Advanced Data Engineering

**Relational And Non-Relational Database**

Priyank Dangwal

Shape

**GDDA 707**

**Lecturer: Mohammad Norouzifard**

**NZSE School of Tech**

Contents

[Assessment 1 - Advanced Data Engineering **Error! Bookmark not defined.**](#_Toc159885386)

[Introduction 3](#_Toc159885387)

[Task – 1: Data Modelling 3](#_Toc159885388)

[Data Modelling Techniques in Relational Database: 3](#_Toc159885389)

[Data Model Structure: 4](#_Toc159885390)

[Identification of anomalies: (Normalisation) 6](#_Toc159885391)

[Non – Relational Data Modelling: 8](#_Toc159885392)

[Documentation for non – Relational Data model. 14](#_Toc159885393)

[Relationships: 14](#_Toc159885394)

[Key Considerations: 14](#_Toc159885395)

[Task 2: Relational and Non-Relational Database Implementation 15](#_Toc159885396)

[Implement the designed data model into a relational database system: 15](#_Toc159885397)

[Customer Entity: 17](#_Toc159885398)

[Genre Entity: 19](#_Toc159885399)

[Movie Entity: 20](#_Toc159885400)

[Rating Entity: 21](#_Toc159885401)

[Implementation of Non – Relational database: 33](#_Toc159885402)

[Conclusion 41](#_Toc159885403)

## Introduction

In today's information systems, data engineering is essential since it helps businesses to use data for analytics, business intelligence, and well-informed decision-making. In this project, data modelling techniques are applied, and relational and non-relational database models are implemented, with an emphasis on the field of data engineering. Building solid data models that are adapted to business needs in a selected scenario is the main goal, with a focus on building databases that can store and process data effectively.   
  
In the sections that follow, we delve into the nuances of relational data modelling, utilizing methods like normalization and entity-relationship (ER) diagrams to thoroughly organize data. Entities such as Customers, Movies, Genres, Directors, and Ratings are all part of the relational model and are connected to one another through predefined relationships. We also explore SQL commands for data integrity checks, handling identity columns, and relationship establishment.   
  
The research goes beyond the conventional domain and explores non-relational data modelling, using MongoDB as an example of a NoSQL database. Non-relational databases are characterized by their flexibility and scalability, as demonstrated by the creation of collections that correspond to defined entities. We guide you through the difficulties that arise throughout the deployment process and offer insights into the benefits and things to keep in mind while using non-relational databases.   
  
With its practical investigation of the ever-evolving topic of data engineering, this project provides hands-on experience in the design and implementation of databases customized to meet business demands. This project seeks to provide people with a thorough understanding of data modelling principles and their practical applications in real-world scenarios by utilizing both relational and non-relational methodologies.

## Task – 1: Data Modelling

## Data Modelling Techniques in Relational Database:

This scenario involves customers interacting with movies, providing ratings, and directors directing movies:

In this movie platform customers can explore movies, rate them, and directors contribute to the creation of movies.

A diagram of a movie data

Description automatically generated

## Data Model Structure:

**Entities and Attributes:**

1. Customer:

* CustomerID (Primary key)
* Name
* Email
* RegistrationDate

Relationship: 1 to N relationship with movie (One customer can watch multiple movie)

1. Movie:

* MovieID (Primary key)
* MovieTitle
* ReleaseDate
* CustomerID (Foreign key)
* GenreID (Foreign key)
* DirectorID (Foreign key)

Relationship: M to 1 relationship (Many movies can be associated with 1 customer)

1. Genre:

* GenreID (Primary key)
* GenreName
* Description
* MovieID (Foreign key)

Relationship: M to N relationship with movie (Genres can be associated with multiple Movies, and each Movie can belong to multiple Genre)

1. Rating:

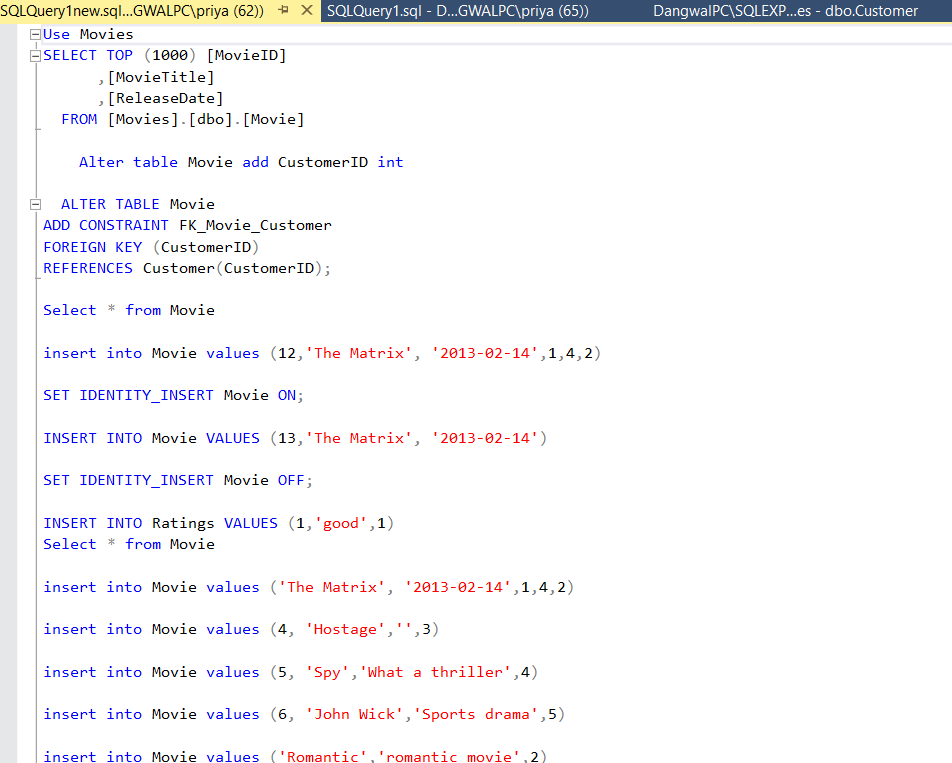
* RatingID (Primary key)
* MovieID (Foreign key)
* Review
* CustomerID (Foreign key)

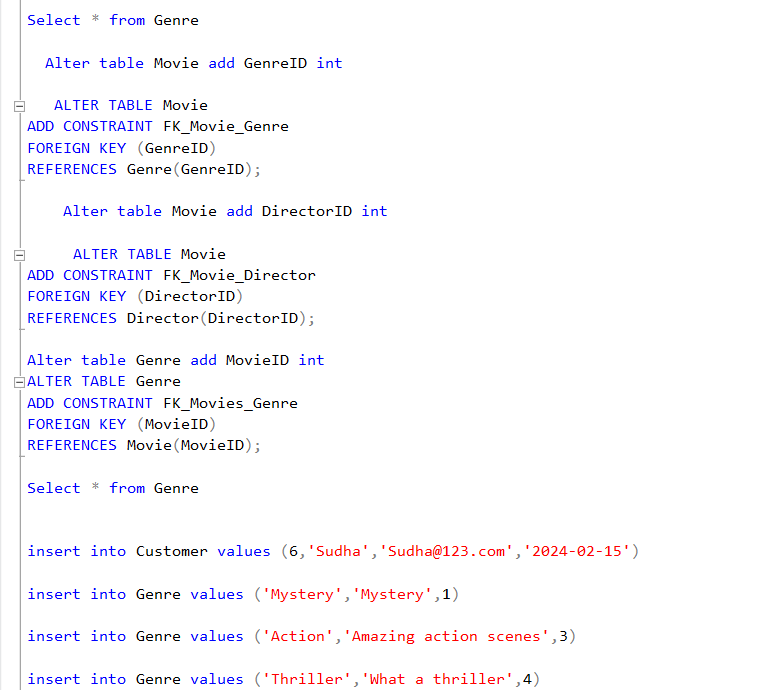
Relationship: M to 1 relation with Customer (Many ratings can be provided by 1 Customer, and many ratings can be provided for 1 Movie)

