ABV-IIITM Gwalior

DBMS Project

B.Tech – 2nd Semester Academic Year 2020-2021

Project -

MOVIE RECOMMENDATION SYSTEM

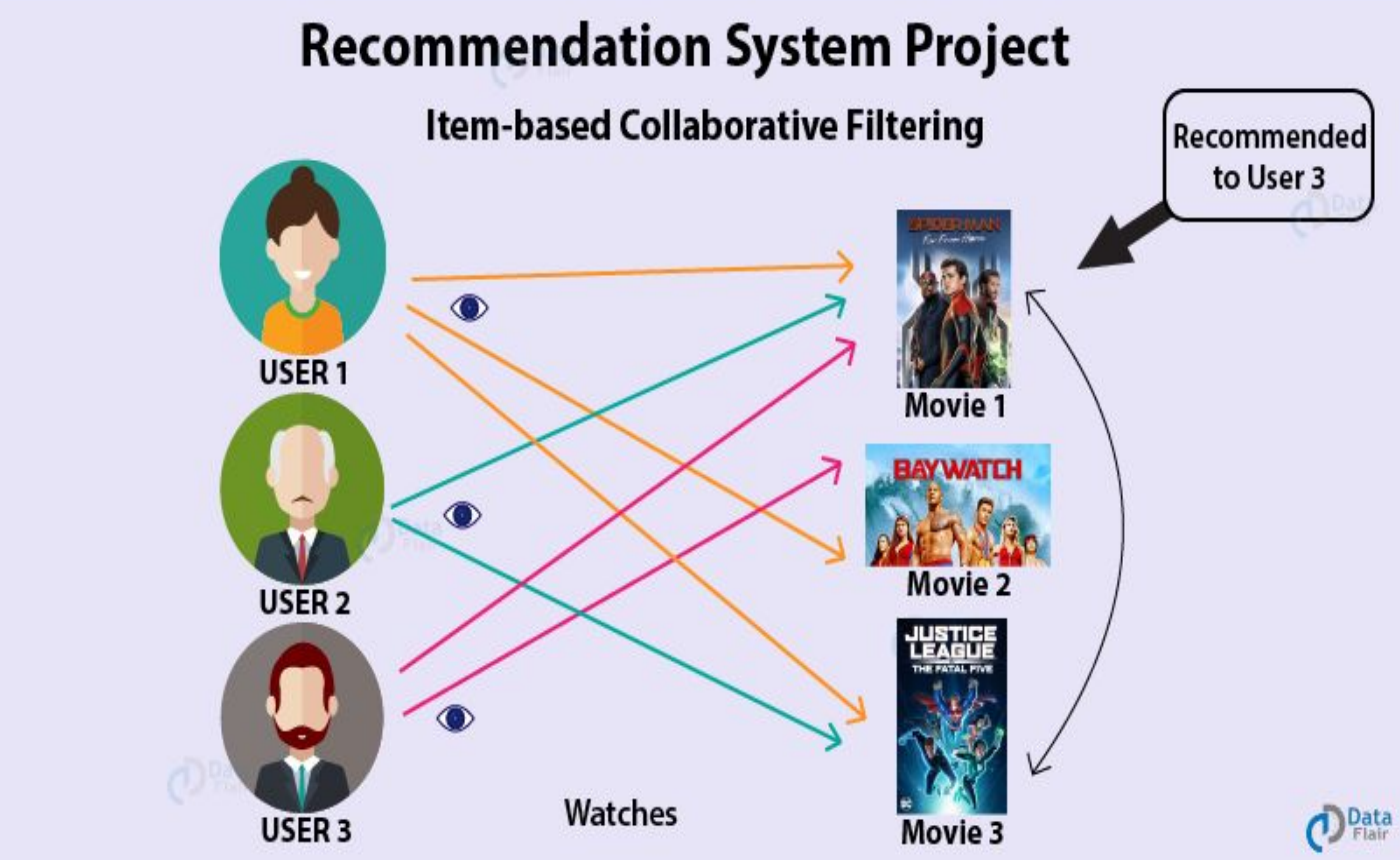
Database Systems by

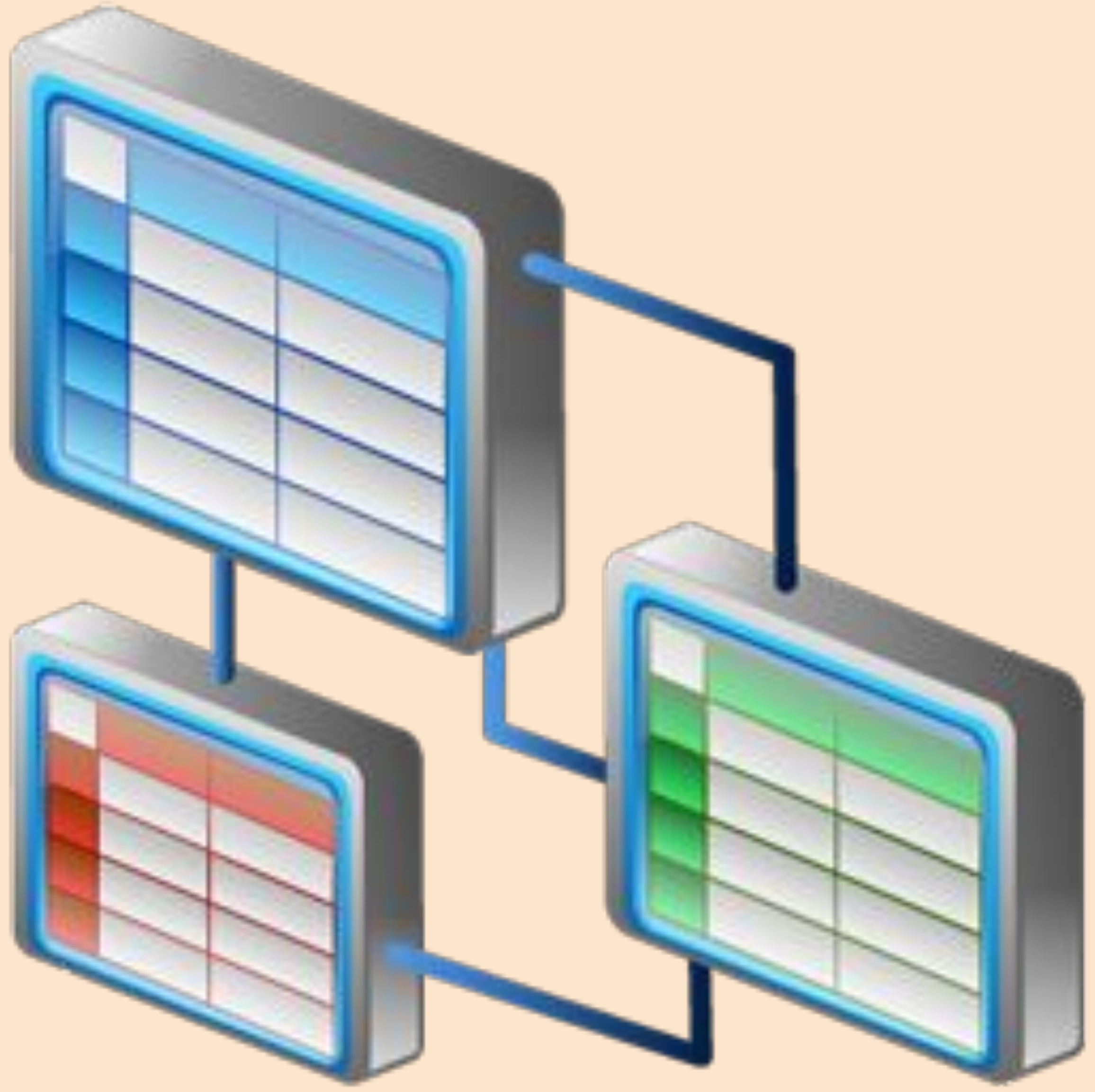
— Dr. Debanjan Sadhya

Submitted By -

Student Roll no.	**Student Name**
2020BCS-013	Anurag Yadav
2020BCS-014	Arun Kumar Rathod
2020BCS-015	Aryan Dadhich
2020BCS-016	Ashish Singh
2020BCS-017	Ashwin Kumar Singh
2020BCS-018	Bachute Sarvesh Vikas

Table of Contents


Recommendation System Project
Item-based Collaborative Filtering
Recommended to User 3
USER 1
USER 2
USER 3
Movie 1
Movie 2
Movie 3
Watches
Data Flair



Project Description

This project is about creating the database about Movie Recommendation System.

The movie recommendation system facilitates the audience to view and research about the movies available on the basis of genre type, language, release year, also they can search according to their favorite actors or directors etc. The aim this project is to design and develop a database maintaining the records of different   movies, specifications etc.. The record of movies includes its release year, name, platform, language, actor directors, genre , ratings , gross market.

Viewers can look for movies with different kind of customised searches. ***For recomendation**, **viewers has to provide the desired industry or genre etc in the form of queries**. Before displaying a movies for a viewer, the availability of movie checked. Similarly, all the search commands are executed and movies are filtered out.. If such combination exist, the list of movies is shown for the viewer to choose..*

Some of assumption and exceptions are also taken since the movie database is very large in reality, it is not feasible to develop the case study to that extent and prepare documentation at that level. Therefore, a small sample database has been created to demonstrate the working of the Movie Recommendation System.

Entities and Attributes

Entities

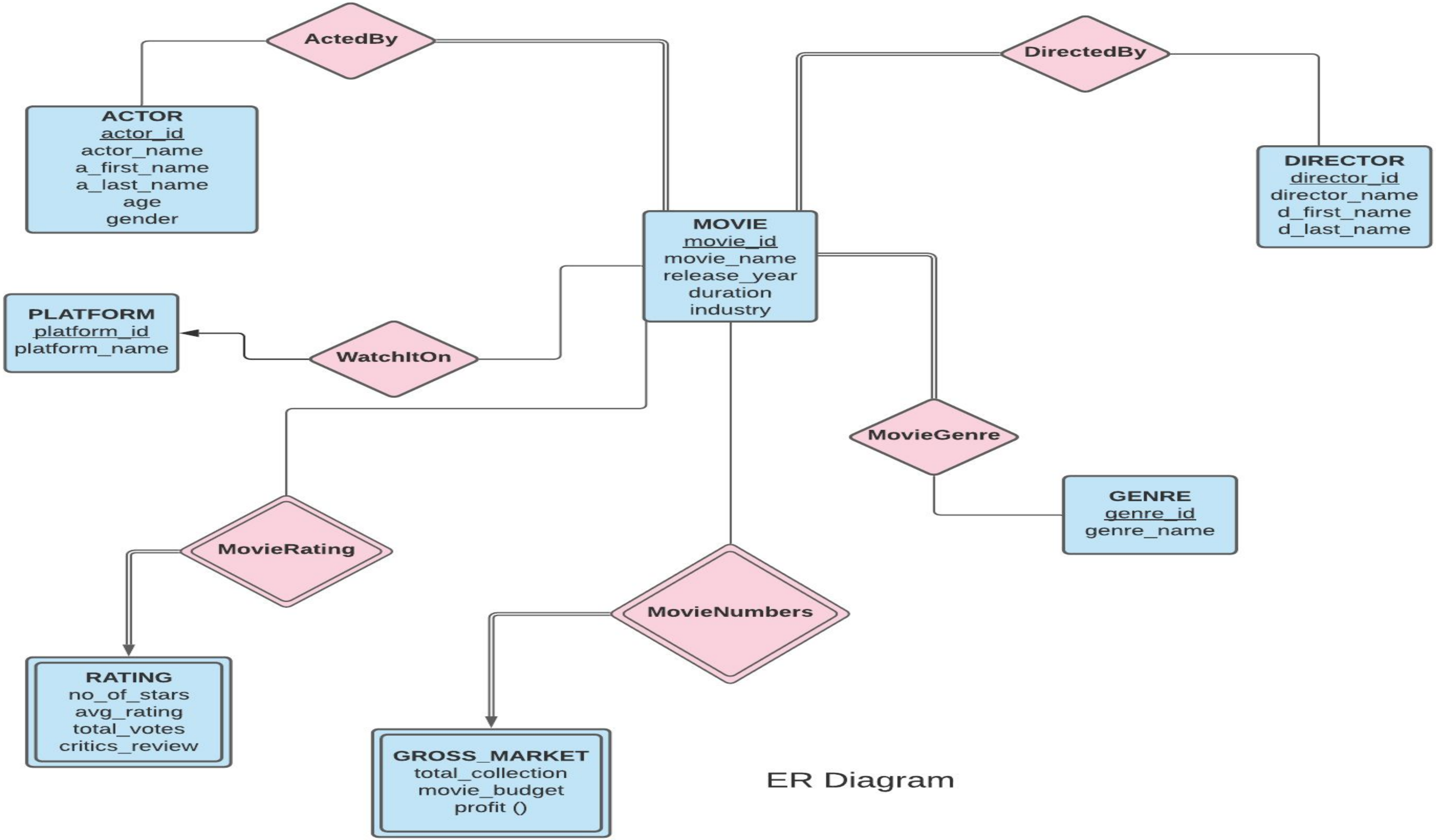
- ❑ Movies
- ❑ Actors
- ❑ Directors
- ❑ Platform
- ❑ Genre
- ❑ Ratings
- ❑ Gross Market

\

Attributes

- movie_id
- movie_name
- release_year
- industry
- duration
- actor_id
- actor_name
- director_id
- director_name
- platform_id
- platform_name
- genre_id
- genre_name
- critics_review
- no_of_stars
- average_rating
- total_votes
- total_collection
- movie_budget
- profit

ER Diagram


ActedBy
DirectedBy
ACTOR
actor_id
actor_name
a_first_name
a_last_name
age
gender
DIRECTOR
director_id
director_name
d_first_name
d_last_name
MOVIE
movie_id
movie_name
release_year
duration
industry
PLATFORM
platform_id
platform_name
WatchItOn
MovieGenre
GENRE
genre_id
genre_name
MovieRating
MovieNumbers
RATING
no_of_stars
avg_rating
total_votes
critics_review
GROSS_MARKET
total_collection
movie_budget
profit ()

ER Diagram

Schema Diagram

MOVIES

movie_id	movie_name	release_year	industry	duration

Primary key

NOTE- as movie names are unique so other things can easily be found by movie name but as primary key is a candidate key which is minimum super key(if one attribute can be identify uniquely so we have to choose either one of them) so we can either choose movie_id or movie_name as primary key and since it is our choice which key to choose, so we have chosen the movie_id

Functional dependencies

Movie_id --> movie_name, release year, industry, duration.
Movie name --> release year, industry, duration, Movie_id

NORMALISED TO BCNF

MOVIES_ONE

movie_id	movie_name

MOVIES_TWO

movie_name	release_ year	industry	duration

Schema Diagram

ACTORS

actor_id	a_first_name	a_last_name	age	gender

Primary key

Functional dependencies

Actor_id -->(a_first_name, a_last_name, age, gender)

NORMALISED TO BCNF

ACTORS

actor_id	a_first_name	a_last_name	age	gender

Schema Diagram

DIRECTORS

director_id	d_first_name	d_last_name

Primary key

Functional dependencies

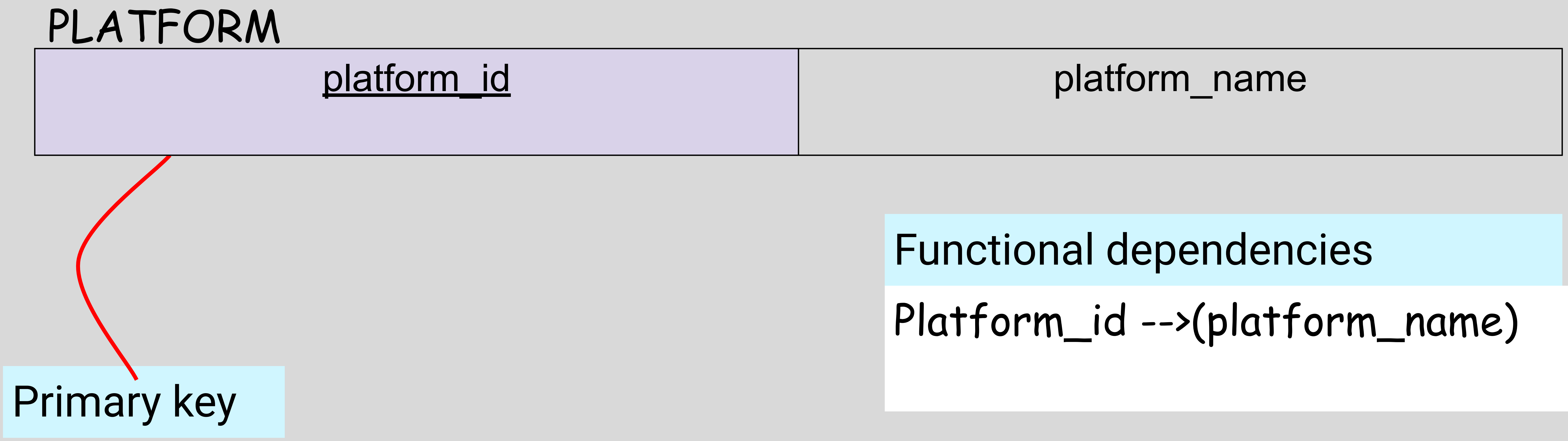
Director_id --> (d_first_name, d_last_name)

NORMALISED TO BCNF

DIRECTORS

director_id	d_first_name	d_last_name

Schema Diagram



PLATFORM

platform_id	platform_name

Primary key

Functional dependencies

Platform_id -->(platform_name)

NORMALISED TO BCNF

PLATFORM

platform_id	platform_name

Schema Diagram

Genre

genre_id	genre_name

Primary key

Functional dependencies

Genre_id -->(genre_name)

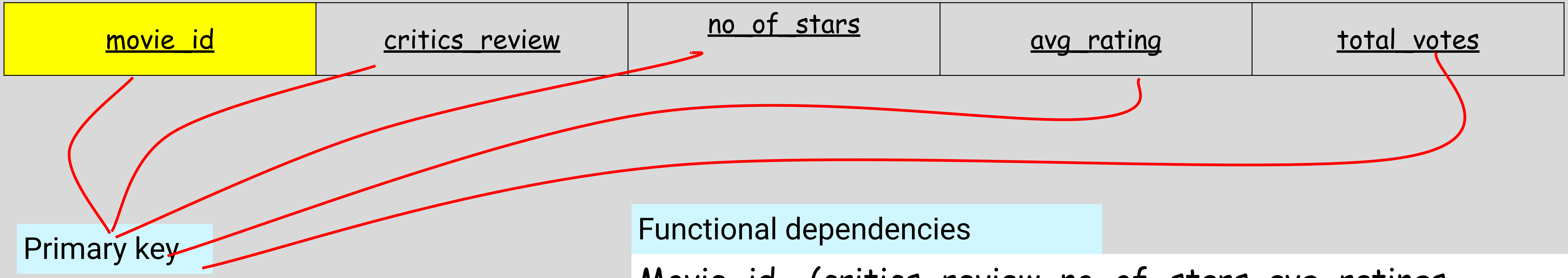
NORMALISED TO BCNF

Genre

genre_id	genre_name

Schema Diagram

RATINGS



movie_id	critics_review	no_of_stars	avg_rating	total_votes

Primary key

Functional dependencies

Movie_id - (critics_review, no_of_stars, avg_ratings, total_votes)

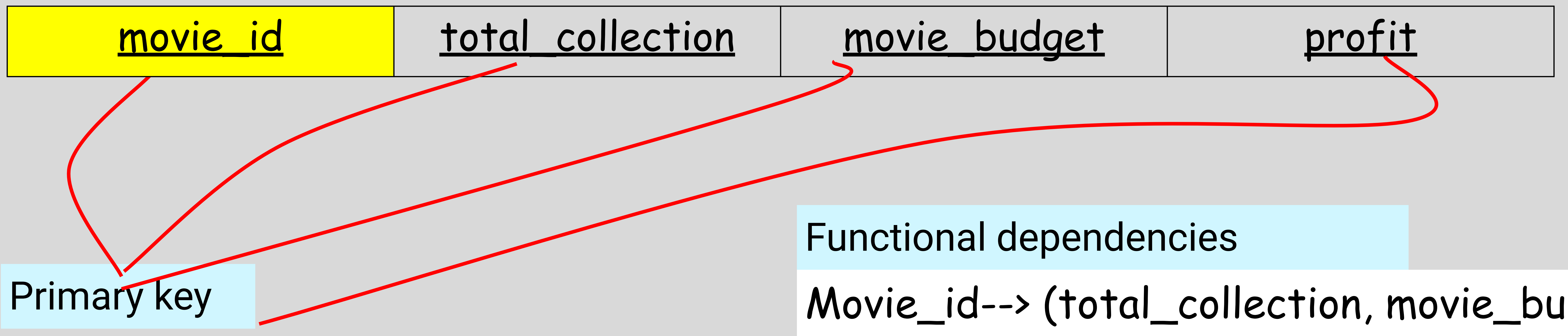
NORMALISED TO BCNF

RATINGS

movie_id	critics_review	no_of_stars	avg_rating	total_votes

Schema Diagram

GROSS_MARKET



movie_id	total_collection	movie_budget	profit

Primary key

Functional dependencies

Movie_id--> (total_collection, movie_budget, profit)

Total_collection, movie_budget → (profit)

NORMALISED TO BCNF

GROSS_MARKET_ONE

movie_id	total_collection	movie_budget

GROSS_MARKET_TWO

total_collection	movie_budget	profit

Schema Diagram

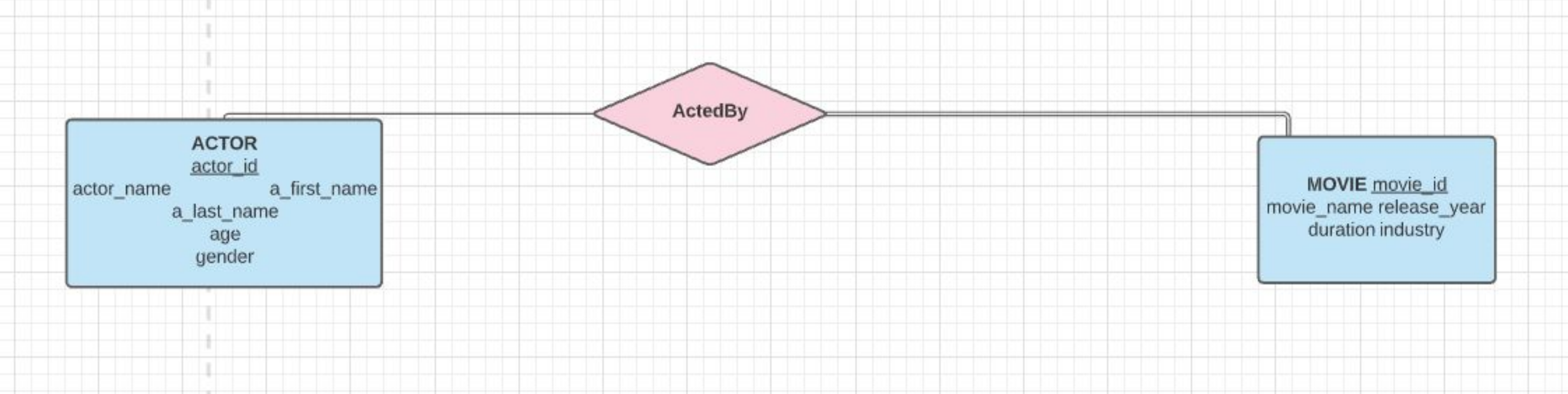
ACTED_BY

movie_id	actor_id

Primary key

Relation description

- denoting relation between movies and actor
- is a many to many relation as movies can have many actors associated also actors can take part in different movies
- there is total participation of movie as each movie should have actors associated with it.



