**SVKM’s NMIMS**

**School of Technology Management & Engineering (Indore Campus)**

**Computer Engineering Department (B Tech/MBATech CE and B Tech AIDS Sem IV)**

**Database Management System**

**Project Report**

|  |  |  |
| --- | --- | --- |
| Program | BTech CE | |
| Semester | 4th | |
| Name of the Project: | **Community-Driven Resource Sharing Platform** | |
|  | | |
| Details of Project Members |  |  |
| Batch | Roll No. | Name |
| 2 | D092 | Uzair Teli |
| 2 | D089 | Tanish Porwal |
| 2 | D090 | Tanmay Jhanjhari |
| Date of Submission: | | |

**Contribution of each project Members:**

|  |  |  |
| --- | --- | --- |
| Roll No. | Name: | Contribution |
| D089 | Tanish Porwal | DBMS |
| D090 | Tanmay Jhanjhari | DBMS |
| D092 | Uzair Teli | DBMS |

**Github link of your project:**

**Note:**

1. Create a readme file if you have multiple files
2. All files must be properly named (Example:R004\_DBMSProject)
3. Submit all relevant files of your work ( Report, all SQL files, Any other files)
4. **Plagiarism is highly discouraged (Your report will be checked for plagiarism)**

**Rubrics for the Project evaluation:**

|  |  |
| --- | --- |
| First phase of evaluation:  Innovative Ideas (5 Marks)  Design and Partial implementation (5 Marks) | 10 marks |
| Final phase of evaluation  Implementation, presentation and viva, Self-Learning and Learning Beyond classroom | 10 marks |

**Project Report**

**Selected Topic**

**by**

**Uzair Teli, Roll number: D092**

**Tanish Porwal, Roll number: D089**

**Tanmay Jhanjhari, Roll number: D090**

**Course: DBMS**

**AY: 2024-25**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr no.** | **Topic** | **Page no.** |
| **1** | Storyline |  |
| **2** | Components of Database Design |  |
| **3** | Entity Relationship Diagram |  |
| **4** | Relational Model |  |
| **5** | Normalization |  |
| **6** | SQL Queries |  |
| **7** | Learning from the Project |  |
| **8** | Project Demonstration |  |
| **9** | Self-learning beyond classroom |  |
| **10** | Learning from the project |  |
| **8** | Challenges faced |  |
| **9** | Conclusion |  |

1. **Storyline**

**Introduction**

In today's world, communities are constantly seeking ways to foster collaboration, sustainability, and mutual support. The "Community-Driven Resource Sharing Platform" is an innovative solution designed to address these needs by providing a platform where community members can share resources like tools, books, or equipment. This platform encourages the efficient use of resources, reduces waste, and promotes a sense of community among users.

**Problem Statement**

Many communities have resources that are underutilized or sitting idle. For example, a homeowner might have gardening tools that are only used a few times a year, or a student might have textbooks that are no longer needed after a semester. These resources could be highly valuable to other community members but are often inaccessible due to a lack of communication and sharing mechanisms.

**Objectives**

The primary objective of the Community-Driven Resource Sharing Platform is to create an online space where community members can list and share resources they are willing to lend. The platform aims to:

* Facilitate the listing and borrowing of resources.
* Track the availability and lending history of resources.
* Send reminders and notifications to users about borrowing deadlines.
* Encourage community engagement and sustainability.

**II. Components of Database Design**

**Entities and Attributes**

**1. User**

* Attributes:
  + UserID
  + Username
  + Password
  + Email
  + ContactInfo

**2. Resource**

* Attributes:
  + ResourceID
  + ResourceName
  + Description
  + Condition

**3. LendingHistory**

* Attributes:
  + LendingID
  + LendDate
  + ReturnDate

**4. Availability**

* Attributes:
  + Available
  + BorrowUntil

**5. Category**

* Attributes:
  + CategoryID
  + CategoryName

**6. Location**

* Attributes:
  + LocationID
  + LocationName
  + Address

**7. Notification**

* Attributes:
  + NotificationID
  + Message
  + NotificationDate

**8. Review**

* Attributes:
  + ReviewID
  + Rating
  + Comment
  + ReviewDate

**Relationships**

**1. User - Resource**

* Relationship: One-to-Many
* Description: A User can own multiple Resources, but each Resource is owned by only one User.
* Cardinality: 1:N
* Participation: Total for Resource (each Resource must be owned by a User)

**2. User - LendingHistory**

* Relationship: One-to-Many
* Description: A User can borrow multiple Resources over time, but each LendingHistory record is associated with one User.
* Cardinality: 1:N
* Participation: Total for LendingHistory (each LendingHistory record must be associated with a User)

**3. Resource - LendingHistory**

* Relationship: One-to-Many
* Description: A Resource can be borrowed multiple times, each time creating a new LendingHistory record.
* Cardinality: 1:N
* Participation: Total for LendingHistory (each LendingHistory record must be associated with a Resource)

**4. Resource - Availability**

* Relationship: One-to-One
* Description: Each Resource has one Availability status indicating if it is currently available or not.
* Cardinality: 1:1
* Participation: Total for both Resource and Availability (each Resource must have an Availability status)

**5. User - Notification**

* Relationship: One-to-Many
* Description: A User can receive multiple Notifications.
* Cardinality: 1:N
* Participation: Total for Notification (each Notification must be associated with a User)

**6. Resource - Notification**

* Relationship: One-to-Many
* Description: A Resource can trigger multiple Notifications.
* Cardinality: 1:N
* Participation: Total for Notification (each Notification must be associated with a Resource)

**7. User - Review**

* Relationship: One-to-Many
* Description: A User can write multiple Reviews.
* Cardinality: 1:N
* Participation: Total for Review (each Review must be associated with a User)

**8. Resource - Review**

* Relationship: One-to-Many
* Description: A Resource can receive multiple Reviews.
* Cardinality: 1:N
* Participation: Total for Review (each Review must be associated with a Resource)

**9. Resource - Category**

* Relationship: Many-to-One
* Description: A Resource belongs to one Category.
* Cardinality: N:1
* Participation: Total for Resource (each Resource must belong to a Category)

