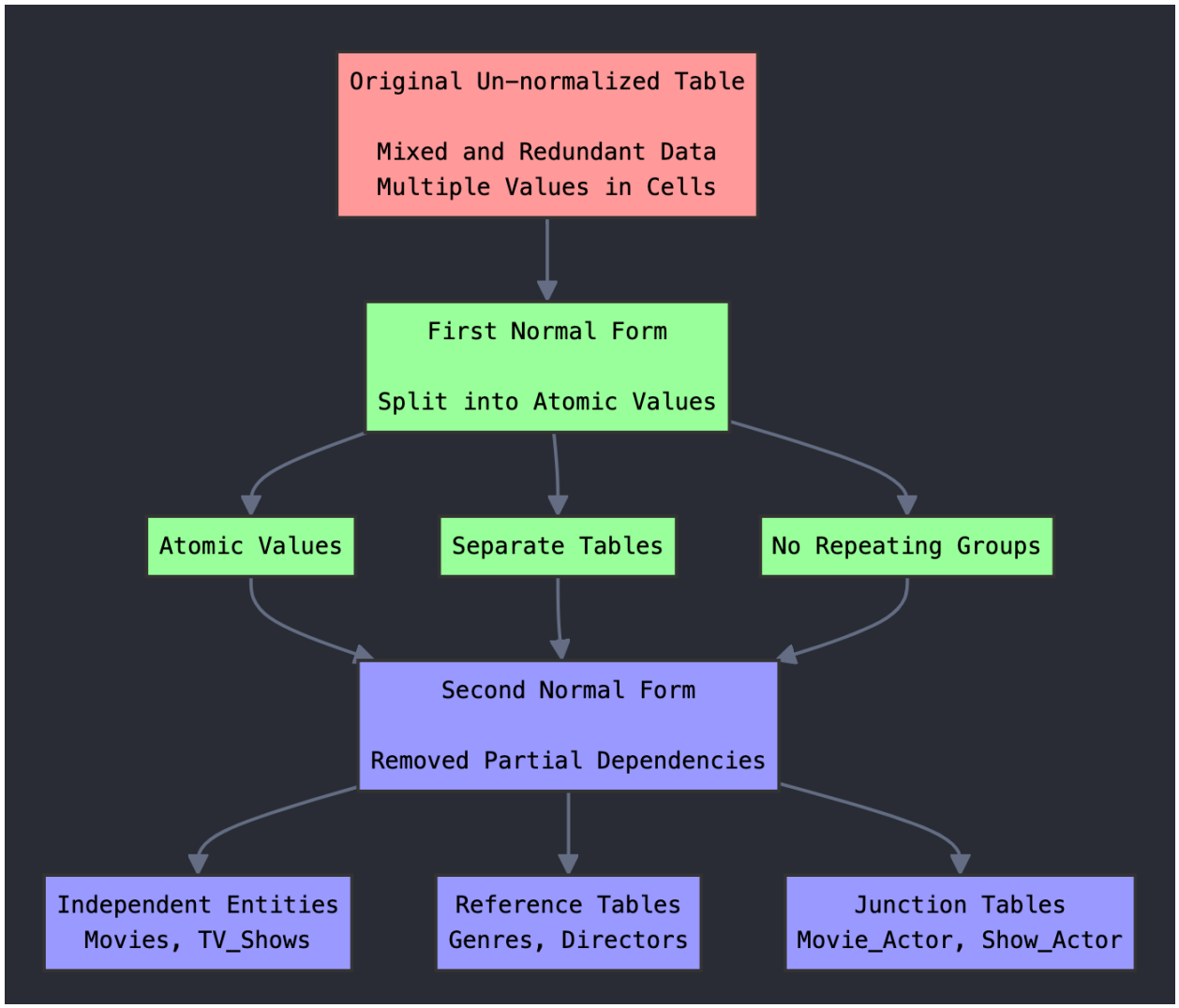
Database Normalization Report

1. Original Un-Normalized

Movie_ Show_ID	Title	Type	Genre	Director_ Info	Production_Info	Actor_ List	Rating	Reviews
1	Inception	Movie	Action, Sci-Fi	C.Nolan, 11 awards, start 1998	Warner Bros., Burbank CS, 900	DiCaprio as Cobb ($20M), Tom Hardy as Eames ($7M)	14A	MovieCritic123:9.0, FilmBuff: 8.8
2	The Crown	Show	Drama, History	Various Directors	Netflix, Los Gatos CA, 500	C. Foy as Queen (250K/ep), M. Smith as Philip (200K)	14+	TVFan: 8.5, SeriesLover: 9.0

Issues:

- Multiple values in single columns as strings
- Mixed data types (ex, salary could be total or per episode)
- Redundancy (ex, Production_Info repeated for each movie
- No relationships to ensure data integrity



2. First Normal Form (1NF)

Adjustments:

- Separate values into individual rows
- Ensure atomic values in each column
- Create separate tables for each entity

Movies/Shows

ID	Title	Type	Rating
1	Inception	Movie	14A
2	The Crown	Show	14+

Genres_Applied

Movie_ID	Genre
1	Action
1	Sci-Fi
2	Drama
2	History

Director_Details

Movie_ID	Director_name	Nationality	Awards	Start_Year
1	C. Nolan	British	11	1998

Production_Details

Movie_ID	Company_Name	Location	Total_Prodcutions
1	Warner Bros	Burbank, CA	900
2	Netflix	Los Gatos, CA	500

Cast_Details

Movie_ID	Actor_Name	Character	Salary	Salary_Typer
1	DiCaprio	Cobb	20000000	Total
1	Hardy	Eames	7000000	Total
2	C. Foy	Queen	250000	Per episode

Reviews

Movie_ID	Reviewer	Rating	Review_Text
1	MovieCritic123	9.0	Amazing!
1	FilmBuff	8.8	Great!