1. Director:

* DirectorID (Primary key)
* DirectorName

Relationship: 1 to 1 relationship with Movie (Only 1 movie can be directed by 1 director at one time)





## Identification of anomalies: (Normalisation)

**Creating Movie Genre table:**

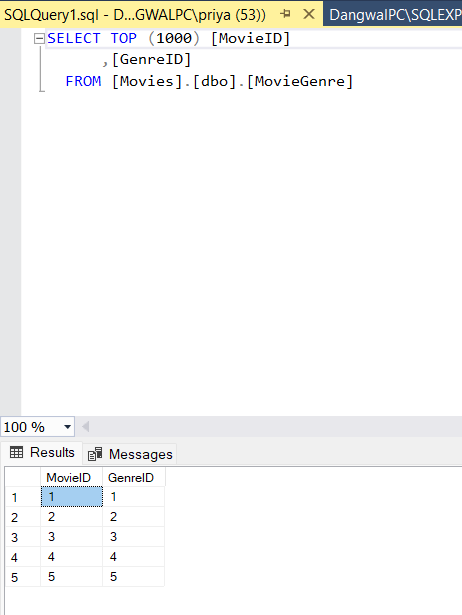
This statement creates the Movie Genre table with MovieID and GenreID as its primary key, and it establishes foreign key relationships with the Movie and Genre tables. This is correct and aligns with the creation of a junction table for a many-to-many relationship.

**Insert Data into Movie Genre Table:**

This statement inserts sample data into the Movie Genre table, representing the relationships between movies and genres. This is appropriate for populating the junction table.

**Provide existing Data into MovieGenre Table:**

This statement attempts to populate the MovieGenre table with existing data from the Movie table.

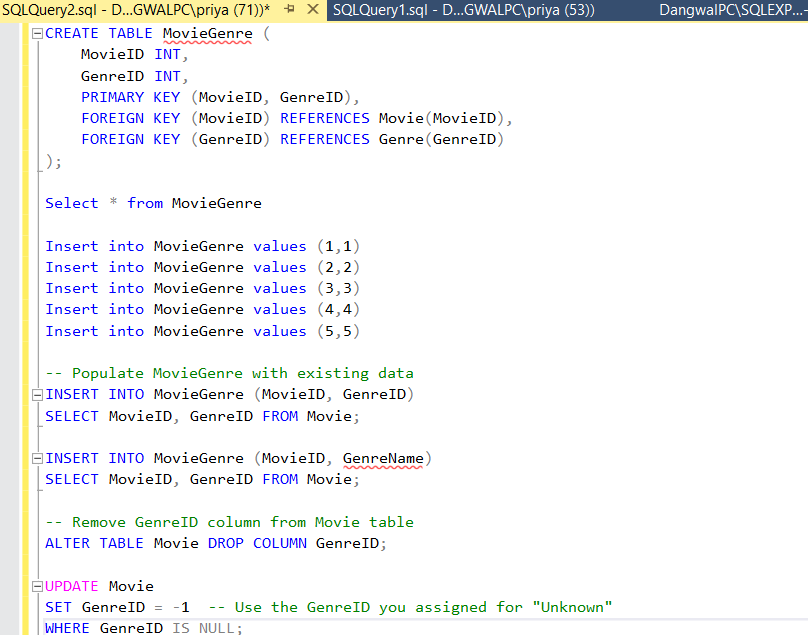


**Remove GenreId table from Movie table:**

This statement removes the GenreID column from Movie table.

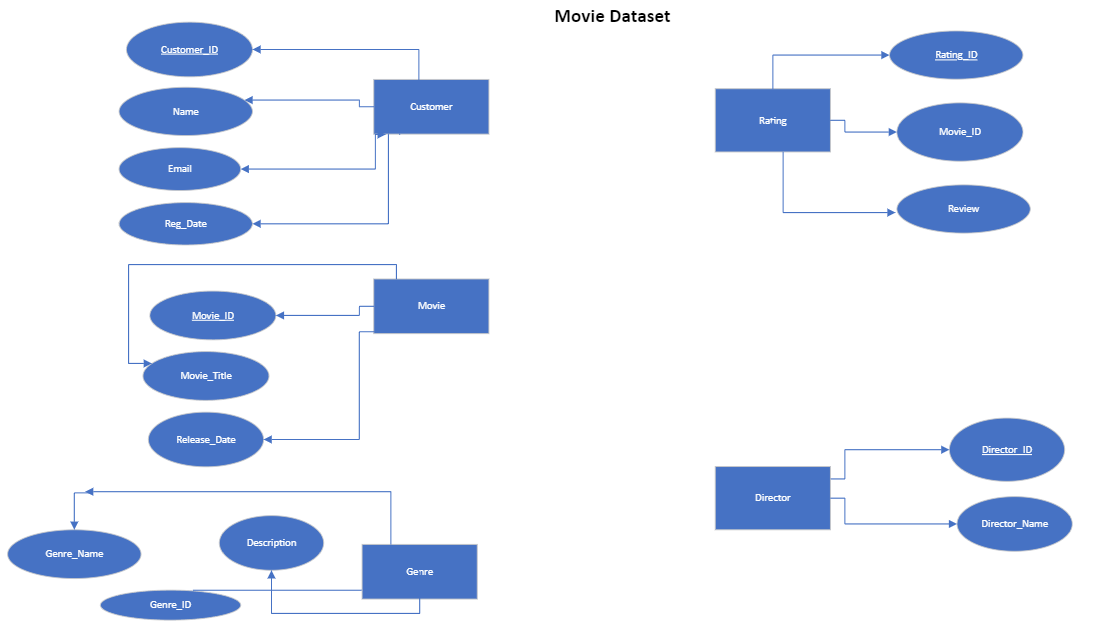
**Update Movie table to handle null values:**

This statement attempts to update the Movie table by setting GenreID to -1 where it is null.



## Non – Relational Data Modelling:

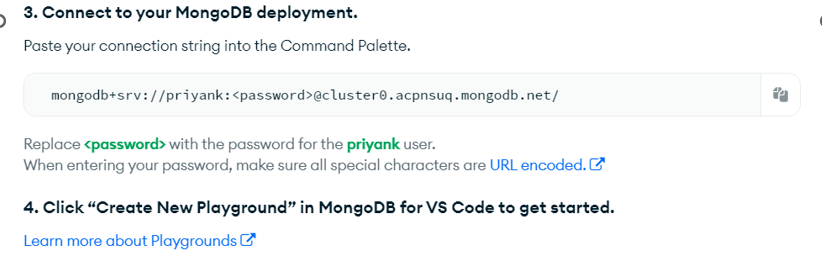
Scenario: Five entities make up the data model of a NoSQL-powered movie rating platform: Customer, Movie, Genre, Rating, and Director. User data, including customerId, name, email, and registrationDate, is stored in a customer document. The Movie document has rooted documents for the Director and Genre in addition to information like movieId, title, and releaseDate. Individual documents having the attributes genreId, name, and description are referred to as genre. The Rating document, which includes the ratingId, customerId, movieId, and ratingValue, is where ratings are recorded. Information including directorId, name, are contained in the director document. The NoSQL concept facilitates scalability and flexibility, enabling effective movie-related information retrieval while taking into account the dynamic nature of user ratings and movie facts. A comprehensive but decentralized picture of the associated entities is ensured through the use of rooted documents and references.