Schema Diagram

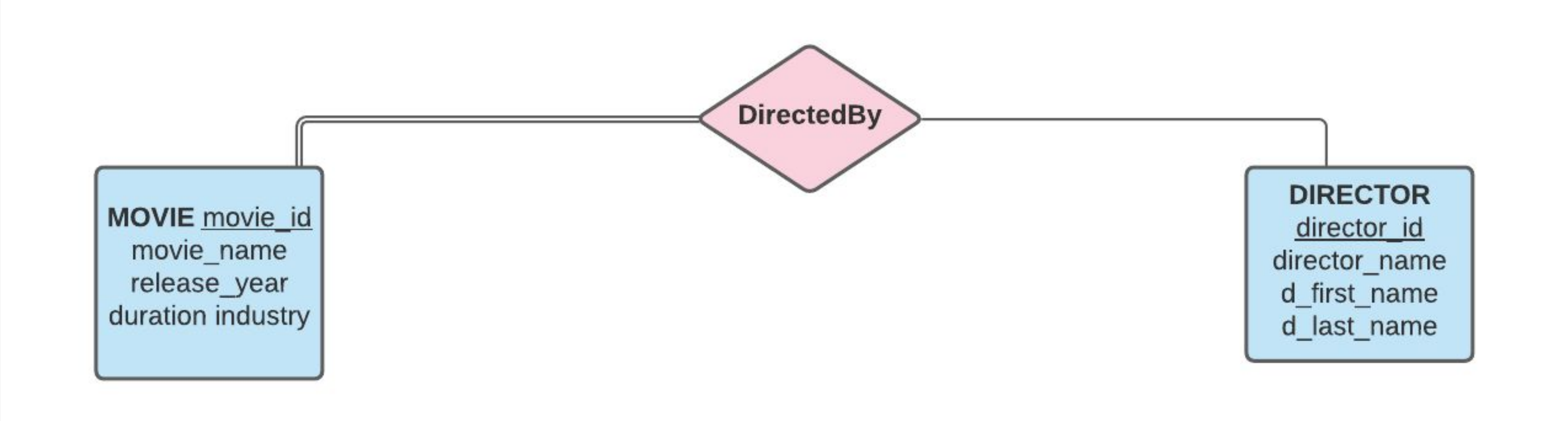
DIRECTED_BY

<u>movie_id</u>	<u>director_id</u>

Primary key

Relation description

- denoting relation between movies and director
- is assumed to be many to one relationship as we consider that each movie can have only one director, also
- there is total participation of movie as each movie should have director associated with it.



Schema Diagram

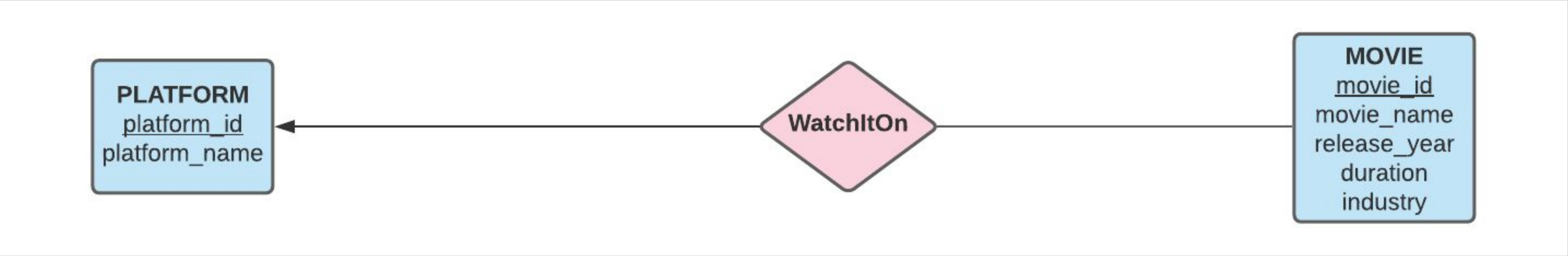
WATCH_IT_ON

platform_id	movie_id

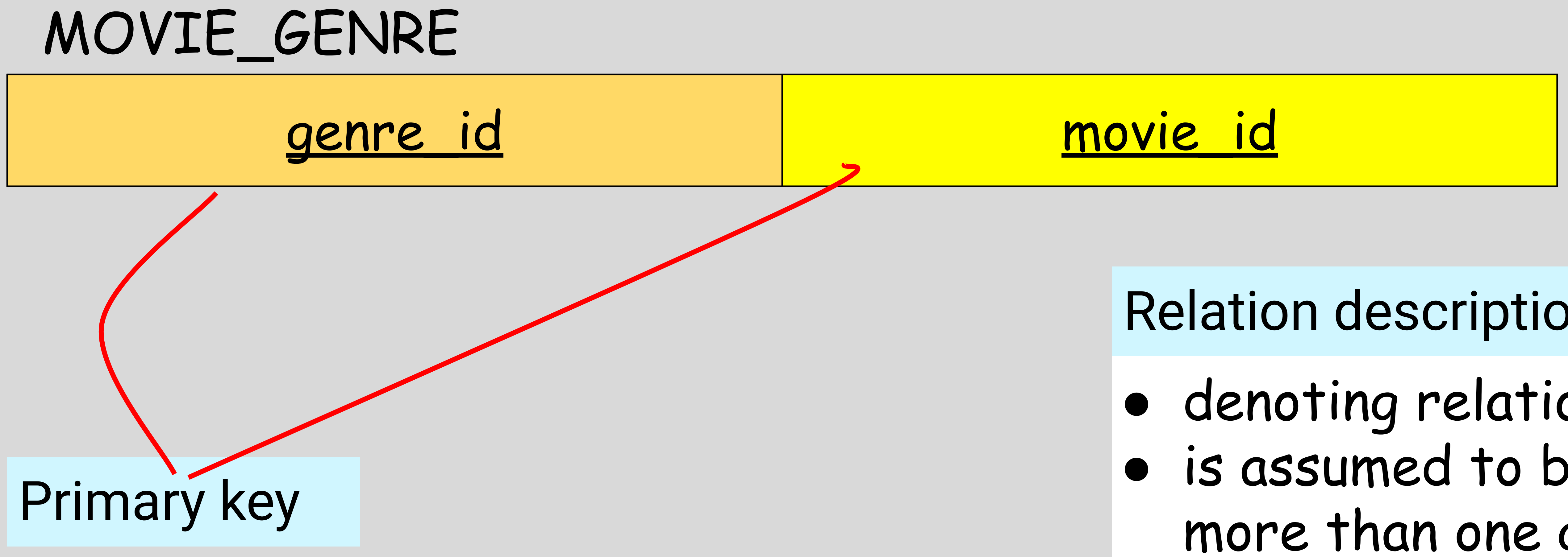
Primary key

Relation description

- denoting relation between movies and platform
- is assumed to be many to one relationship as we consider that each movie is available on only one platform and a platform can have multiple movies



Schema Diagram



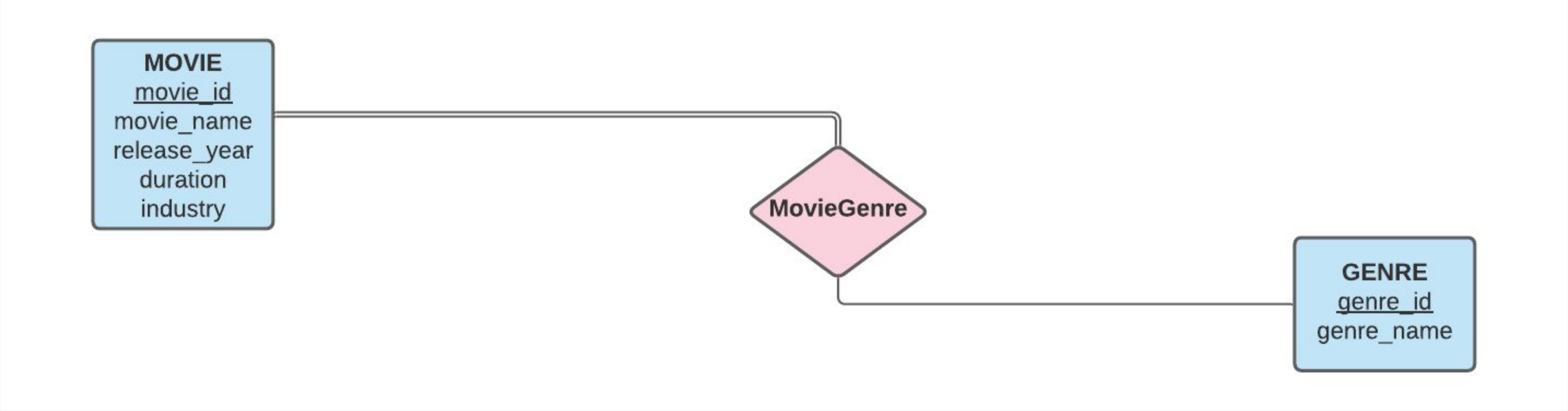
MOVIE_GENRE

genre_id	movie_id

Primary key

Relation description

- denoting relation between movies and genre
- is assumed to be many to many relationship as movies can have more than one genre also genres are associated with more than one movie, also
- there is total participation of movies as each movie should have atleast one genre



Schema Diagram

MOVIE_RATING

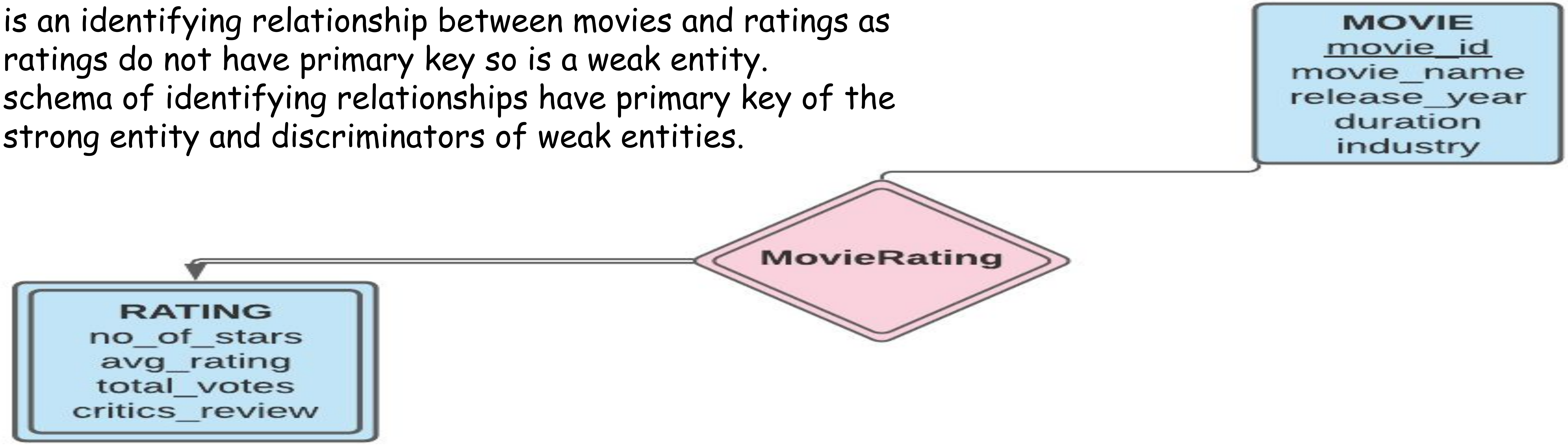
movie_id	critics_review	no_of_stars	avg_rating	total_votes

Primary key

This relation is redundant as it is identifying relation and the schema for ratings already contains all the information.

Relation description

- is an identifying relationship between movies and ratings as ratings do not have primary key so is a weak entity.
- schema of identifying relationships have primary key of the strong entity and discriminators of weak entities.



Schema Diagram

MOVIE NUMBERS

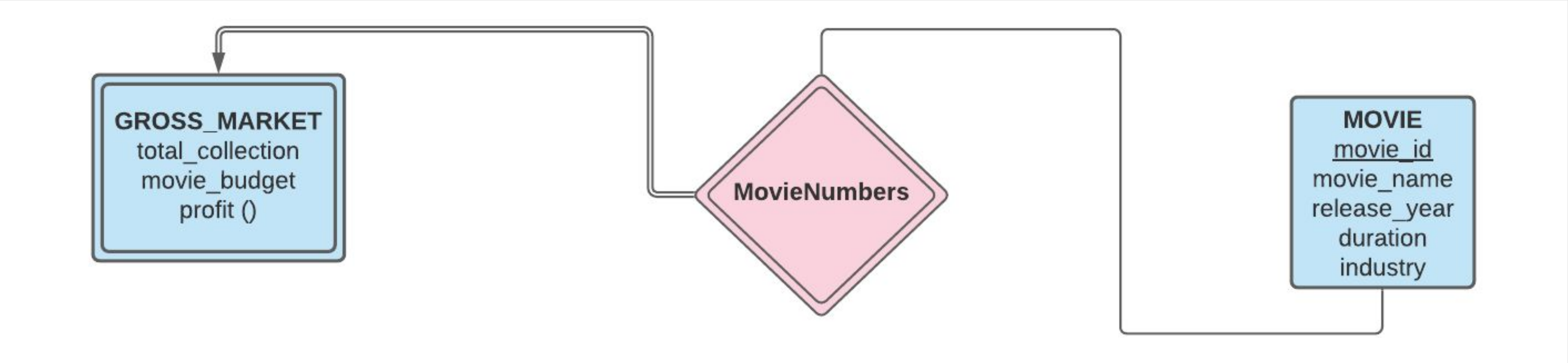
movie_id	total_collection	movie_budget	profit

Primary key

Relation description

- is an identifying relationship between movies and gross market as gross market do not have primary key so is a weak entity.
- profit is a derived attribute of budget and collection

This relation is redundant as it is identifying relation and the schema for gross market already contains all the information.



GROSS_MARKET
total_collection
movie_budget
profit ()

MovieNumbers

MOVIE
movie_id
movie_name
release_year
duration
industry

Explanation for Schema Diagram :

For the Entity “Movies”: “movie_id” is the primary key hence every other attribute is dependent on it.Also “movie_name” attribute can also take us to all the other attributes. Hence here normalization is required.

For the Entity “Actors”: “actor_id” is the only primary key and here normalization is not required. It is already in BCNF as there is no other dependencies.

For the Entity “Directors”: “director_id” is the only primary key and here normalization is not required. It is already in BCNF as there is no other dependencies.

For the Entity “Platform”: “platform_id” is the only primary key and here normalization is not required. It is already in BCNF as there is no other dependencies.

Explanation for Schema Diagram:

For the Entity "Genre": "genre_id" is the only primary key and here normalization is not required. It is already in BCNF.

For the Entity "Ratings": "movie_id" is the primary key and as it is weak entity, all other attributes are dependent on primary key. No other attribute can give other attribute. Hence it is also in BCNF.

For the Entity "Gross_Market": "movie_id" is the only primary key. Also "profit" attribute can be derived from "total_collection" and "Movie_budget". Hence here normalisation required.

For the Relation "Acted_By": "movie_id" and "actor_id" are the primary keys/candidate keys(Many to Many Relation) and here normalization is not required. It is already in BCNF

Explanation for Schema Diagram:

For the Relation "Directed_By": "movie_id" and "directed_id" is the primary key and here normalization is not required. It is already in BCNF. This is also Many to Many Relation.

For the relation "Watch_It_On": "movie_id" is the primary key and "platform_id" is dependent on it. And Normalization not required. This is Many to One Relation.

For the relation "movie_genre": "movie_id" and "genre_id" both are primary keys as it is many to many relation and hence it is already in BCNF.

For the Relation "movie_ratings": This is the weak entity relation of entities "Movies" and "Ratings". Hence identifying strong entity is "Movies" and "movie_id" is the primary key along with discriminators as all other entities. And this is redundant and should not be included in schema diagram.

Explanation for Schema Diagram:

For the Relation “Movie_Numbers”: This relation is similar as the relation “movie_ratings”. Hence here also “movie_id” is the primary key along with discriminators as all other attributes of entity “Gross_Market”. This is also redundant and should not be included in schema diagram.

Schema Diagram

MOVIES

movie_id	movie_name	release_year	industry	duration

ACTORS

actor_id	a_first_name	a_last_name	age	gender

DIRECTORS

director_id	d_first_name	d_last_name

PLATFORM

platform_id	platform_name

Genre

genre_id	genre_name

RATINGS

movie_id	critics_review	no_of_stars	avg_rating	total_votes

GROSS_MARKET

movie_id	total_collection	movie_budget	profit

ACTED_BY

movie_id	actor_id

DIRECTED_BY

movie_id	director_id

WATCH_IT_ON

platform_id	movie_id

MOVIE_GENRE

genre_id	movie_id

Normalised Schema

MOVIES_ONE

movie_id	movie_name

MOVIES_TWO

movie_name	release_ year	industry	duration

ACTORS

actor_id	a_first_name	a_last_name	age	gender

DIRECTORS

director_id	d_first_name	d_last_name

PLATFORM

platform_id	platform_name

Genre

genre_id	genre_name

RATINGS

movie_id	critics_review	no_of_stars	avg_rating	total_votes

GROSS_MARKET_ONE

movie_id	total_collection	movie_budget

GROSS_MARKET_TWO

total_collection	movie_budget	profit

ACTED_BY

movie_id	actor_id

DIRECTED_BY

movie_id	director_id

WATCH_IT_ON

platform_id	movie_id

MOVIE_GENRE

genre_id	movie_id

Explanation for Normalization:

“Movies” Entity: Here “movie_id” is the primary key and hence all other attributes are dependent on it. But “movie_name” attribute also determines all other attributes. Hence decomposition is done and now both “Movies_One” ans “Movies_Two” are in BCNF.

“Actors” Entity: It is already in BCNF.
“Directors” Entity : It is also already in BCNF.
“Platform” Entity : It is also already in BCNF.
“Genre” Entity : It is also already in BCNF.
“Ratings” Entity : It is also already in BCNF as it is weak entity and all functional dependencies are according to it.