3. Second Normal Form (2NF)

Adjustments:

- Remove partial dependencies (ex. Production_Info now depends on Company_ID)
    - Attributes must depend on entire primary key
- Create independent tables for reusable (ex. Genres, Directors, Production_Companies, Actors
- Create bridge tables for scalability
- Use foreign keys (REFERENCES) to maintain relationships
- Add constraints (ex. CHECK, NOT NULL) for data validity

Genres (reference table for genres)

Genre_ID	Genre_Name	Description
1	Action	High-energy films
2	Sci-Fi	Futuristic stories

Directors (independent entity for directors)

Director_ID	First_name	Last_Name	Nationality	Awards_Count	Start_Year
1	Christopher	Nolan	British	11	1998

Production_Companies (independent entity for companies)

Company_ID	Company_Name	Location	Total_Prodcutions
1	Warner Bros	Burbank, CA	900
2	Netflix	Los Gatos, CA	500

Actors (independent entity for actors)

Actor_ID	First_name	Last_Name	Nationality
1	Leonardo	DiCaprio	American
2	Tom	Hardy	British

Movies (main entity with foreign keys)

Movie_ID	Title	Type	Rating	Director_ID	Company_ID
1	Inception	Movie	14A	1	1

Movie_Actor (bridge table to deal with many-to-many relationships)

Movie_ID	Actor_ID	Character	Salary	is_Lead_Role
1	1	Cobb	20000000	1
1	2	Eames	7000000	0

Table Creation SQL code

```
-- Create Genres table
CREATE TABLE Genres (
    genre_id INT PRIMARY KEY,
    genre_name VARCHAR2(50) NOT NULL UNIQUE,
    description VARCHAR2(4000),
    created_date DATE DEFAULT SYSDATE,
    is_active NUMBER(1) DEFAULT 1,
    target_audience VARCHAR2(50)
);

-- Create Production Companies table
CREATE TABLE Production_Companies (
    company_id INT PRIMARY KEY,
    company_name VARCHAR(100) NOT NULL UNIQUE,
    founding_date DATE,
    headquarters VARCHAR(100),
    website VARCHAR(200),
    total_productions INT DEFAULT 0
);

-- Create Directors table
CREATE TABLE Directors (
    director_id INT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    birth_date DATE,
    nationality VARCHAR(50),
    awards_count INT DEFAULT 0,
    start_year INT CHECK (start_year >= 1900)
```

```
);

-- Create Actors table
CREATE TABLE Actors (
    actor_id INT PRIMARY KEY,
    first_name VARCHAR2(50) NOT NULL,
    last_name VARCHAR2(50) NOT NULL,
    birth_date DATE,
    nationality VARCHAR2(50),
    biography VARCHAR2(4000),
    awards_count INT DEFAULT 0
);

-- Create Movies table
CREATE TABLE Movies (
    movie_id INT PRIMARY KEY,
    title VARCHAR2(200) NOT NULL,
    release_date DATE,
    duration_minutes INT CHECK (duration_minutes > 0),
    genre_id INT REFERENCES Genres(genre_id) ON DELETE CASCADE,
    director_id INT REFERENCES Directors(director_id) ON DELETE SET NULL,
    company_id INT REFERENCES Production_Companies(company_id) ON DELETE SET NULL,
    budget NUMBER(15,2),
    box_office NUMBER(15,2),
    rating VARCHAR2(10) CHECK (rating IN ('G', 'PG', '14A', '18A', 'R', 'A')),
    CONSTRAINT movie_release_check CHECK (release_date >= TO_DATE('1900-01-01', 'YYYY-MM-
DD'))
);

-- Create TV Shows table
CREATE TABLE TV_Shows (
    show_id INT PRIMARY KEY,
    title VARCHAR(200) NOT NULL,
    start_date DATE,
    end_date DATE,
    genre_id INT REFERENCES Genres(genre_id) ON DELETE CASCADE,
    company_id INT REFERENCES Production_Companies(company_id) ON DELETE SET NULL,
    total_seasons INT DEFAULT 1,
    status VARCHAR(20) CHECK (status IN ('Running', 'Ended', 'Cancelled', 'On Hiatus')),
    rating VARCHAR(10) CHECK (rating IN ('C', 'C8', 'G', 'PG', '14+', '18+')),
    CONSTRAINT show_dates_check CHECK (end_date >= start_date)
);

-- Create Episodes table
CREATE TABLE Episodes (
    episode_id INT PRIMARY KEY,
    show_id INT REFERENCES TV_Shows(show_id) ON DELETE CASCADE,
    title VARCHAR(200) NOT NULL,
    season_number INT CHECK (season_number > 0),
    episode_number INT CHECK (episode_number > 0),
    air_date DATE,
    duration_minutes INT CHECK (duration_minutes > 0),
```

```
    director_id INT REFERENCES Directors(director_id) ON DELETE SET NULL,
    rating DECIMAL(3,1) CHECK (rating >= 0 AND rating <= 10),
    UNIQUE(show_id, season_number, episode_number)
);

-- Create Movie_Actor bridge table
CREATE TABLE Movie_Actor (
    movie_id INT REFERENCES Movies(movie_id) ON DELETE CASCADE,
    actor_id INT REFERENCES Actors(actor_id) ON DELETE CASCADE,
    character_name VARCHAR2(100),
    is_lead_role NUMBER(1) DEFAULT 0,
    salary NUMBER(15,2),
    PRIMARY KEY (movie_id, actor_id)
);

-- Create Show_Actor bridge table
CREATE TABLE Show_Actor (
    show_id INT REFERENCES TV_Shows(show_id) ON DELETE CASCADE,
    actor_id INT REFERENCES Actors(actor_id) ON DELETE CASCADE,
    character_name VARCHAR2(100),
    is_main_cast NUMBER(1) DEFAULT 0,
    salary_per_episode NUMBER(15,2),
    PRIMARY KEY (show_id, actor_id)
);

-- Create Actor_Season bridge table
CREATE TABLE Actor_Season (
    show_id INT REFERENCES TV_Shows(show_id) ON DELETE CASCADE,
    actor_id INT REFERENCES Actors(actor_id) ON DELETE CASCADE,
    season_number INT CHECK (season_number > 0),
    PRIMARY KEY (show_id, actor_id, season_number)
);

-- Create Reviews table
CREATE TABLE Reviews (
    review_id INT PRIMARY KEY,
    content_type VARCHAR2(10) CHECK (content_type IN ('Movie', 'TV Show')),
    content_id INT,
    rating NUMBER(3,1) CHECK (rating >= 0 AND rating <= 10),
    review_text VARCHAR2(4000),
    review_date DATE DEFAULT SYSDATE,
    reviewer_name VARCHAR2(100)
);
```