**10. Resource - Location**

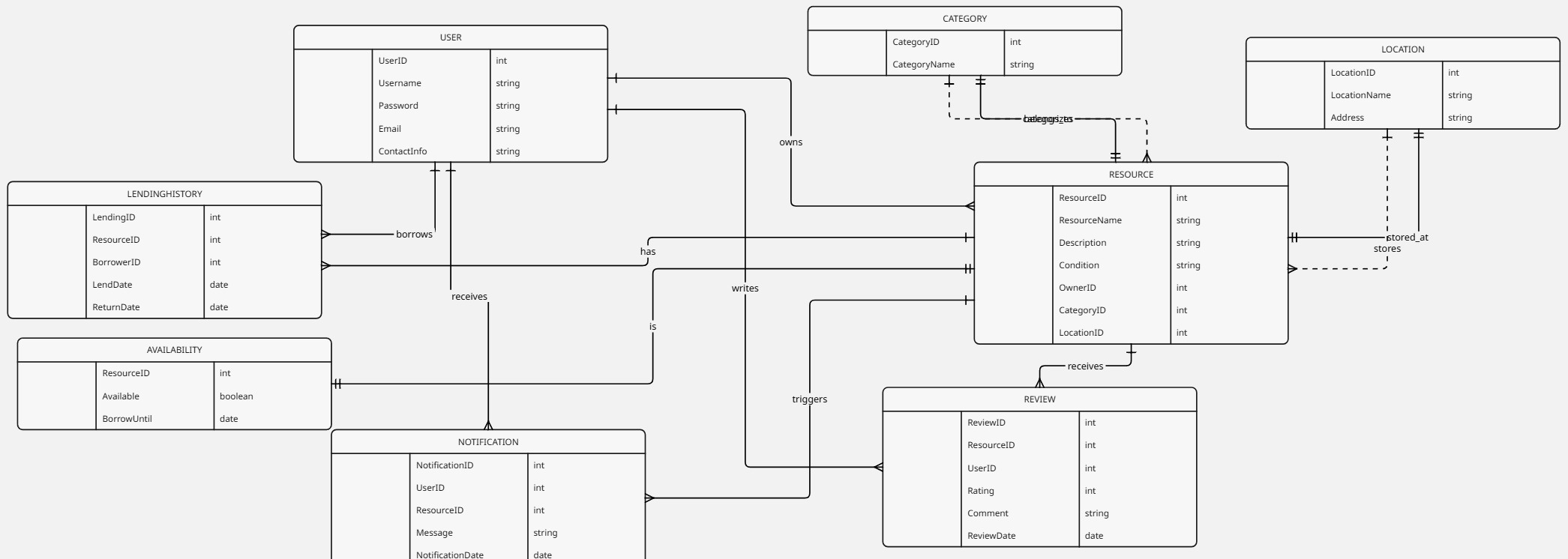
* Relationship: Many-to-One
* Description: A Resource is stored at one Location.
* Cardinality: N:1
* Participation: Total for Resource (each Resource must be stored at a Location)

**11. Category - Resource**

* Relationship: One-to-Many
* Description: A Category can have multiple Resources.
* Cardinality: 1:N
* Participation: Optional for Category (a Category can exist without Resources)

**12. Location - Resource**

* Relationship: One-to-Many
* Description: A Location can store multiple Resources.
* Cardinality: 1:N
* Participation: Optional for Location (a Location can exist without Resources)

**III. Entity Relationship Diagram**

**IV. Relational Model**

**Relational Model**

**User Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| UserID | INT | PRIMARY KEY |
| Username | VARCHAR | NOT NULL, UNIQUE |
| Password | VARCHAR | NOT NULL |
| Email | VARCHAR | NOT NULL, |
| ContactInfo | VARCHAR |  |

**Resource Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| ResourceID | INT | PRIMARY KEY |
| ResourceName | VARCHAR | NOT NULL |
| Description | TEXT |  |
| Condition | VARCHAR |  |
| OwnerID | INT | FOREIGN KEY REFERENCES User(UserID) |
| CategoryID | INT | FOREIGN KEY REFERENCES Category(CategoryID) |
| LocationID | INT | FOREIGN KEY REFERENCES Location(LocationID) |

**LendingHistory Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| LendingID | INT | PRIMARY KEY |
| ResourceID | INT | FOREIGN KEY REFERENCES Resource(ResourceID) |
| BorrowerID | INT | FOREIGN KEY REFERENCES User(UserID) |
| LendDate | DATE | NOT NULL |
| ReturnDate | DATE |  |

**Availability Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| ResourceID | INT | PRIMARY KEY, FOREIGN KEY REFERENCES Resource(ResourceID) |
| Available | BOOLEAN | NOT NULL |
| BorrowUntil | DATE |  |

**Category Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| CategoryID | INT | PRIMARY KEY |
| CategoryName | VARCHAR | NOT NULL, UNIQUE |

**Location Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| LocationID | INT | PRIMARY KEY |
| LocationName | VARCHAR | NOT NULL |
| Address | VARCHAR |  |

**Notification Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| NotificationID | INT | PRIMARY KEY |
| UserID | INT | FOREIGN KEY REFERENCES User(UserID) |
| ResourceID | INT | FOREIGN KEY REFERENCES Resource(ResourceID) |
| Message | TEXT | NOT NULL |
| NotificationDate | DATE | NOT NULL |

**Review Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| ReviewID | INT | PRIMARY KEY |
| ResourceID | INT | FOREIGN KEY REFERENCES Resource(ResourceID) |
| UserID | INT | FOREIGN KEY REFERENCES User(UserID) |
| Rating | INT | CHECK (Rating >= 1 AND Rating <= 5) |
| Comment | TEXT |  |
| ReviewDate | DATE | NOT NULL |

**List of Tables**

1. **User**
2. **Resource**
3. **LendingHistory**
4. **Availability**
5. **Category**
6. **Location**
7. **Notification**
8. **Review**

**V. Normalization**

All the tables in the database are already in 1NF, 2NF, 3NF, and BCNF. This ensures that the database is well-structured, free of redundancy, and maintains data integrity.

**Further Normalization Suggestions**

1. **User Table**:
   * Currently well-structured, but if the ContactInfo contains multiple pieces of information (e.g., phone number, address), it can be split into a separate table.
2. **Resource Table**:
   * The Condition attribute can be normalized into a separate table if there is a fixed set of conditions (e.g., New, Good, Fair, Poor).
3. **LendingHistory Table**:
   * Already well-structured, no further changes needed.
4. **Availability Table**:
   * Already well-structured, no further changes needed.
5. **Category Table**:
   * Already well-structured, no further changes needed.
6. **Location Table**:
   * Already well-structured, no further changes needed.
7. **Notification Table**:
   * The Message attribute can be standardized if there are predefined messages (e.g., Borrowing reminder, Return reminder).
8. **Review Table**:
   * Already well-structured, no further changes needed.