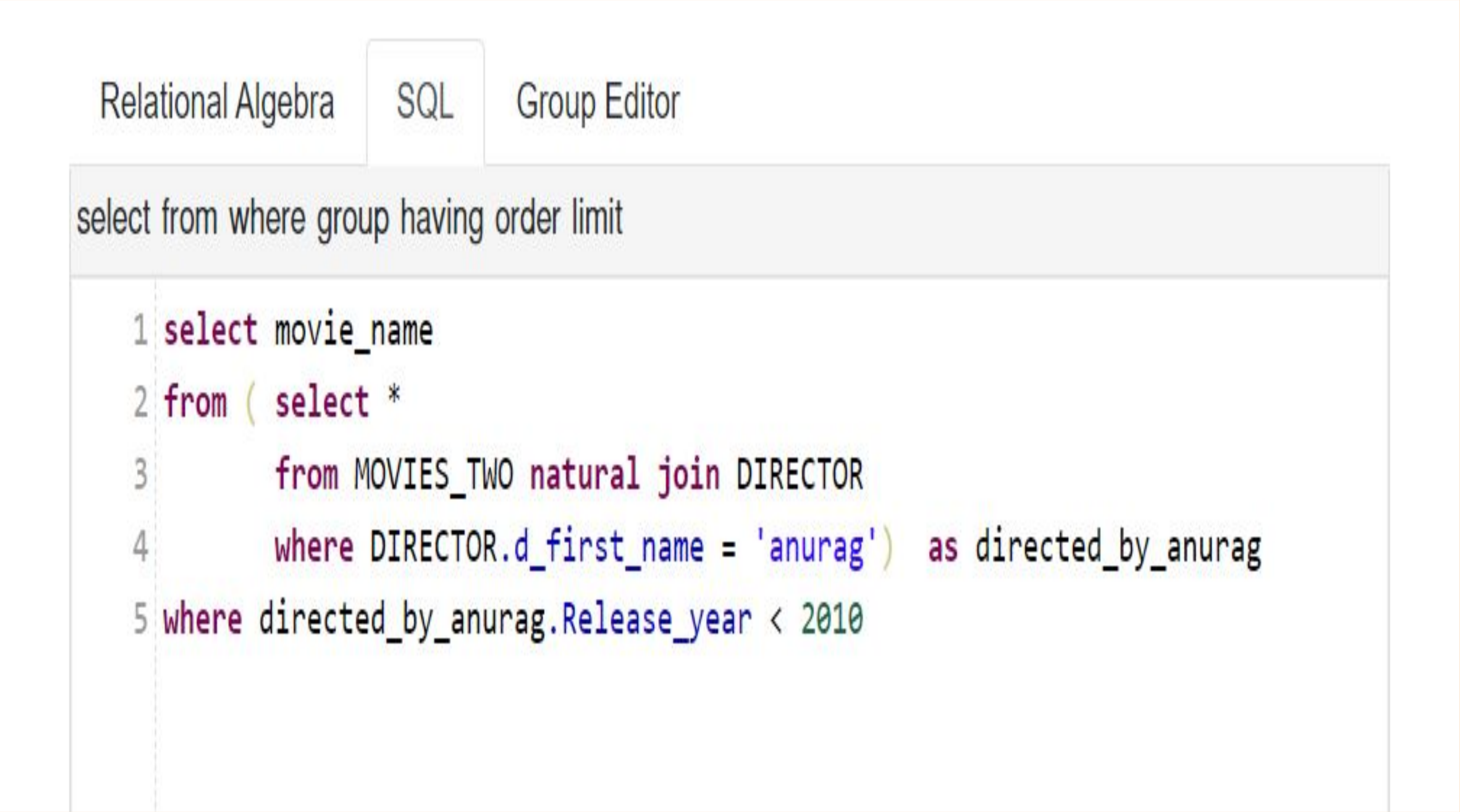
Explanation for Normalization:

“Gross_Market” Entity: Here “movie_id” is the primary key. But the non-prime attributes “total_collection” and “movie_budget” are deciding another non-prime attribute i.e. “profit”. Hence decomposition has been performed.
Now, both tables “Gross_Market_One” and “Gross_Market_Two” are in BCNF.

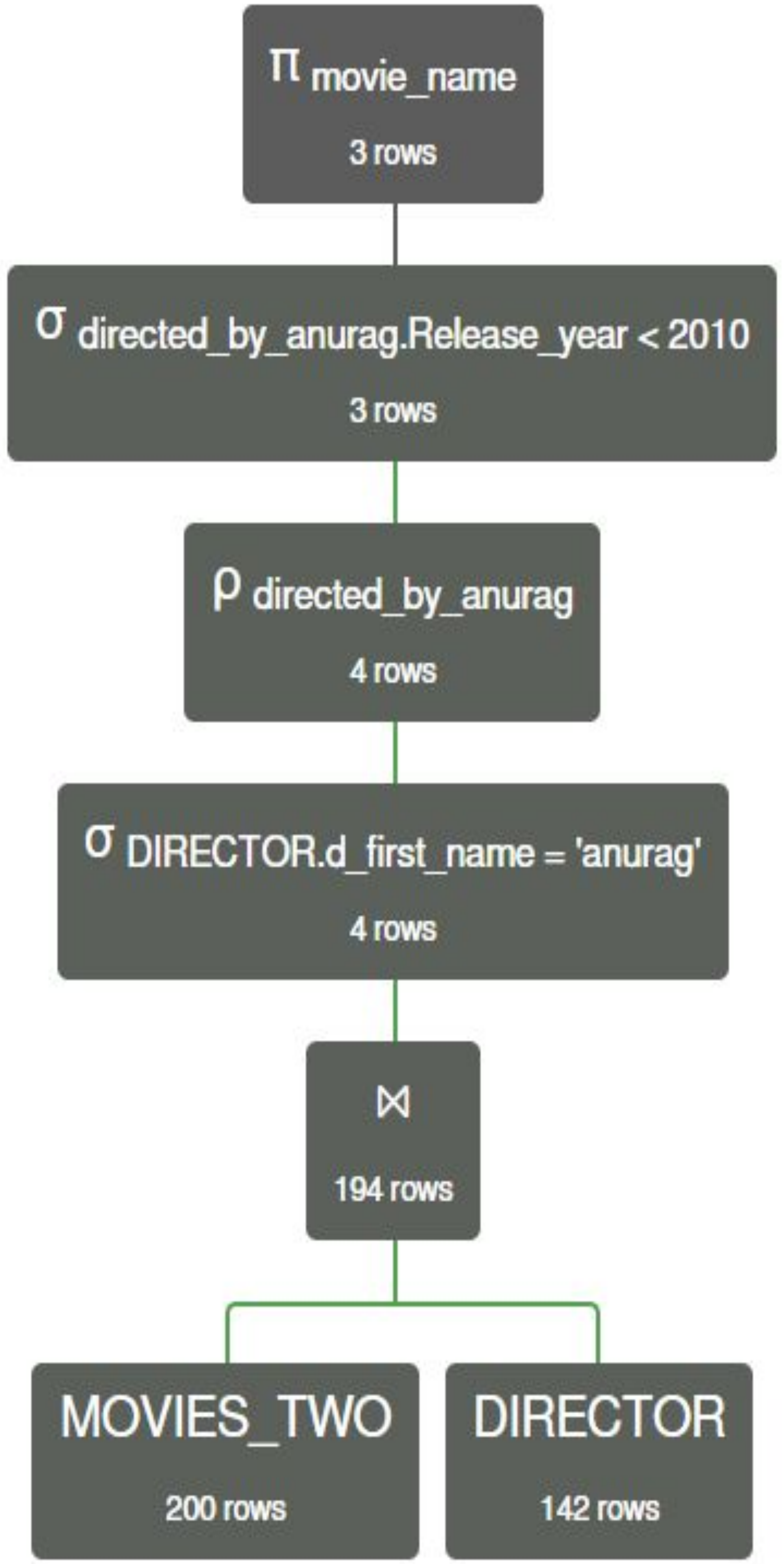
Similarly respective relation tables are already in/normalized into BCNF.
and relations of weak entity are redundant are not included in schema diagram.

Relational and SQL Queries

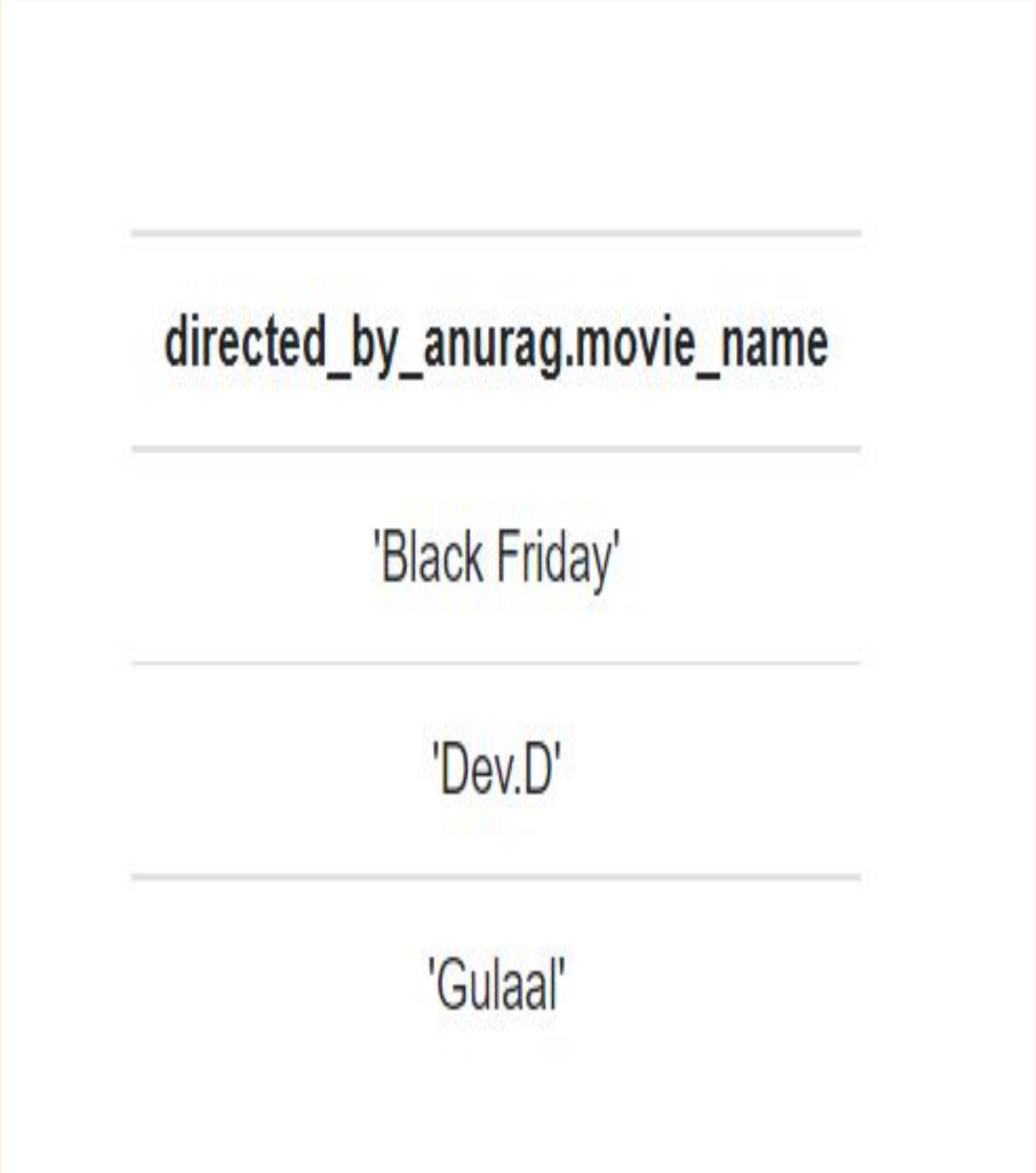
Q. Finding the movies directed by anurag before 2010



```
select movie_name
from ( select *
       from MOVIES_TWO natural join DIRECTOR
       where DIRECTOR.d_first_name = 'anurag')  as directed_by_anurag
where directed_by_anurag.Release_year < 2010
```

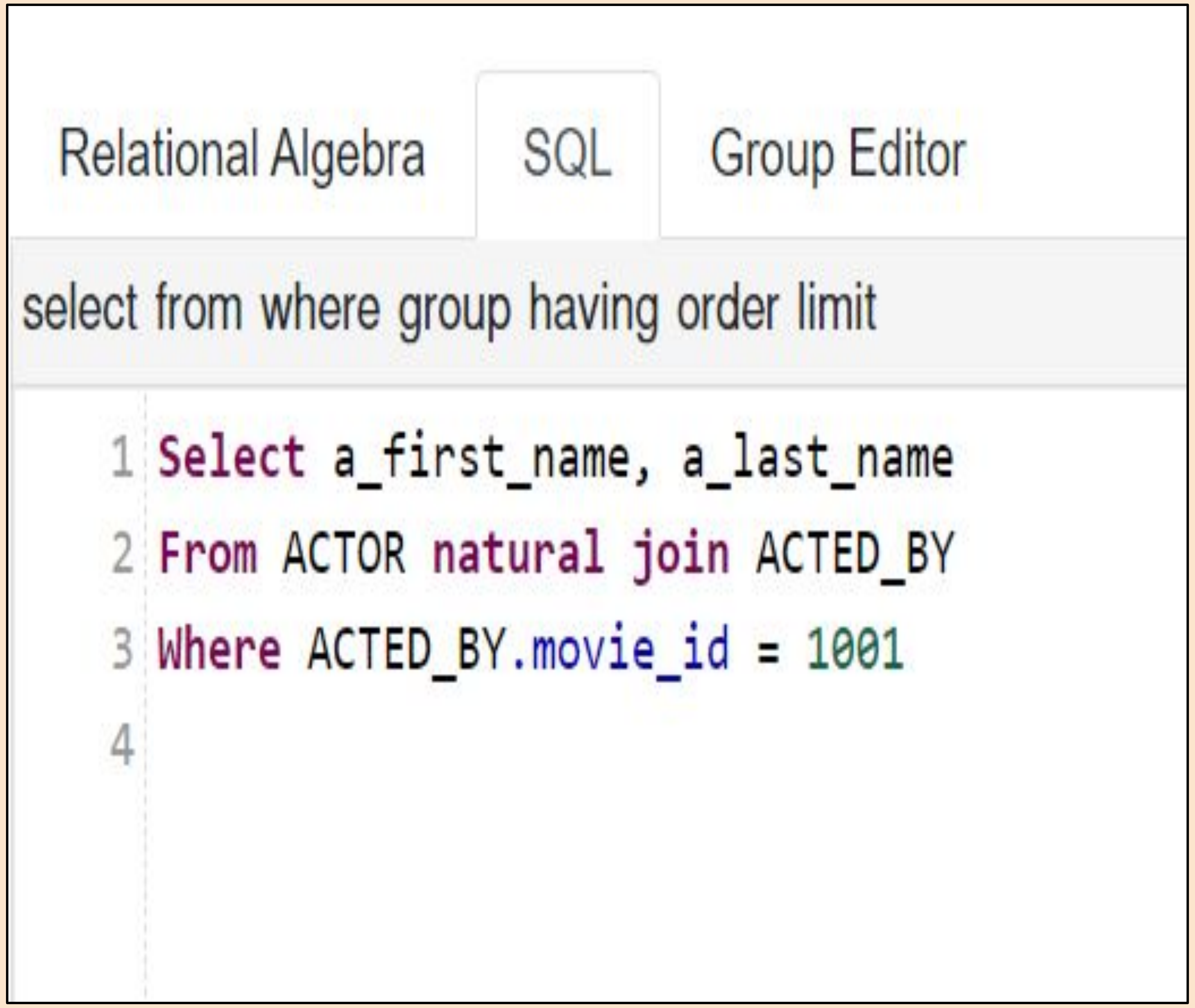


$$\pi_{\text{movie\_name}}\ \sigma_{\text{directed\_by\_anurag.Release\_year} < 2010}\ \rho_{\text{directed\_by\_anurag}}\ (\ \sigma_{\text{DIRECTOR.d\_first\_name = 'anurag'}}\ (\ \text{MOVIES\_TWO} \bowtie \text{DIRECTOR}\ )\ )$$

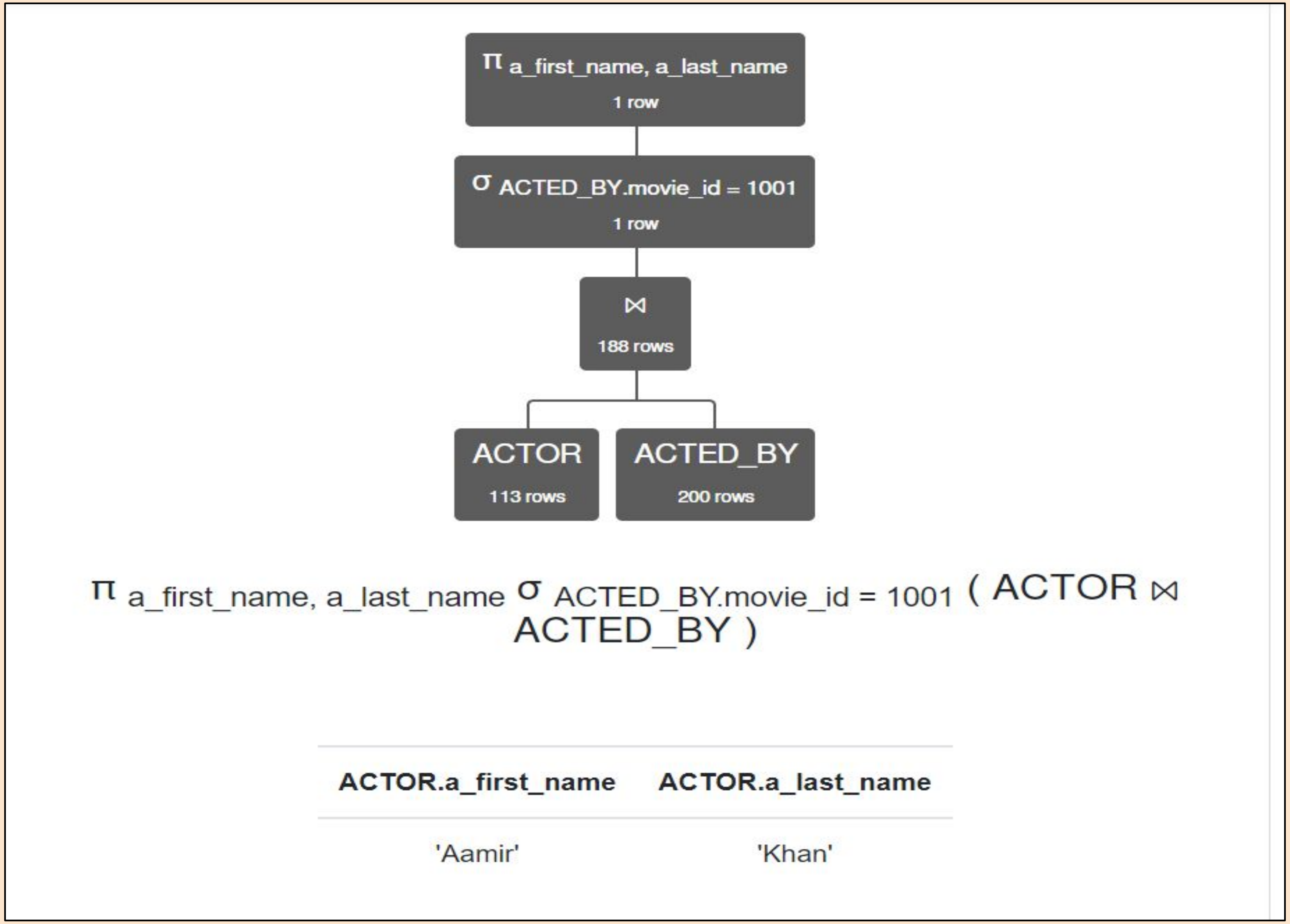


directed_by_anurag.movie_name
'Black Friday'
'Dev.D'
'Gulaal'

