University of Engineering & Technology, Lahore

Department of Computer Science

Project Movie Market (Database Management System)

Moazam Ali

Supervisor: Ms. Rida Saeed

A report submitted in partial fulfilment of the requirements of
the University of Engineering & Technology, Lahore for the degree of
Bachelors of Science in *Computer Science*

April 18, 2025

Declaration

I, Moazam Ali, of the Department of Computer Science, University of Engineering & Technology, Lahore, confirm that this is my own work and figures, tables, equations, code snippets, artworks, and illustrations in this report are original and have not been taken from any other person's work, except where the works of others have been explicitly acknowledged, quoted, and referenced. I understand that if failing to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly.

I give consent to a copy of my report being shared with future students as an exemplar.

I give consent for my work to be made available more widely to members of UET and public with interest in teaching, learning and research.

Moazam Ali
April 18, 2025

Abstract

The Movie Market project presents the design and implementation of a relational database system built using MySQL to manage movie rentals and purchases. The goal of the project is to create a robust backend capable of tracking customers, inventory, rental transactions, sales, and payments, while maintaining data integrity and adhering to best practices in database normalization. The system integrates core SQL operations along with advanced features such as triggers, views, and stored procedures to automate processes and enhance data analysis. The project includes the development of an Entity Relationship Diagram (ERD), Enhanced ERD (EERD), and normalization up to Third Normal Form (3NF). A comprehensive suite of SQL queries was developed to support reporting needs, including administrative insights and customer behavior tracking. Functional testing and data-driven validations were performed using test logs. The final outcome is a scalable, cleanly structured database system suitable for integration with modern applications. This report provides full documentation of the design process, implementation logic, and performance testing for the Movie Market system.

Keywords: MySQL, database design, normalization, ERD, movie rental system

Contents

List of Figures

List of Tables

List of Abbreviations

DBMS	Database Management System
SQL	Structured Query Language
DDL	Data Definition Language
DML	Data Manipulation Language
ERD	Entity Relationship Diagram
EERD	Enhanced Entity Relationship Diagram
PK	Primary Key
FK	Foreign Key
3NF	Third Normal Form
API	Application Programming Interface
UI	User Interface
UML	Unified Modeling Language

Chapter 1

Introduction

1.1 Background

Digital platforms have transformed how users rent and purchase movies, requiring efficient backend systems to manage customer data, inventory, and transactions. This project, titled *Movie Market*, aims to replicate such a system using MySQL, applying core database design principles to create a scalable, relational backend.

1.2 Problem Statement

Small-scale media businesses often rely on fragmented tools to manage operations, leading to redundancy, errors, and poor data insights. There is a need for a centralized, normalized database system that supports rentals and sales while maintaining integrity and enabling reporting and automation.

1.3 Aims and Objectives

Aim: To design and implement a normalized database system for managing movie rentals and sales.

Objectives:

- Create ERD and EERD diagrams.
- Normalize tables to 3NF and implement them in MySQL.
- Develop queries, views, and triggers for core operations.
- Validate functionality through test cases and admin reports.

1.4 Solution Approach

The solution includes:

- Designing ERD/EERD models to capture system relationships.
- Building the schema via SQL scripts and inserting test data.
- Developing complex queries, views, and triggers for automation and analytics.
- Conducting test logs to ensure functional correctness.

1.5 Summary of Contributions

This project delivers:

- A clean, normalized MySQL database design.
- Advanced database operations including views, triggers, and procedures.
- Test cases that validate functionality.
- Admin-level reporting queries for business insights.

1.6 Organization of the Report

This report is structured as follows:

Chapter 1: Introduction – *Provides background, objectives, and methodology.*

Chapter 2: Database Design & Modeling – *Overview of entities, designs and normalization justification.*

Chapter 3: System Implementation – *Table definitions, sample data, constraints, keys and relationships.*

Chapter 4: Query Development & Features – *Functional Queries (joins, nested, aggregates), Views, Stored Procedures, Triggers*

Chapter 5: Reporting & Admin Tools – *Admin Reports (daily rentals, genre stats, out-of-stock etc.), Views for Management, High-value customer insights*

Chapter 6: Testing & Validation – *Test Scenarios, Sample Executions*

Chapter 7: Conclusion – *Summary of Achievements, Reflection on the Development Process, Potential Improvements*

Chapter 2

Database Design & Modeling

2.1 Entities Overview

The Movie Market database consists of five core entities, each representing a key part of the rental and sales workflow:

- **Customers** – Holds customer details including name, email, phone, and join date.
- **Movies** – Stores movie data such as title, genre, release year, stock, price, and availability.
- **Rentals** – Records customer rentals including rental date, return date, and any late fees.
- **Sales** – Logs purchases made by customers, linked to payment records.
- **Payments** – Tracks all payment transactions related to rentals or purchases.