**Revised Relational Model**

**User Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| **UserID** | **INT** | **PRIMARY KEY** |
| **Password** | **VARCHAR** | **NOT NULL** |

**UserDetails Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| **DetailID** | **INT** | **PRIMARY KEY** |
| **UserID** | **INT** | **FOREIGN KEY REFERENCES User(UserID)** |
| **Username** | **VARCHAR** | **NOT NULL, UNIQUE** |
| **Email** | **VARCHAR** | **NOT NULL, UNIQUE** |

**ContactInfo Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| ContactInfoID | INT | PRIMARY KEY |
| UserID | INT | FOREIGN KEY REFERENCES User(UserID) |
| PhoneNumber | VARCHAR |  |
| Address | VARCHAR |  |

**Resource Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| ResourceID | INT | PRIMARY KEY |
| ResourceName | VARCHAR | NOT NULL |
| Description | TEXT |  |
| ConditionID | INT | FOREIGN KEY REFERENCES Condition(ConditionID) |
| OwnerID | INT | FOREIGN KEY REFERENCES User(UserID) |
| CategoryID | INT | FOREIGN KEY REFERENCES Category(CategoryID) |
| LocationID | INT | FOREIGN KEY REFERENCES Location(LocationID) |

**Condition Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| ConditionID | INT | PRIMARY KEY |
| ConditionName | VARCHAR | NOT NULL, UNIQUE |

**LendingHistory Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| LendingID | INT | PRIMARY KEY |
| ResourceID | INT | FOREIGN KEY REFERENCES Resource(ResourceID) |
| BorrowerID | INT | FOREIGN KEY REFERENCES User(UserID) |
| LendDate | DATE | NOT NULL |
| ReturnDate | DATE |  |

**Availability Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| ResourceID | INT | PRIMARY KEY, FOREIGN KEY REFERENCES Resource(ResourceID) |
| Available | BOOLEAN | NOT NULL |
| BorrowUntil | DATE |  |

**Category Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| CategoryID | INT | PRIMARY KEY |
| CategoryName | VARCHAR | NOT NULL, UNIQUE |

**Location Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| LocationID | INT | PRIMARY KEY |
| LocationName | VARCHAR | NOT NULL |
| Address | VARCHAR |  |

**Notification Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| NotificationID | INT | PRIMARY KEY |
| UserID | INT | FOREIGN KEY REFERENCES User(UserID) |
| ResourceID | INT | FOREIGN KEY REFERENCES Resource(ResourceID) |
| MessageID | INT | FOREIGN KEY REFERENCES Message(MessageID) |
| NotificationDate | DATE | NOT NULL |

**Message Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| MessageID | INT | PRIMARY KEY |
| MessageContent | TEXT | NOT NULL, UNIQUE |

**Review Table**

| **Column Name** | **Data Type** | **Constraints** |
| --- | --- | --- |
| ReviewID | INT | PRIMARY KEY |
| ResourceID | INT | FOREIGN KEY REFERENCES Resource(ResourceID) |
| UserID | INT | FOREIGN KEY REFERENCES User(UserID) |
| Rating | INT | CHECK (Rating >= 1 AND Rating <= 5) |
| Comment | TEXT |  |
| ReviewDate | DATE | NOT NULL |

**List of Tables**

1. **User**
2. **UserDetails**
3. **ContactInfo**
4. **Resource**
5. **Condition**
6. **LendingHistory**
7. **Availability**
8. **Category**
9. **Location**
10. **Notification**
11. **Message**
12. **Review**

**Summary**

### Normalization Check for 1NF:

| **Table** | **1NF Status** | **Explanation** |
| --- | --- | --- |
| **User** | ✅ Yes | All values are atomic (Username, Password, Email). No repeating groups or multi-valued attributes. |
| **ContactInfo** | ✅ Yes | Each field holds a single value (PhoneNumber, Address). No repeating groups. |
| **Condition** | ✅ Yes | All values are atomic (ConditionName). |
| **Category** | ✅ Yes | Each field holds a single value (CategoryName). No multi-valued attributes. |
| **Location** | ✅ Yes | All values are atomic (LocationName, Address). |
| **Resource** | ✅ Yes | Each field holds a single value (ResourceName, Description, ConditionID, etc.). No repeating groups. |
| **LendingHistory** | ✅ Yes | All attributes are atomic (ResourceID, BorrowerID, LendDate, etc.). |
| **Availability** | ✅ Yes | All values are atomic (Available, BorrowUntil). |
| **Notification** | ✅ Yes | Each field holds a single value (UserID, ResourceID, MessageID, etc.). |
| **Message** | ✅ Yes | All attributes are atomic (MessageContent). |
| **Review** | ✅ Yes | Each field holds a single value (Rating, Comment, ReviewDate, etc.). |
| ✔️ **All tables satisfy 1NF** |  |  |

### Normalization Check for 2NF:

| **Table** | **2NF Status** | **Explanation** |
| --- | --- | --- |
| **User** | ✅ Yes | All non-key attributes (Username, Password, Email) depend entirely on the primary key (UserID). |
| **ContactInfo** | ✅ Yes | All non-key attributes (PhoneNumber, Address) depend on the primary key (ContactInfoID) and not partially. |
| **Condition** | ✅ Yes | The non-key attribute (ConditionName) fully depends on the primary key (ConditionID). |
| **Category** | ✅ Yes | The non-key attribute (CategoryName) fully depends on the primary key (CategoryID). |
| **Location** | ✅ Yes | All non-key attributes (LocationName, Address) fully depend on the primary key (LocationID). |
| **Resource** | ✅ Yes | All non-key attributes fully depend on the primary key (ResourceID) (no partial dependency exists). |
| **LendingHistory** | ✅ Yes | All non-key attributes fully depend on the primary key (LendingID). |
| **Availability** | ✅ Yes | All non-key attributes (Available, BorrowUntil) depend on the primary key (ResourceID). |
| **Notification** | ✅ Yes | All non-key attributes fully depend on the primary key (NotificationID). |
| **Message** | ✅ Yes | The non-key attribute (MessageContent) depends entirely on the primary key (MessageID). |
| **Review** | ✅ Yes | All non-key attributes fully depend on the primary key (ReviewID). |
| ✔️ **All tables satisfy 2NF** |  |  |

### Normalization Check for 3NF:

| **Table** | **3NF Status** | **Explanation** |
| --- | --- | --- |
| **User** | ✅ Yes | No transitive dependency exists. All non-key attributes (Username, Password, Email) directly depend on the primary key. |
| **ContactInfo** | ✅ Yes | No transitive dependency exists. All non-key attributes directly depend on the primary key (ContactInfoID). |
| **Condition** | ✅ Yes | No transitive dependency exists. The non-key attribute (ConditionName) directly depends on the primary key. |
| **Category** | ✅ Yes | No transitive dependency exists. The non-key attribute (CategoryName) directly depends on the primary key. |
| **Location** | ✅ Yes | No transitive dependency exists. All non-key attributes directly depend on the primary key (LocationID). |
| **Resource** | ✅ Yes | No transitive dependency exists. All non-key attributes directly depend on the primary key (ResourceID). |
| **LendingHistory** | ✅ Yes | No transitive dependency exists. All non-key attributes directly depend on the primary key (LendingID). |
| **Availability** | ✅ Yes | No transitive dependency exists. All non-key attributes directly depend on the primary key (ResourceID). |
| **Notification** | ✅ Yes | No transitive dependency exists. All non-key attributes directly depend on the primary key (NotificationID). |
| **Message** | ✅ Yes | No transitive dependency exists. The non-key attribute (MessageContent) directly depends on the primary key. |
| **Review** | ✅ Yes | No transitive dependency exists. All non-key attributes directly depend on the primary key (ReviewID). |
| ✔️ **All tables satisfy 3NF** |  |  |

### Summary

* **1NF**: All tables satisfy 1NF as all values are atomic and no repeating groups exist.
* **2NF**: All tables satisfy 2NF as there are no partial dependencies.
* **3NF**: All tables satisfy 3NF as there are no transitive dependencies.