Q. Finding the names of the actors who were a part of a particular movie



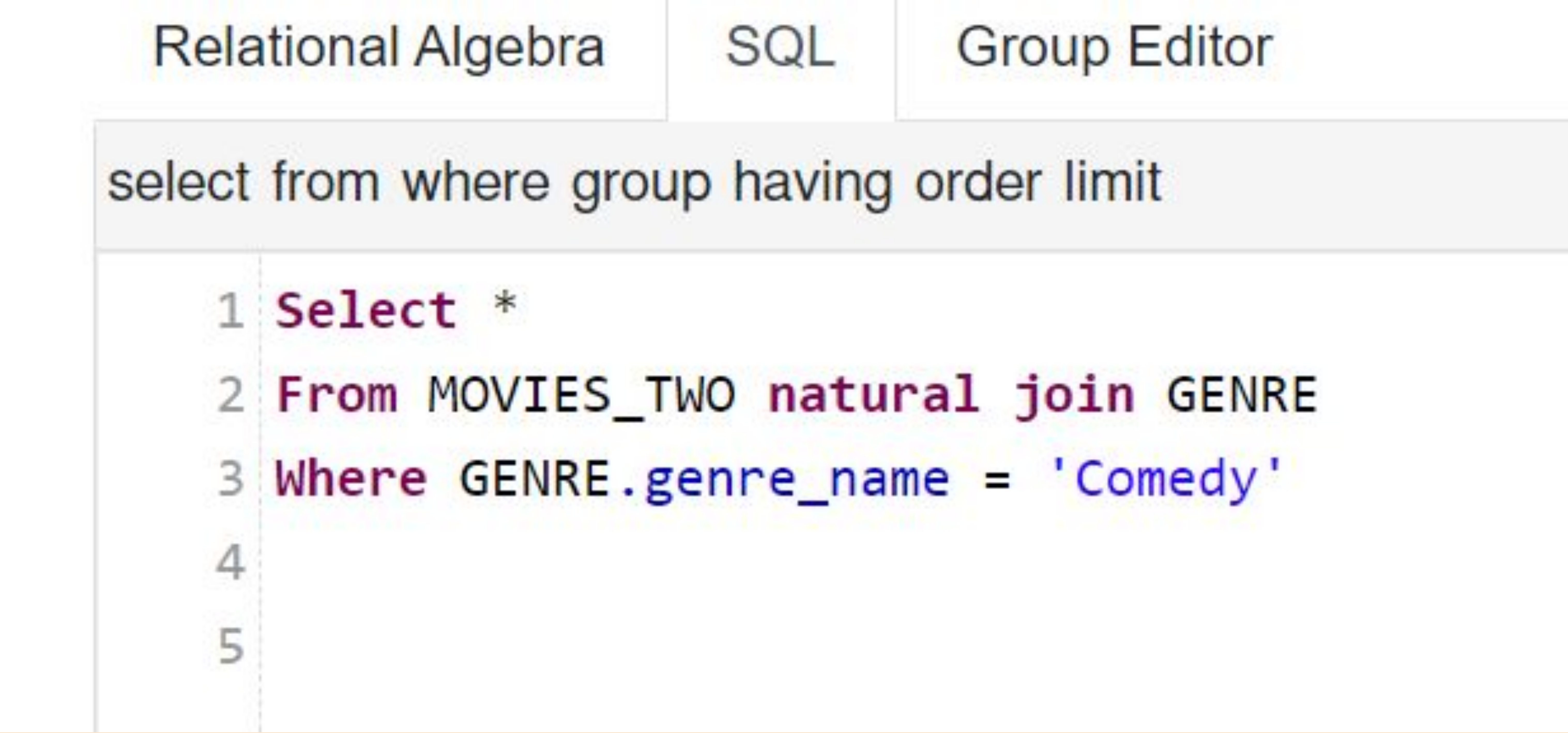
```sql
Select a_first_name, a_last_name
From ACTOR natural join ACTED_BY
Where ACTED_BY.movie_id = 1001

```



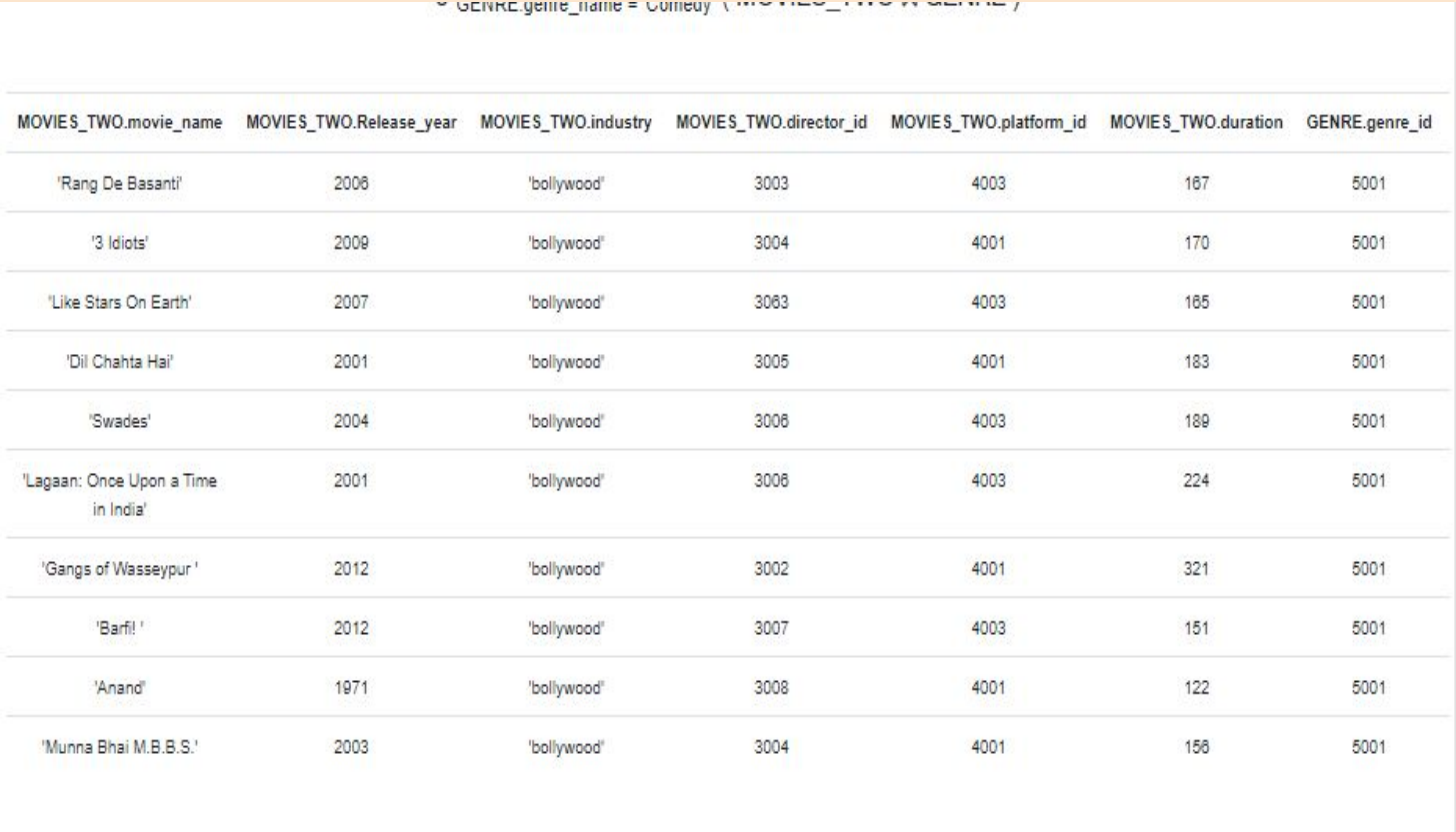
$\pi_{\text{a\_first\_name, a\_last\_name}}\ \sigma_{\text{ACTED\_BY.movie\_id = 1001}}\ (\text{ACTOR} \bowtie \text{ACTED\_BY})$

ACTOR.a_first_name	ACTOR.a_last_name
'Aamir'	'Khan'

Q. Finding the names of the movies with comedy genre



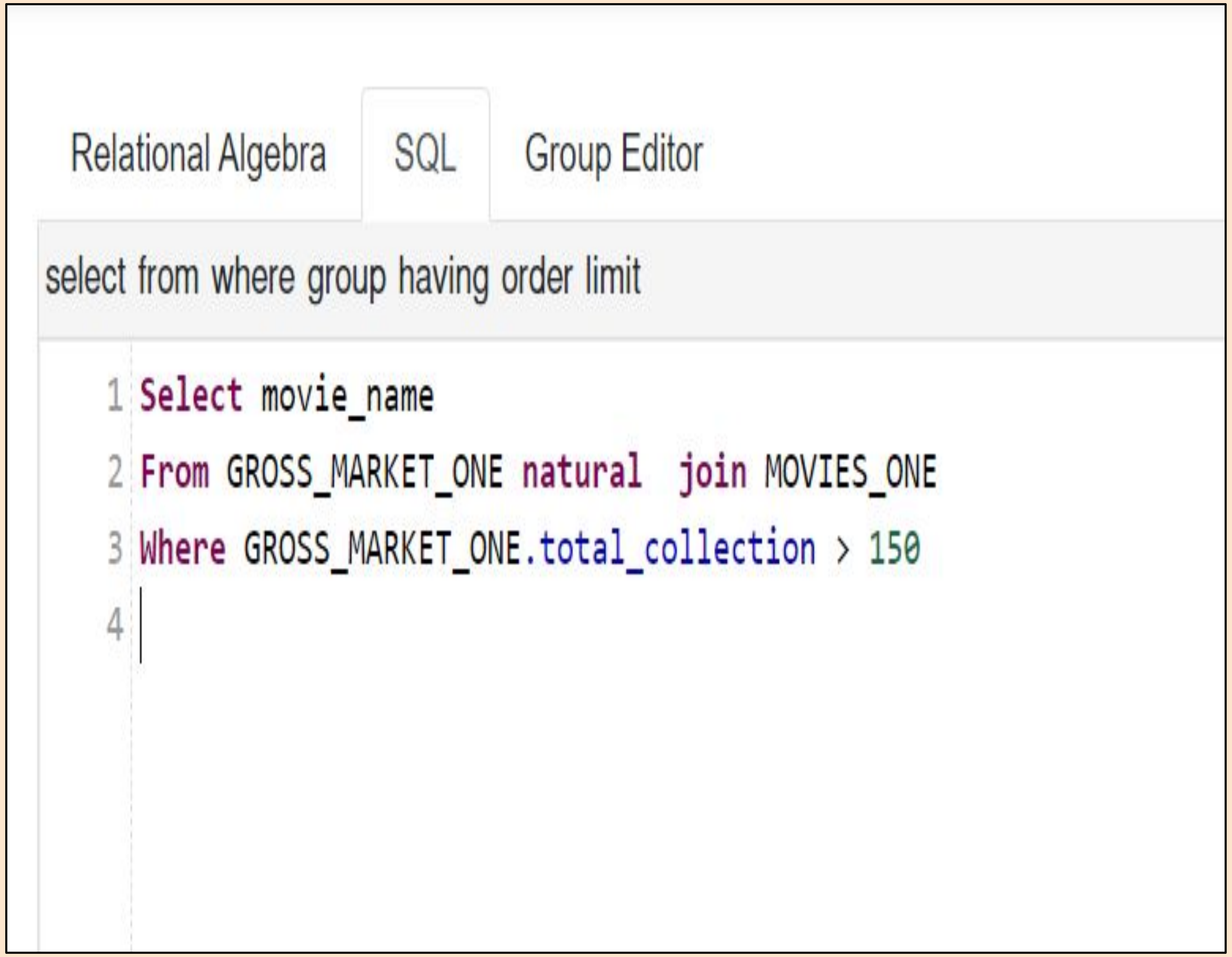
```
Select *
From MOVIES_TWO natural join GENRE
Where GENRE.genre_name = 'Comedy'
```



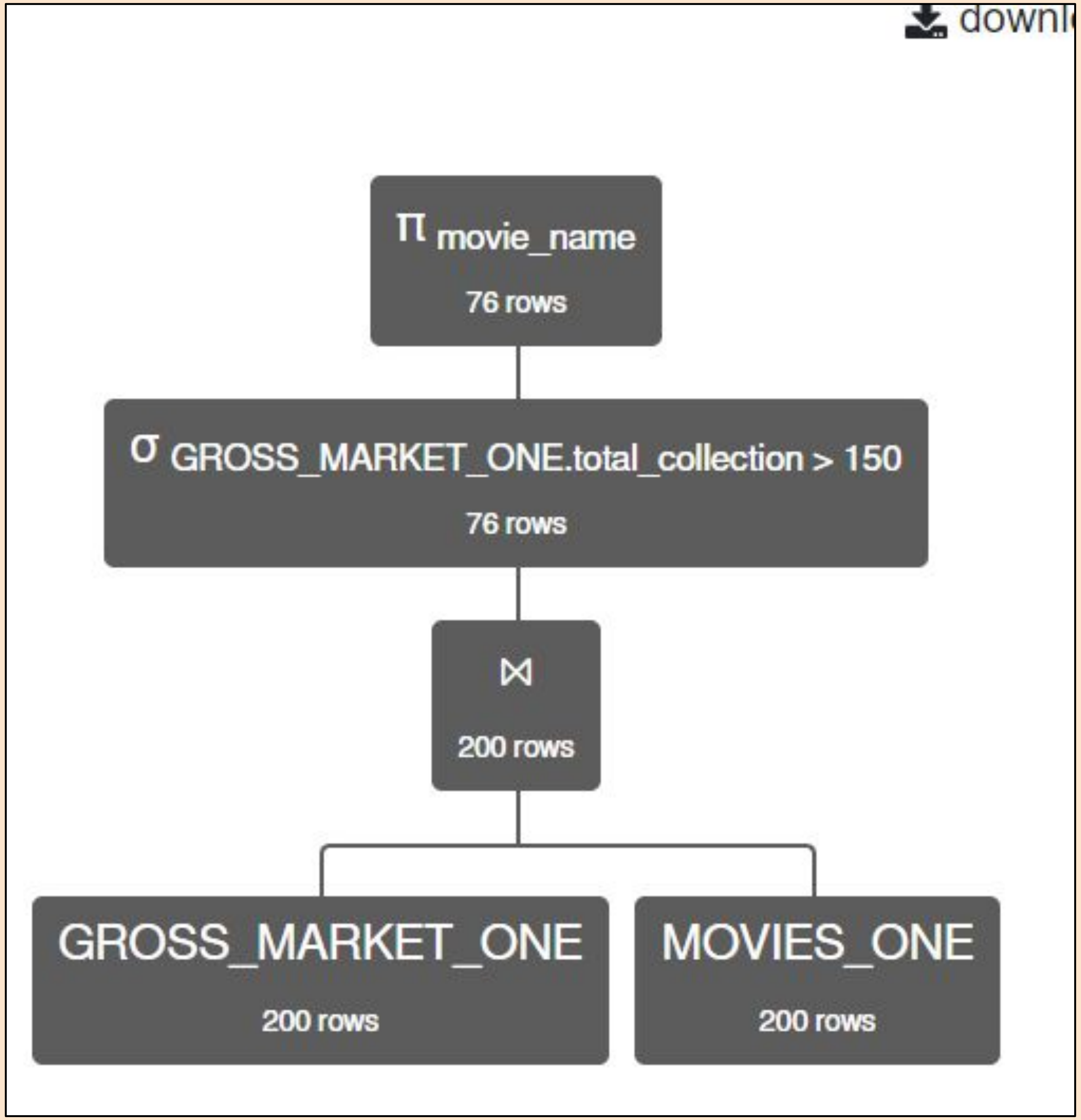
σ GENRE.genre_name = 'Comedy' (MOVIES_TWO ⋈ GENRE)

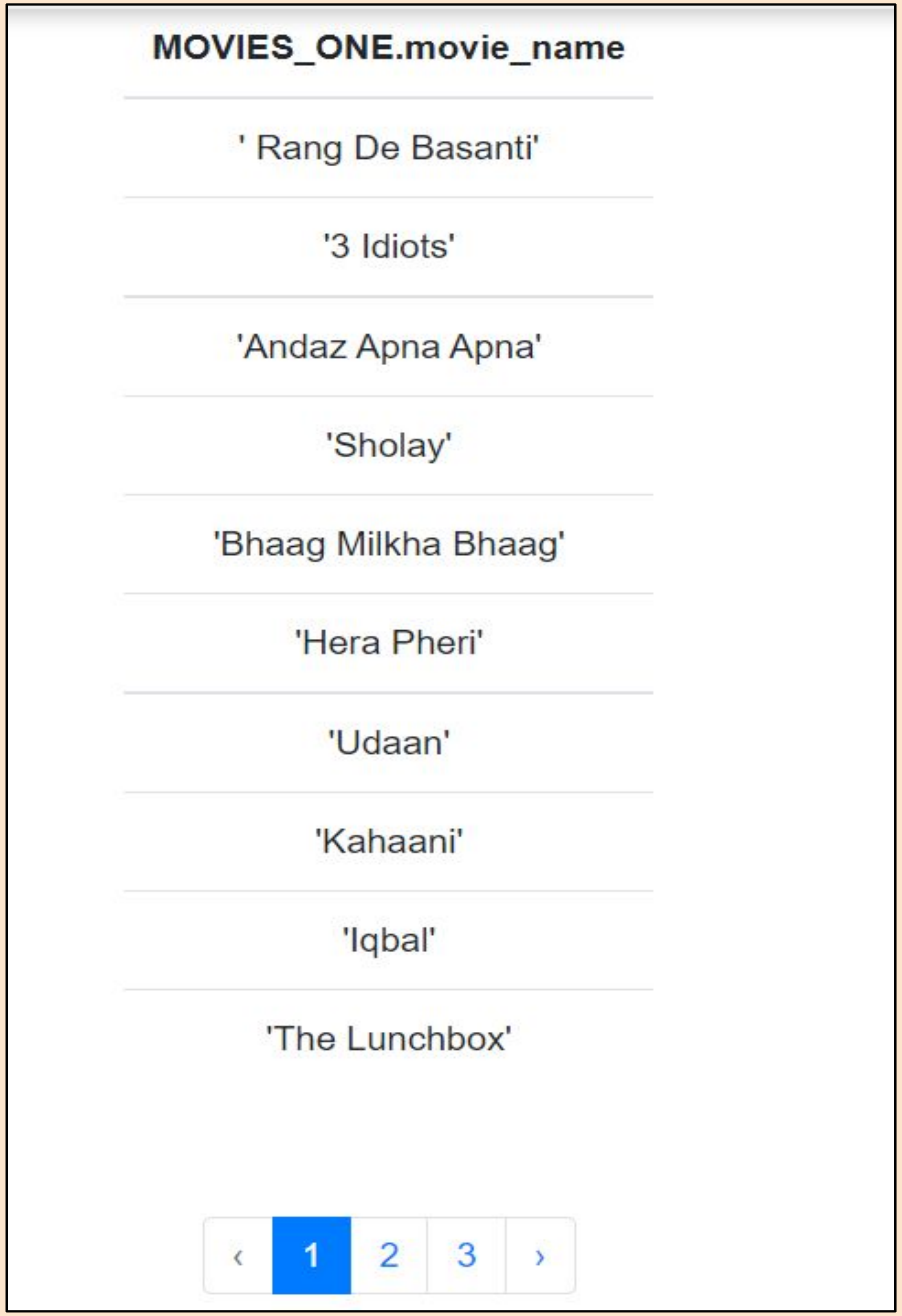
MOVIES_TWO.movie_name	MOVIES_TWO.Release_year	MOVIES_TWO.industry	MOVIES_TWO.director_id	MOVIES_TWO.platform_id	MOVIES_TWO.duration	GENRE.genre_id
'Rang De Basanti'	2006	'bollywood'	3003	4003	167	5001
'3 Idiots'	2009	'bollywood'	3004	4001	170	5001
'Like Stars On Earth'	2007	'bollywood'	3063	4003	165	5001
'Dil Chahta Hai'	2001	'bollywood'	3005	4001	183	5001
'Swades'	2004	'bollywood'	3006	4003	189	5001
'Lagaan: Once Upon a Time in India'	2001	'bollywood'	3006	4003	224	5001
'Gangs of Wasseypur '	2012	'bollywood'	3002	4001	321	5001
'Barfi! '	2012	'bollywood'	3007	4003	151	5001
'Anand'	1971	'bollywood'	3008	4001	122	5001
'Munna Bhai M.B.B.S.'	2003	'bollywood'	3004	4001	156	5001

Q. Finding the movies with a total collection greater than 150cr

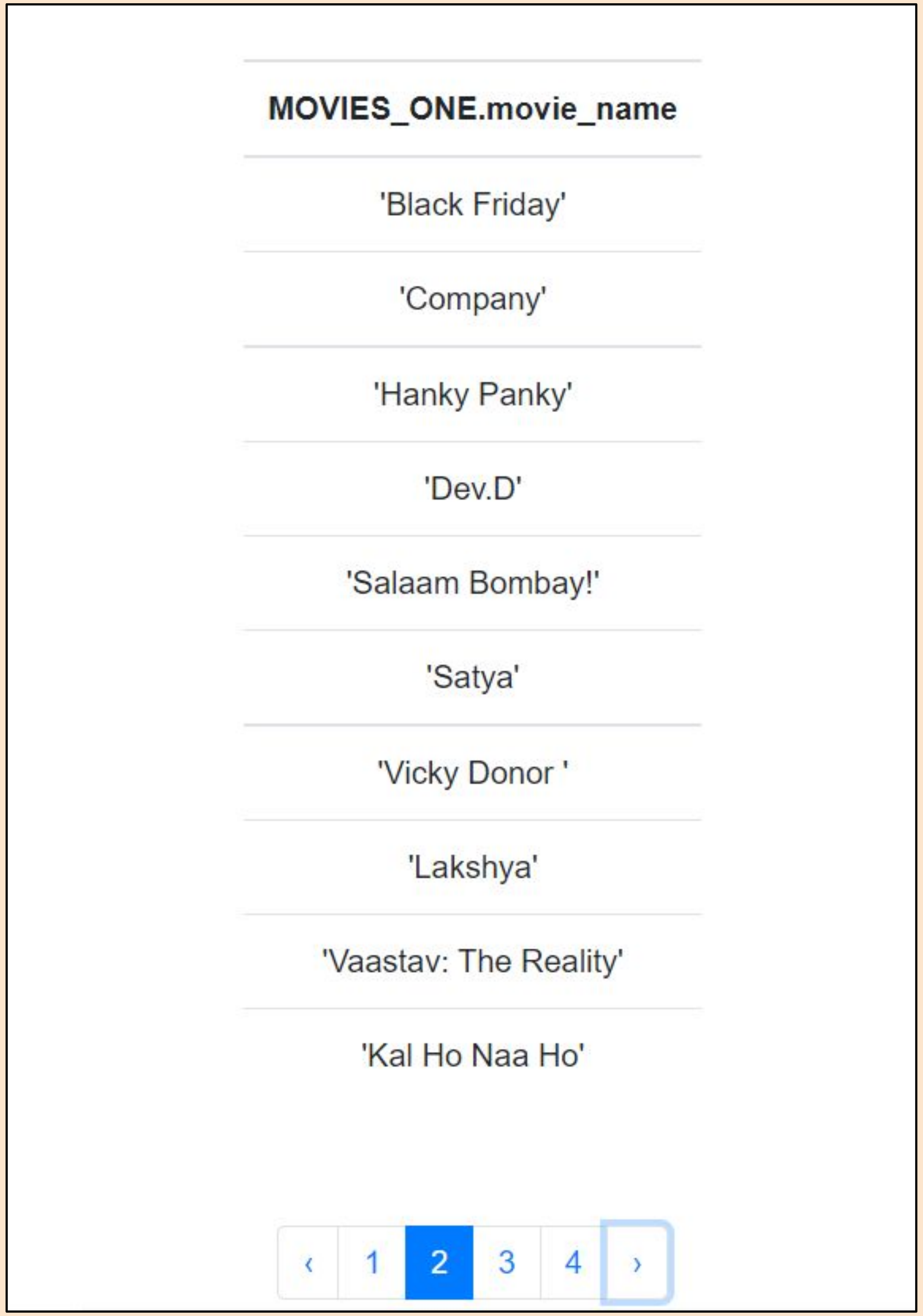


```
Select movie_name
From GROSS_MARKET_ONE natural  join MOVIES_ONE
Where GROSS_MARKET_ONE.total_collection > 150
```