Data Insertion SQL code

```sql
-- Genres Data
INSERT INTO Genres (genre_id, genre_name, description, created_date, is_active, target_audience)
VALUES (1, 'Action', 'High-energy films with chase scenes and stunts', SYSDATE, 1, 'Young Adults');

INSERT INTO Genres (genre_id, genre_name, description, created_date, is_active, target_audience)
VALUES (2, 'Drama', 'Character-driven stories with emotional themes', SYSDATE, 1, 'Adults');

INSERT INTO Genres (genre_id, genre_name, description, created_date, is_active, target_audience)
VALUES (3, 'Comedy', 'Light-hearted entertainment with humor', SYSDATE, 1, 'All Ages');

INSERT INTO Genres (genre_id, genre_name, description, created_date, is_active, target_audience)
VALUES (4, 'Science Fiction', 'Futuristic stories with advanced technology', SYSDATE, 1, 'Young Adults');

INSERT INTO Genres (genre_id, genre_name, description, created_date, is_active, target_audience)
VALUES (5, 'Fantasy', 'Magical and supernatural elements', SYSDATE, 1, 'All Ages');

-- Production Companies Data
INSERT INTO Production_Companies (company_id, company_name, founding_date, headquarters, website, total_productions)
VALUES (1, 'Universal Studios', TO_DATE('1912-04-30', 'YYYY-MM-DD'), 'Universal City, CA', 'www.universalstudios.com',
1000);

INSERT INTO Production_Companies (company_id, company_name, founding_date, headquarters, website, total_productions)
VALUES (2, 'Warner Bros.', TO_DATE('1923-04-04', 'YYYY-MM-DD'), 'Burbank, CA', 'www.warnerbros.com', 900);

INSERT INTO Production_Companies (company_id, company_name, founding_date, headquarters, website, total_productions)
VALUES (3, 'Netflix Studios', TO_DATE('1997-08-29', 'YYYY-MM-DD'), 'Los Gatos, CA', 'www.netflix.com', 500);

INSERT INTO Production_Companies (company_id, company_name, founding_date, headquarters, website, total_productions)
VALUES (4, 'HBO', TO_DATE('1972-11-08', 'YYYY-MM-DD'), 'New York, NY', 'www.hbo.com', 300);

INSERT INTO Production_Companies (company_id, company_name, founding_date, headquarters, website, total_productions)
VALUES (5, 'Paramount Pictures', TO_DATE('1912-05-08', 'YYYY-MM-DD'), 'Hollywood, CA', 'www.paramount.com', 800);

-- Directors Data
INSERT INTO Directors (director_id, first_name, last_name, birth_date, nationality, awards_count, start_year)
VALUES (1, 'Christopher', 'Nolan', TO_DATE('1970-07-30', 'YYYY-MM-DD'), 'British', 11, 1998);

INSERT INTO Directors (director_id, first_name, last_name, birth_date, nationality, awards_count, start_year)
VALUES (2, 'Ava', 'DuVernay', TO_DATE('1972-08-24', 'YYYY-MM-DD'), 'American', 8, 2006);

INSERT INTO Directors (director_id, first_name, last_name, birth_date, nationality, awards_count, start_year)
VALUES (3, 'Martin', 'Scorsese', TO_DATE('1942-11-17', 'YYYY-MM-DD'), 'American', 20, 1967);

INSERT INTO Directors (director_id, first_name, last_name, birth_date, nationality, awards_count, start_year)
VALUES (4, 'Kathryn', 'Bigelow', TO_DATE('1951-11-27', 'YYYY-MM-DD'), 'American', 9, 1978);

INSERT INTO Directors (director_id, first_name, last_name, birth_date, nationality, awards_count, start_year)
VALUES (5, 'Bong', 'Joon-ho', TO_DATE('1969-09-14', 'YYYY-MM-DD'), 'South Korean', 15, 1994);

INSERT INTO Directors (director_id, first_name, last_name, birth_date, nationality, awards_count, start_year)
VALUES (6, 'Quentin', 'Tarantino', TO_DATE('1963-03-27', 'YYYY-MM-DD'), 'American', 32, 1987);

INSERT INTO Directors (director_id, first_name, last_name, birth_date, nationality, awards_count, start_year)
VALUES (7, 'Greta', 'Gerwig', TO_DATE('1983-08-04', 'YYYY-MM-DD'), 'American', 12, 2008);

INSERT INTO Directors (director_id, first_name, last_name, birth_date, nationality, awards_count, start_year)
VALUES (8, 'Denis', 'Villeneuve', TO_DATE('1967-10-03', 'YYYY-MM-DD'), 'Canadian', 15, 1994);

INSERT INTO Directors (director_id, first_name, last_name, birth_date, nationality, awards_count, start_year)
VALUES (9, 'Chloé', 'Zhao', TO_DATE('1982-03-31', 'YYYY-MM-DD'), 'Chinese', 10, 2010);
```

```sql
INSERT INTO Directors (director_id, first_name, last_name, birth_date, nationality, awards_count, start_year)
VALUES (10, 'Jordan', 'Peele', TO_DATE('1979-02-21', 'YYYY-MM-DD'), 'American', 8, 2012);

-- Actors Data
INSERT INTO Actors (actor_id, first_name, last_name, birth_date, nationality, biography, awards_count)
VALUES (1, 'Leonardo', 'DiCaprio', TO_DATE('1974-11-11', 'YYYY-MM-DD'), 'American', 'Award-winning actor known for
dramatic roles', 56);

INSERT INTO Actors (actor_id, first_name, last_name, birth_date, nationality, biography, awards_count)
VALUES (2, 'Meryl', 'Streep', TO_DATE('1949-06-22', 'YYYY-MM-DD'), 'American', 'Versatile actress with numerous
accolades', 94);

INSERT INTO Actors (actor_id, first_name, last_name, birth_date, nationality, biography, awards_count)
VALUES (3, 'Denzel', 'Washington', TO_DATE('1954-12-28', 'YYYY-MM-DD'), 'American', 'Acclaimed actor and director',
60);

INSERT INTO Actors (actor_id, first_name, last_name, birth_date, nationality, biography, awards_count)
VALUES (4, 'Viola', 'Davis', TO_DATE('1965-08-11', 'YYYY-MM-DD'), 'American', 'Powerful dramatic actress', 71);

INSERT INTO Actors (actor_id, first_name, last_name, birth_date, nationality, biography, awards_count)
VALUES (5, 'Tom', 'Hardy', TO_DATE('1977-09-15', 'YYYY-MM-DD'), 'British', 'Known for intense character portrayals',
24);

INSERT INTO Actors (actor_id, first_name, last_name, birth_date, nationality, biography, awards_count)
VALUES (6, 'Cate', 'Blanchett', TO_DATE('1969-05-14', 'YYYY-MM-DD'), 'Australian', 'Versatile actress known for both
stage and screen', 85);

INSERT INTO Actors (actor_id, first_name, last_name, birth_date, nationality, biography, awards_count)
VALUES (7, 'Samuel L.', 'Jackson', TO_DATE('1948-12-21', 'YYYY-MM-DD'), 'American', 'Prolific actor with iconic roles',
45);

INSERT INTO Actors (actor_id, first_name, last_name, birth_date, nationality, biography, awards_count)
VALUES (8, 'Margot', 'Robbie', TO_DATE('1990-07-02', 'YYYY-MM-DD'), 'Australian', 'Rising star known for diverse
roles', 28);

INSERT INTO Actors (actor_id, first_name, last_name, birth_date, nationality, biography, awards_count)
VALUES (9, 'Timothée', 'Chalamet', TO_DATE('1995-12-27', 'YYYY-MM-DD'), 'American', 'Young actor known for dramatic
performances', 15);

INSERT INTO Actors (actor_id, first_name, last_name, birth_date, nationality, biography, awards_count)