**VI. SQL Queries**

-- Create and use the database

CREATE DATABASE IF NOT EXISTS ggdbms;

USE ggdbms;

-- User Table

CREATE TABLE User (

UserID INT PRIMARY KEY,

Password VARCHAR(255) NOT NULL

);

-- UserDetails Table (Normalized)

CREATE TABLE UserDetails (

DetailID INT PRIMARY KEY,

UserID INT NOT NULL,

Username VARCHAR(255) NOT NULL UNIQUE,

Email VARCHAR(255) NOT NULL UNIQUE,

FOREIGN KEY (UserID) REFERENCES User(UserID)

);

-- ContactInfo table

CREATE TABLE ContactInfo (

ContactInfoID INT PRIMARY KEY,

UserID INT,

PhoneNumber VARCHAR(255),

Address VARCHAR(255),

FOREIGN KEY (UserID) REFERENCES User(UserID)

);

-- ResourceCondition table (renamed from "Condition")

CREATE TABLE ResourceCondition (

ConditionID INT PRIMARY KEY,

ConditionName VARCHAR(255) NOT NULL UNIQUE

);

-- Category table

CREATE TABLE Category (

CategoryID INT PRIMARY KEY,

CategoryName VARCHAR(255) NOT NULL UNIQUE

);

-- Location table

CREATE TABLE Location (

LocationID INT PRIMARY KEY,

LocationName VARCHAR(255) NOT NULL,

Address VARCHAR(255)

);

-- Resource table

CREATE TABLE Resource (

ResourceID INT PRIMARY KEY,

ResourceName VARCHAR(255) NOT NULL,

Description TEXT,

ConditionID INT,

OwnerID INT,

CategoryID INT,

LocationID INT,

FOREIGN KEY (ConditionID) REFERENCES ResourceCondition(ConditionID),

FOREIGN KEY (OwnerID) REFERENCES User(UserID),

FOREIGN KEY (CategoryID) REFERENCES Category(CategoryID),

FOREIGN KEY (LocationID) REFERENCES Location(LocationID)

);

-- LendingHistory table

CREATE TABLE LendingHistory (

LendingID INT PRIMARY KEY,

ResourceID INT,

BorrowerID INT,

LendDate DATE NOT NULL,

ReturnDate DATE,

FOREIGN KEY (ResourceID) REFERENCES Resource(ResourceID),

FOREIGN KEY (BorrowerID) REFERENCES User(UserID)

);

-- Availability table

CREATE TABLE Availability (

ResourceID INT PRIMARY KEY,

Available BOOLEAN NOT NULL,

BorrowUntil DATE,

FOREIGN KEY (ResourceID) REFERENCES Resource(ResourceID)

);

-- Message table

CREATE TABLE Message (

MessageID INT PRIMARY KEY,

MessageContent VARCHAR(255) NOT NULL UNIQUE

);

-- Notification table

CREATE TABLE Notification (

NotificationID INT PRIMARY KEY,

UserID INT,

ResourceID INT,

MessageID INT,

NotificationDate DATE NOT NULL,

FOREIGN KEY (UserID) REFERENCES User(UserID),

FOREIGN KEY (ResourceID) REFERENCES Resource(ResourceID),

FOREIGN KEY (MessageID) REFERENCES Message(MessageID)

);

-- Review table

CREATE TABLE Review (

ReviewID INT PRIMARY KEY,

ResourceID INT,

UserID INT,

Rating INT CHECK (Rating >= 1 AND Rating <= 5),

Comment TEXT,

ReviewDate DATE NOT NULL,

FOREIGN KEY (ResourceID) REFERENCES Resource(ResourceID),

FOREIGN KEY (UserID) REFERENCES User(UserID)

);

INSERT INTO User (UserID, Password)

VALUES

(1, 'pass1'),

(2, 'pass2'),

(3, 'pass3'),

(4, 'pass4'),

(5, 'pass5');

INSERT INTO UserDetails (DetailID, UserID, Username, Email)

VALUES

(1, 1, 'user1', 'user1@example.com'),

(2, 2, 'user2', 'user2@example.com'),

(3, 3, 'user3', 'user3@example.com'),

(4, 4, 'user4', 'user4@example.com'),

(5, 5, 'user5', 'user5@example.com');

INSERT INTO ContactInfo (ContactInfoID, UserID, PhoneNumber, Address) VALUES

(1, 1, '9999911111', 'City A'),

(2, 2, '9999911112', 'City B'),

(3, 3, '9999911113', 'City C'),

(4, 4, '9999911114', 'City D'),

(5, 5, '9999911115', 'City E'),

(6, 6, '9999911116', 'City F'),

(7, 7, '9999911117', 'City G'),

(8, 8, '9999911118', 'City H'),

(9, 9, '9999911119', 'City I'),

(10, 10, '9999911120', 'City J'),

(11, 11, '9999911121', 'City K'),

(12, 12, '9999911122', 'City L'),

(13, 13, '9999911123', 'City M'),

(14, 14, '9999911124', 'City N'),

(15, 15, '9999911125', 'City O'),

(16, 16, '9999911126', 'City P'),

(17, 17, '9999911127', 'City Q'),

(18, 18, '9999911128', 'City R'),

(19, 19, '9999911129', 'City S'),

(20, 20, '9999911130', 'City T'),

(21, 21, '9999911131', 'City U'),

(22, 22, '9999911132', 'City V'),

(23, 23, '9999911133', 'City W'),

(24, 24, '9999911134', 'City X'),

(25, 25, '9999911135', 'City Y');

INSERT INTO ResourceCondition (ConditionID, ConditionName) VALUES

(1, 'New'),

(2, 'Good'),

(3, 'Fair'),

(4, 'Poor');