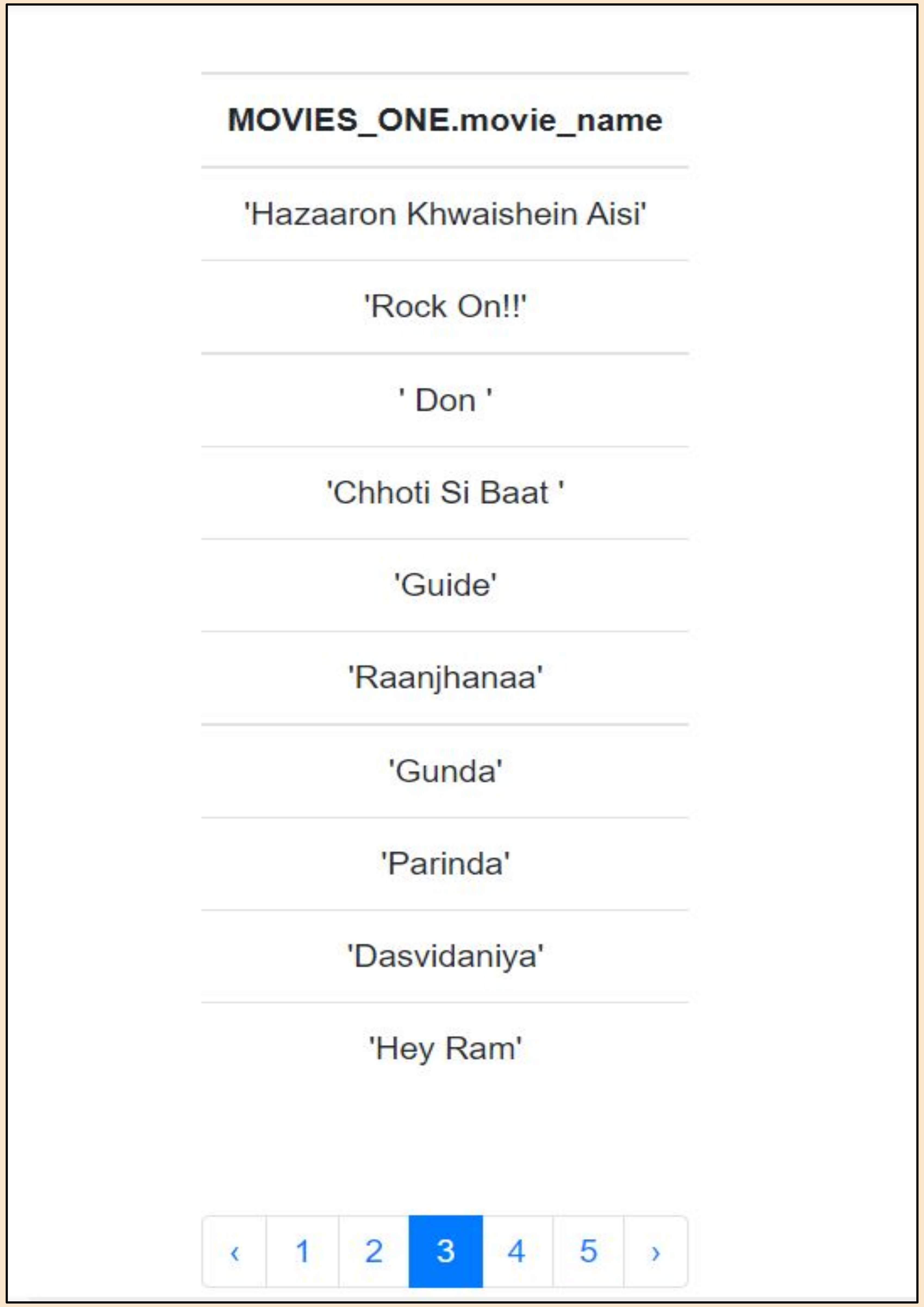




MOVIES_ONE.movie_name
' Rang De Basanti'
'3 Idiots'
'Andaz Apna Apna'
'Sholay'
'Bhaag Milkha Bhaag'
'Hera Pheri'
'Udaan'
'Kahaani'
'Iqbal'
'The Lunchbox'

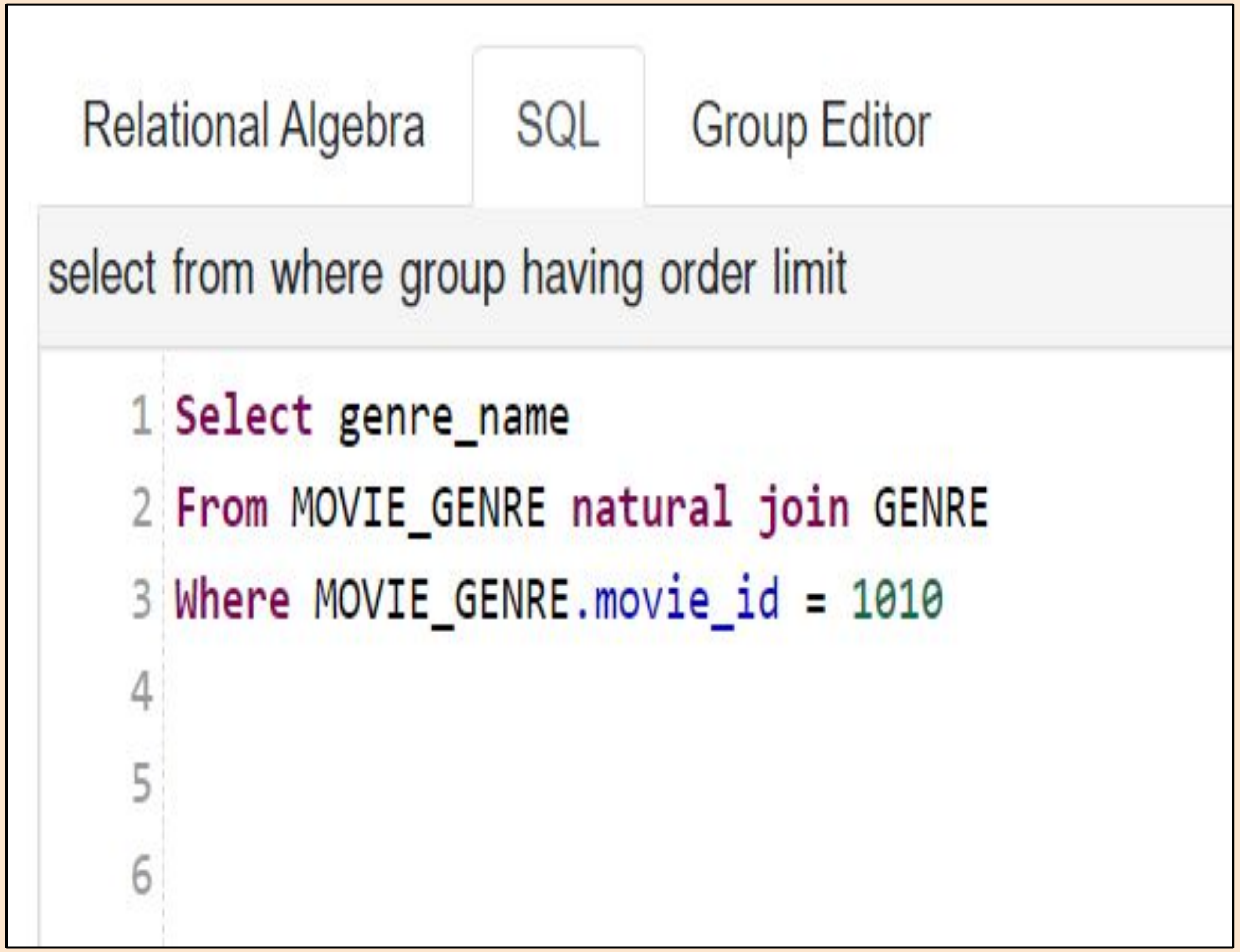


MOVIES_ONE.movie_name
'Black Friday'
'Company'
'Hanky Panky'
'Dev.D'
'Salaam Bombay!'
'Satya'
'Vicky Donor '
'Lakshya'
'Vaastav: The Reality'
'Kal Ho Naa Ho'

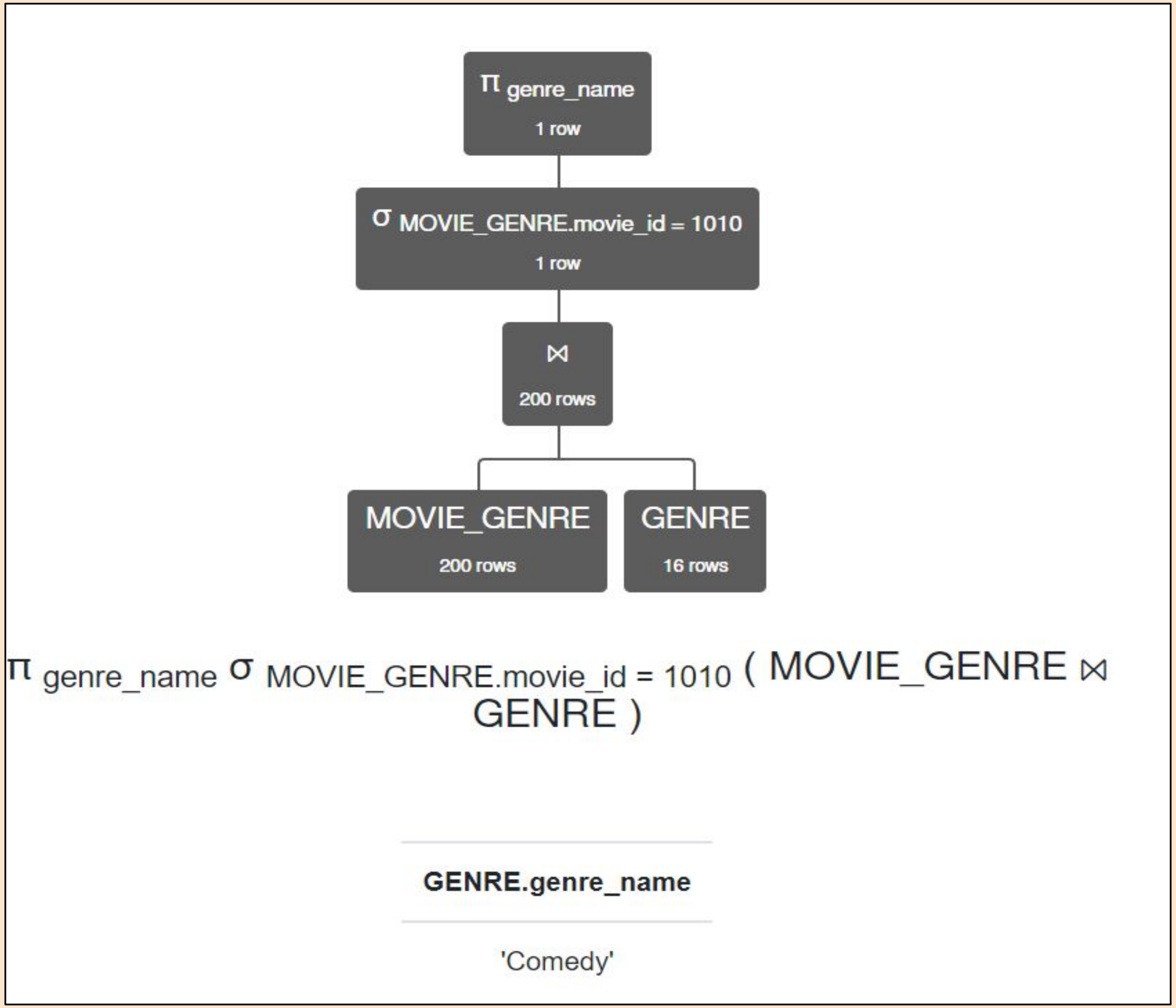


MOVIES_ONE.movie_name
'Hazaaron Khwaishein Aisi'
'Rock On!!'
' Don '
'Chhoti Si Baat '
'Guide'
'Raanjhanaa'
'Gunda'
'Parinda'
'Dasvidaniya'
'Hey Ram'

Q. Finding the genres of the movie



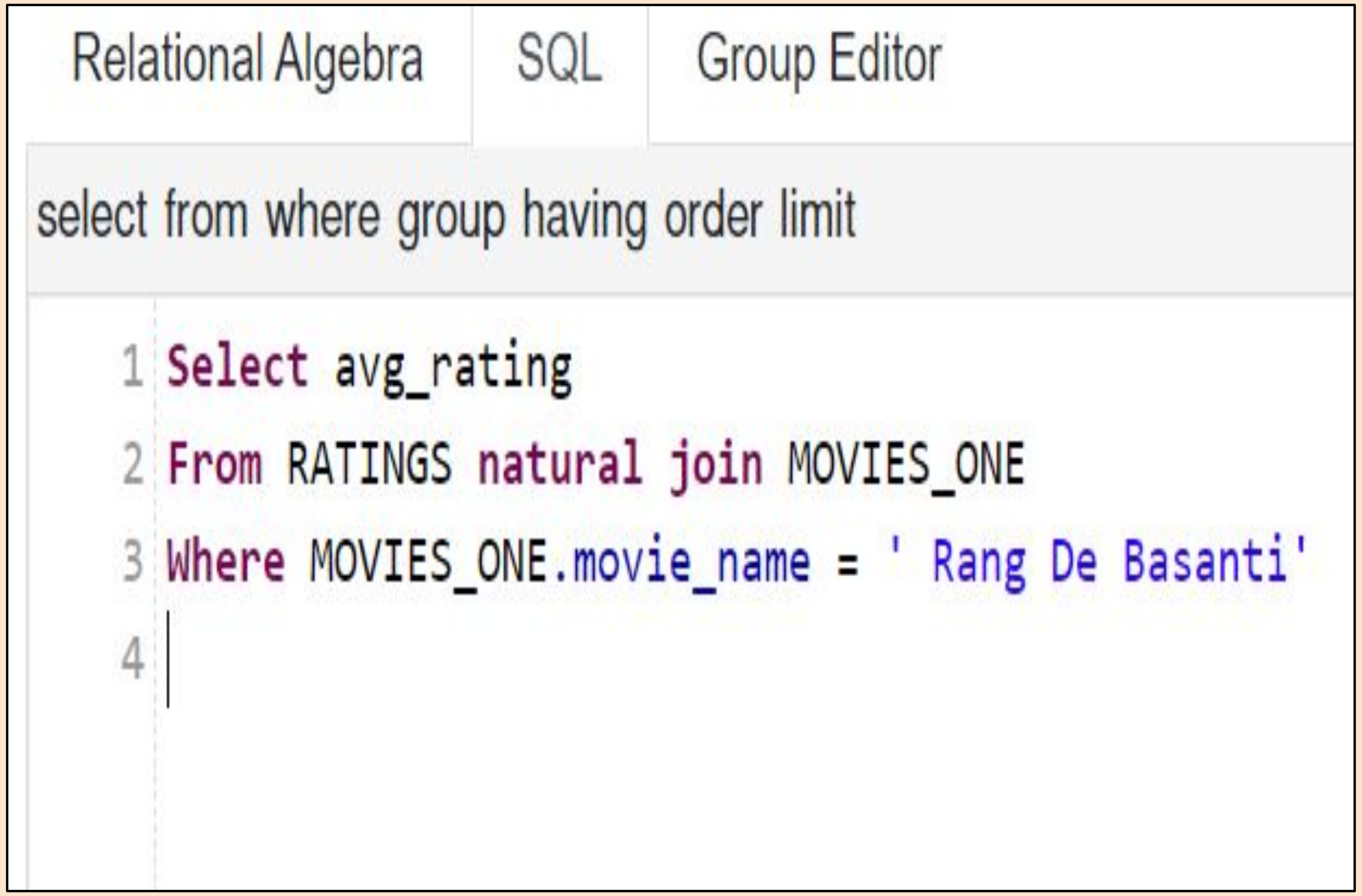
```
Select genre_name
From MOVIE_GENRE natural join GENRE
Where MOVIE_GENRE.movie_id = 1010
```



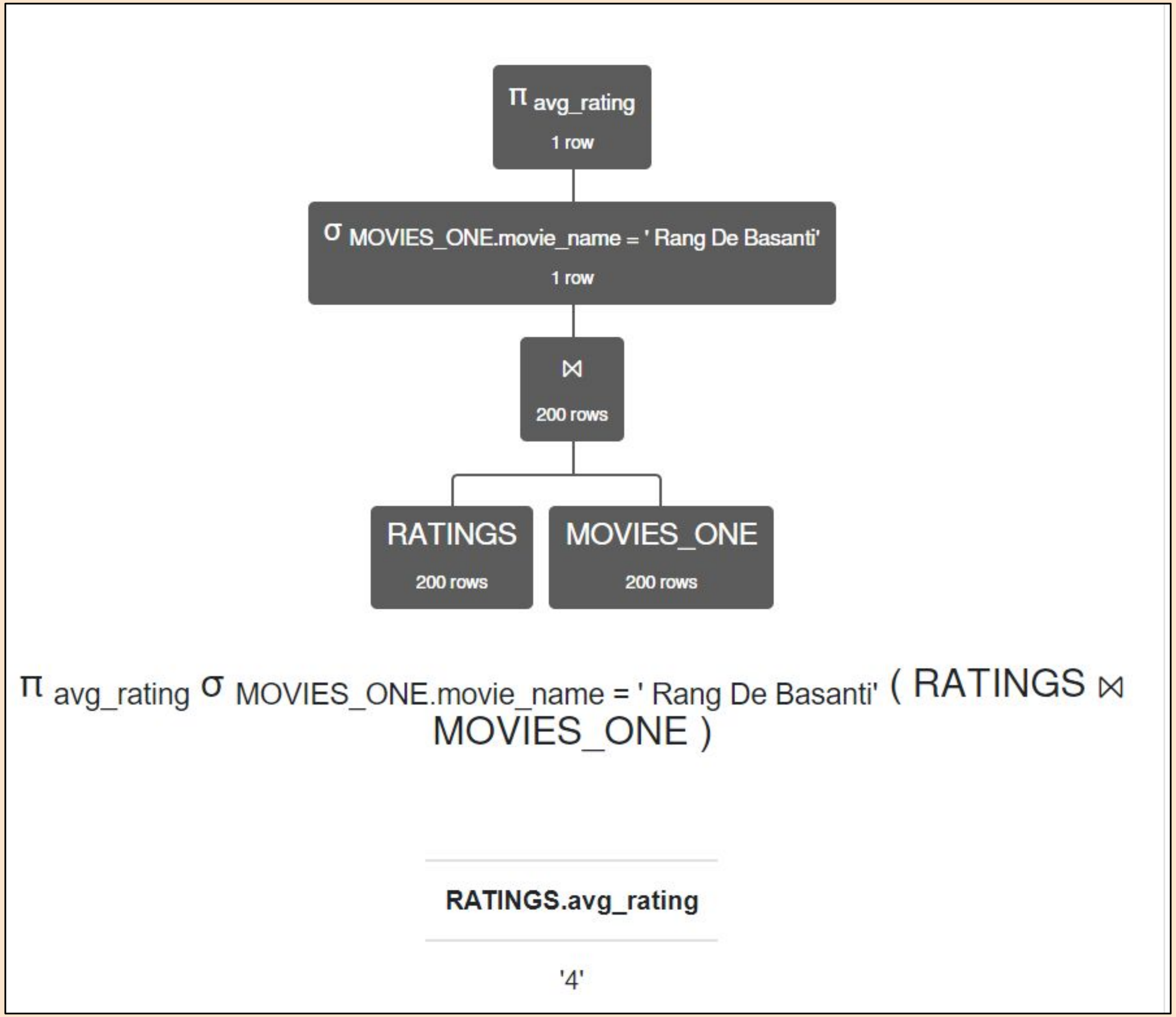
$\pi_{\text{genre\_name}}\ \sigma_{\text{MOVIE\_GENRE.movie\_id} = 1010}$ (MOVIE_GENRE ⋈ GENRE)