VALUES (10, 'Zendaya', 'Coleman', TO_DATE('1996-09-01', 'YYYY-MM-DD'), 'American', 'Multi-talented actress and
performer', 22);

-- Movies Data
INSERT INTO Movies (movie_id, title, release_date, duration_minutes, genre_id, director_id, company_id, budget,
box_office, rating)
VALUES (1, 'Inception', TO_DATE('2010-07-16', 'YYYY-MM-DD'), 148, 4, 1, 2, 160000000, 836800000, '14A');

INSERT INTO Movies (movie_id, title, release_date, duration_minutes, genre_id, director_id, company_id, budget,
box_office, rating)
VALUES (2, 'The Post', TO_DATE('2017-12-22', 'YYYY-MM-DD'), 116, 2, 3, 1, 50000000, 179000000, 'PG');

INSERT INTO Movies (movie_id, title, release_date, duration_minutes, genre_id, director_id, company_id, budget,
box_office, rating)
VALUES (3, 'Training Day', TO_DATE('2001-10-05', 'YYYY-MM-DD'), 122, 2, 4, 2, 45000000, 104900000, '18A');

INSERT INTO Movies (movie_id, title, release_date, duration_minutes, genre_id, director_id, company_id, budget,
box_office, rating)
VALUES (4, 'Parasite', TO_DATE('2019-05-30', 'YYYY-MM-DD'), 132, 2, 5, 5, 11400000, 258773645, '14A');

INSERT INTO Movies (movie_id, title, release_date, duration_minutes, genre_id, director_id, company_id, budget,
box_office, rating)
VALUES (5, 'The Devil Wears Prada', TO_DATE('2006-06-30', 'YYYY-MM-DD'), 109, 3, 2, 1, 35000000, 326700000, 'PG');

INSERT INTO Movies (movie_id, title, release_date, duration_minutes, genre_id, director_id, company_id, budget,
box_office, rating)
VALUES (6, 'Pulp Fiction', TO_DATE('1994-10-14', 'YYYY-MM-DD'), 154, 2, 6, 5, 8000000, 213928762, '18A');
```

```
INSERT INTO Movies (movie_id, title, release_date, duration_minutes, genre_id, director_id, company_id, budget,
box_office, rating)
VALUES (7, 'Little Women', TO_DATE('2019-12-25', 'YYYY-MM-DD'), 135, 2, 7, 1, 40000000, 218844555, 'PG');

INSERT INTO Movies (movie_id, title, release_date, duration_minutes, genre_id, director_id, company_id, budget,
box_office, rating)
VALUES (8, 'Dune', TO_DATE('2021-10-22', 'YYYY-MM-DD'), 155, 4, 8, 2, 165000000, 401847917, 'PG');

INSERT INTO Movies (movie_id, title, release_date, duration_minutes, genre_id, director_id, company_id, budget,
box_office, rating)
VALUES (9, 'Nomadland', TO_DATE('2021-02-19', 'YYYY-MM-DD'), 108, 2, 9, 3, 5000000, 39601466, 'PG');

INSERT INTO Movies (movie_id, title, release_date, duration_minutes, genre_id, director_id, company_id, budget,
box_office, rating)
VALUES (10, 'Get Out', TO_DATE('2017-02-24', 'YYYY-MM-DD'), 104, 4, 10, 1, 4500000, 255407969, '14A');

-- TV Shows Data
INSERT INTO TV_Shows (show_id, title, start_date, end_date, genre_id, company_id, total_seasons, status, rating)
VALUES (1, 'Stranger Things', TO_DATE('2016-07-15', 'YYYY-MM-DD'), NULL, 4, 3, 4, 'Running', '14+');

INSERT INTO TV_Shows (show_id, title, start_date, end_date, genre_id, company_id, total_seasons, status, rating)
VALUES (2, 'Game of Thrones', TO_DATE('2011-04-17', 'YYYY-MM-DD'), TO_DATE('2019-05-19', 'YYYY-MM-DD'), 5, 4, 8,
'Ended', '18+');

INSERT INTO TV_Shows (show_id, title, start_date, end_date, genre_id, company_id, total_seasons, status, rating)
VALUES (3, 'The Office', TO_DATE('2005-03-24', 'YYYY-MM-DD'), TO_DATE('2013-05-16', 'YYYY-MM-DD'), 3, 1, 9, 'Ended',
'PG');

INSERT INTO TV_Shows (show_id, title, start_date, end_date, genre_id, company_id, total_seasons, status, rating)
VALUES (4, 'Breaking Bad', TO_DATE('2008-01-20', 'YYYY-MM-DD'), TO_DATE('2013-09-29', 'YYYY-MM-DD'), 2, 3, 5, 'Ended',
'18+');

INSERT INTO TV_Shows (show_id, title, start_date, end_date, genre_id, company_id, total_seasons, status, rating)
VALUES (5, 'The Crown', TO_DATE('2016-11-04', 'YYYY-MM-DD'), NULL, 2, 3, 5, 'Running', '14+');

INSERT INTO TV_Shows (show_id, title, start_date, end_date, genre_id, company_id, total_seasons, status, rating)
VALUES (6, 'The Mandalorian', TO_DATE('2019-11-12', 'YYYY-MM-DD'), NULL, 5, 3, 3, 'Running', '14+');

INSERT INTO TV_Shows (show_id, title, start_date, end_date, genre_id, company_id, total_seasons, status, rating)
VALUES (7, 'Succession', TO_DATE('2018-06-03', 'YYYY-MM-DD'), TO_DATE('2023-05-28', 'YYYY-MM-DD'), 2, 4, 4, 'Ended',
'18+');

INSERT INTO TV_Shows (show_id, title, start_date, end_date, genre_id, company_id, total_seasons, status, rating)
VALUES (8, 'Ted Lasso', TO_DATE('2020-08-14', 'YYYY-MM-DD'), TO_DATE('2023-05-31', 'YYYY-MM-DD'), 3, 3, 3, 'Ended',
'14+');

INSERT INTO TV_Shows (show_id, title, start_date, end_date, genre_id, company_id, total_seasons, status, rating)
VALUES (9, 'The Last of Us', TO_DATE('2023-01-15', 'YYYY-MM-DD'), NULL, 2, 4, 1, 'Running', '18+');

INSERT INTO TV_Shows (show_id, title, start_date, end_date, genre_id, company_id, total_seasons, status, rating)
VALUES (10, 'Wednesday', TO_DATE('2022-11-23', 'YYYY-MM-DD'), NULL, 5, 3, 1, 'Running', '14+');

-- Episodes Data
INSERT INTO Episodes (episode_id, show_id, title, season_number, episode_number, air_date, duration_minutes,
director_id, rating)
VALUES (1, 1, 'Chapter One: The Vanishing of Will Byers', 1, 1, TO_DATE('2016-07-15', 'YYYY-MM-DD'), 48, 2, 8.7);

INSERT INTO Episodes (episode_id, show_id, title, season_number, episode_number, air_date, duration_minutes,
director_id, rating)
VALUES (2, 2, 'Winter Is Coming', 1, 1, TO_DATE('2011-04-17', 'YYYY-MM-DD'), 62, 3, 9.1);

INSERT INTO Episodes (episode_id, show_id, title, season_number, episode_number, air_date, duration_minutes,
director_id, rating)
VALUES (3, 3, 'Pilot', 1, 1, TO_DATE('2005-03-24', 'YYYY-MM-DD'), 23, 4, 7.6);

INSERT INTO Episodes (episode_id, show_id, title, season_number, episode_number, air_date, duration_minutes,
```

```
director_id, rating)
VALUES (4, 4, 'Pilot', 1, 1, TO_DATE('2008-01-20', 'YYYY-MM-DD'), 58, 5, 9.0);

INSERT INTO Episodes (episode_id, show_id, title, season_number, episode_number, air_date, duration_minutes,
director_id, rating)
VALUES (5, 5, 'Wolferton Splash', 1, 1, TO_DATE('2016-11-04', 'YYYY-MM-DD'), 58, 1, 8.7);

INSERT INTO Episodes (episode_id, show_id, title, season_number, episode_number, air_date, duration_minutes,
director_id, rating)
VALUES (6, 6, 'Chapter 1: The Mandalorian', 1, 1, TO_DATE('2019-11-12', 'YYYY-MM-DD'), 39, 8, 8.8);

INSERT INTO Episodes (episode_id, show_id, title, season_number, episode_number, air_date, duration_minutes,
director_id, rating)
VALUES (7, 7, 'Succession', 1, 1, TO_DATE('2018-06-03', 'YYYY-MM-DD'), 62, 7, 8.9);

INSERT INTO Episodes (episode_id, show_id, title, season_number, episode_number, air_date, duration_minutes,