INSERT INTO Category (CategoryID, CategoryName) VALUES

(1, 'Books'),

(2, 'Tools'),

(3, 'Electronics'),

(4, 'Furniture'),

(5, 'Games');

INSERT INTO Location (LocationID, LocationName, Address) VALUES

(1, 'Location1', 'Street A'),

(2, 'Location2', 'Street B'),

(3, 'Location3', 'Street C'),

(4, 'Location4', 'Street D'),

(5, 'Location5', 'Street E'),

(6, 'Location6', 'Street F'),

(7, 'Location7', 'Street G'),

(8, 'Location8', 'Street H'),

(9, 'Location9', 'Street I'),

(10, 'Location10', 'Street J'),

(11, 'Location11', 'Street K'),

(12, 'Location12', 'Street L'),

(13, 'Location13', 'Street M'),

(14, 'Location14', 'Street N'),

(15, 'Location15', 'Street O'),

(16, 'Location16', 'Street P'),

(17, 'Location17', 'Street Q'),

(18, 'Location18', 'Street R'),

(19, 'Location19', 'Street S'),

(20, 'Location20', 'Street T'),

(21, 'Location21', 'Street U'),

(22, 'Location22', 'Street V'),

(23, 'Location23', 'Street W'),

(24, 'Location24', 'Street X'),

(25, 'Location25', 'Street Y');

INSERT INTO Resource (ResourceID, ResourceName, Description, ConditionID, OwnerID, CategoryID, LocationID) VALUES

(1, 'Resource1', 'Drill machine', 2, 1, 2, 1),

(2, 'Resource2', 'Screwdriver set', 3, 2, 2, 2),

(3, 'Resource3', 'Study Table', 4, 3, 4, 3),

(4, 'Resource4', 'Python Programming Book', 1, 4, 1, 4),

(5, 'Resource5', 'Chess Board', 2, 5, 5, 5),

(6, 'Resource6', 'Projector', 1, 6, 3, 6),

(7, 'Resource7', 'Camping Tent', 2, 7, 4, 7),

(8, 'Resource8', 'C++ Book', 3, 8, 1, 8),

(9, 'Resource9', 'Power Bank', 2, 9, 3, 9),

(10, 'Resource10', 'PlayStation', 1, 10, 5, 10),

(11, 'Resource11', 'Chair Set', 4, 11, 4, 11),

(12, 'Resource12', 'Table Lamp', 3, 12, 3, 12),

(13, 'Resource13', 'Microcontroller Kit', 2, 13, 3, 13),

(14, 'Resource14', 'DSA Book', 1, 14, 1, 14),

(15, 'Resource15', 'Ladder', 2, 15, 2, 15),

(16, 'Resource16', 'Football', 2, 16, 5, 16),

(17, 'Resource17', 'Router', 1, 17, 3, 17),

(18, 'Resource18', 'Sofa', 4, 18, 4, 18),

(19, 'Resource19', 'Textbook - DBMS', 1, 19, 1, 19),

(20, 'Resource20', 'Whiteboard', 2, 20, 4, 20),

(21, 'Resource21', 'Arduino Board', 1, 21, 3, 21),

(22, 'Resource22', 'Hammer', 3, 22, 2, 22),

(23, 'Resource23', 'Gaming Controller', 2, 23, 5, 23),

(24, 'Resource24', 'Scanner', 2, 24, 3, 24),

(25, 'Resource25', 'Screw Gun', 1, 25, 2, 25);

INSERT INTO LendingHistory (LendingID, ResourceID, BorrowerID, LendDate, ReturnDate) VALUES

(1, 1, 5, '2023-05-01', '2023-05-15'),

(2, 2, 10, '2023-06-10', '2023-06-25'),

(3, 3, 15, '2023-07-05', '2023-07-20'),

(4, 4, 20, '2023-08-01', '2023-08-10'),

(5, 5, 2, '2023-09-12', '2023-09-20'),

(6, 6, 8, '2023-10-03', '2023-10-17'),

(7, 7, 18, '2023-11-11', '2023-11-25'),

(8, 8, 4, '2023-12-01', '2023-12-12'),

(9, 9, 6, '2024-01-05', '2024-01-18'),

(10, 10, 12, '2024-02-14', '2024-02-28'),

(11, 11, 7, '2024-03-01', '2024-03-10'),

(12, 12, 21, '2023-11-01', '2023-11-10'),

(13, 13, 22, '2023-06-15', '2023-06-30'),

(14, 14, 19, '2023-08-10', '2023-08-25'),

(15, 15, 24, '2023-09-01', '2023-09-15'),

(16, 16, 23, '2023-10-12', '2023-10-26'),

(17, 17, 25, '2023-11-05', '2023-11-20'),

(18, 18, 3, '2023-12-01', '2023-12-18'),

(19, 19, 14, '2024-01-10', '2024-01-20'),

(20, 20, 16, '2024-02-01', '2024-02-15'),

(21, 21, 13, '2024-03-01', '2024-03-10'),

(22, 22, 17, '2023-06-01', '2023-06-15'),

(23, 23, 11, '2023-07-10', '2023-07-20'),

(24, 24, 9, '2023-08-01', '2023-08-12'),

(25, 25, 1, '2023-09-05', '2023-09-19');

INSERT INTO Availability (ResourceID, Available, BorrowUntil) VALUES

(1, 1, NULL),

(2, 0, '2025-05-01'),

(3, 1, NULL),

(4, 1, NULL),

(5, 0, '2025-06-10'),

(6, 1, NULL),

(7, 0, '2025-04-30'),

(8, 1, NULL),

(9, 1, NULL),

(10, 0, '2025-07-01'),

(11, 1, NULL),

(12, 0, '2025-08-15'),

(13, 1, NULL),

(14, 1, NULL),

(15, 0, '2025-09-01'),

(16, 1, NULL),

(17, 1, NULL),

(18, 0, '2025-05-15'),

(19, 1, NULL),

(20, 0, '2025-06-01'),

(21, 1, NULL),

(22, 1, NULL),

(23, 0, '2025-07-20'),

(24, 1, NULL),

(25, 0, '2025-08-10');

INSERT INTO Message (MessageID, MessageContent) VALUES

(1, 'Borrow reminder'),

(2, 'Return reminder'),

(3, 'Resource available'),

(4, 'Overdue notice');

INSERT INTO Notification (NotificationID, UserID, ResourceID, MessageID, NotificationDate) VALUES