GENRE.genre_name
'Comedy'

Q. Finding the average ratings of a movie



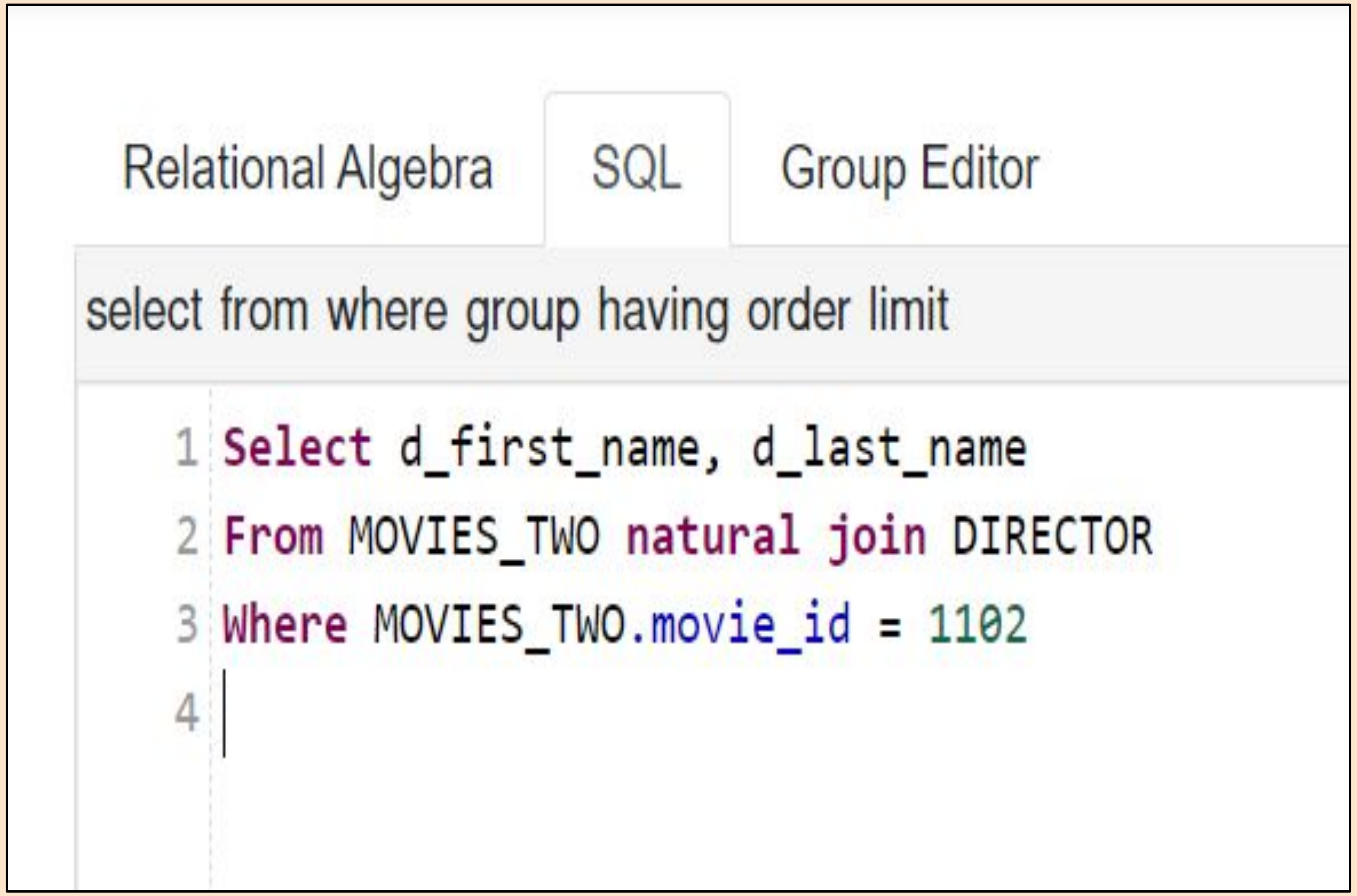
```
Select avg_rating
From RATINGS natural join MOVIES_ONE
Where MOVIES_ONE.movie_name = ' Rang De Basanti'
```



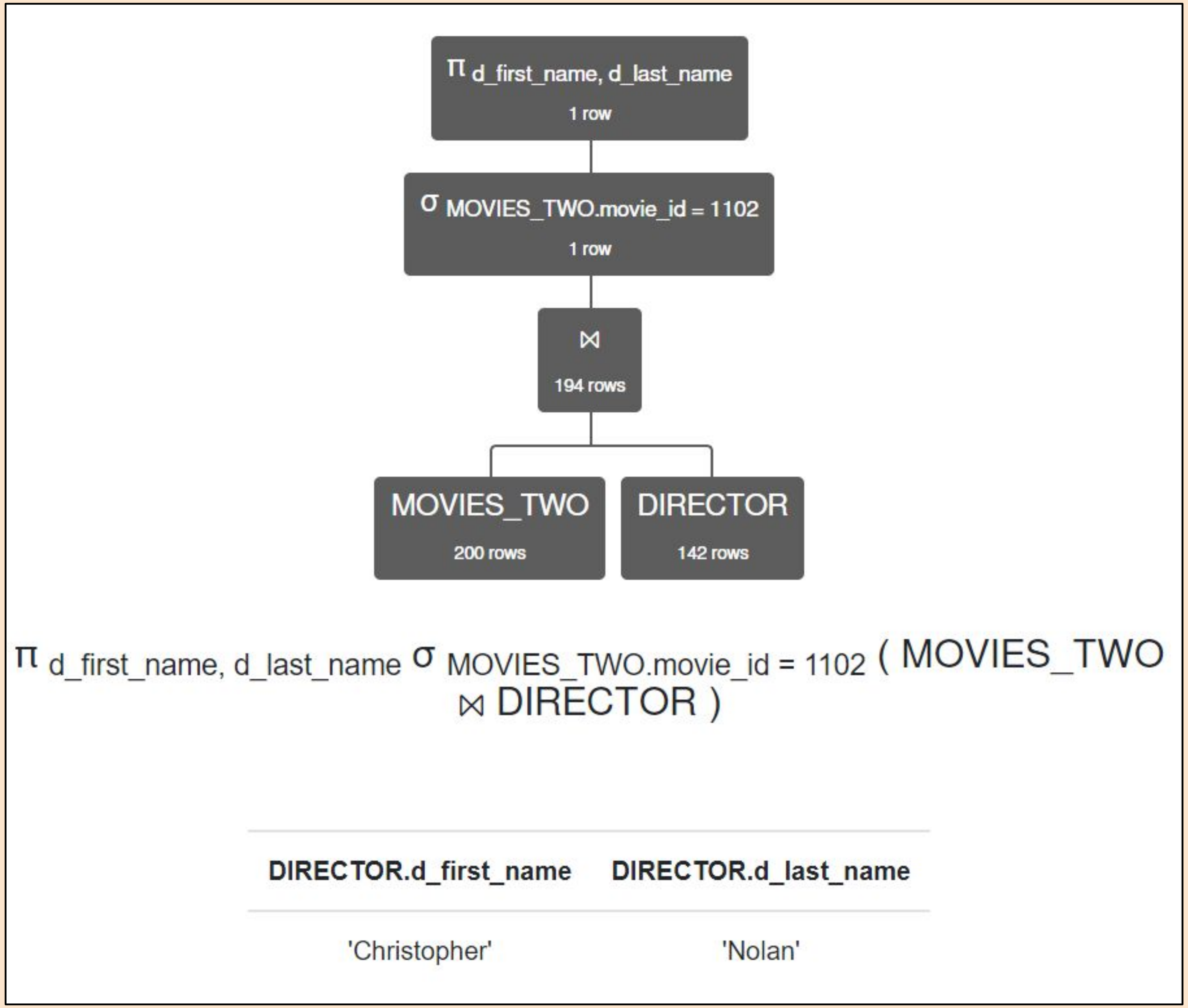
$$\pi_{\text{avg\_rating}}\ \sigma_{\text{MOVIES\_ONE.movie\_name = ' Rang De Basanti'}}\ (\text{RATINGS} \bowtie \text{MOVIES\_ONE})$$

RATINGS.avg_rating
'4'

Q. Finding the names of the directors of a particular movie



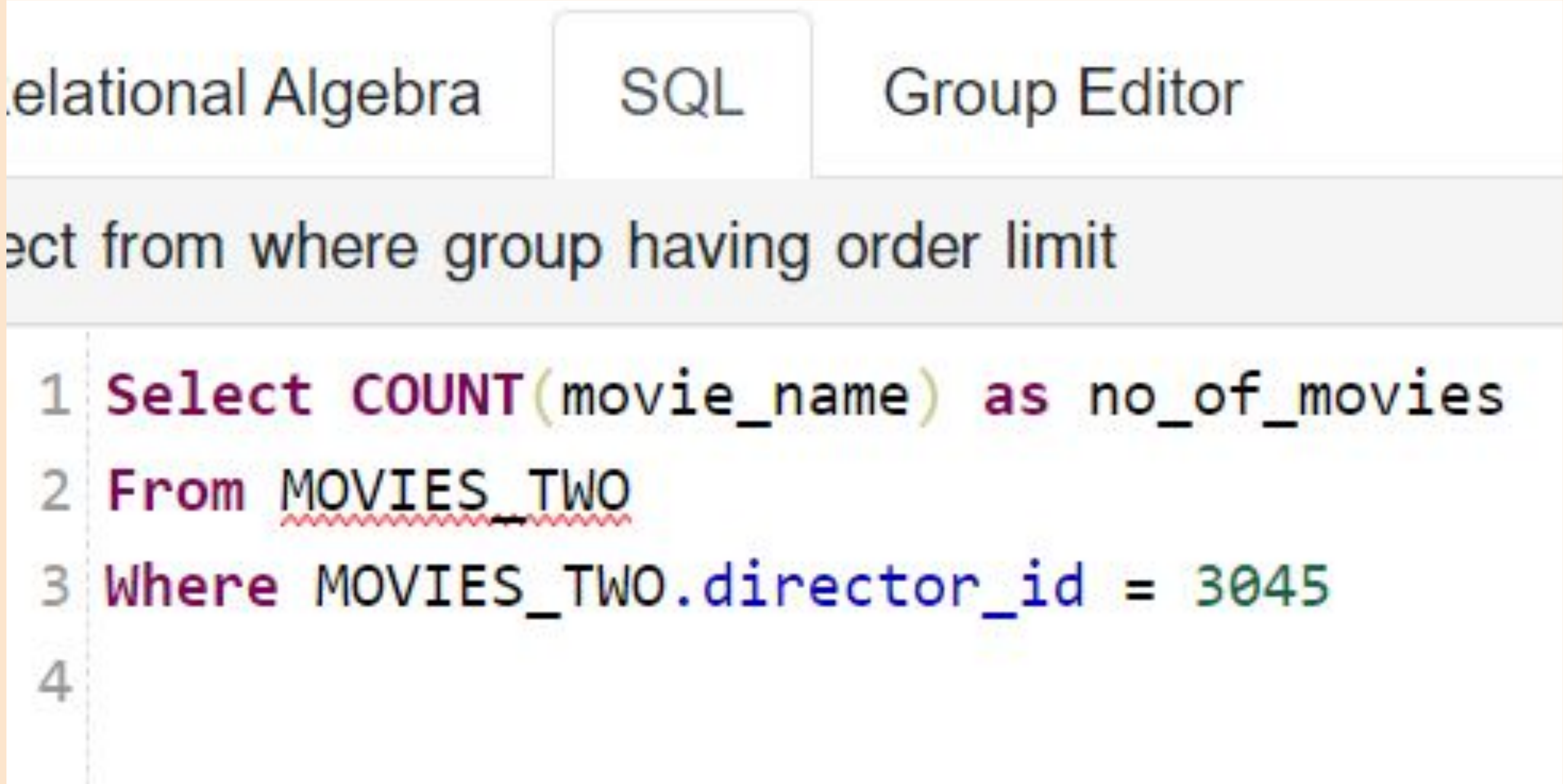
```sql
Select d_first_name, d_last_name
From MOVIES_TWO natural join DIRECTOR
Where MOVIES_TWO.movie_id = 1102
```



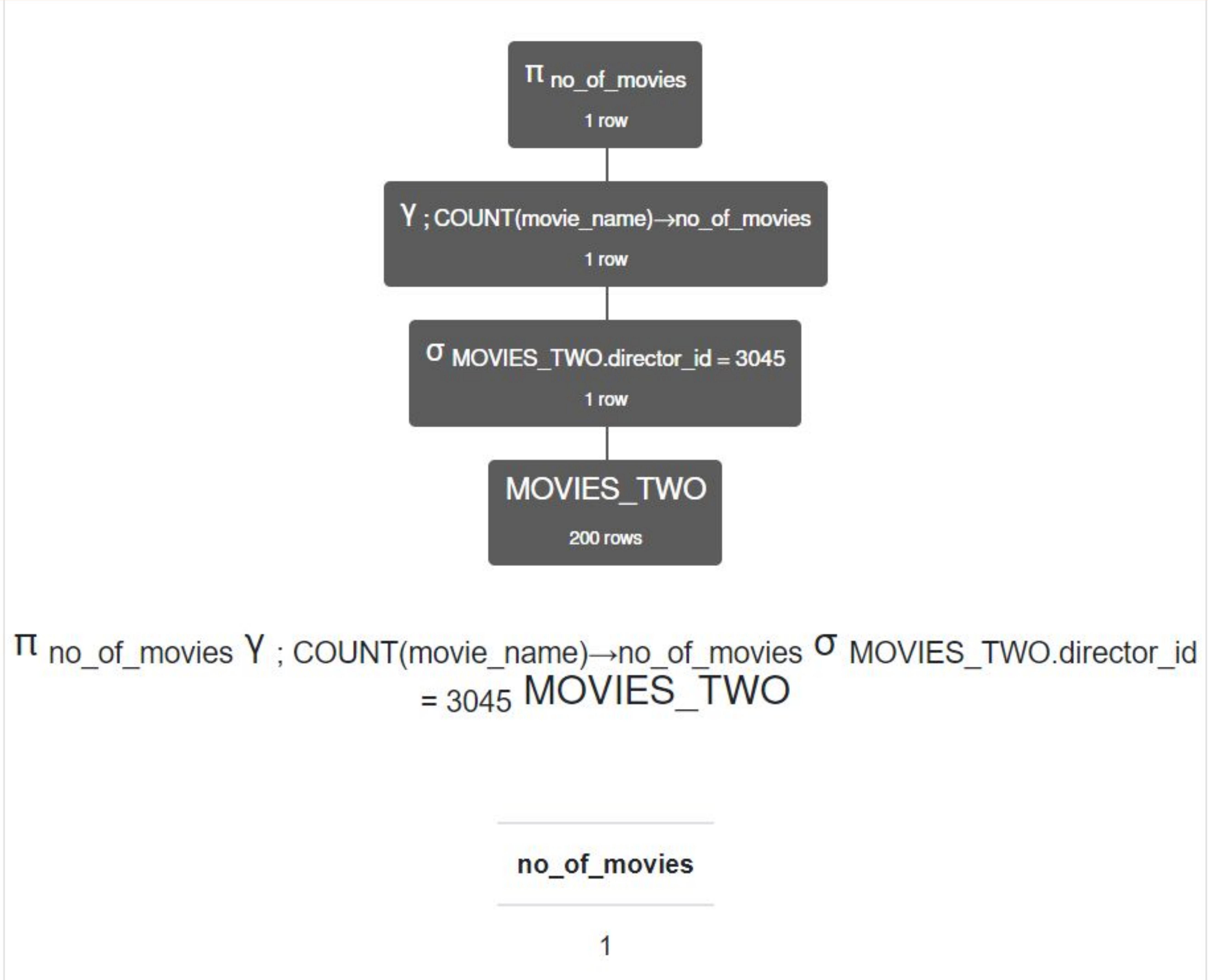
$$\pi_{\text{d\_first\_name, d\_last\_name}}\ \sigma_{\text{MOVIES\_TWO.movie\_id} = 1102}\ (\text{MOVIES\_TWO} \bowtie \text{DIRECTOR})$$

DIRECTOR.d_first_name	DIRECTOR.d_last_name
'Christopher'	'Nolan'

Q. Finding the number of movies that are in the database directed by a particular director



```
Select COUNT(movie_name) as no_of_movies
From MOVIES_TWO
Where MOVIES_TWO.director_id = 3045
```

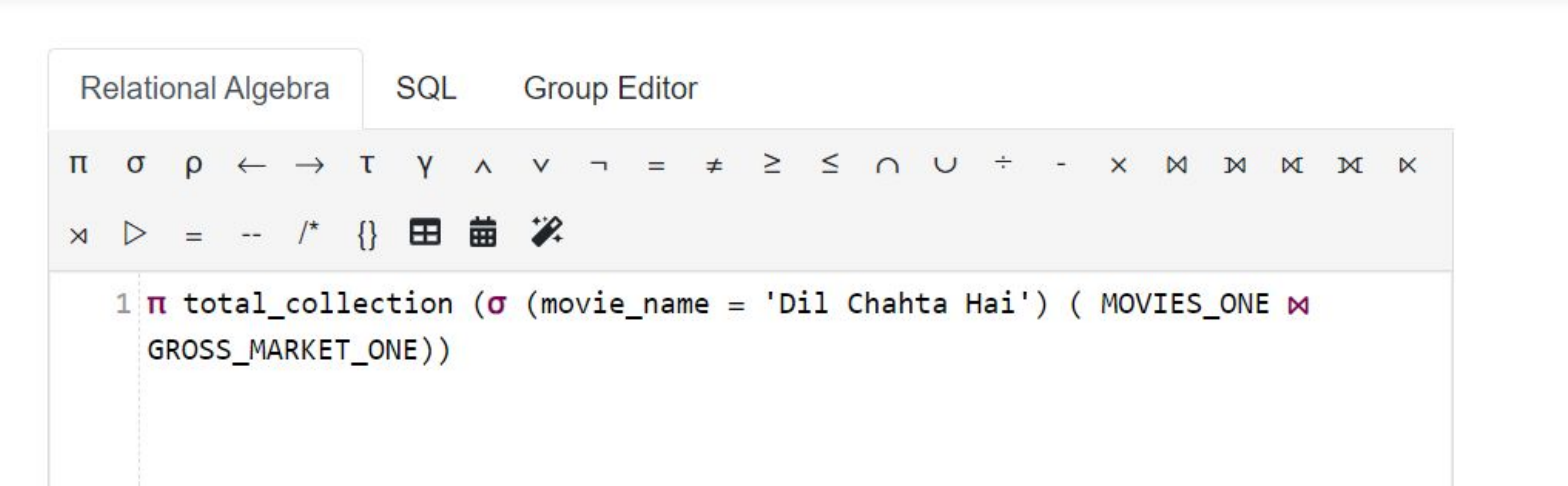


π no_of_movies
1 row
γ ; COUNT(movie_name)→no_of_movies
1 row
σ MOVIES_TWO.director_id = 3045
1 row
MOVIES_TWO
200 rows

