

# Database Management Systems

## Group Project – Module 1

### Group 6

Team member 1:  
Team member 2:  
Team member 3:  
Team member 4: Anshita Thakkar

### **About the Dataset:**

#### ***Introduction:***

In the dynamic and ever-evolving world of the film industry, understanding the factors that contribute to a movie's success is crucial for producers and stakeholders especially after covid and the recent strike in Hollywood. One important facet of a movie's performance is its budget and how it corresponds with audience reaction as evidenced in IMDb ratings. The allocation of financial resources can have a considerable impact on a film's production values, star cast, and overall quality, all of which influence viewer opinions and ratings.

To explore deeper into this link, we have taken the design and implementation of a relational database that captures and organizes information about movies, their budgets, IMDb ratings, and genres. This database tries to provide useful insights into the relationship between budget and IMDb ratings across various movie genres. By analyzing this data, producers can make informed decisions about budget allocations, potentially optimizing their investments and improving the overall quality and success of their productions.

### **The Dataset:**

The "Movie Database: Ratings, Budgets, and Box Office Earnings" is a robust SQL dataset that provides a wealth of information for movie enthusiasts, analysts, and researchers. This dataset includes essential details such as movie names, rating timelines, unique movie IDs, production budgets, viewer votes, movie types, and worldwide as well as domestic box office earnings.

#### ***Key Features:***

- **Rating Timeline:** Explore the ratings of movies over time, allowing us to analyse how audience perceptions change.
- **Budget and Box Office Data:** Gain insights into the financial aspects of the movies, including production budgets and their box office performance, both domestically and globally.
- **Movie Classification:** Understand the genres or types of movies available in the dataset, enabling us to segment and analyse different film categories.
- **Comprehensive Information:** With movie IDs, you can easily link and query data, ensuring comprehensive and accurate analysis.

This dataset is invaluable for a wide range of applications, including movie industry analysis, trend forecasting, and understanding the financial dynamics of the film industry. Whether you're a data scientist, movie critic, or just a movie enthusiast, this dataset provides a treasure trove of information for your research and exploration.

Movie Genres

**Dataset obtained from Kaggle:**

<https://www.kaggle.com/datasets/shahjhanalam/movie-data-analytics-dataset>

**Database problem:**

Producers in the film industry are constantly challenged with optimizing budget allocations to increase the success of a picture. The IMDb rating of a film, which measures audience pleasure, is an important part of its performance.

Producers want to know if there is a link between a movie's budget and its IMDb rating, taking into account the influence of different genres. A relational database will help record information about movies, their budgets, IMDb ratings, and genres to make decisions.

**Opportunities:**

1. Budget Optimization Across Genres
2. Genre-specific Audience Preference
3. Risk Mitigation in Budgeting
4. Strategic Investment Planning
5. Marketing and Promotion
6. Feedback Loop for Improvement
7. Customized content Strategy

**Database Design:**

**1. Entities and Relationships:**

**1. Movie Entity:**

- Fields: `MovieID` (Primary Key), `Title`, `ReleaseDate`, `Budget`, `IMDbRating`, `GenreID` (Foreign Key)
- Relationship: One movie can belong to one genre, but one genre can have multiple movies (One-to-Many).

**2. Genre1 Entity:**

- Fields: `GenreID` (Primary Key), `GenreName`
- Relationship: One genre can have multiple movies, but each movie belongs to only one genre (Many-to-One).

**2. Fields and Data Types:**

**1. Movie Table:**

- `MovieID`: Integer (Primary Key)
- `Title`: VARCHAR(255)
- `ReleaseDate`: Date

- `Budget`: Decimal or Integer (based on currency and precision needed)
- `IMDbRating`: Decimal (e.g., 7.5)
- `GenreID`: Integer (Foreign Key referencing Genre.GenreID)