(1, 1, 1, 1, '2024-01-05'),

(2, 2, 2, 2, '2024-01-10'),

(3, 3, 3, 3, '2024-01-15'),

(4, 4, 4, 4, '2024-01-20'),

(5, 5, 5, 1, '2024-01-25'),

(6, 6, 6, 2, '2024-02-01'),

(7, 7, 7, 3, '2024-02-05'),

(8, 8, 8, 4, '2024-02-10'),

(9, 9, 9, 1, '2024-02-15'),

(10, 10, 10, 2, '2024-02-20'),

(11, 11, 11, 3, '2024-02-25'),

(12, 12, 12, 4, '2024-03-01'),

(13, 13, 13, 1, '2024-03-05'),

(14, 14, 14, 2, '2024-03-10'),

(15, 15, 15, 3, '2024-03-15'),

(16, 16, 16, 4, '2024-03-20'),

(17, 17, 17, 1, '2024-03-25'),

(18, 18, 18, 2, '2024-03-30'),

(19, 19, 19, 3, '2024-04-01'),

(20, 20, 20, 4, '2024-04-05'),

(21, 21, 21, 1, '2024-04-10'),

(22, 22, 22, 2, '2024-04-15'),

(23, 23, 23, 3, '2024-04-20'),

(24, 24, 24, 4, '2024-04-25'),

(25, 25, 25, 1, '2024-04-30');

INSERT INTO Review (ReviewID, ResourceID, UserID, Rating, Comment, ReviewDate) VALUES

(1, 1, 2, 5, 'Very helpful tool!', '2024-02-01'),

(2, 2, 3, 4, 'Good condition.', '2024-02-05'),

(3, 3, 4, 3, 'Could be better.', '2024-02-10'),

(4, 4, 5, 5, 'Excellent book.', '2024-02-15'),

(5, 5, 6, 4, 'Useful and clean.', '2024-02-20'),

(6, 6, 7, 5, 'Nice projector.', '2024-02-25'),

(7, 7, 8, 2, 'Old but works.', '2024-03-01'),

(8, 8, 9, 4, 'Good for learning.', '2024-03-05'),

(9, 9, 10, 3, 'Battery life is low.', '2024-03-10'),

(10, 10, 11, 5, 'Great gaming fun!', '2024-03-15'),

(11, 11, 12, 2, 'Worn out chairs.', '2024-03-20'),

(12, 12, 13, 3, 'Decent lamp.', '2024-03-25'),

(13, 13, 14, 4, 'Well maintained.', '2024-03-30'),

(14, 14, 15, 5, 'Amazing book.', '2024-04-01'),

(15, 15, 16, 3, 'Rusty ladder.', '2024-04-05'),

(16, 16, 17, 4, 'Kids loved it.', '2024-04-10'),

(17, 17, 18, 5, 'Very fast router.', '2024-04-15'),

(18, 18, 19, 1, 'Broken leg.', '2024-04-20'),

(19, 19, 20, 4, 'Informative textbook.', '2024-04-25'),

(20, 20, 21, 5, 'Perfect whiteboard.', '2024-04-30'),

(21, 21, 22, 4, 'Awesome hardware.', '2024-05-01'),

(22, 22, 23, 3, 'OK hammer.', '2024-05-05'),

(23, 23, 24, 4, 'Responsive controls.', '2024-05-10'),

(24, 24, 25, 5, 'Clear scans.', '2024-05-15'),

(25, 25, 1, 3, 'Not powerful enough.', '2024-05-20');

SET SQL\_SAFE\_UPDATES = 0;

START TRANSACTION;

-- all your queries

-- 1. List all users

SELECT \* FROM User;

-- 2. Show all resources with their availability

SELECT r.ResourceName, a.Available, a.BorrowUntil

FROM Resource r

JOIN Availability a ON r.ResourceID = a.ResourceID;

-- 3. List all resources along with category and location

SELECT r.ResourceName, c.CategoryName, l.LocationName

FROM Resource r

JOIN Category c ON r.CategoryID = c.CategoryID

JOIN Location l ON r.LocationID = l.LocationID;

-- 4. Show resources and their condition

SELECT r.ResourceName, rc.ConditionName

FROM Resource r

JOIN ResourceCondition rc ON r.ConditionID = rc.ConditionID;

-- 5. Get all users who have written reviews

SELECT DISTINCT u.Username

FROM User u

JOIN Review r ON u.UserID = r.UserID;

-- 6. Show average rating for each resource

SELECT r.ResourceName, AVG(rv.Rating) AS AvgRating

FROM Review rv

JOIN Resource r ON rv.ResourceID = r.ResourceID

GROUP BY r.ResourceName;

-- 7. Get top 5 highest-rated resources

SELECT r.ResourceName, AVG(rv.Rating) AS AvgRating

FROM Review rv

JOIN Resource r ON rv.ResourceID = r.ResourceID

GROUP BY r.ResourceName

ORDER BY AvgRating DESC

LIMIT 5;

-- 8. Count how many resources are available

SELECT COUNT(\*) AS AvailableCount

FROM Availability

WHERE Available = 1;

-- 9. Show all users and how many resources they own

SELECT u.Username, COUNT(r.ResourceID) AS TotalResources

FROM User u

LEFT JOIN Resource r ON u.UserID = r.OwnerID

GROUP BY u.Username;

-- 10. Show borrowing history of a resource

SELECT lh.LendDate, lh.ReturnDate, u.Username

FROM LendingHistory lh

JOIN User u ON lh.BorrowerID = u.UserID

WHERE lh.ResourceID = 5;

-- 11. List all notifications with user and message

SELECT n.NotificationID, u.Username, m.MessageContent, n.NotificationDate

FROM Notification n

JOIN User u ON n.UserID = u.UserID

JOIN Message m ON n.MessageID = m.MessageID;

-- 12. Find resources not currently available

SELECT r.ResourceName

FROM Resource r

JOIN Availability a ON r.ResourceID = a.ResourceID

WHERE a.Available = 0;

-- 13. List all categories with number of resources

SELECT c.CategoryName, COUNT(r.ResourceID) AS Total

FROM Category c

LEFT JOIN Resource r ON c.CategoryID = r.CategoryID

GROUP BY c.CategoryName;

-- 14. Show all resources that are due to be returned after today

SELECT r.ResourceName, a.BorrowUntil

FROM Resource r

JOIN Availability a ON r.ResourceID = a.ResourceID

WHERE a.BorrowUntil > CURDATE();

-- 15. Find users with more than 1 review

SELECT u.Username, COUNT(rv.ReviewID) AS ReviewCount

FROM User u

JOIN Review rv ON u.UserID = rv.UserID

GROUP BY u.Username

HAVING COUNT(rv.ReviewID) > 1;

-- 16. List all reviews with usernames and resource names

SELECT u.Username, r.ResourceName, rv.Rating, rv.Comment

FROM Review rv

JOIN User u ON rv.UserID = u.UserID

JOIN Resource r ON rv.ResourceID = r.ResourceID;