director_id, rating)
VALUES (8, 8, 'Pilot', 1, 1, TO_DATE('2020-08-14', 'YYYY-MM-DD'), 30, 6, 8.5);

INSERT INTO Episodes (episode_id, show_id, title, season_number, episode_number, air_date, duration_minutes,
director_id, rating)
VALUES (9, 9, 'When You''re Lost in the Darkness', 1, 1, TO_DATE('2023-01-15', 'YYYY-MM-DD'), 81, 9, 9.3);

INSERT INTO Episodes (episode_id, show_id, title, season_number, episode_number, air_date, duration_minutes,
director_id, rating)
VALUES (10, 10, 'Wednesday''s Child Is Full of Woe', 1, 1, TO_DATE('2022-11-23', 'YYYY-MM-DD'), 47, 10, 8.2);

-- Movie_Actor Data
INSERT INTO Movie_Actor (movie_id, actor_id, character_name, is_lead_role, salary)
VALUES (1, 1, 'Dom Cobb', 1, 20000000);

INSERT INTO Movie_Actor (movie_id, actor_id, character_name, is_lead_role, salary)
VALUES (1, 5, 'Eames', 0, 7000000);

INSERT INTO Movie_Actor (movie_id, actor_id, character_name, is_lead_role, salary)
VALUES (2, 2, 'Kay Graham', 1, 15000000);

INSERT INTO Movie_Actor (movie_id, actor_id, character_name, is_lead_role, salary)
VALUES (3, 3, 'Alonzo Harris', 1, 12000000);

INSERT INTO Movie_Actor (movie_id, actor_id, character_name, is_lead_role, salary)
VALUES (4, 4, 'Special Appearance', 0, 5000000);

INSERT INTO Movie_Actor (movie_id, actor_id, character_name, is_lead_role, salary)
VALUES (6, 7, 'Jules Winnfield', 1, 1000000);

INSERT INTO Movie_Actor (movie_id, actor_id, character_name, is_lead_role, salary)
VALUES (7, 8, 'Amy March', 1, 8000000);

INSERT INTO Movie_Actor (movie_id, actor_id, character_name, is_lead_role, salary)
VALUES (8, 9, 'Paul Atreides', 1, 12000000);

INSERT INTO Movie_Actor (movie_id, actor_id, character_name, is_lead_role, salary)
VALUES (9, 6, 'Fern', 1, 5000000);

INSERT INTO Movie_Actor (movie_id, actor_id, character_name, is_lead_role, salary)
VALUES (10, 10, 'Chris Washington', 1, 500000);

-- Show_Actor Data
INSERT INTO Show_Actor (show_id, actor_id, character_name, is_main_cast, salary_per_episode)
VALUES (1, 5, 'Guest Role', 0, 100000);

INSERT INTO Show_Actor (show_id, actor_id, character_name, is_main_cast, salary_per_episode)
VALUES (2, 4, 'Regular Character', 1, 150000);

INSERT INTO Show_Actor (show_id, actor_id, character_name, is_main_cast, salary_per_episode)
VALUES (3, 2, 'Guest Star', 0, 100000);
```

```sql
INSERT INTO Show_Actor (show_id, actor_id, character_name, is_main_cast, salary_per_episode)
VALUES (4, 3, 'Guest Role', 0, 200000);

INSERT INTO Show_Actor (show_id, actor_id, character_name, is_main_cast, salary_per_episode)
VALUES (5, 1, 'Special Appearance', 0, 250000);

INSERT INTO Show_Actor (show_id, actor_id, character_name, is_main_cast, salary_per_episode)
VALUES (6, 7, 'Moff Gideon', 1, 200000);

INSERT INTO Show_Actor (show_id, actor_id, character_name, is_main_cast, salary_per_episode)
VALUES (7, 6, 'Guest Star', 0, 350000);

INSERT INTO Show_Actor (show_id, actor_id, character_name, is_main_cast, salary_per_episode)
VALUES (8, 8, 'Guest Role', 0, 150000);

INSERT INTO Show_Actor (show_id, actor_id, character_name, is_main_cast, salary_per_episode)
VALUES (9, 9, 'Guest Appearance', 0, 300000);

-- Actor_Season Data
INSERT INTO Actor_Season (show_id, actor_id, season_number)
VALUES (1, 5, 4);

INSERT INTO Actor_Season (show_id, actor_id, season_number)
VALUES (2, 4, 1);

INSERT INTO Actor_Season (show_id, actor_id, season_number)
VALUES (2, 4, 2);

INSERT INTO Actor_Season (show_id, actor_id, season_number)
VALUES (2, 4, 3);

INSERT INTO Actor_Season (show_id, actor_id, season_number)
VALUES (3, 2, 7);

INSERT INTO Actor_Season (show_id, actor_id, season_number)
VALUES (4, 3, 5);

INSERT INTO Actor_Season (show_id, actor_id, season_number)
VALUES (5, 1, 3);

INSERT INTO Actor_Season (show_id, actor_id, season_number)
VALUES (6, 7, 2);

INSERT INTO Actor_Season (show_id, actor_id, season_number)
VALUES (6, 7, 3);

INSERT INTO Actor_Season (show_id, actor_id, season_number)
VALUES (7, 6, 4);

INSERT INTO Actor_Season (show_id, actor_id, season_number)
VALUES (8, 8, 2);

INSERT INTO Actor_Season (show_id, actor_id, season_number)
VALUES (9, 9, 1);

INSERT INTO Actor_Season (show_id, actor_id, season_number)
VALUES (10, 10, 1);

-- Reviews Data
INSERT INTO Reviews (review_id, content_type, content_id, rating, review_text, reviewer_name, review_date)
VALUES (1, 'Movie', 1, 9.0, 'A mind-bending masterpiece that challenges reality.', 'MovieCritic123', SYSDATE - 10);

INSERT INTO Reviews (review_id, content_type, content_id, rating, review_text, reviewer_name, review_date)
VALUES (2, 'TV Show', 1, 8.5, 'Nostalgic and thrilling series that keeps you on the edge.', 'SeriesLover', SYSDATE -
8);
```

```sql
INSERT INTO Reviews (review_id, content_type, content_id, rating, review_text, reviewer_name, review_date)
VALUES (3, 'Movie', 4, 9.5, 'A perfect blend of social commentary and dark humor.', 'FilmAnalyst', SYSDATE - 5);

INSERT INTO Reviews (review_id, content_type, content_id, rating, review_text, reviewer_name, review_date)
VALUES (4, 'TV Show', 2, 9.3, 'Epic storytelling with amazing character development.', 'TVFanatic', SYSDATE - 3);

INSERT INTO Reviews (review_id, content_type, content_id, rating, review_text, reviewer_name, review_date)
VALUES (5, 'Movie', 2, 8.0, 'Powerful portrayal of journalism and freedom of press.', 'CinemaExpert', SYSDATE - 1);

INSERT INTO Reviews (review_id, content_type, content_id, rating, review_text, reviewer_name, review_date)
VALUES (6, 'Movie', 6, 9.4, 'A groundbreaking classic that redefined cinema.', 'CinemaVeteran', SYSDATE - 15);

INSERT INTO Reviews (review_id, content_type, content_id, rating, review_text, reviewer_name, review_date)
VALUES (7, 'TV Show', 6, 8.9, 'Fresh take on the Star Wars universe.', 'SciFiFanatic', SYSDATE - 12);

INSERT INTO Reviews (review_id, content_type, content_id, rating, review_text, reviewer_name, review_date)
VALUES (8, 'Movie', 8, 8.8, 'Visually stunning adaptation of the classic novel.', 'FilmScholar', SYSDATE - 9);

INSERT INTO Reviews (review_id, content_type, content_id, rating, review_text, reviewer_name, review_date)
VALUES (9, 'TV Show', 9, 9.2, 'Masterful adaptation of the video game.', 'SeriesEnthusiast', SYSDATE - 6);

INSERT INTO Reviews (review_id, content_type, content_id, rating, review_text, reviewer_name, review_date)
VALUES (10, 'Movie', 10, 9.0, 'Revolutionary horror film with social commentary.', 'HorrorBuff', SYSDATE - 4);
```