These tables are designed to minimize redundancy and maintain clear relationships using foreign keys.

2.2 Entity Relationship Diagram (ERD)

The ERD models the relationships between the five core entities. Each table has a primary key and is connected via foreign keys where necessary. For example, the `Rentals` table links `Customers` and `Movies`, while the `Sales` table also references `Payments`.

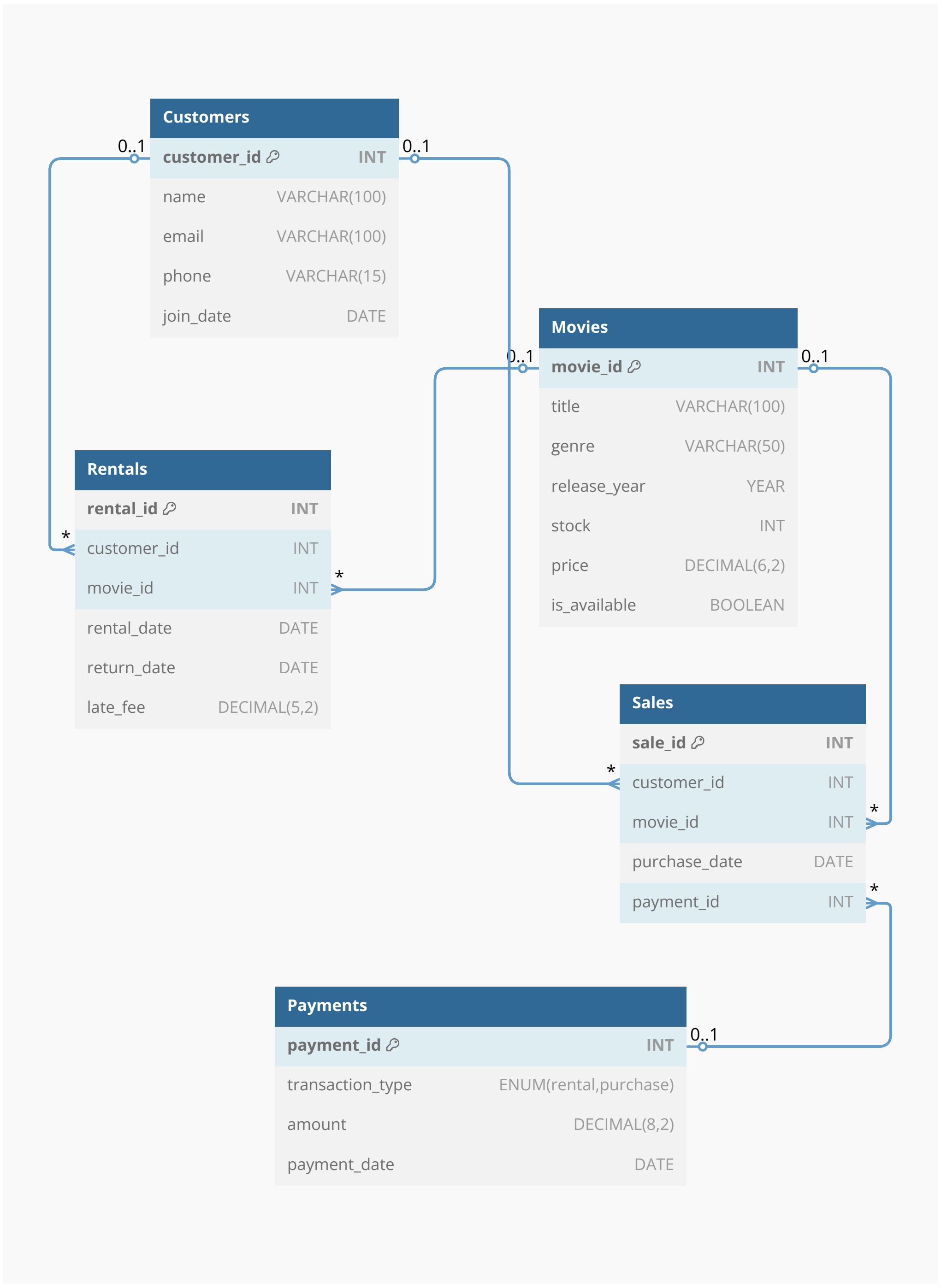

Figure 2.1: Entity Relationship Diagram (ERD) of the Movie Market Database

This diagram clearly defines one-to-many relationships such as:

- One customer may rent or purchase multiple movies.
- One movie may be rented or sold to many customers.
- One sale links to one payment.