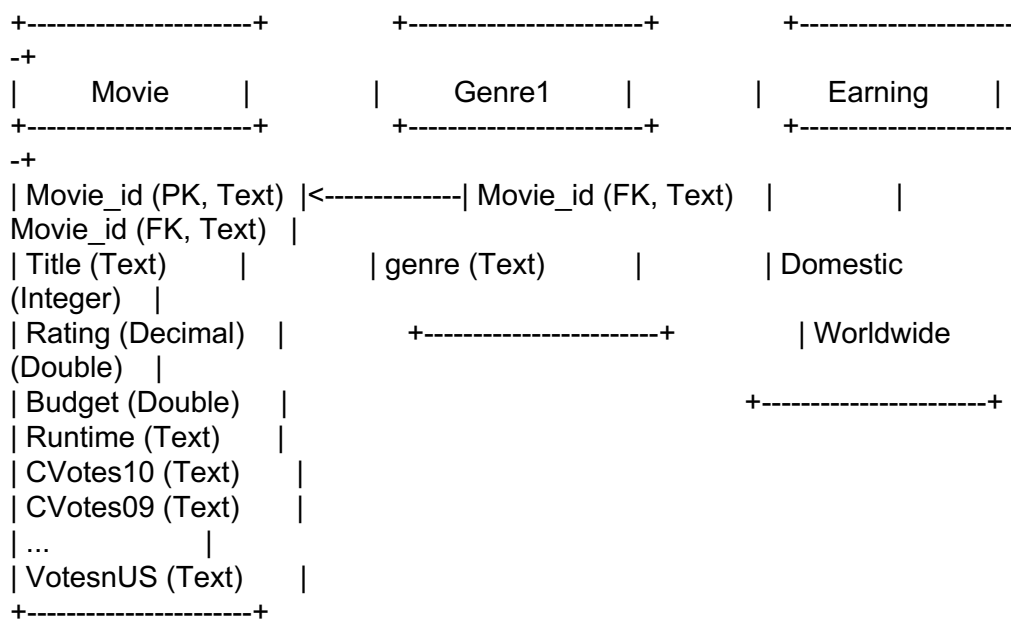
## 2. Genre1 Table:

- `GenreID`: Integer (Primary Key)
- `GenreName`: VARCHAR(50)

## 3. Table: Earning

- Fields:
  - Movie\_id (Foreign Key referencing Movie, Text)
  - Domestic (Integer)
  - Worldwide (Double)
- Foreign Key: \*\* Movie\_id references Movie (Text)

## 3. Entity relationship Diagram (ERD):



### Relational Integrity Constraints:

- The `Movie.GenreID` field is a foreign key referencing `Genre.GenreID`, ensuring that each movie is associated with a valid genre.
- Primary keys are unique identifiers for each record in their respective tables.
- The relationship between the Movie and Genre tables ensures referential integrity.

## Data Summarization:

**Project Choice:**

The chosen project is to create a relational database for analysing the correlation between the budget of movies and their IMDb ratings, taking genres into account.

IMDb ratings are crucial for audiences and filmmakers, serving as a trusted benchmark for a movie's quality and popularity. These ratings significantly influence viewers' movie choices and are often used in film marketing. Considering the importance of IMDb ratings, the information given in the dataset can be highly beneficial for movie experts and producers.

**Important Highlights:**

- The Movie table stores information about each movie, including its title, release date, budget, IMDb rating, and genre.
- The Genre table contains unique genres, allowing for consistent categorization.
- The relationship between the Movie and Genre tables enables querying movies based on genre and facilitates correlation analysis.

**Overview:**

This project involves creating a relational database to analyse the correlation between movie budgets, IMDb ratings, and genres. The Movie and Genre tables are designed with appropriate fields, data types, and relational integrity constraints to support this analysis. The database structure allows for efficient querying and exploration of the correlation between budget and IMDb rating, considering different genres.

**Key Components:**

1. Movie Entity: Captures details about each movie, including title, release date, budget, IMDb rating, and genre and it serves as the database's main hub and contains data on every single movie. It has the following attributes:

- Movie ID: A unique identifier for each movie.
- Title: The title of the movie.
- Release Date: The date the movie was released.
- Budget: The production budget of the movie.
- IMDb Rating: The average rating of the movie on IMDb.
- Genre: The genre(s) of the movie.