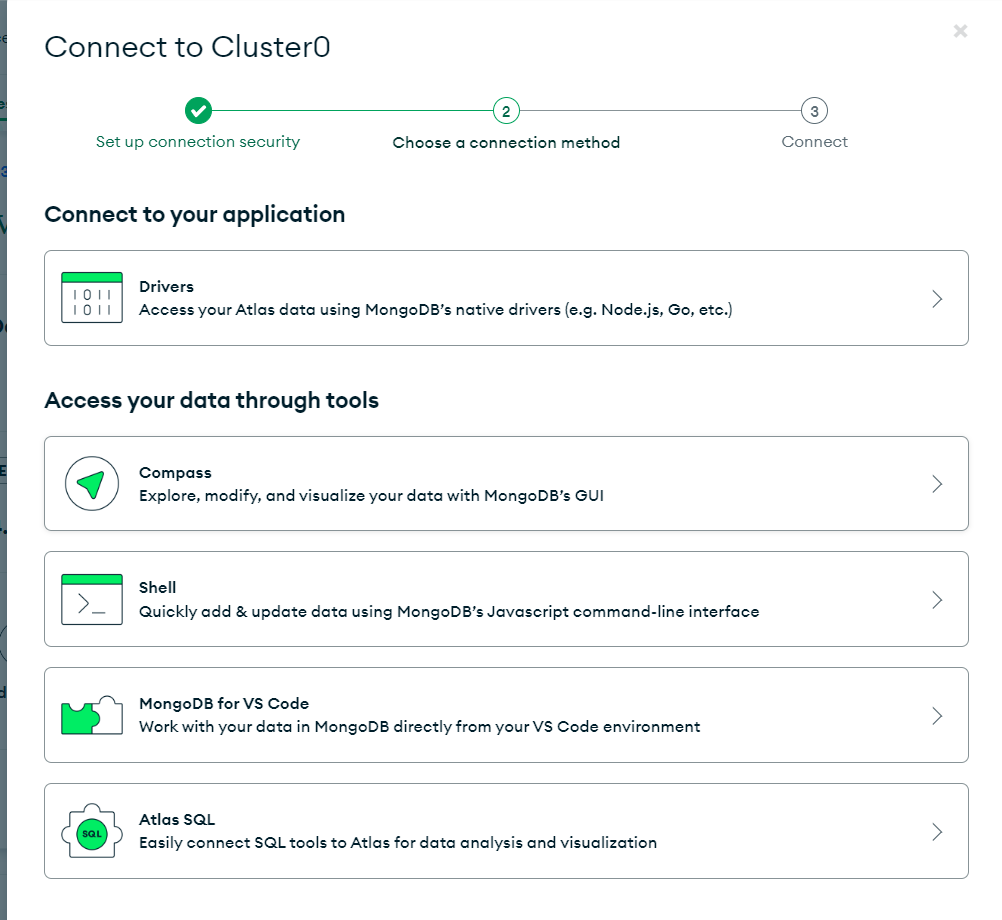


To employ non - relational technique for above scenario I am using MongoDB Platform which can implement the designed data.

At first, I have connected to Cluster0. Then I am using MongoDB for VS Code and connecting it.

Below is the visualisation to connect.





A screenshot of a computer

Description automatically generated



Screen Shot of the Customer Entity in VS code



Screenshot for Movies Entity in VS code



Screenshot for Genre Entity in VS code

A screen shot of a computer

Description automatically generated

Screenshot for Rating Entity in VS code

A screen shot of a computer

Description automatically generated

Screenshot for Director Entity in VS code

# Documentation for non – Relational Data model.

**Entities:**

1. Customer:

Attributes: customerId (ObjectId), name (string), email (string), registration Date (ISODate).

Relationships: Ratings (embedded array).

2. Movie:

Attributes: movieId (ObjectId), title (string), release Date (ISODate).

Relationships: Genre (embedded document), Director (embedded document), Ratings (referenced array).

3. Genre:

Attributes: GenreId (ObjectId), genre name (string), description (string).

Relationships: Movies (referenced array).

4. Rating:

Attributes: RatingId (ObjectId), movieId (ObjectId), review (int).

5. Director:

Attributes: directorId (ObjectId), name (string).

Relationships: Movies (referenced array).

## Relationships:

Customer to Ratings: One-to-Many (embedded array).

Movie to Genre: One-to-One (embedded document).

Movie to Director: One-to-One (embedded document).

Movie to Ratings: One-to-Many (referenced array).

Genre to Movies: One-to-Many (referenced array).

Director to Movies: One-to-Many (referenced array).

## Key Considerations:

All entities use ObjectId as their primary key. References and embedded documents are used to manage relationships. The data model permits flexible changes while guaranteeing effective information retrieval.   
The entities, properties, and relationships inside the non-relational data model for the movie rating platform are clearly outlined in this structured documentation.

A screenshot of a computer program

Description automatically generated

As shown in above screenshot in my NoSQL database scenario, while using MongoDB, relationship between entities has been represented by embedding technique.

Embedded Ratings in Customer document.

# Task 2: Relational and Non-Relational Database Implementation

## Implement the designed data model into a relational database system:

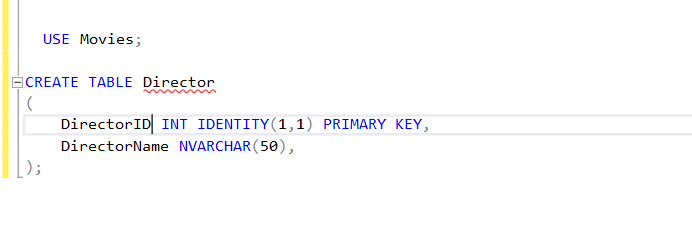
1. **Below are the tables for 5 entities.**