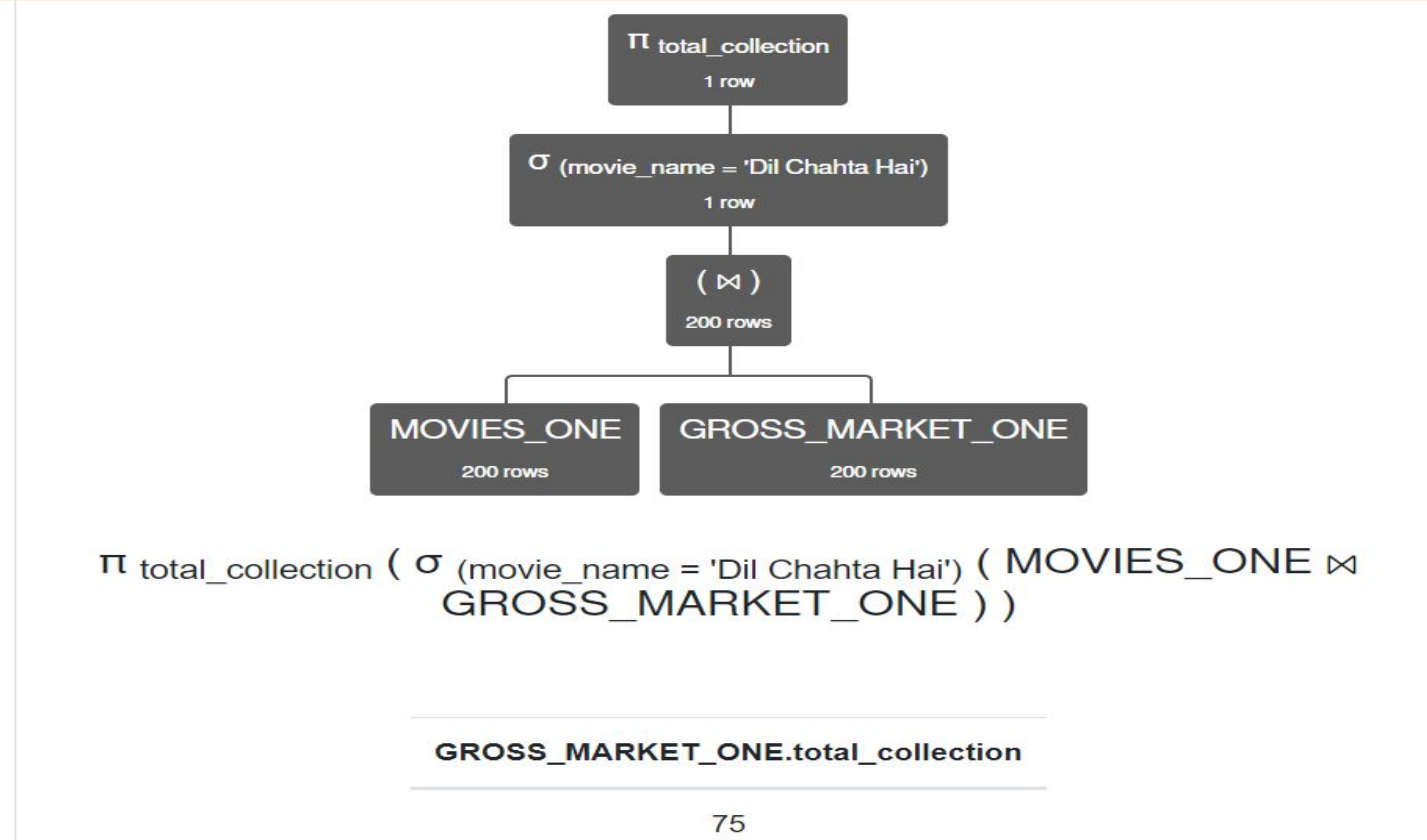
π no_of_movies γ ; COUNT(movie_name)→no_of_movies σ MOVIES_TWO.director_id = 3045 MOVIES_TWO

no_of_movies
1

Q. Finding the toal collection of the movie “dil chahta he”



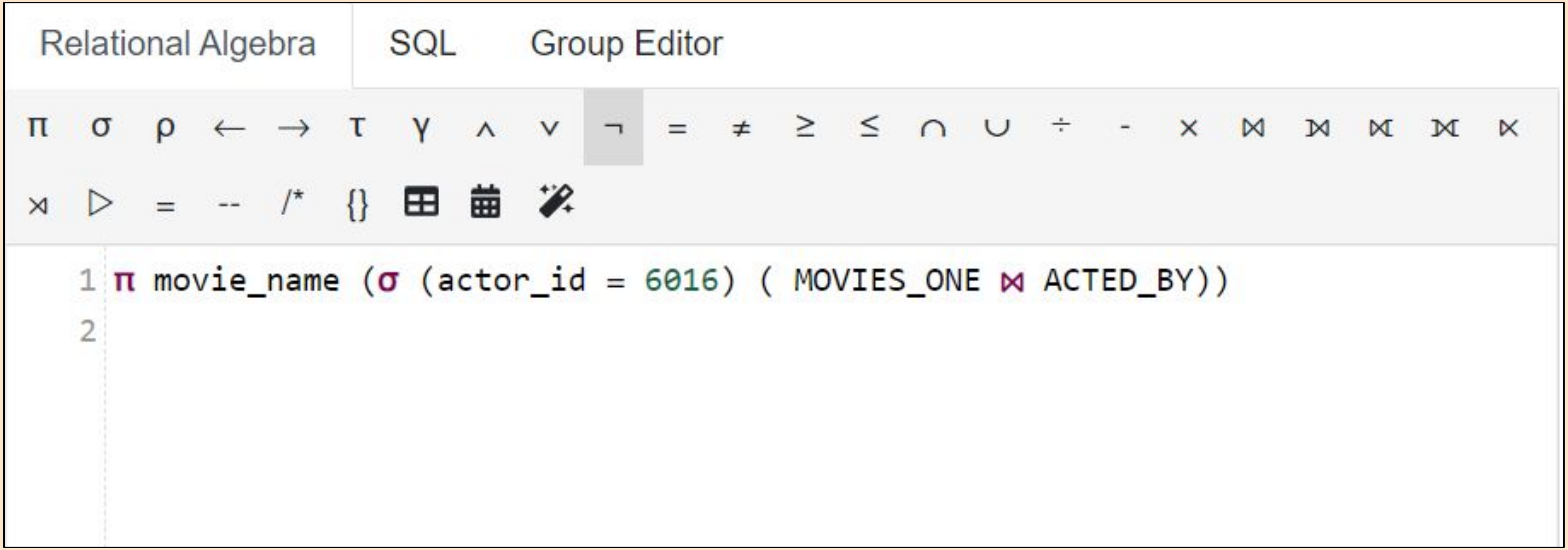
```
π total_collection (σ (movie_name = 'Dil Chahta Hai') ( MOVIES_ONE ⋈
GROSS_MARKET_ONE))
```



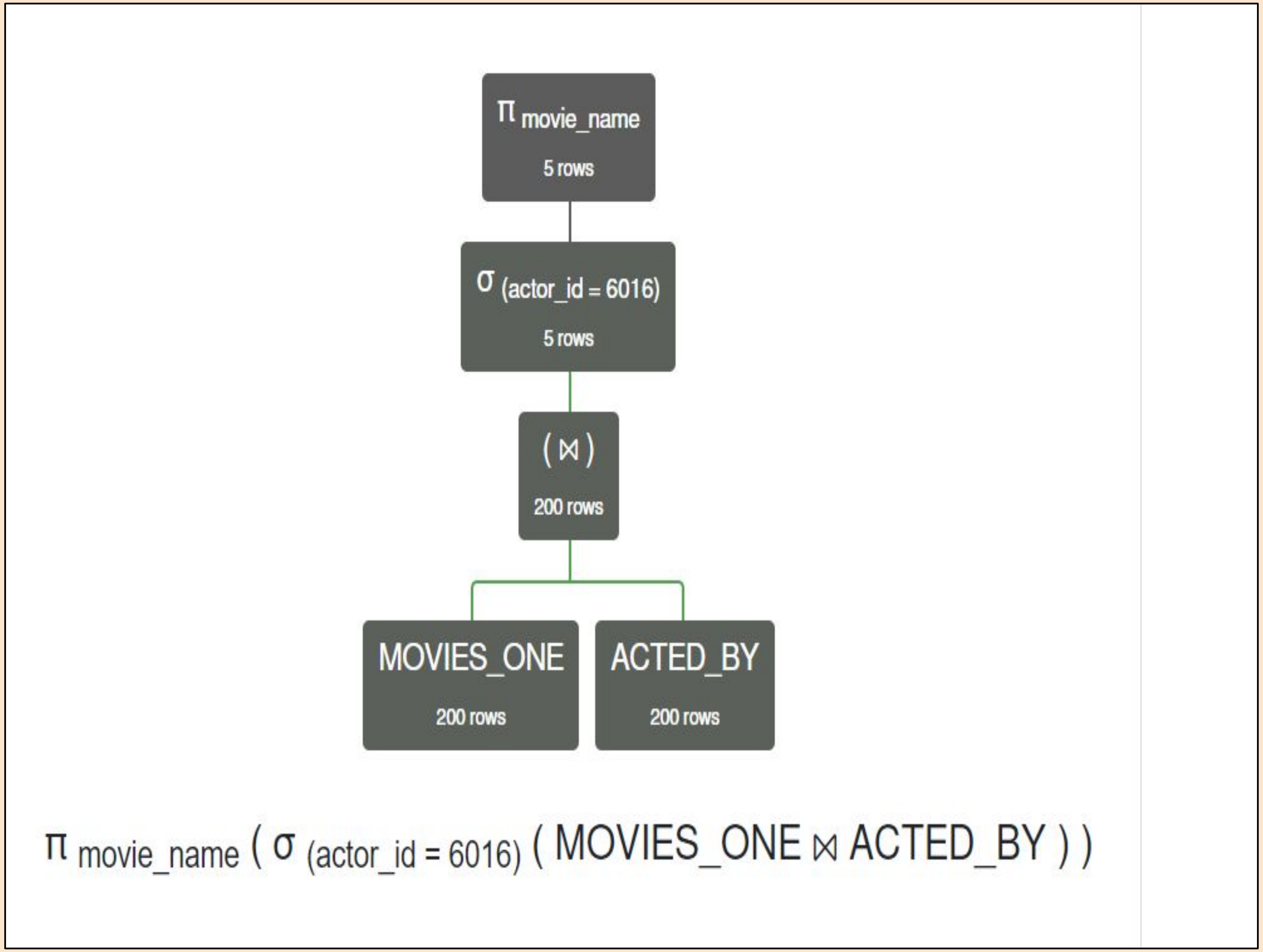
$\pi_{\text{total\_collection}}$ ($\sigma_{\text{(movie\_name = 'Dil Chahta Hai')}}$ (MOVIES_ONE ⋈ GROSS_MARKET_ONE))

GROSS_MARKET_ONE.total_collection
75

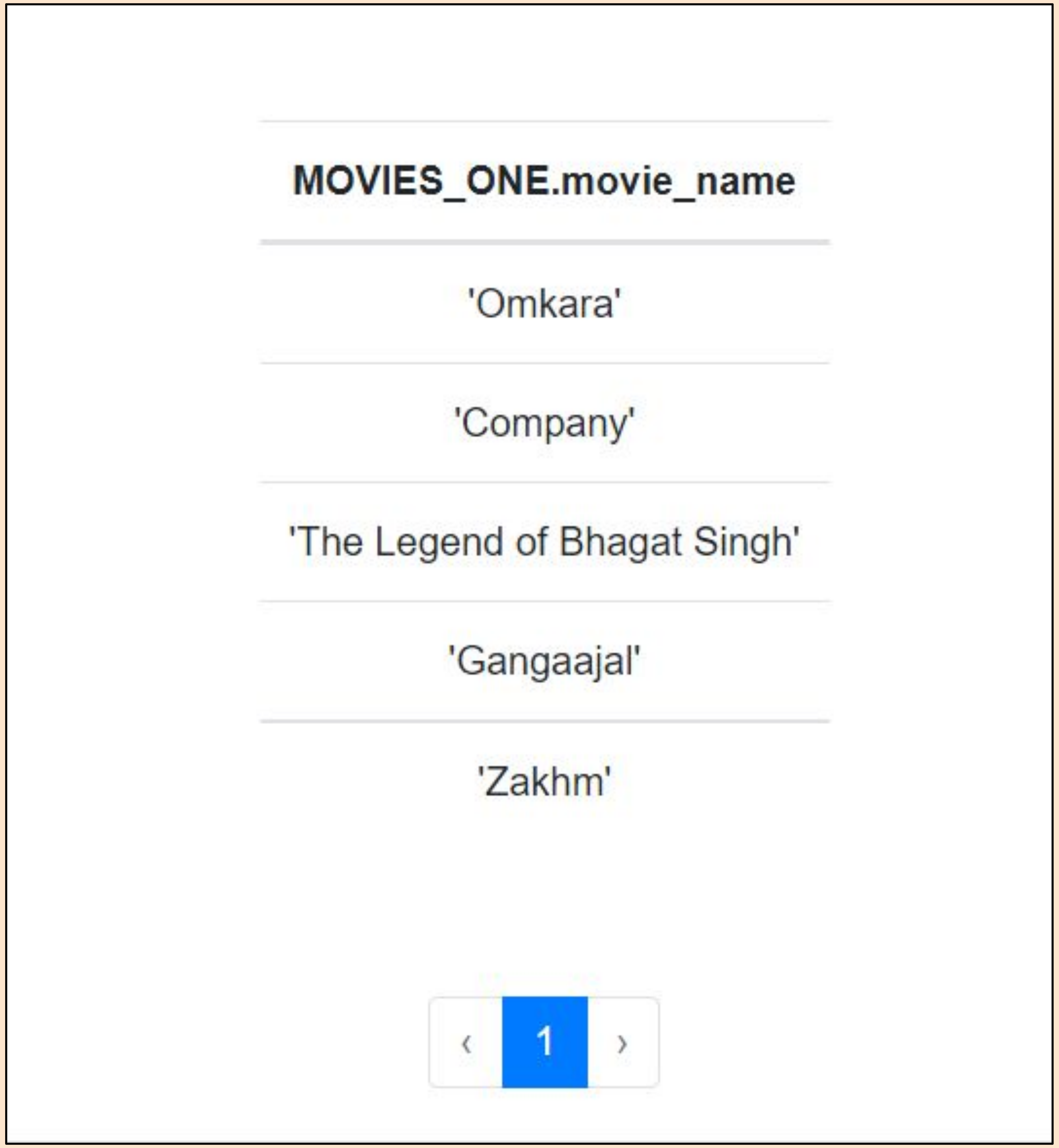
Q. Finding the names of the movies in which a particular actor worked on



```
π movie_name (σ (actor_id = 6016) ( MOVIES_ONE ⋈ ACTED_BY))
```

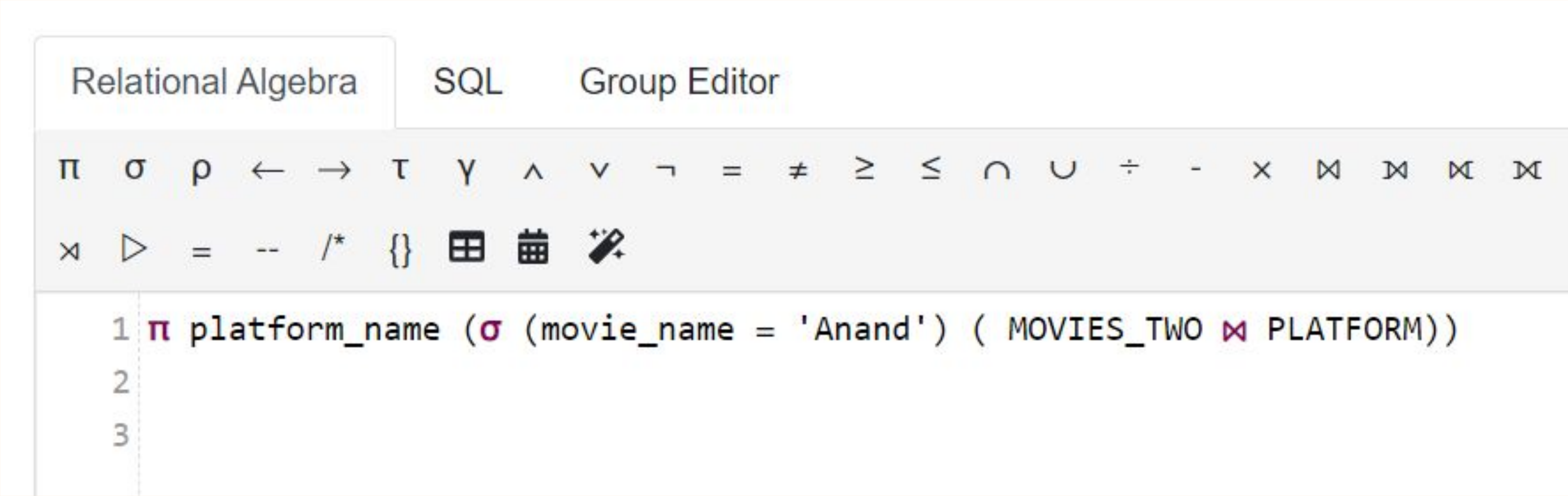


$\pi_{movie\_name} (\sigma_{(actor\_id = 6016)} (MOVIES\_ONE \bowtie ACTED\_BY))$

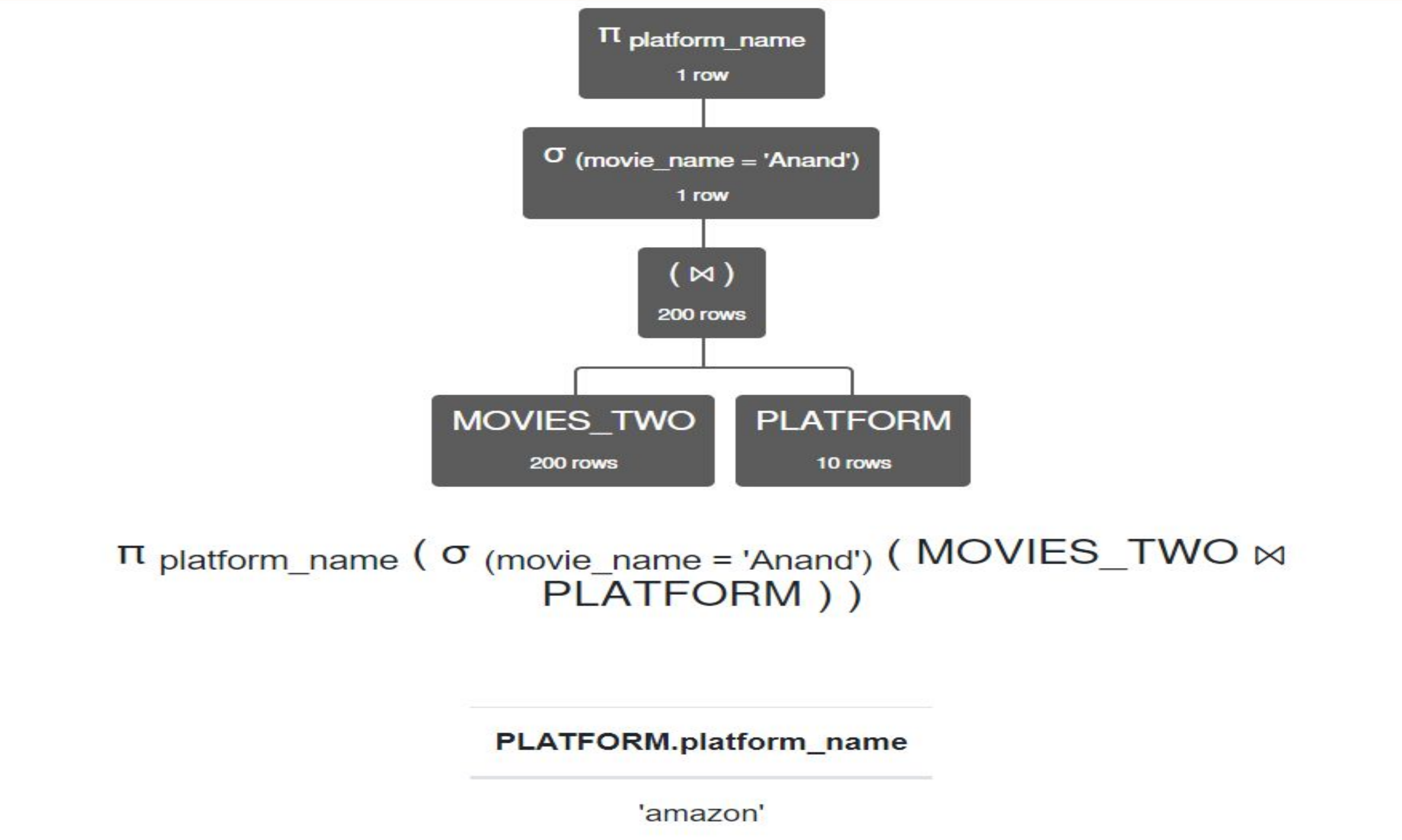


MOVIES_ONE.movie_name
'Omkara'
'Company'
'The Legend of Bhagat Singh'
'Gangaajal'
'Zakhm'