2. Genre Entity: Defines unique genres that movies can belong to. It includes the following attributes:

- Genre ID: A unique identifier for each genre.
- Genre Name: The name of the genre.

3. Relationships: One-to-Many relationship between Movie and Genre tables, allowing for genre-based analysis. This means that a movie can belong to multiple genres, but a genre can only be associated with one or more movies. This relationship allows for genre-based analysis, such as identifying the most popular genres or the average rating of movies within a particular genre.

**Objective:**

The primary goal is to enable producers to explore the correlation between movie budgets and IMDb ratings by considering different genres. The relational database provides a structured and efficient platform for this analysis.

It is important to know how the rating of moves varies based on genre and budget combination, especially for the beginner movie producers as it can help them make better decisions of which direction to choose to enhance their career and what results, they can achieve within certain investment options.

**Next Steps:**

1. Populate the database with relevant movie data.
2. Implement queries and analyses to explore the correlation between budget and IMDb rating within specific genres.
3. Continuously update and refine the database as new movie data becomes available.

This relational database offers a foundation for insightful analyses that can guide producers in decision-making processes related to budget allocation and genre-specific IMDb ratings.

## Module 2:

### 1. Populating the above table:

First, we created the Genre table.

#### Create Table Genre1:

```
CREATE TABLE Genre1(  
    GenreID INT PRIMARY KEY,  
    GenreName VARCHAR(255) NOT NULL  
);
```

```
12 • ○ CREATE TABLE Genre1 (  
13     GenreID INT PRIMARY KEY,  
14     GenreName VARCHAR(50)  
15 );
```

#### Genre1 table:

```
INSERT INTO Genre1 (GenreID, GenreName)
```

```
VALUES
```

```
(1, 'Action'),  
(2, 'Drama'),  
(3, 'Comedy'),  
(4, 'Sci-Fi'),  
(5, 'Thriller'),  
(6, 'Romance'),  
(7, 'Adventure'),  
(8, 'Fantasy'),  
(9, 'Horror'),  
(10, 'Mystery'),  
(11, 'Animation'),  
(12, 'Crime'),  
(13, 'Family'),  
(14, 'History'),  
(15, 'Documentary');
```

GenreID	GenreName
1	Action
2	Drama
3	Comedy
4	Sci-Fi
5	Thriller
6	Romance
7	Adventure
8	Fantasy
9	Horror
10	Mystery
11	Animation
12	Crime
13	Family
14	History
15	Document...
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	
31	
32	
33	
34	
35	
36	
37	
38	
39	
40	
41	
42	
43	
44	
45	
46	
47	
48	
49	
50	
51	
52	
53	
54	
55	
56	
57	
58	
59	
60	
61	
62	
63	
64	
65	
66	
67	
68	
69	
70	
71	
72	
73	
74	
75	
76	
77	
78	
79	
80	
81	
82	
83	
84	
85	
86	
87	
88	
89	
90	
91	
92	
93	
94	
95	
96	
97	
98	
99	
100	

Time	Action	Response
49 19:16:50	SELECT * FROM Earning LIMIT 0, 1000	40 row(s) returned
50 19:19:17	SELECT * FROM Genre1 LIMIT 0, 1000	15 row(s) returned

Then, we created the movie table referencing GenreID as the foreign key.