A screenshot of a computer

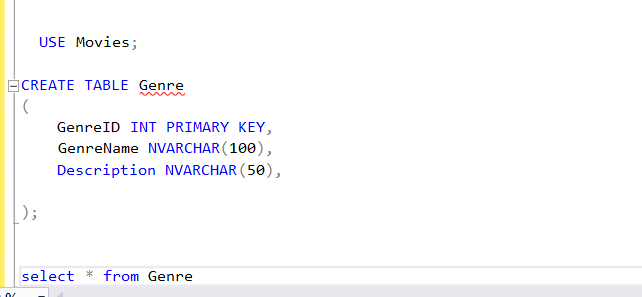
Description automatically generated

A screenshot of a computer

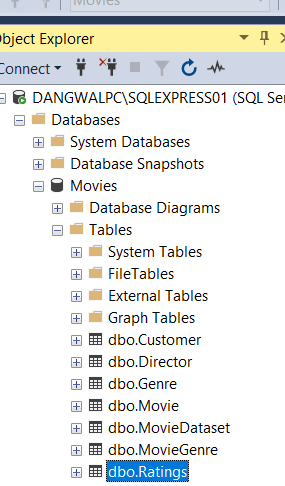
Description automatically generated







1. **Below are the sample records for each entity table.**

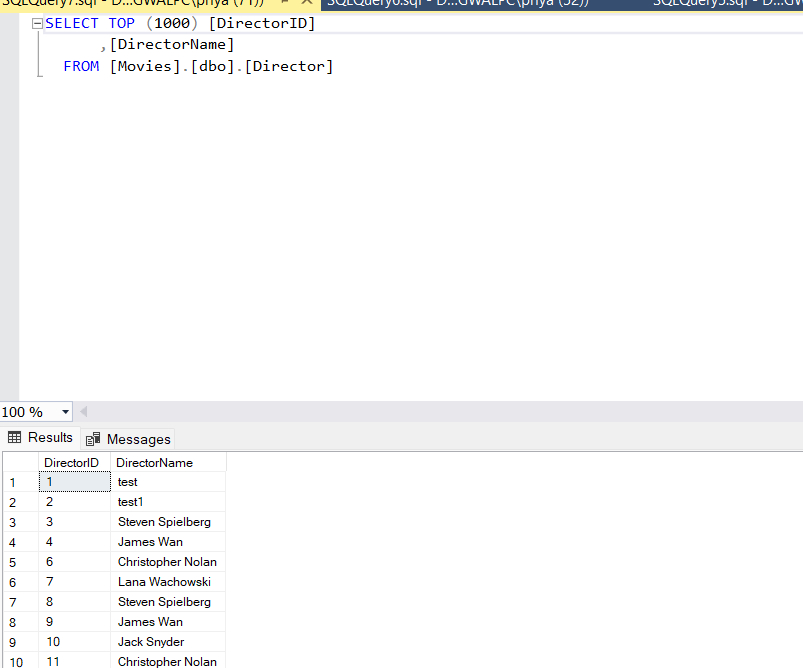


## Customer Entity:

A screenshot of a computer

Description automatically generated

Director Entity:



## Genre Entity:

A screenshot of a computer

Description automatically generated

## Movie Entity:

A screenshot of a computer

Description automatically generated

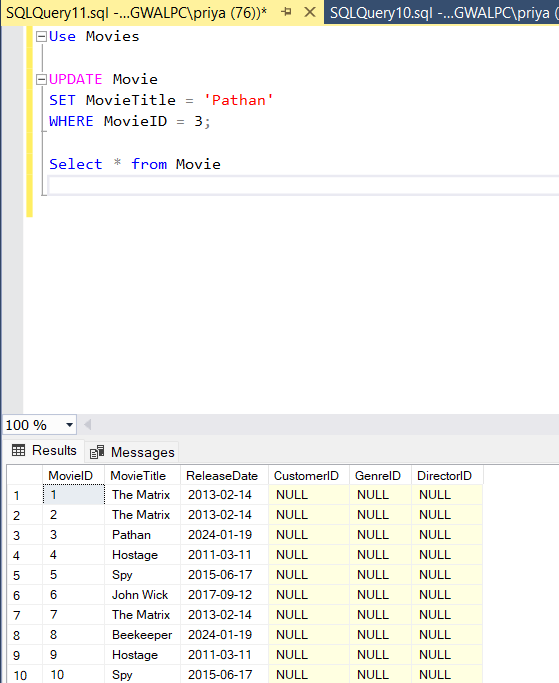
## Rating Entity:

A screenshot of a computer

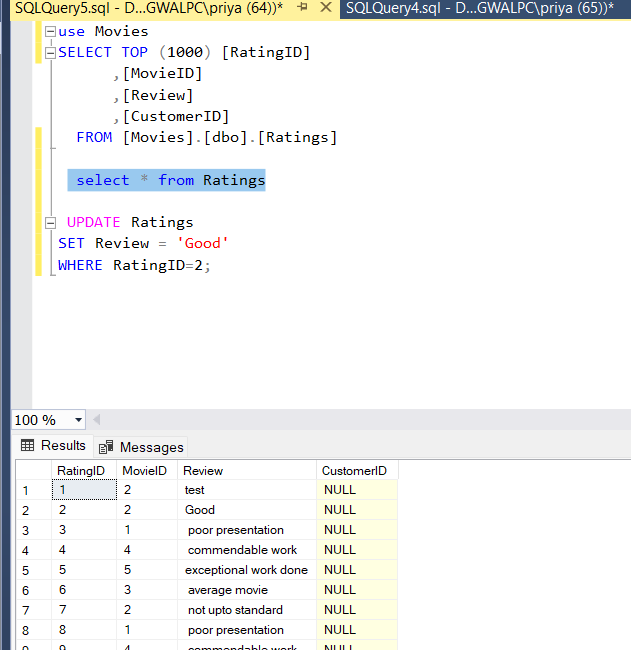
Description automatically generated

1. **Update queries:**

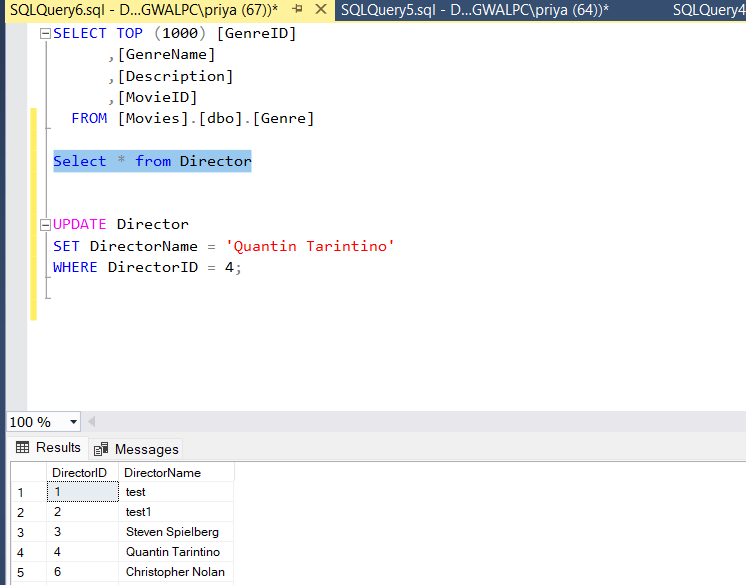
While performing update function in movie table for Title of the movie has been changed from Beekeeper to Pathan.



While updating Ratings table the review for ratingID 2 has been changed from test to good.



Now for third query while updating director Table in DirectorID 4 we have changed the name from James Wan to Quantin Tarintino.