2.3 Enhanced ERD (EERD)

To represent conceptual design improvements, an Enhanced ERD (EERD) introduces a generalized entity called `Transaction`, which combines attributes common to both rentals and sales. This abstraction allows for greater flexibility and extensibility in future implementations.

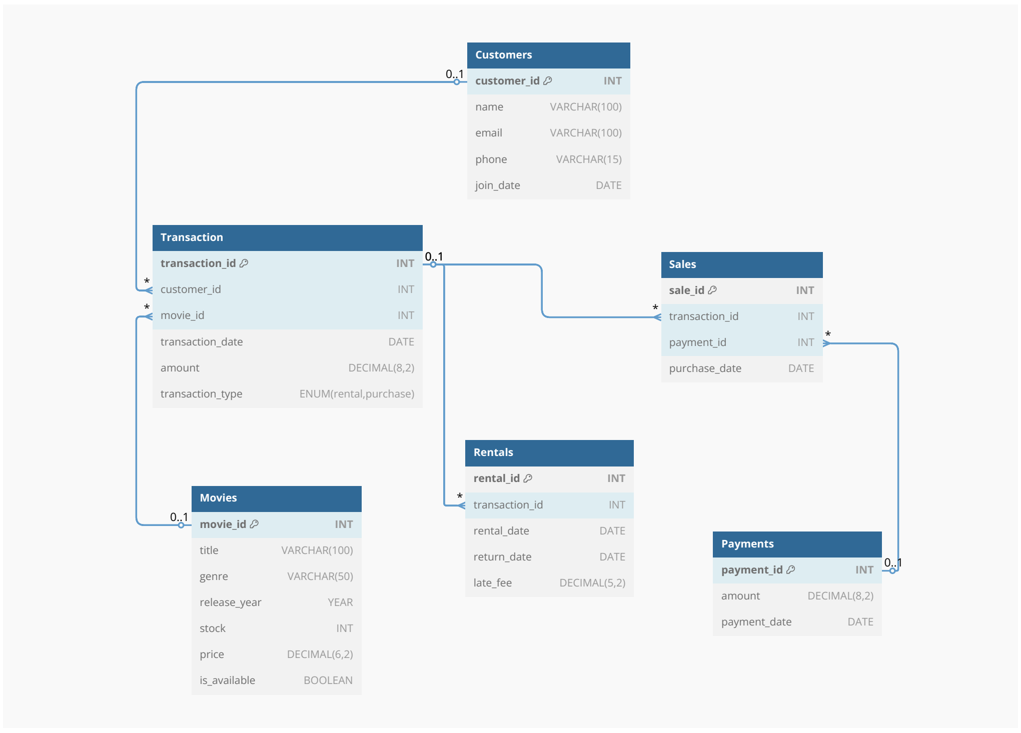

Figure 2.2: Enhanced Entity Relationship Diagram (EERD) of the Movie Market Database

The EERD introduces:

- **Transaction** – A generalized entity holding common fields like customer ID, movie ID, transaction date, amount, and transaction type.
- **Rentals** and **Sales** – Specialized entities that extend `Transaction`, adding fields like late fee or payment ID.

2.4 Normalization

All tables in the database were normalized to Third Normal Form (3NF), ensuring minimal redundancy and maximum integrity.

2.4.1 First Normal Form (1NF)

All data is atomic, and no multi-valued or repeating fields exist in any table.

2.4.2 Second Normal Form (2NF)

Every non-key attribute is fully dependent on the primary key. No partial dependencies are present.

2.4.3 Third Normal Form (3NF)

There are no transitive dependencies. Each non-key attribute depends only on the primary key, and not on other non-key attributes.
Normalization was a guiding principle throughout the schema design process, ensuring data clarity and consistency across the system.

Chapter 3

System Implementation

3.1 Overview

The implementation of the Movie Market database system was carried out using MySQL. It involved translating the ERD and EERD into structured SQL code for defining tables, inserting data, and enforcing integrity constraints. All SQL scripts are organized in modular files to separate design, data population, and testing logic.