### Movies Table:

```
CREATE TABLE Movie (
    MovieID INT PRIMARY KEY,
    Title VARCHAR(255),
    ReleaseDate DATE,
    Budget DECIMAL(15, 2),
    IMDbRating DECIMAL(3, 1),
    GenreID INT,
    FOREIGN KEY (GenreID) REFERENCES Genre1(GenreID)
);
```

```
1 • CREATE TABLE Movie (
2     MovieID INT PRIMARY KEY,
3     Title VARCHAR(255),
4     ReleaseDate DATE,
5     Budget DECIMAL(15, 2),
6     IMDbRating DECIMAL(3, 1),
7     GenreID INT,
8     FOREIGN KEY (GenreID) REFERENCES Genre1(GenreID)
9 );
```

```
INSERT INTO Movie (MovieID, Title, ReleaseDate, Budget, IMDbRating, GenreID)
VALUES
```

```
(1, 'Inception', '2010-07-16', 160000000, 8.8, 1),
(2, 'The Shawshank Redemption', '1994-09-23', 25000000, 9.3, 2),
(3, 'The Dark Knight', '2008-07-18', 185000000, 9.0, 1),
(4, 'Pulp Fiction', '1994-10-14', 8000000, 8.9, 3),
(5, 'Forrest Gump', '1994-07-06', 55000000, 8.8, 2),
(6, 'The Matrix', '1999-03-31', 63000000, 8.7, 4),
(7, 'The Godfather', '1972-03-24', 6000000, 9.2, 3),
(8, 'The Shawshank Redemption', '1994-09-23', 25000000, 9.3, 2),
(9, 'The Dark Knight Rises', '2012-07-20', 250000000, 8.4, 1),
(10, 'The Lord of the Rings: The Fellowship of the Ring', '2001-12-19', 93000000, 8.8, 7),
(11, 'Inglourious Basterds', '2009-08-21', 75000000, 8.3, 5),
(12, 'The Departed', '2006-10-06', 90000000, 8.5, 10),
(13, 'The Silence of the Lambs', '1991-02-14', 19000000, 8.6, 5),
(14, 'Fight Club', '1999-10-15', 63000000, 8.8, 5),
(15, 'The Godfather: Part II', '1974-12-20', 13000000, 9.0, 3),
(16, 'The Lord of the Rings: The Two Towers', '2002-12-18', 94000000, 8.7, 7),
(17, 'The Matrix Reloaded', '2003-05-15', 150000000, 7.2, 4),
(18, 'The Matrix Revolutions', '2003-11-05', 150000000, 6.8, 4),
(19, 'The Lord of the Rings: The Return of the King', '2003-12-17', 94000000, 8.9, 7),
(20, 'The Social Network', '2010-10-01', 40000000, 7.7, 6),
(21, 'The Grand Budapest Hotel', '2014-03-28', 25000000, 8.1, 6),
(22, 'The Shining', '1980-05-23', 19000000, 8.4, 9),
(23, 'Blade Runner', '1982-06-25', 28000000, 8.1, 4),
(24, 'The Big Lebowski', '1998-03-06', 15000000, 8.1, 3),
(25, 'Interstellar', '2014-11-07', 165000000, 8.6, 1),
(26, 'The Pianist', '2002-09-24', 35000000, 8.5, 2),
(27, 'The Green Mile', '1999-12-10', 60000000, 8.6, 2),
(28, 'Schindler's List', '1993-11-30', 22000000, 8.9, 3),
(29, 'The Usual Suspects', '1995-09-15', 6000000, 8.5, 10),
(30, 'The Great Gatsby', '2013-05-10', 105000000, 7.2, 6),
(31, 'The Dark Knight Rises', '2012-07-20', 250000000, 8.4, 1),
(32, 'Django Unchained', '2012-12-25', 100000000, 8.4, 5),
(33, 'The Godfather: Part III', '1990-12-25', 54000000, 7.6, 3),
(34, 'The Wolf of Wall Street', '2013-12-25', 100000000, 8.2, 6),
(35, 'The Martian', '2015-10-02', 108000000, 8.0, 4),
(36, 'American Beauty', '1999-10-01', 15000000, 8.3, 3),
(37, 'Amélie', '2001-04-25', 10000000, 8.3, 6),
(38, 'City of God', '2002-02-08', 3300000, 8.6, 2),
(39, 'The Sixth Sense', '1999-08-06', 40000000, 8.1, 5),
(40, 'Life Is Beautiful', '1997-12-20', 20000000, 8.6, 13);
```