-- 17. Show resources in 'Books' category

SELECT r.ResourceName

FROM Resource r

JOIN Category c ON r.CategoryID = c.CategoryID

WHERE c.CategoryName = 'Books';

-- 18. Show users who haven’t posted any review

SELECT u.Username

FROM User u

LEFT JOIN Review rv ON u.UserID = rv.UserID

WHERE rv.ReviewID IS NULL;

-- 19. Show last 5 notifications

SELECT \* FROM Notification ORDER BY NotificationDate DESC LIMIT 5;

-- 20. Show resource usage frequency (how many times borrowed)

SELECT r.ResourceName, COUNT(lh.LendingID) AS TimesLent

FROM Resource r

LEFT JOIN LendingHistory lh ON r.ResourceID = lh.ResourceID

GROUP BY r.ResourceName;

-- 21. Update a user's email

UPDATE User SET Email = 'newemail@example.com' WHERE UserID = 1;

-- 22. Mark a resource as available

UPDATE Availability SET Available = 1, BorrowUntil = NULL WHERE ResourceID = 4;

-- 23. Delete a review with bad rating

DELETE FROM Review WHERE Rating = 1;

-- 24. Find all resources owned by user 'user3'

SELECT r.ResourceName

FROM Resource r

JOIN User u ON r.OwnerID = u.UserID

WHERE u.Username = 'user3';

-- 25. Show all usernames and their contact numbers

SELECT u.Username, ci.PhoneNumber

FROM User u

JOIN ContactInfo ci ON u.UserID = ci.UserID;

-- 26. List all usernames who have borrowed a resource

SELECT DISTINCT u.Username

FROM User u

JOIN LendingHistory lh ON u.UserID = lh.BorrowerID;

-- 27. Count total number of reviews per rating

SELECT Rating, COUNT(\*) AS Total

FROM Review

GROUP BY Rating;

-- 28. Show users who received notifications in the past 30 days

SELECT DISTINCT u.Username

FROM Notification n

JOIN User u ON n.UserID = u.UserID

WHERE n.NotificationDate >= CURDATE() - INTERVAL 30 DAY;

-- 29. Resources in 'Poor' condition

SELECT r.ResourceName

FROM Resource r

JOIN ResourceCondition rc ON r.ConditionID = rc.ConditionID

WHERE rc.ConditionName = 'Poor';

-- 30. Create a view with full resource summary

CREATE VIEW ResourceSummary AS

SELECT r.ResourceName, u.Username AS Owner, c.CategoryName, l.LocationName, rc.ConditionName, a.Available

FROM Resource r

JOIN User u ON r.OwnerID = u.UserID

JOIN Category c ON r.CategoryID = c.CategoryID

JOIN Location l ON r.LocationID = l.LocationID

JOIN ResourceCondition rc ON r.ConditionID = rc.ConditionID

JOIN Availability a ON r.ResourceID = a.ResourceID;

-- 31. Query the ResourceSummary view

SELECT \* FROM ResourceSummary;

-- 32. Get the highest rated resource

SELECT r.ResourceName

FROM Review rv

JOIN Resource r ON rv.ResourceID = r.ResourceID

GROUP BY r.ResourceName

ORDER BY AVG(rv.Rating) DESC

LIMIT 1;

-- 33. Get all resources borrowed but not returned yet

SELECT r.ResourceName, lh.BorrowerID

FROM LendingHistory lh

JOIN Resource r ON lh.ResourceID = r.ResourceID

WHERE lh.ReturnDate IS NULL;

-- 34. Change the name of a resource

UPDATE Resource SET ResourceName = 'High-Powered Drill' WHERE ResourceID = 1;

-- 35. Delete a user and their contact info

DELETE FROM ContactInfo WHERE UserID = 25;

DELETE FROM User WHERE UserID = 25;

-- 36. Count of messages used in notifications

SELECT m.MessageContent, COUNT(n.NotificationID) AS TimesUsed

FROM Message m

JOIN Notification n ON m.MessageID = n.MessageID

GROUP BY m.MessageContent;

-- 37. Show reviews containing word 'helpful'

SELECT \* FROM Review WHERE Comment LIKE '%helpful%';

-- 38. Find resources never borrowed

SELECT r.ResourceName

FROM Resource r

LEFT JOIN LendingHistory lh ON r.ResourceID = lh.ResourceID

WHERE lh.LendingID IS NULL;

-- 39. Show most recently added review

SELECT \* FROM Review ORDER BY ReviewDate DESC LIMIT 1;

-- 40. Get count of users per email domain

SELECT SUBSTRING\_INDEX(Email, '@', -1) AS Domain, COUNT(\*) AS UserCount

FROM User

GROUP BY Domain;

SELECT u.UserID, ud.Username, ud.Email, u.Password

FROM User u

JOIN UserDetails ud ON u.UserID = ud.UserID;

SELECT ud.Username, ud.Email, u.Password

FROM User u

JOIN UserDetails ud ON u.UserID = ud.UserID

WHERE ud.Username = 'user1';

SELECT ud.Username, ud.Email, u.Password

FROM User u

JOIN UserDetails ud ON u.UserID = ud.UserID

WHERE ud.Email = 'user1@example.com';

UPDATE User

SET Password = 'newpassword'

WHERE UserID = 1;

UPDATE UserDetails

SET Email = 'newemail@example.com'

WHERE Username = 'user1';

COMMIT;

SELECT ResourceID, Available, BorrowUntil FROM Availability LIMIT 10;

SELECT \* FROM Review WHERE Rating = 1;

SELECT u.Username, COUNT(r.ResourceID) AS TotalOwned

FROM User u

LEFT JOIN Resource r ON u.UserID = r.OwnerID

GROUP BY u.Username;

SELECT \* FROM User;

SELECT \* FROM ContactInfo;

SELECT \* FROM ResourceCondition;

SELECT \* FROM Category;

SELECT \* FROM Location;

SELECT \* FROM Resource;

SELECT \* FROM LendingHistory;