3.2 Table Creation

The core database schema is defined in the file `layout_ddl.sql`. Each table was created with a focus on normalization, clarity, and efficient indexing.

- **Customers** – Includes fields for customer ID, name, email, phone number, and join date.
- **Movies** – Includes movie title, genre, release year, stock count, price, and availability.
- **Rentals** – Tracks movie rentals with rental and return dates, including late fees.
- **Sales** – Records movie purchases with references to payment IDs.
- **Payments** – Stores financial transaction details.

All primary keys are auto-incrementing, and foreign key constraints were used to maintain referential integrity between tables.

3.3 Sample Data

The file `sample_mm_data.sql` contains test data used to simulate real-world operations. It includes:

- 5 sample customers with valid contact details.
- A catalog of movies across different genres.
- Rental transactions, including some with late returns to trigger late fee logic.
- Purchase entries with corresponding payment records.

This data was used for testing query functionality and system behavior under typical operations.

3.4 Keys and Constraints

Each table enforces integrity via the following constraints:

- **Primary keys** ensure entity uniqueness.
- **Foreign keys** link related records across tables (e.g., customers to rentals).
- **Unique constraints** prevent duplication of critical fields such as email.
- **Default values** (e.g., `CURRENT_DATE`) reduce input complexity.

These constraints prevent anomalies and preserve data consistency.

3.5 Relationships and Referential Integrity

All one-to-many relationships defined in the ERD were enforced using foreign keys. For example:

- One customer can have many rentals and sales.
- One movie can be referenced by many rentals or purchases.
- One sale links to one unique payment.

This relational structure supports complex queries and reporting, while preserving data integrity.

Chapter 4

Query Development & Features

4.1 Functional Queries

The file `queries.sql` contains the primary set of SQL queries developed for the Movie Market system. These queries demonstrate how the database supports everyday business functions and analytical insights.

Key queries include:

- Listing all rentals by customers with details like return dates and late fees.
- Displaying movies currently available for rent.
- Calculating total rentals and late fees per customer.
- Identifying the top 3 most rented movies.
- Viewing customer purchase history with movie titles and purchase dates.
- Calculating total revenue generated from both rentals and purchases.
- Nested queries for identifying repeat customers and movies never rented.

These queries demonstrate the use of **JOINs**, **aggregates**, **subqueries**, and **filters** to extract meaningful data.

4.2 Views

Views were created to simplify frequent or complex queries. The primary view defined is:

- `CustomerRentalSummary` – Aggregates total rentals and late fees per customer.

Views enhance maintainability and make reporting more straightforward by encapsulating logic in reusable components.

4.3 Stored Procedures

A stored procedure, defined in `advance_ops.sql`, was implemented to generate rental summaries for any customer by ID.

- `GetCustomerRentalSummary(IN cust_id INT)` – Returns total rentals and total late fees for the specified customer.

Stored procedures provide flexibility and abstraction, making data operations reusable and secure.

4.4 Triggers

Two triggers were developed to enforce automation and consistency:

- `trg_calculate_late_fee` – Automatically calculates the late fee when a return date is updated, based on days overdue.
- `trg_update_movie_availability` – Automatically updates a movie's availability status when its stock is updated to or from zero.

Triggers help maintain business rules at the database level without requiring external logic.

4.5 Summary

The implementation of queries, views, procedures, and triggers brings the Movie Market system to life. These features go beyond data storage, enabling dynamic interactions, automated operations, and real-time analysis. They collectively transform the schema into a fully functional, real-world data system.

Chapter 5

Reporting & Admin Tools

5.1 Overview

Effective database systems go beyond storage — they support business decisions through meaningful reports and insights. The Movie Market project includes a suite of admin-level SQL queries, written in `admin_reps.sql`, designed to analyze operational data and guide strategic decisions.

5.2 Admin Reports

Key reports include:

- **Daily Rentals Report** – Shows the number of rentals per day.
- **Genre Popularity Report** – Aggregates rental counts by movie genre.
- **Out-of-Stock Movies** – Lists movies with zero inventory remaining.
- **Latest Purchases** – Displays the most recent purchase for each customer.

Table 5.1: Sample Output: Daily Rentals Report

Rental Date	Total Rentals
2024-03-01	2
2024-03-03	3
2024-03-06	1
2024-03-07	1

Table 5.2: Sample Output: Genre Popularity Report

Genre	Rental Count
Sci-Fi	4
Romance	1
Crime	1

5.3 Management Views

In addition to ad hoc queries, management-friendly views were created for regular reporting. For example:

- `CustomerPurchaseHistory` – Summarizes all purchases by customer, including payment data and movie details.