MovieID	Title	ReleaseDate	Budget	IMDbRating	GenreID
3	The Dark Knight	2008-07-18	185000000.00	9.0	1
4	Pulp Fiction	1994-10-14	8000000.00	8.9	3
5	Forrest Gump	1994-07-06	55000000.00	8.8	2
6	The Matrix	1999-03-31	63000000.00	8.7	4
7	The Godfather	1972-03-24	6000000.00	9.2	3
8	The Shawshank Redemption	1994-09-23	25000000.00	8.3	2
9	The Dark Knight Rises	2012-07-20	250000000.00	8.4	1
10	The Lord of the Rings: The Fellowship of the Ring	2001-12-19	93000000.00	8.8	7
11	Inglourious Basterds	2009-08-21	75000000.00	8.3	5
12	The Departed	2006-10-06	90000000.00	8.5	10
13	The Silence of the Lambs	1991-02-14	19000000.00	8.6	5
14	Fight Club	1999-10-15	63000000.00	8.8	5
15	The Godfather: Part II	1974-12-20	13000000.00	9.0	3
16	The Lord of the Rings: The Two Towers	2002-12-18	94000000.00	8.7	7
17	The Matrix Reloaded	2003-05-15	150000000.00	7.2	4
18	The Matrix Revolutions	2003-11-05	150000000.00	6.8	4
19	The Lord of the Rings: The Return of the King	2003-12-17	94000000.00	8.9	7
20	The Social Network	2010-10-01	40000000.00	7.7	6
21	The Grand Budapest Hotel	2014-03-28	25000000.00	8.1	6
22	The Shining	1980-05-23	19000000.00	8.4	9
23	Blade Runner	1982-06-25	28000000.00	8.1	4
24	The Big Lebowski	1998-03-06	15000000.00	8.1	3
25	Interstellar	2014-11-07	165000000.00	8.6	1
26	The Pianist	2002-09-24	35000000.00	8.5	2
27	The Green Mile	1999-12-10	60000000.00	8.6	2
28	Schindler's List	1993-11-30	22000000.00	8.9	3
29	The Usual Suspects	1995-09-15	6000000.00	8.5	10
30	The Great Gatsby	2013-05-10	105000000.00	7.2	6
31	The Dark Knight Rises	2012-07-20	250000000.00	8.4	1
32	Django Unchained	2012-12-25	100000000.00	8.4	5
33	The Godfather: Part III	1990-12-25	54000000.00	7.6	3
34	The Wolf of Wall Street	2013-12-25	100000000.00	8.2	6
35	The Martian	2015-10-02	108000000.00	8.0	4
36	American Beauty	1999-10-01	15000000.00	8.3	3
37	Amélie	2001-04-25	10000000.00	8.3	6
38	City of God	2002-02-08	3300000.00	8.6	2
39	The Sixth Sense	1999-08-06	40000000.00	8.1	5
40	Life is Beautiful	1997-12-20	20000000.00	8.6	13

Finally, we created the Earning table, referencing Movie\_id as the FOREIGN KEY.

### Earning Table:

```
CREATE TABLE Earning (
    Movie_id INT PRIMARY KEY,
    Domestic BIGINT,
    Worldwide DECIMAL(15, 2),
    FOREIGN KEY (Movie_id) REFERENCES Movie (MovieID)
);
```

```
18 • CREATE TABLE Earning (
19     Movie_id INT PRIMARY KEY,
20     Domestic BIGINT,
21     Worldwide DECIMAL(15, 2),
22     FOREIGN KEY (Movie_id) REFERENCES Movie (MovieID)
23 );
```

```
INSERT INTO Earning (Movie_id, Domestic, Worldwide)
VALUES
('1', 292576195, 829895144),
('2', 28341469, 589439162),
('3', 534858444, 1004558444),
('4', 107928762, 214179088),
('5', 330252182, 786635413),
('6', 171479930, 463517383),
('7', 292576195, 829895144),
('8', 200000000, 750000000),
('9', 120000000, 500000000),
('10', 180000000, 800000000),
```