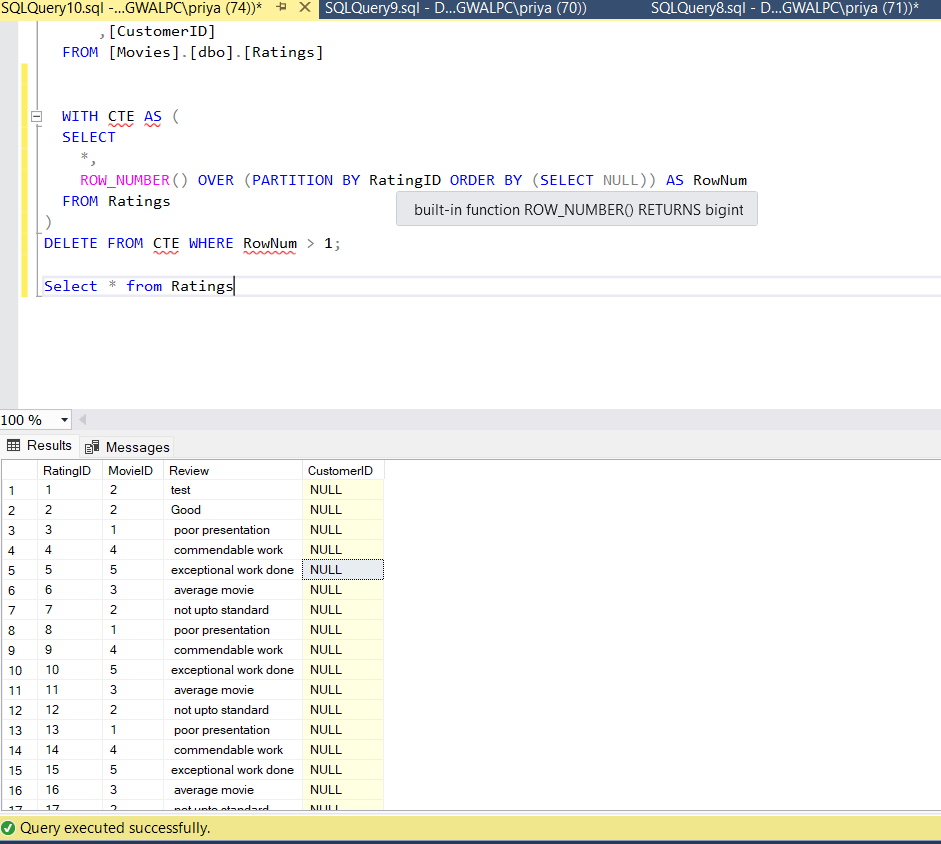
1. **Removal of duplicate documents in data columns:**

To remove duplicates records from table, I am using Row Number() window function along with Common Table Expression(CTE). The following method will show us how to remove any duplicate from Table ‘Ratings’.

While using the above method there’s no duplicate record in Ratings table thus, it doesn’t show any difference.

Now I am using one more table named Customers to see if there’s any duplicates in the table.

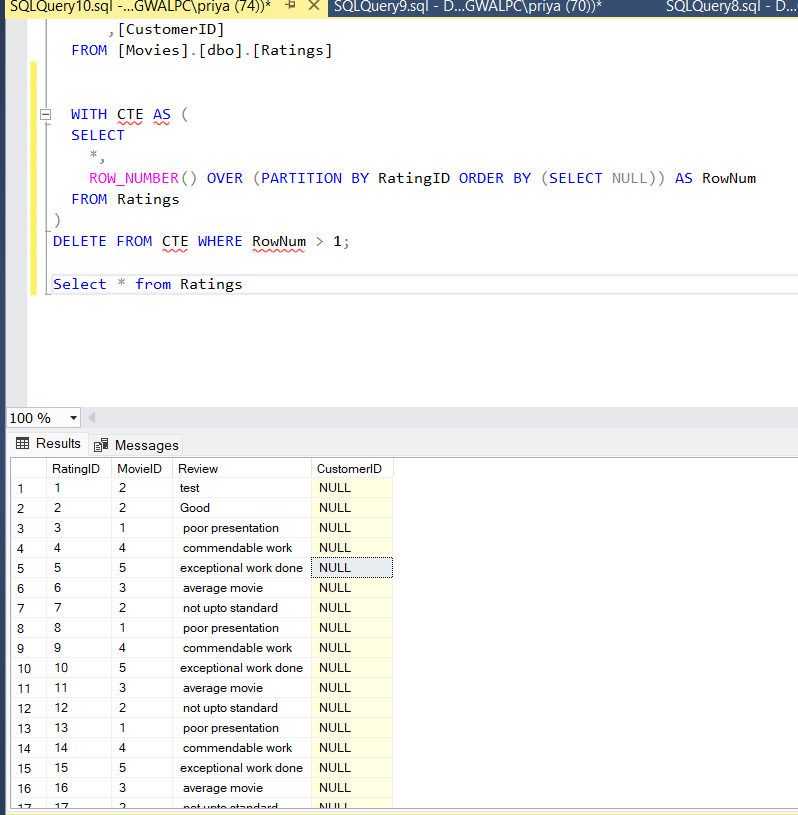
While using the remove duplicate method in Customer table, there’s no difference in the original table and after.