Complex SELECT queries

1. Movie Profitability Analysis:

This query:

- evaluates the financial performance of each movie,
- calculates its profit (difference between box office revenue and budget) and return on investment (ROI),
- links each movie to its genre, director, and production company to provide a complete picture of contributing factors to profitability.

```sql
SELECT
    m.title,                          -- Movie name
    g.genre_name,                     -- Genre of the movie
    m.budget,                         -- Original budget
    m.box_office,                     -- Total earnings
    (m.box_office - m.budget) as profit,  -- Simple profit calculation
    ROUND(((m.box_office - m.budget) / m.budget * 100), 2) as roi_percentage,  -- Return on Investment
    d.first_name || ' ' || d.last_name as director_name,  -- Combined director name
    pc.company_name                   -- Production company
FROM Movies m
JOIN Genres g ON m.genre_id = g.genre_id                -- Link to get genre info
JOIN Directors d ON m.director_id = d.director_id        -- Link to get director info
JOIN Production_Companies pc ON m.company_id = pc.company_id  -- Link to get company info
WHERE m.box_office > 0                                  -- Only include movies with earnings
ORDER BY roi_percentage DESC;                           -- Sort by most profitable first
```

- The JOIN clauses bring in data from related tables like Genres, Directors, and Production_Companies.
- The profit is calculated as m.box_office - m.budget, and ROI is derived by dividing the profit by the budget.
- The WHERE clause excludes movies without box office earnings.
- Sorting by roi_percentage ensures the most profitable movies appear first.