('11', 160000000, 700000000),  
('12', 100000000, 450000000),  
('13', 130000000, 600000000),  
('14', 90000000, 400000000),  
('15', 110000000, 550000000),  
('16', 170000000, 700000000),  
('17', 200000000, 850000000),  
('18', 130000000, 600000000),  
('19', 160000000, 700000000),  
('20', 140000000, 650000000),  
('21', 80000000, 300000000),  
('22', 120000000, 500000000),  
('23', 100000000, 400000000),  
('24', 70000000, 250000000),  
('25', 190000000, 800000000),  
('26', 80000000, 300000000),  
('27', 120000000, 500000000),  
('28', 110000000, 450000000),  
('29', 95000000, 350000000),  
('30', 160000000, 600000000),  
('31', 180000000, 800000000),  
('32', 200000000, 850000000),  
('33', 110000000, 450000000),  
('34', 150000000, 650000000),  
('35', 210000000, 900000000),  
('36', 70000000, 300000000),  
('37', 90000000, 400000000),  
('38', 50000000, 200000000),  
('39', 120000000, 500000000),  
('40', 100000000, 400000000);

Result Grid			Filter Rows: <input type="text" value="Search"/>	Edit:	Export/Import:
Movie_id	Domestic	Worldwide			
2	28341489	589439182.00			
3	534858444	100458444.00			
4	107928762	214179088.00			
5	330252182	786835413.00			
6	171479830	463517388.00			
7	282576195	829865144.00			
8	200000000	750000000.00			
9	120000000	500000000.00			
10	180000000	800000000.00			
11	160000000	700000000.00			
12	100000000	450000000.00			
13	130000000	600000000.00			
14	90000000	400000000.00			
15	110000000	550000000.00			
16	170000000	700000000.00			
17	200000000	850000000.00			
18	130000000	600000000.00			
19	160000000	700000000.00			
20	140000000	650000000.00			
21	80000000	300000000.00			
22	120000000	500000000.00			
23	100000000	400000000.00			
24	70000000	250000000.00			
25	190000000	800000000.00			
26	80000000	300000000.00			
27	120000000	500000000.00			
28	110000000	450000000.00			
29	95000000	350000000.00			
30	160000000	600000000.00			
31	180000000	800000000.00			
32	200000000	850000000.00			
33	110000000	450000000.00			
34	150000000	650000000.00			
35	210000000	900000000.00			
36	70000000	250000000.00			
37	80000000	300000000.00			
38	50000000	200000000.00			
39	120000000	500000000.00			
40	100000000	400000000.00			

Earning 8			Apply	Revert
Action	Time	Response		
50	19:19:17	SELECT * FROM Genre1 LIMIT 0, 1000	15 row(s) returned	
51	19:21:06	SELECT * FROM Earning LIMIT 0, 1000	40 row(s) returned	

## 2. Data Normalization:

None of the tables need to be normalized, please see the explanation below:

### Movie Table

1NF: Achieved as all fields contain atomic values.

2NF: Since MovieID is a simple primary key, all attributes are fully functionally dependent on it.

3NF: There are no transitive dependencies; all non-key attributes depend only on the primary key.

### Genre1 Table

1NF: Contains atomic values in each field.

2NF: Automatically achieved as there is a single-column primary key.

3NF: No transitive dependencies; GenreName depends only on GenreID.

### Earning Table

1NF: Each field has atomic values.

2NF: With Movie\_id as the primary key, other attributes depend on the whole key.

3NF: No transitive dependencies present.

### 3. Queries:

#### Query 1:

#### Top 10 IMDB Rating Movies with Genres

This query gives viewers a visual of the genres of the top 10 rates movies by IMDB. The query pulls data from the Movie and Genre 1 table.

```
SELECT
    M.MovieID,
    M.Title,
    M.IMDbRating,
    G1.GenreName
FROM
    Movie M
    JOIN Genre1 G1 ON M.GenreID = G1.GenreID
ORDER BY
    M.IMDbRating DESC
LIMIT 10;
```

261 • SELECT  
262 M.MovieID,  
263 M.Title,  
264 M.IMDbRating,  
265 G1.GenreName  
266 FROM  
267 Movie M  
268 JOIN Genre1 G1 ON M.GenreID = G1.GenreID  
269 ORDER BY  
270 M.IMDbRating DESC  
271 LIMIT 10;  
272