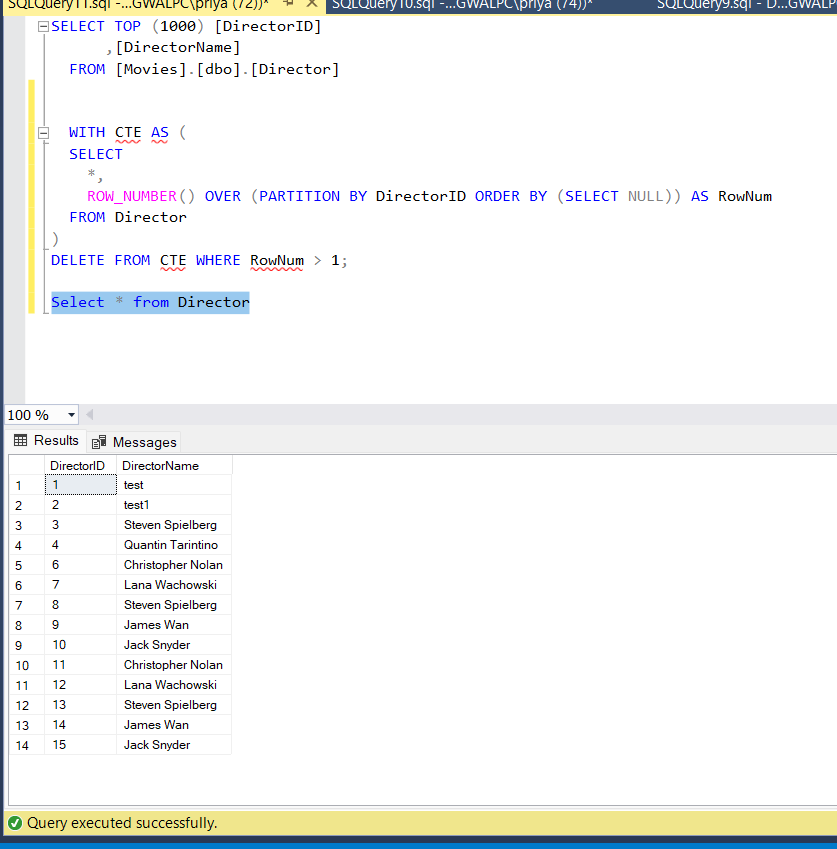
A screenshot of a computer

Description automatically generated

**For Genre table:**

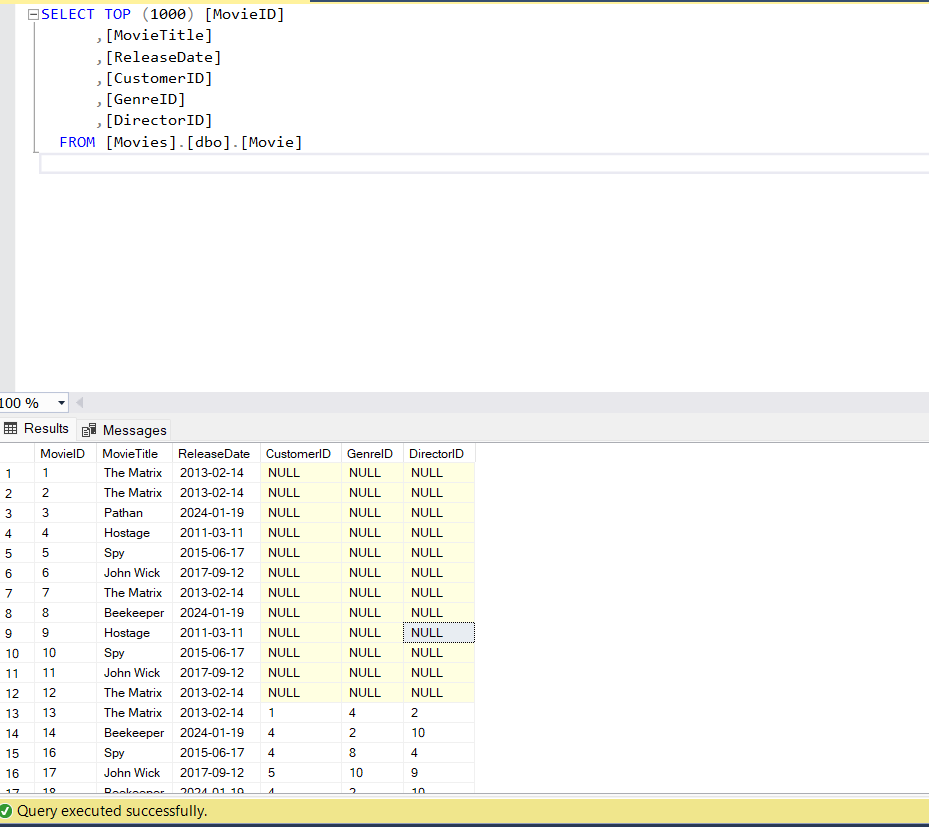


**For Director table:**

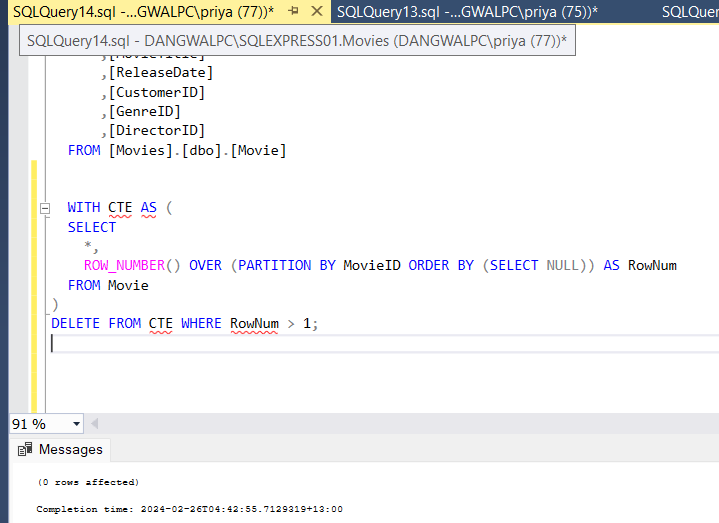


**For Movie table:**

**Original table:**



**After using the remove function:**



As the Screenshot shows 0 rows affected in completion time, means there’s no duplicate documents.

1. **Performing table joins:**

**In my first screenshot:**

While performing table joins to retrieve records for data aggregation, I am joining Movies, Genre and Ratings table to perform inner join. The main feature of my table is MovieId as it will create a relationship for Movie, genre and ratings. (Joining of Data)

The chosen column in result is MovieId and Movie title from movie table, Genre Name from genre table and Review from rating table. (Chosen Column)

As I got multiple rows with same MovieID in Genre and Rating table, each combination producing separate row for output. (for Duplicates)

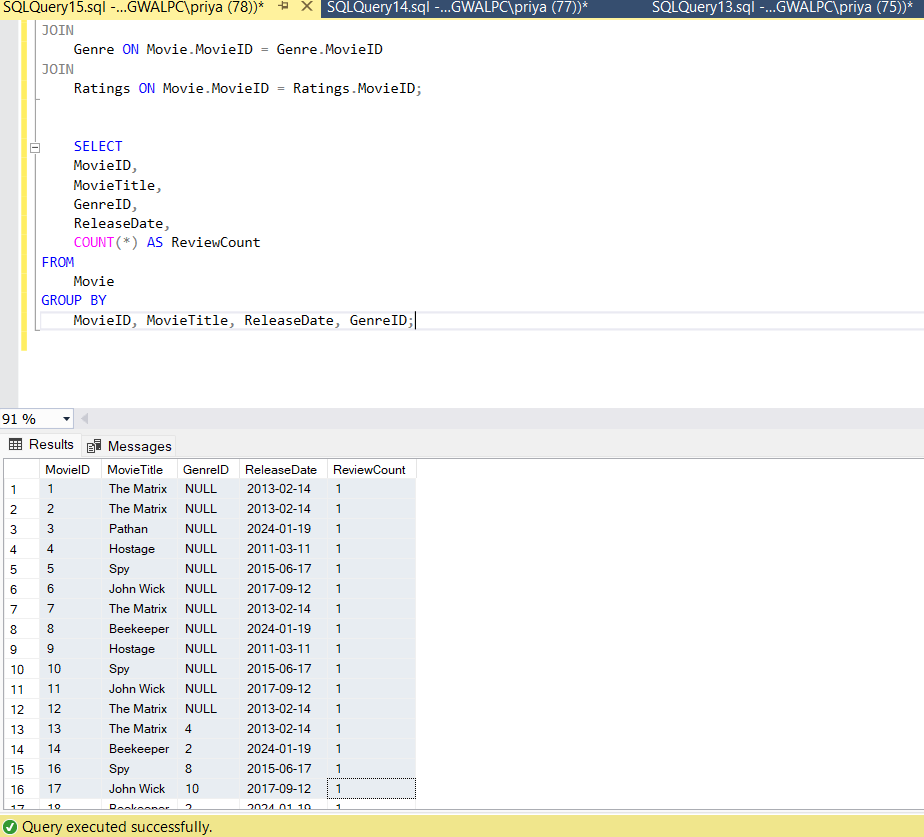
**For second screenshot:**

In this to get aggregated data I am using Groupby method. From Movie table for columns MovieID, Movie Title, Release Date and GenreID, to count the number of reviews for each.

In Groupby clause I am using Count (\*) function as review count to count the number of reviews in each group.