Q. Finding the platform name on which a particular movie is available



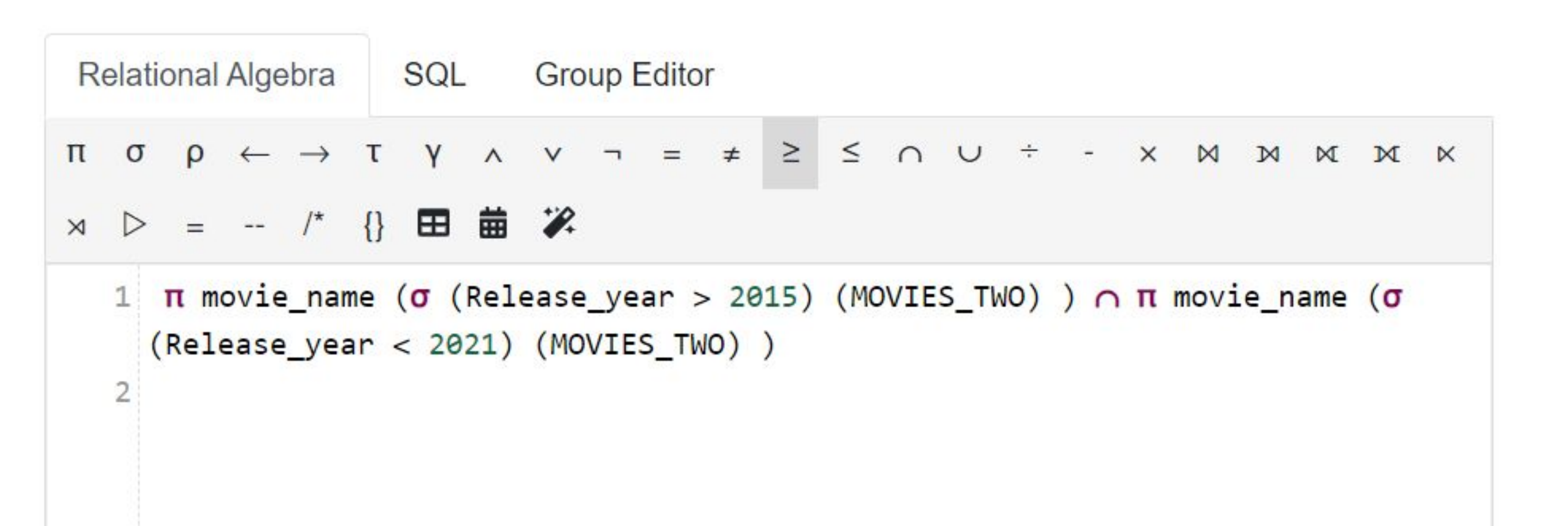
```
π platform_name (σ (movie_name = 'Anand') ( MOVIES_TWO ⋈ PLATFORM))
```



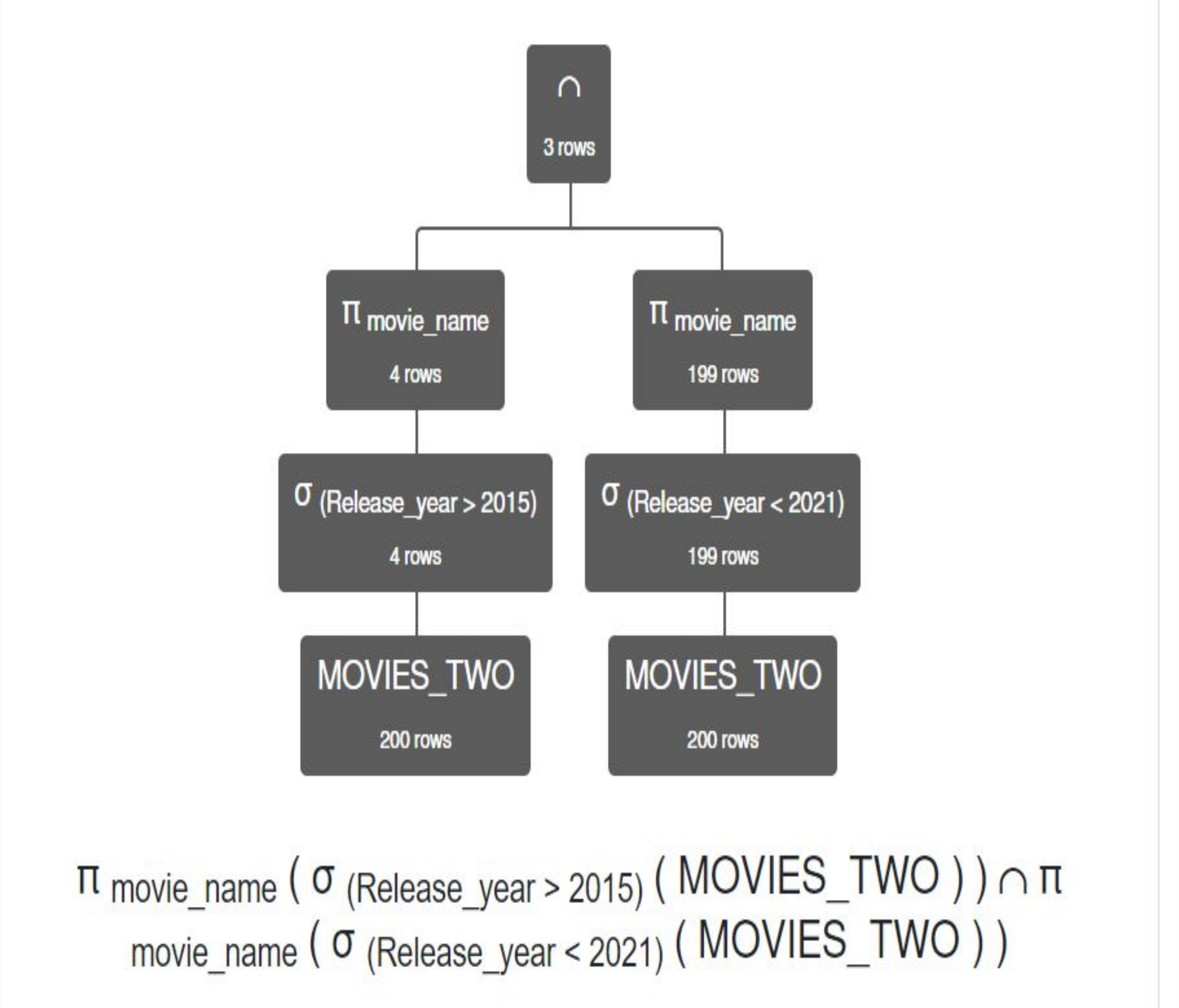
$$\pi_{\text{platform\_name}} (\sigma_{(\text{movie\_name} = \text{'Anand'})} (\text{MOVIES\_TWO} \bowtie \text{PLATFORM}))$$

PLATFORM.platform_name
'amazon'

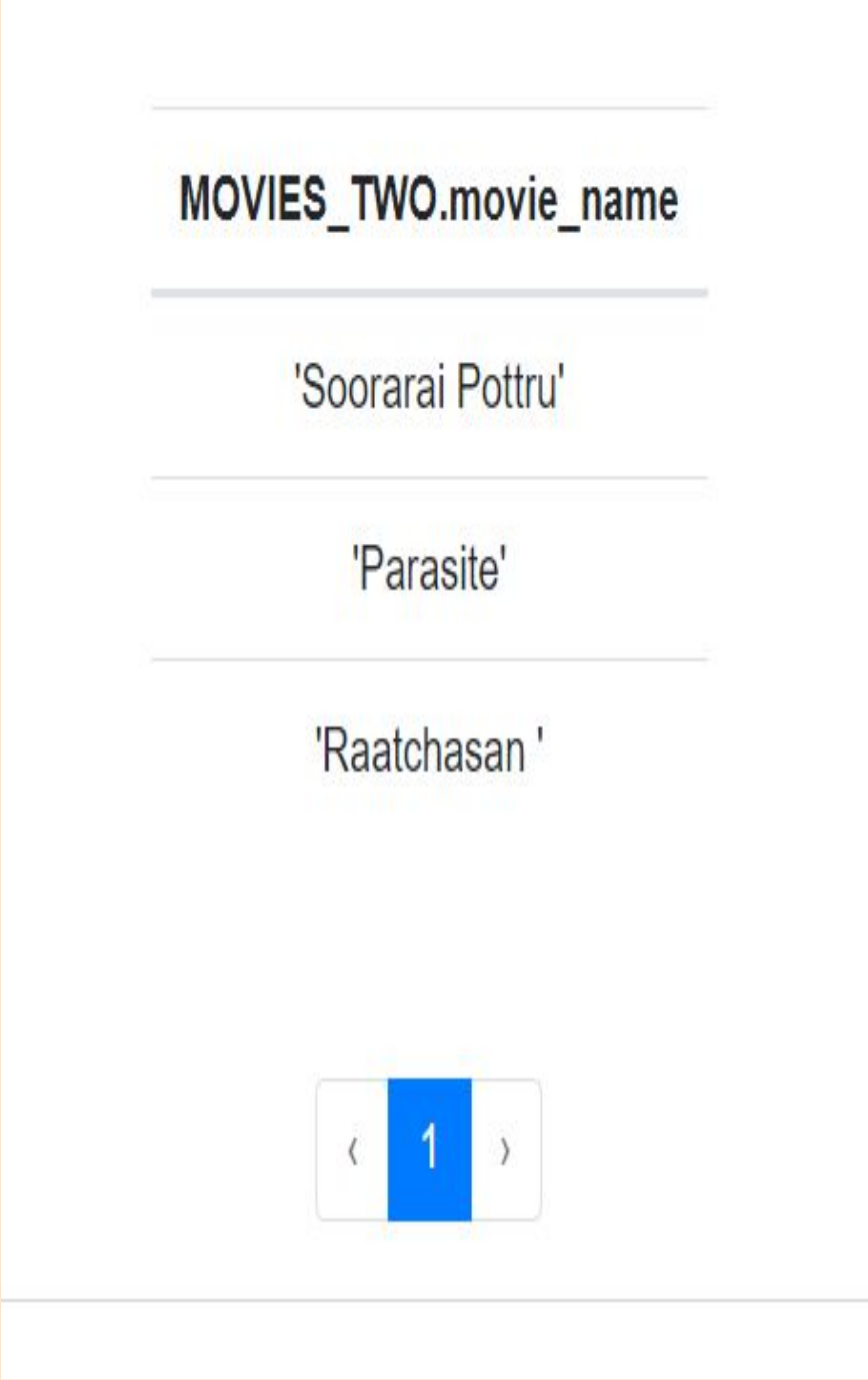
Q. Finding the name of the movies that were released after 2015 and before 2021



```
π movie_name (σ (Release_year > 2015) (MOVIES_TWO) ) ∩ π movie_name (σ (Release_year < 2021) (MOVIES_TWO) )
```

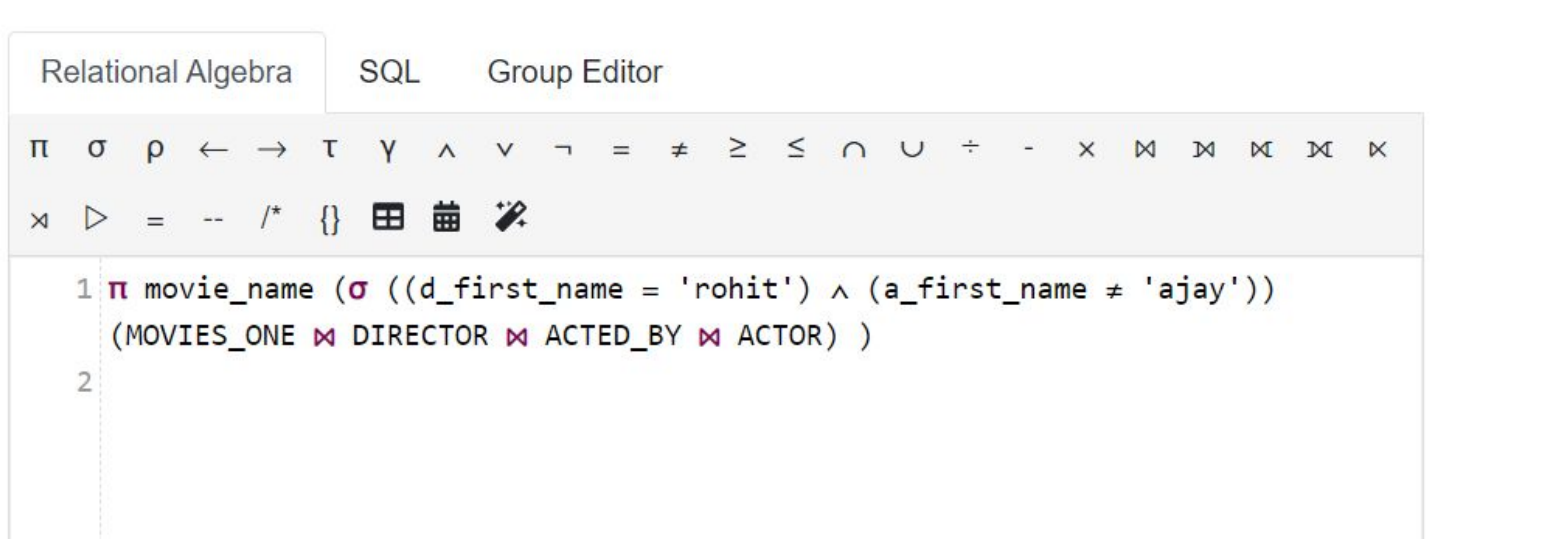


$\pi_{movie\_name}\ (\ \sigma_{(Release\_year > 2015)}\ (\ MOVIES\_TWO\ )\ )\ \cap\ \pi_{movie\_name}\ (\ \sigma_{(Release\_year < 2021)}\ (\ MOVIES\_TWO\ )\ )$

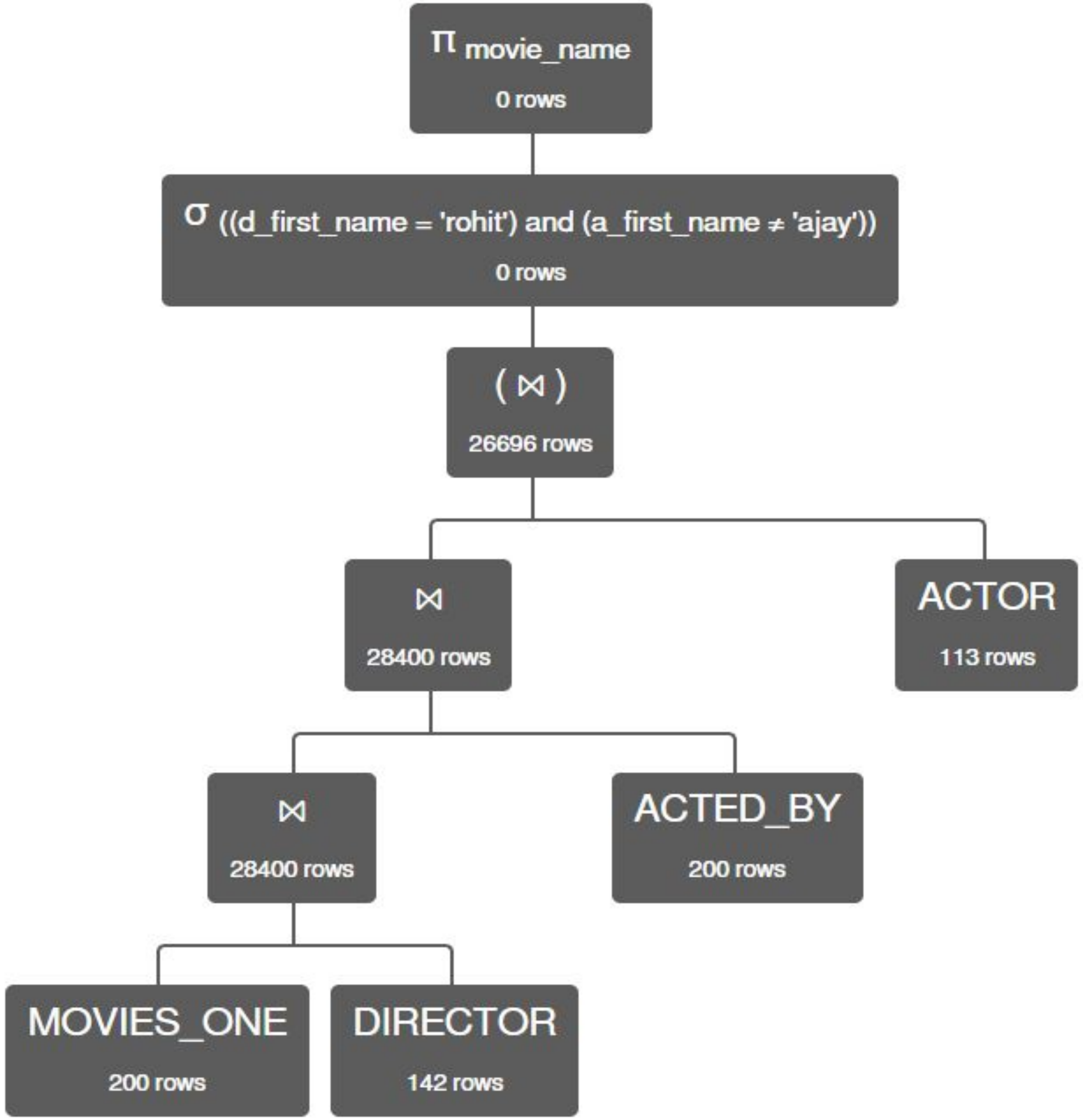


MOVIES_TWO.movie_name
'Soorarai Pottru'
'Parasite'
'Raatchasan '

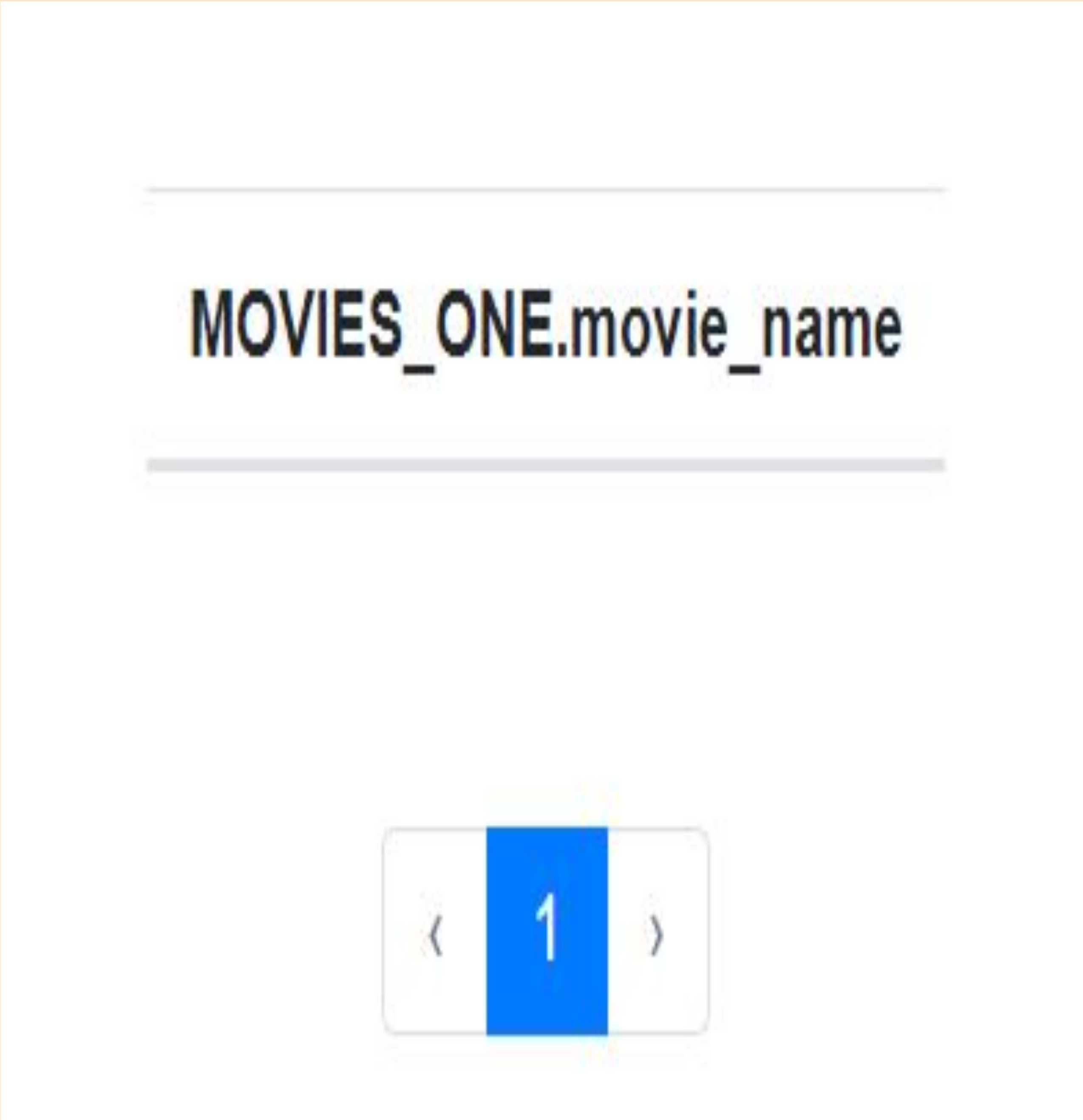
Q. Finding the names of the movies for which the director’s first name was’ rohit’ but the actor with first name ‘ajay’ was not a part of the movie.



```
π movie_name (σ ((d_first_name = 'rohit') ∧ (a_first_name ≠ 'ajay')) (MOVIES_ONE ⨝ DIRECTOR ⨝ ACTED_BY ⨝ ACTOR) )
```



$\pi_{\text{movie\_name}} (\sigma_{((\text{d\_first\_name} = \text{'rohit'}) \text{ and } (\text{a\_first\_name} \neq \text{'ajay'}))} (((\text{MOVIES\_ONE} \bowtie \text{DIRECTOR}) \bowtie \text{ACTED\_BY}) \bowtie \text{ACTOR}))$



MOVIES_ONE.movie_name