100% 10:271

Result Grid

MovieID	Title	IMDbRating	GenreName
2	The Shawshank Redemption	9.3	Drama
8	The Shawshank Redemption	9.3	Drama
7	The Godfather	9.2	Comedy
3	The Dark Knight	9.0	Action
15	The Godfather: Part II	9.0	Comedy
4	Pulp Fiction	8.9	Comedy
28	Schindler's List	8.9	Comedy
19	The Lord of the Rings: The Return of the King	8.9	Adventure
1	Inception	8.8	Action
5	Forrest Gump	8.8	Drama

Result 22 Read Only

Action Output

	Time	Action	Response
67	20:01:39	SELECT M.MovieID, M.Title, M.ReleaseDate, M.Budget, M.IMDbRating, G1.GenreName FROM Mov...	40 row(s) returned
68	20:02:28	SELECT M.MovieID, M.Title, M.IMDbRating, G1.GenreName FROM Movie M JOIN Genre1 G1 ON M.Ge...	10 row(s) returned

## Query 2:

### Average IMDb Rating and Budget by Genre:

This query is useful because it shows the average IMDB rating and average budget by genre, proving useful to directors when they decide which movie to make that will be popular and maximize their profits.

```
SELECT
    G1.GenreName,
    AVG(M.IMDbRating) AS AverageRating,
    AVG(M.Budget) AS AverageBudget
FROM
    Movie M
    JOIN Genre1 G1 ON M.GenreID = G1.GenreID
GROUP BY
    G1.GenreName;
```

The screenshot shows a database query editor with a SQL query and its results. The query is as follows:

```
250 • SELECT
251     G1.GenreName,
252     AVG(M.IMDbRating) AS AverageRating,
253     AVG(M.Budget) AS AverageBudget
254 FROM
255     Movie M
256     JOIN Genre1 G1 ON M.GenreID = G1.GenreID
257 GROUP BY
258     G1.GenreName;
259
```

The result grid displays the following data:

GenreName	AverageRating	AverageBudget
Action	8.64000	202000000.000000
Drama	8.85000	33883333.333333
Comedy	8.57143	19000000.000000
Sci-Fi	7.76000	99800000.000000
Adventure	8.80000	93666666.666667
Thriller	8.44000	59400000.000000
Mystery	8.50000	48000000.000000
Romance	7.90000	56000000.000000
Horror	8.40000	19000000.000000
Family	8.60000	20000000.000000

The bottom section of the screenshot shows the 'Action Output' table, which contains two rows of execution details:

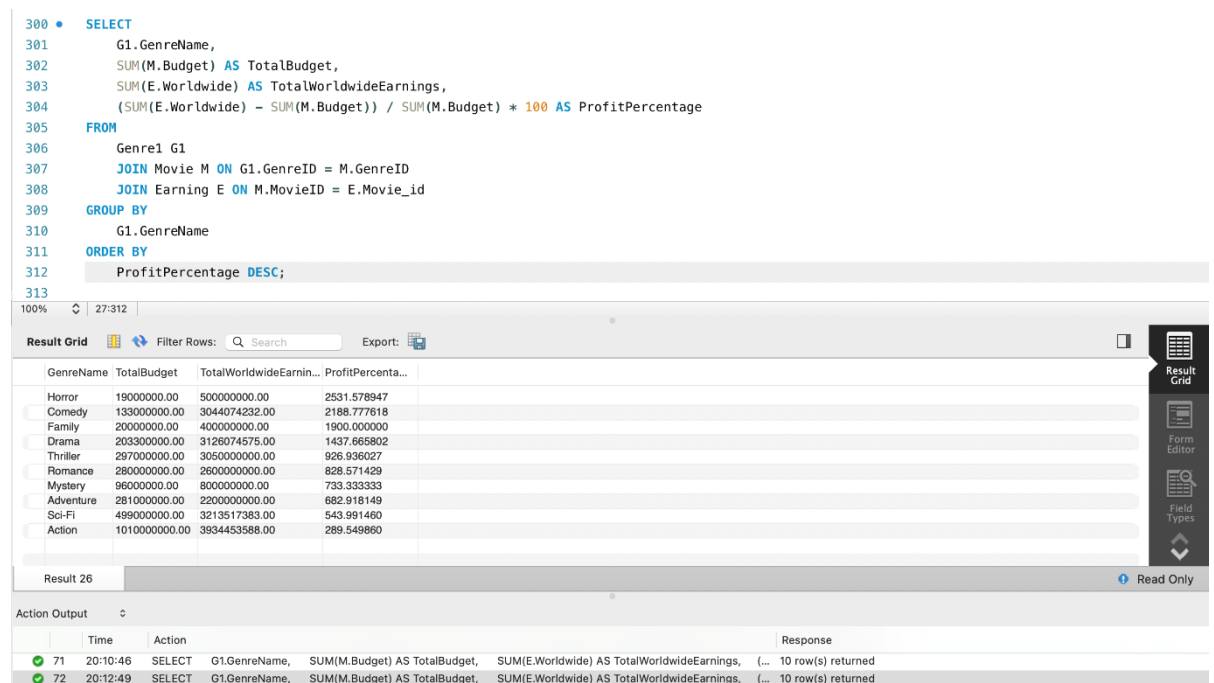
	Time	Action	Response
68	20:02:28	SELECT M.MovieID, M.Title, M.IMDbRating, G1.GenreName FROM Movie M JOIN Genre1 G1 ON M.Ge...	10 row(s) returned
69	20:03:18	SELECT G1.GenreName, AVG(M.IMDbRating) AS AverageRating, AVG(M.Budget) AS AverageBudget FROM...	10 row(s) returned