Insights:

- Identifies the top-performing movies based on ROI.
- Demonstrates how different elements, like genre and production companies, contribute to financial success.
- Enables stakeholders to make data-driven decisions about future investments.

	TITLE	GENRE_NAME	BUDGET	BOX_OFFICE	PROFIT	ROI_PERCENTAGE	DIRECTOR_NAME	COMPANY_NAME
1	Get Out	Science Fiction	4500000	255407969	250907969	5575.73	Jordan Peele	Universal Studios
2	Pulp Fiction	Drama	8000000	213928762	205928762	2574.11	Quentin Tarantino	Paramount Pictures
3	Parasite	Drama	11400000	258773645	247373645	2169.94	Bong Joon-ho	Paramount Pictures
4	The Devil Wears Prada	Comedy	35000000	326700000	291700000	833.43	Ava DuVernay	Universal Studios
5	Nomadland	Drama	5000000	39601466	34601466	692.03	Chloé Zhao	Netflix Studios
6	Little Women	Drama	40000000	218844555	178844555	447.11	Greta Gerwig	Universal Studios
7	Inception	Science Fiction	160000000	836800000	676800000	423	Christopher Nolan	Warner Bros.
8	The Post	Drama	50000000	179000000	129000000	258	Martin Scorsese	Universal Studios
9	Dune	Science Fiction	165000000	401847917	236847917	143.54	Denis Villeneuve	Warner Bros.
10	Training Day	Drama	45000000	104900000	59900000	133.11	Kathryn Bigelow	Warner Bros.

2. Actor Career Summary:

This query:

- Provides an overview of each actor's career.
- Counts their participation in movies and TV shows.
- Calculates their total earnings and highlights the number of awards they have won.

```sql
SELECT
    a.first_name || ' ' || a.last_name as actor_name,    -- Actor's full name
    COUNT(DISTINCT ma.movie_id) as movie_count,          -- Number of movies acted in
    COUNT(DISTINCT sa.show_id) as show_count,            -- Number of TV shows acted in
    SUM(ma.salary) as total_movie_earnings,              -- Total earnings from movies
    AVG(sa.salary_per_episode) as avg_episode_salary,    -- Average per-episode salary
    a.awards_count                                       -- Number of awards won
FROM Actors a
LEFT JOIN Movie_Actor ma ON a.actor_id = ma.actor_id     -- Link to movie roles
LEFT JOIN Show_Actor sa ON a.actor_id = sa.actor_id      -- Link to TV show roles
GROUP BY a.actor_id, a.first_name, a.last_name, a.awards_count
HAVING COUNT(DISTINCT ma.movie_id) > 0 OR COUNT(DISTINCT sa.show_id) > 0;  -- Only active actors
```

- LEFT JOIN ensures actors without roles in one medium (movies or TV) are still included in the results.
- COUNT(DISTINCT) prevents double-counting roles across movies or shows.

- Aggregates like SUM and AVG summarize financial data for each actor.
- The HAVING clause filters out actors with no roles in either medium.

Insights:

- Highlights prolific actors and their career scope across mediums.
- Provides a comparison of movie vs. TV show involvement.
- Helps identify award-winning actors for casting or promotional decisions.
- we can better understand their impact on the industry and identify top performers across different mediums.

	ACTOR_NAME	MOVIE_COUNT	SHOW_COUNT	TOTAL_MOVIE_EARNINGS	AVG_EPISODE_SALARY	AWARDS_COUNT
1	Leonardo DiCaprio	1	1	20000000	250000	56
2	Meryl Streep	1	1	15000000	100000	94
3	Denzel Washington	1	1	12000000	200000	60
4	Viola Davis	1	1	5000000	150000	71
5	Tom Hardy	1	1	7000000	100000	24
6	Cate Blanchett	1	1	5000000	350000	85
7	Samuel L. Jackson	1	1	1000000	200000	45
8	Margot Robbie	1	1	8000000	150000	28
9	Timothée Chalamet	1	1	12000000	300000	15
10	Zendaya Coleman	1	0	500000	(null)	22
11	Jenna Ortega	0	1	(null)	250000	5

3. TV Show Performance Analysis:

This query:

- Evaluates the performance of TV shows based on episode ratings, season count, and cast size.
- Provides insights into average actor salaries and other performance metrics.

```sql
SELECT
    t.title,                                    -- Show name
    t.total_seasons,                            -- Number of seasons
    COUNT(e.episode_id) as total_episodes,      -- Count of episodes
    AVG(e.rating) as avg_episode_rating,        -- Average episode rating
    COUNT(DISTINCT sa.actor_id) as cast_size,   -- Number of actors
    g.genre_name,                               -- Genre of show
    t.status,                                   -- Current status (Running/Ended)
    AVG(sa.salary_per_episode) as avg_actor_salary  -- Average actor salary
FROM TV_Shows t
JOIN Episodes e ON t.show_id = e.show_id           -- Link to episodes
JOIN Genres g ON t.genre_id = g.genre_id           -- Link to genre
LEFT JOIN Show_Actor sa ON t.show_id = sa.show_id  -- Link to actors
GROUP BY t.show_id, t.title, t.total_seasons, g.genre_name, t.status
ORDER BY avg_episode_rating DESC;
```

- The JOIN with Episodes allows the calculation of total episodes and average ratings.
- LEFT JOIN with Show_Actor includes salary data and cast size for each show.
- GROUP BY ensures metrics are calculated per show, while ORDER BY ranks them by episode ratings.

Insights:

- Ranks shows based on episode ratings and performance metrics.
- Analyzes cast size and salary trends across shows.
- Assists in identifying successful genres and high-performing content.