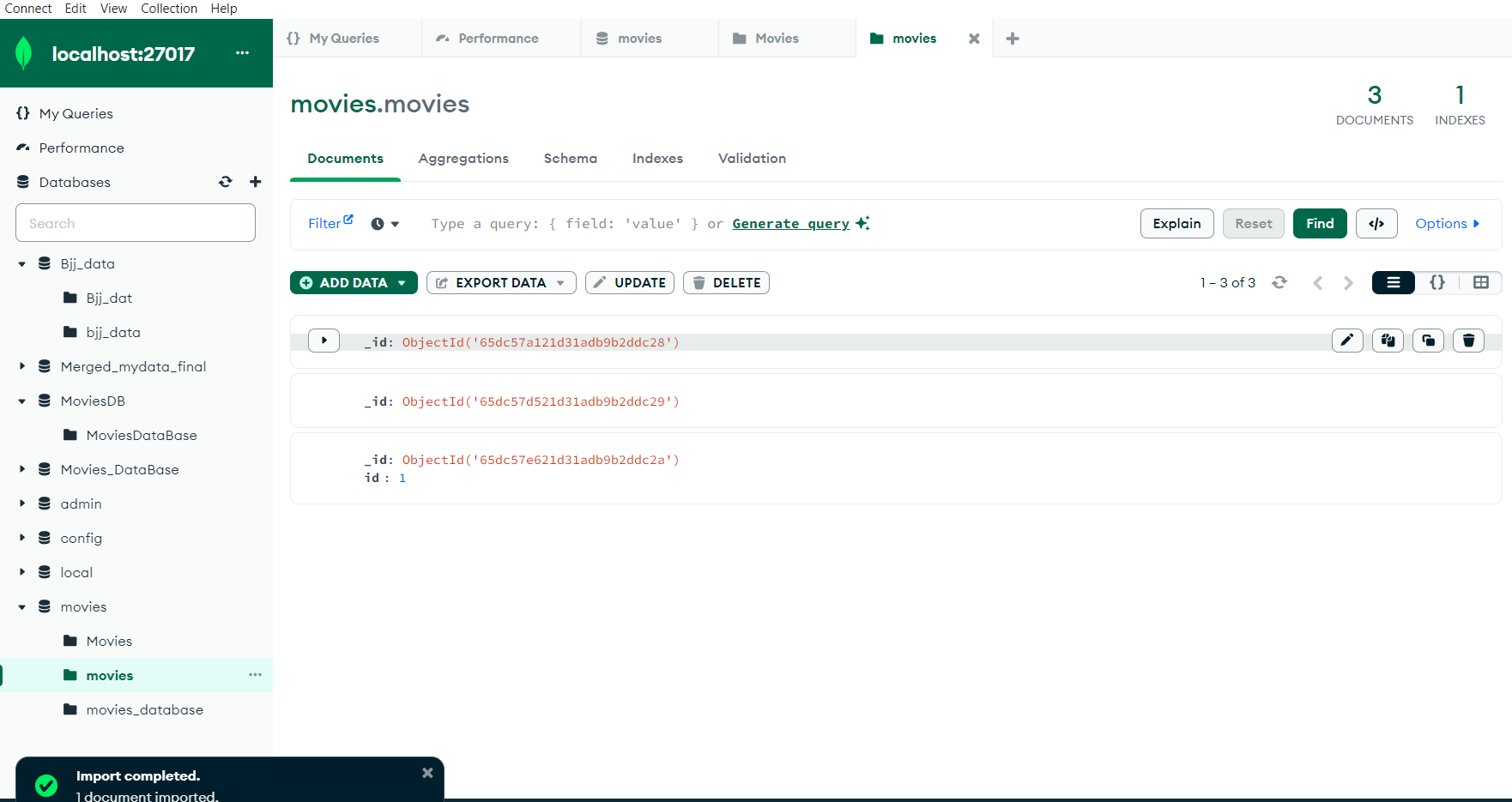
A screenshot of a computer

Description automatically generated



## Implementation of Non – Relational database:

1. **Creating Collection: Movies**

****

Five collections have been created –

* Customer collection
* Director collection
* Genre collection
* Ratings collection
* Movies collection

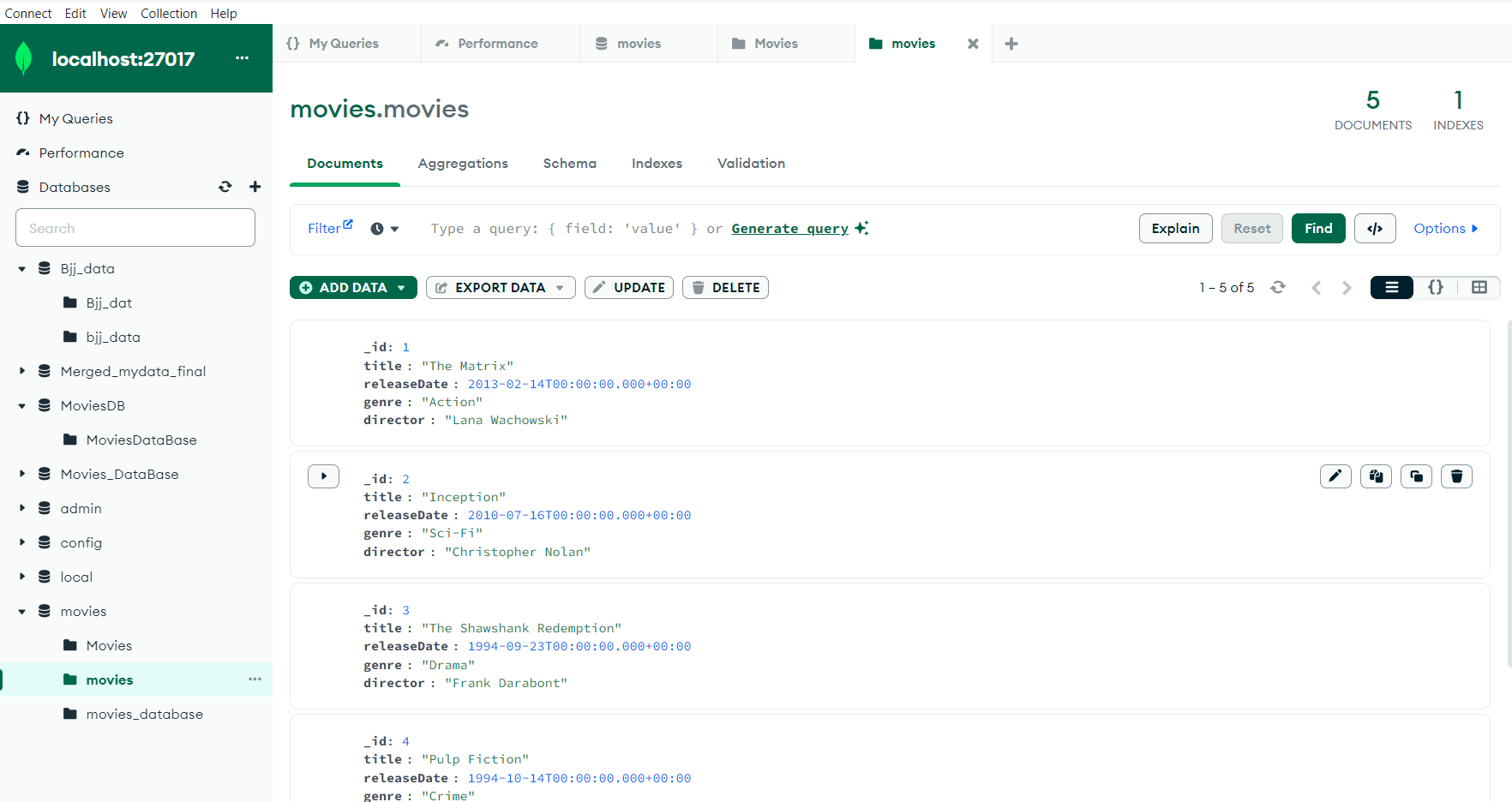
1. **Inserting 5 samples in each collections**

Command to insert 5 entities in Movies Collection

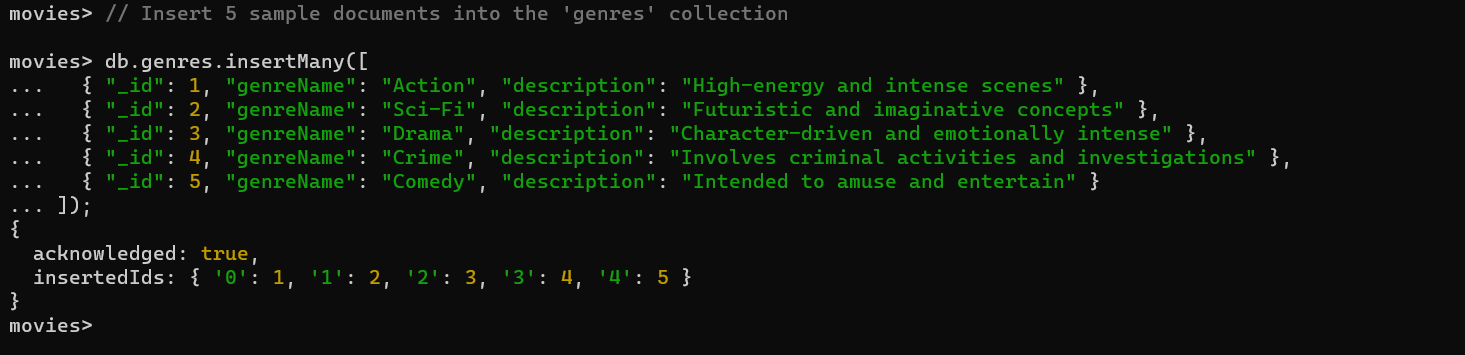
A computer screen with green and white text