### Query 3:

#### Highest to lowest profit percentage by Genre:

This query shows us the top 10 genres, the total budget, total revenue and the profit percentage in descending order. This is useful for directors looking to maximize profit as it shows which genre movie is the most lucrative to make. Based on the findings, the Horror genre is the most profitable as it made 2531% profit followed by Comedy and Family at 2188% and 1900% respectively. Action films are the least profitable, making only 289% profit based on this dataset.

```
SELECT
    G1.GenreName,
    SUM(M.Budget) AS TotalBudget,
    SUM(E.Worldwide) AS TotalWorldwideEarnings,
    (SUM(E.Worldwide) - SUM(M.Budget)) / SUM(M.Budget) * 100 AS
ProfitPercentage
FROM
    Genre1 G1
    JOIN Movie M ON G1.GenreID = M.GenreID
    JOIN Earning E ON M.MovieID = E.Movie_id
GROUP BY
    G1.GenreName
ORDER BY
    ProfitPercentage DESC;
```



```
300 • SELECT
301     G1.GenreName,
302     SUM(M.Budget) AS TotalBudget,
303     SUM(E.Worldwide) AS TotalWorldwideEarnings,
304     (SUM(E.Worldwide) - SUM(M.Budget)) / SUM(M.Budget) * 100 AS ProfitPercentage
305 FROM
306     Genre1 G1
307     JOIN Movie M ON G1.GenreID = M.GenreID
308     JOIN Earning E ON M.MovieID = E.Movie_id
309 GROUP BY
310     G1.GenreName
311 ORDER BY
312     ProfitPercentage DESC;
313
```

GenreName	TotalBudget	TotalWorldwideEarnings	ProfitPercentage
Horror	19000000.00	500000000.00	2531.578947
Comedy	133000000.00	3044074232.00	2188.777618
Family	200000000.00	400000000.00	1900.000000
Drama	203300000.00	3126074575.00	1437.665802
Thriller	297000000.00	305000000.00	926.936027
Romance	280000000.00	260000000.00	828.571429
Mystery	96000000.00	80000000.00	733.333333
Adventure	281000000.00	220000000.00	682.918149
Sci-Fi	499000000.00	3213517383.00	543.991460
Action	1010000000.00	3934453588.00	289.549880

Result 26

Action Output

	Time	Action	Response
71	20:10:46	SELECT	G1.GenreName, SUM(M.Budget) AS TotalBudget, SUM(E.Worldwide) AS TotalWorldwideEarnings, (... 10 row(s) returned
72	20:12:49	SELECT	G1.GenreName, SUM(M.Budget) AS TotalBudget, SUM(E.Worldwide) AS TotalWorldwideEarnings, (... 10 row(s) returned