Table 5.3: Sample Output: Customer Purchase History View

Customer	Movie	Purchase Date	Amount ($)
Alice Johnson	Joker	2024-03-08	14.99
Bob Smith	Interstellar	2024-03-09	17.25
Charlie Davis	Inception	2024-03-10	13.75

5.4 Customer Insights

Advanced queries were also implemented to extract business intelligence, such as:

- **High-Value Customers** – Identifies customers who spent more than a defined threshold (e.g., $30).
- **Repeat Renters** – Uses nested queries to find users with multiple rentals.

Table 5.4: Sample Output: High-Value Customers

Customer	Total Spent ($)
Charlie Davis	41.25

5.5 Summary

The reporting and admin tools implemented in this project elevate the database from a basic transaction tracker to a decision-support system. These features enable operational monitoring, inventory planning, customer segmentation, and executive-level overviews — making the system truly business-ready.

Chapter 6

Testing & Validation

6.1 Overview

Testing was an integral part of the development process to ensure the Movie Market database operated as expected. All core functionalities were verified using real SQL test cases written in the file `test_logs.sql`. These tests covered data integrity, business logic (e.g., triggers), and query output correctness.

6.2 Test Scenarios

The following functionalities were tested:

- Adding new customers and verifying their presence in the database.
- Renting a movie and checking stock reduction.
- Returning a movie and validating late fee calculation via trigger.
- Inserting payment and purchase data.
- Running views and procedures for accurate data retrieval.
- Validating movie availability based on stock status.

6.3 Sample Test Executions

Sample SQL commands used during testing included:

- `INSERT INTO Customers (...) VALUES (...);` – Added test user.
- `UPDATE Rentals SET return_date = ...;` – Triggered late fee calculation.
- `CALL GetCustomerRentalSummary(1);` – Invoked stored procedure for summary.
- `SELECT * FROM CustomerRentalSummary;` – Queried view output.

Table 6.1: Test Case Log (Selected Examples)

Test Case	**Action**	**Expected Result**	**Status**
Insert customer	Add new record to `Customers`	Row appears	PASS
Rent movie	Add to `Rentals` and update stock	Stock -1	PASS
Return late	Set return date 5+ days later	Late fee > 0	PASS
Invoke procedure	Call rental summary	Correct data	PASS
Check out-of-stock	Set stock $= 0$ for movie	`is_available = FALSE`	PASS

6.4 Outcome

All test scenarios passed successfully, verifying the accuracy of table logic, data relationships, and business rules. Triggers executed automatically as intended, stored procedures returned valid outputs, and all queries produced correct results.

6.5 Conclusion

The Movie Market database has been fully validated through structured test cases, demonstrating its reliability, accuracy, and readiness for real-world use. Testing confirms the success of the project's design, implementation, and automation objectives.

Chapter 7

Conclusion

7.1 Summary of Achievements

The Movie Market project successfully demonstrates the development of a fully functional, normalized relational database system using MySQL. From designing the ERD and EERD to implementing tables, constraints, views, and triggers, every aspect was handled with real-world application in mind.

Key accomplishments include:

- A normalized database structure adhering to 3NF standards.
- A complete schema including customers, movies, rentals, sales, and payments.
- SQL scripts for sample data, testing, queries, views, triggers, and procedures.
- A suite of admin reports and analytics for business intelligence.
- Full validation through test case execution with successful results.

7.2 Reflection on the Development Process

Building this project was a hands-on learning journey in database design and implementation. It provided practical experience in:

- Structuring real-world data relationships.
- Writing and organizing SQL scripts for various functions.
- Thinking critically about data flow, constraints, and automation.
- Ensuring reliability through thorough testing.

This project bridged theoretical knowledge with technical application in an engaging and structured way.

7.3 Future Enhancements

While the backend system is fully functional, future work could include:

- Developing a frontend web interface for customer interaction.

- Integrating authentication and role-based access.
- Adding an API layer for external application integration.
- Implementing real-time stock alerts or customer notifications.

These enhancements would elevate the Movie Market system into a complete, full-stack application.

7.4 Final Remarks

Movie Market is more than just a project — it's a practical showcase of how databases power real-world systems. With clean design, robust logic, and automation features, it provides a strong foundation for future applications in retail, rentals, and media systems.