Description automatically generated

Result showing the added samples



**Command to insert 5 entities in Genre Collection**



Out put of the Genre Collection sample

A screenshot of a computer

Description automatically generated

Command to insert 5 entities in Ratings Collection

A screenshot of a computer program

Description automatically generated

Output of the Ratings Collection

A screenshot of a computer

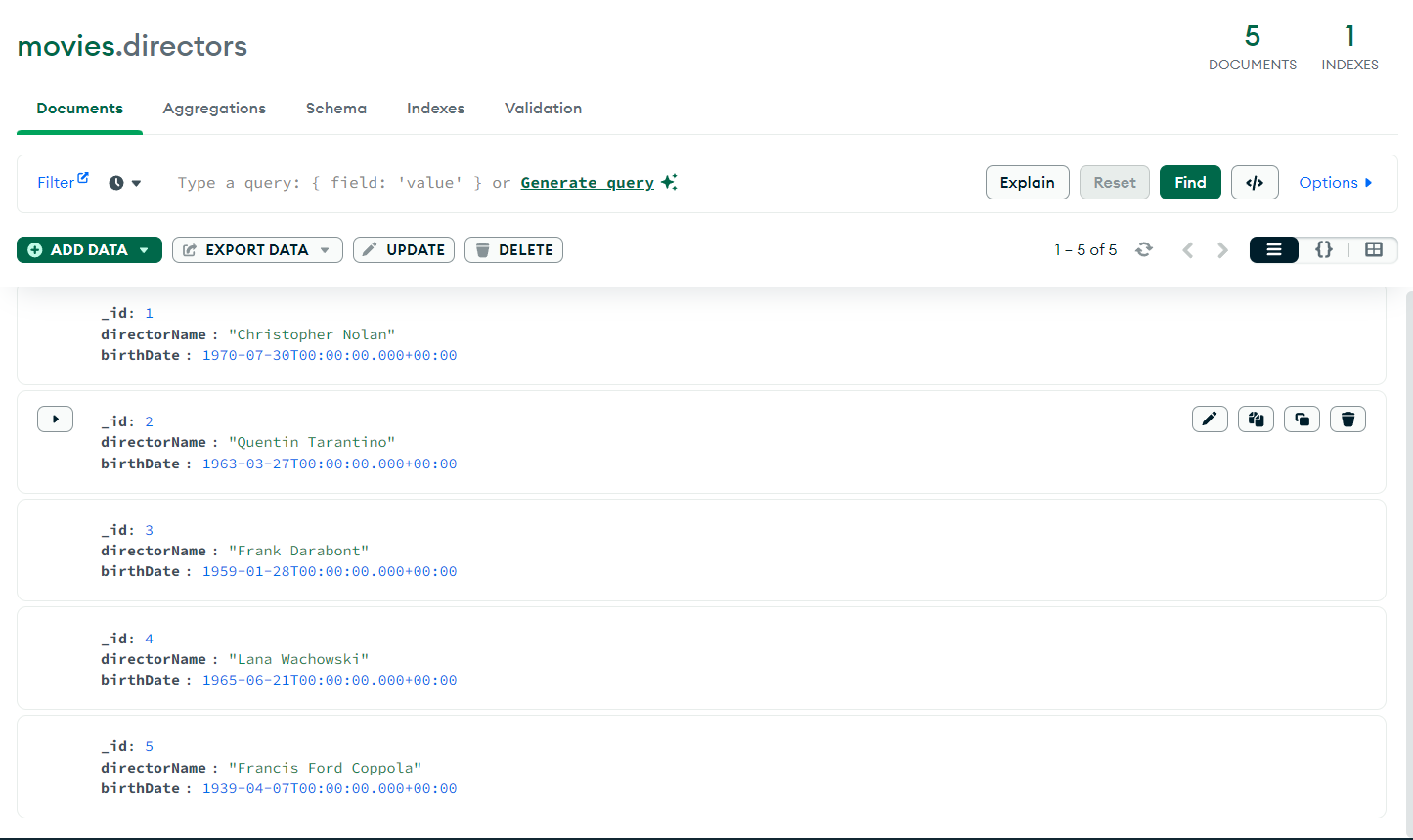
Description automatically generated

**Command to insert 5 entities in Director Collection**

A screenshot of a computer

Description automatically generated

**Output of the Director Collection**



**Command to insert 5 entities in Customer Collection**

A computer screen with green and white text

Description automatically generated

Output of the Customer Collection

A screenshot of a computer

Description automatically generated

1. **Update command 1.**

Before

A screenshot of a computer

Description automatically generated

**After update command we will change the email ID of customer with id = 1**

A screen shot of a computer

Description automatically generated

Output after update comand

A white background with blue text

Description automatically generated

**2. updating the review column in Ratings collection where id = 1**

Before

A screenshot of a computer

Description automatically generated

Command in the MongoDB shell

A black screen with white text

Description automatically generated

Out put after the update command

A screenshot of a computer

Description automatically generated

**Updating the director name to “new director name”**



1. **Command to add duplicate data in the collections**

A computer screen with many green and yellow text

Description automatically generated

Output of duplicate data

A screenshot of a computer

Description automatically generated

**Command to remove duplicate data**

A computer screen shot of a computer screen

Description automatically generated

**Output of removing the duplicate samples**

A screenshot of a computer

Description automatically generated

## Conclusion

To sum up, the goal of the data engineering project was to use relational and non-relational database models and data modelling methodologies to match certain business objectives in a selected scenario. To store and handle data effectively, the project concentrated on building strong data models and putting databases in place.   
  
Entities like Customer, Movie, Genre, Director, and Rating were identified for the relational data model. To illustrate the connections between these entities, an Entity-Relationship (ER) diagram was created. All aspects of the data model structure, including entities, properties, and connections, were thoroughly documented. Through normalization up to the third normal form (3NF), potential irregularities were corrected.  
  
  
During the normalization process, there were difficulties, such as handling identification columns and the requirement to add foreign keys. Primary and foreign keys were established, and linkages between tables were created using SQL statements.   
  
MongoDB was used in the non-relational data model's implementation. Collections were established in relation to recognized entities, including Customers, Movies, Genres, Ratings, and Directors. To illustrate how data is represented in a non-relational database, sample documents were added to each collection.   
  
During the procedure, there were problems, such as duplicate document removal and identification issues with the MongoDB shell. Nevertheless, workarounds and solutions were offered to deal with these problems.   
  
Notwithstanding the difficulties, the project effectively illustrated how data modeling approaches can be used to relational and non-relational databases, highlighting the adaptability and factors to be considered when creating scalable and effective data storage solutions. The knowledge acquired from this project advances our comprehension of data engineering concepts and procedures.