	TITLE	TOTAL_SEASONS	TOTAL_EPISODES	AVG_EPISODE_RATING	CAST_SIZE	GENRE_NAME	STATUS	AVG_ACTOR_SALARY
1	The Last of Us	1	1	9.3	1	Drama	Running	300000
2	Game of Thrones	8	1	9.1	1	Fantasy	Ended	150000
3	Breaking Bad	5	1	9	1	Drama	Ended	200000
4	Succession	4	1	8.9	1	Drama	Ended	350000
5	The Mandalorian	3	1	8.8	1	Fantasy	Running	200000
6	Stranger Things	4	1	8.7	1	Science Fiction	Running	100000
7	The Crown	5	1	8.7	1	Drama	Running	250000
8	Ted Lasso	3	1	8.5	1	Comedy	Ended	150000
9	Wednesday	1	1	8.2	1	Fantasy	Running	250000
10	The Office	9	1	7.6	1	Comedy	Ended	100000

"This query captures the dynamics of TV show success, from audience reception to production details, making it invaluable for evaluating the small-screen landscape."

4. Director Success Metrics:

This query:

- Assesses the success of directors across movies and TV shows.
- measures their success using metrics like the number of projects they've worked on, box office performance, ratings, and awards.

```
SELECT
    d.first_name || ' ' || d.last_name as director_name,  -- Director's full name
    COUNT(DISTINCT m.movie_id) as movies_directed,        -- Number of movies directed
    COUNT(DISTINCT e.episode_id) as episodes_directed,    -- Number of TV episodes directed
    AVG(m.box_office) as avg_box_office,                 -- Average box office earnings
    SUM(m.box_office) as total_box_office,               -- Total box office earnings
    d.awards_count,                                      -- Number of awards
    AVG(e.rating) as avg_episode_rating                  -- Average TV episode rating
FROM Directors d
LEFT JOIN Movies m ON d.director_id = m.director_id         -- Link to movies
LEFT JOIN Episodes e ON d.director_id = e.director_id       -- Link to episodes
GROUP BY d.director_id, d.first_name, d.last_name, d.awards_count
HAVING COUNT(DISTINCT m.movie_id) > 0 OR COUNT(DISTINCT e.episode_id) > 0;  -- Only active
directors
```

- The LEFT JOIN ensures directors without projects in one category (movies or TV) are still included.
- COUNT(DISTINCT) avoids overcounting in cases where directors worked on multiple episodes of the same show.
- Aggregations like AVG and SUM summarize financial and rating data.
- The HAVING clause filters for directors with at least one movie or TV episode.

Insights:

- Highlights directors excelling in movies, TV shows, or both.
- Tracks financial performance and audience reception.
- Provides benchmarks for measuring director success.

	DIRECTOR_NAME	MOVIES_DIRECTED	EPISODES_DIRECTED	AVG_BOX_OFFICE	TOTAL_BOX_OFFICE	AWARDS_COUNT	AVG_EPISODE_RATING
1	Christopher Nolan	1	1	836800000	836800000	11	8.7
2	Ava DuVernay	1	1	326700000	326700000	8	8.7
3	Martin Scorsese	1	1	179000000	179000000	20	9.1
4	Kathryn Bigelow	1	1	104900000	104900000	9	7.6
5	Bong Joon-ho	1	1	258773645	258773645	15	9
6	Quentin Tarantino	1	1	213928762	213928762	32	8.5
7	Greta Gerwig	1	1	218844555	218844555	12	8.9
8	Denis Villeneuve	1	1	401847917	401847917	15	8.8
9	Chloé Zhao	1	1	39601466	39601466	10	9.3
10	Jordan Peele	1	1	255407969	255407969	8	8.2

5. Content Review Analysis:

This query:

- Aggregates reviews for movies and TV shows.
- audience feedback to identify critically acclaimed works and provides insights into rating distributions.
- Calculates the average, highest, and lowest ratings for each piece of content.

```
SELECT
    CASE
        WHEN r.content_type = 'Movie' THEN m.title  -- Show movie title if it's a movie
        ELSE t.title                                -- Show TV show title if it's a show
    END as content_title,
    r.content_type,                                 -- Movie or TV Show
    AVG(r.rating) as avg_rating,                    -- Average rating
    COUNT(r.review_id) as review_count,             -- Number of reviews
    MIN(r.rating) as lowest_rating,                 -- Lowest rating received
    MAX(r.rating) as highest_rating,                -- Highest rating received
    g.genre_name                                    -- Genre of content
FROM Reviews r
LEFT JOIN Movies m ON r.content_id = m.movie_id AND r.content_type = 'Movie'        -- Link to movies
LEFT JOIN TV_Shows t ON r.content_id = t.show_id AND r.content_type = 'TV Show'     -- Link to TV shows
LEFT JOIN Genres g ON (m.genre_id = g.genre_id OR t.genre_id = g.genre_id)         -- Link to genres
GROUP BY
    CASE
```

```
        WHEN r.content_type = 'Movie' THEN m.title
        ELSE t.title
    END,
    r.content_type,
    g.genre_name
ORDER BY avg_rating DESC;
```

- The CASE statement handles movies and TV shows separately, dynamically selecting the title based on content_type.
- LEFT JOIN ensures all reviews are included, even if linked data is missing.
- Aggregations like AVG, MIN, and MAX summarize audience feedback.
- Sorting by avg_rating ranks content by audience acclaim.

Insights:

- Identifies the most critically acclaimed movies and TV shows.
- Provides a breakdown of ratings and genres.
- Assists in highlighting top-rated content for marketing or investment decisions.

	CONTENT_TITLE	CONTENT_TYPE	AVG_RATING	REVIEW_COUNT	LOWEST_RATING	HIGHEST_RATING	GENRE_NAME
1	Parasite	Movie	9.5	1	9.5	9.5	Drama
2	Pulp Fiction	Movie	9.4	1	9.4	9.4	Drama
3	Game of Thrones	TV Show	9.3	1	9.3	9.3	Fantasy
4	The Last of Us	TV Show	9.2	1	9.2	9.2	Drama
5	Get Out	Movie	9	1	9	9	Science Fiction
6	Inception	Movie	9	1	9	9	Science Fiction
7	The Mandalorian	TV Show	8.9	1	8.9	8.9	Fantasy
8	Dune	Movie	8.8	1	8.8	8.8	Science Fiction
9	Stranger Things	TV Show	8.5	1	8.5	8.5	Science Fiction
10	The Post	Movie	8	1	8	8	Drama

ADDITIONAL DETAILS

Case Example: Frame a hypothetical scenario, ex) A production studio could use this database to decide whether to invest in a sci-fi series with a high-budget director or a low-budget comedy.
Industry Context: Relate your findings to real-world trends, ex) the rise of streaming services or genre shifts post-pandemic.
Future Expansion: Mention how the database could evolve to include streaming metrics, global audience data, or real-time social media feedback. ex) building a database for a site like Letterboxd