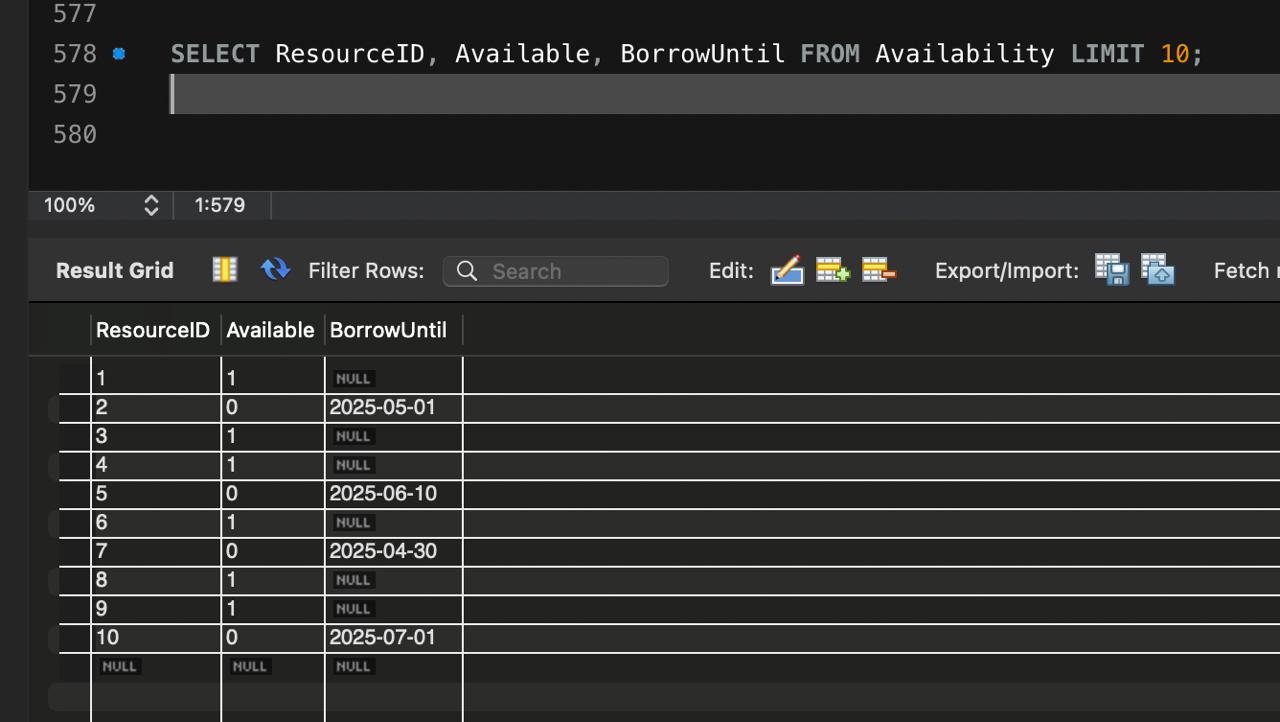
SELECT \* FROM Availability;

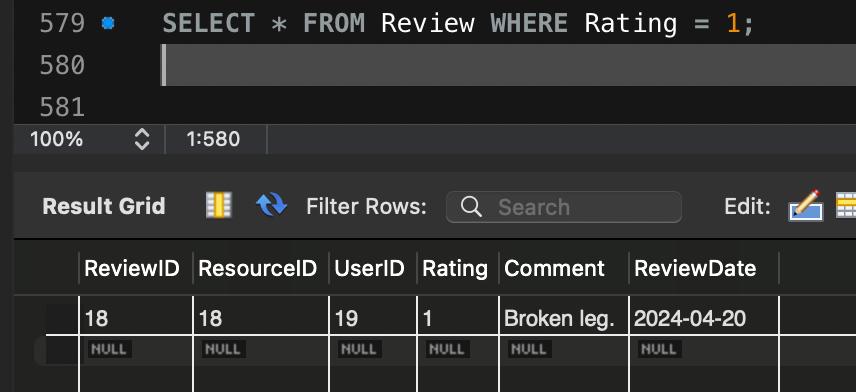
SELECT \* FROM Message;

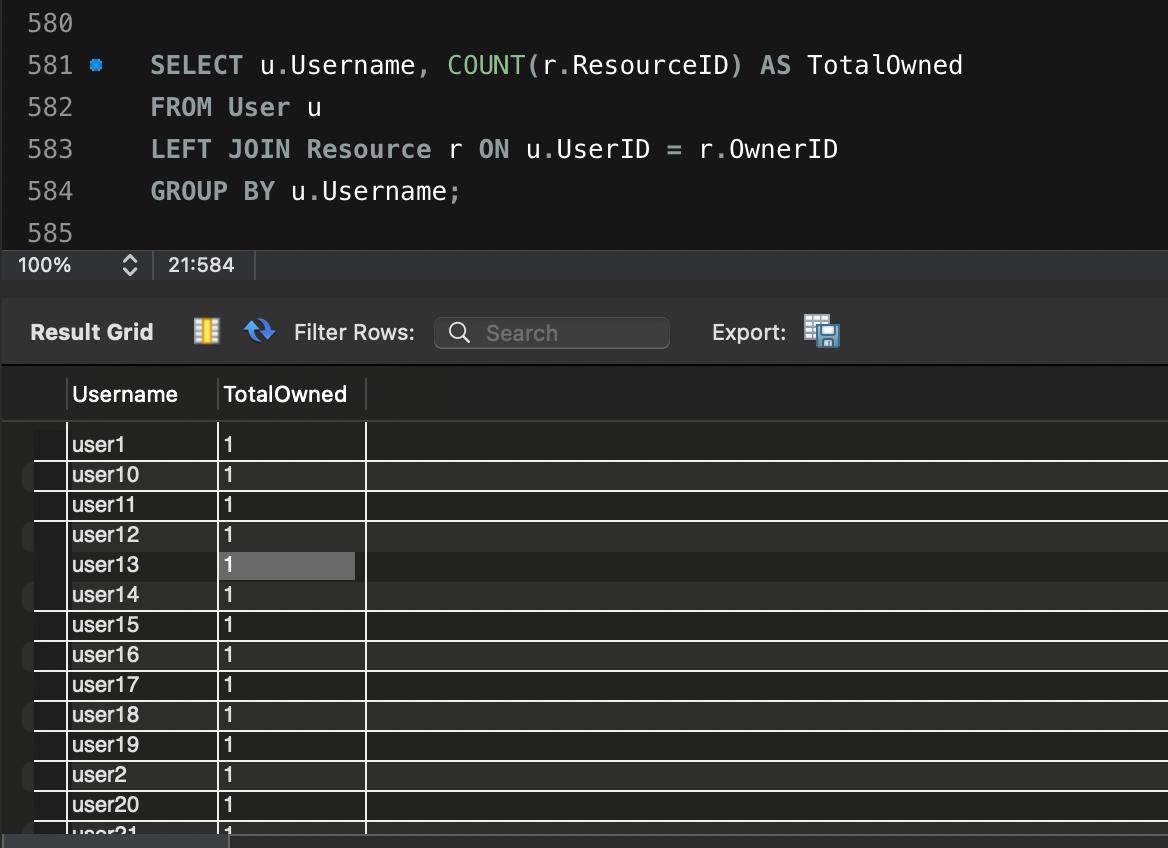
SELECT \* FROM Notification;

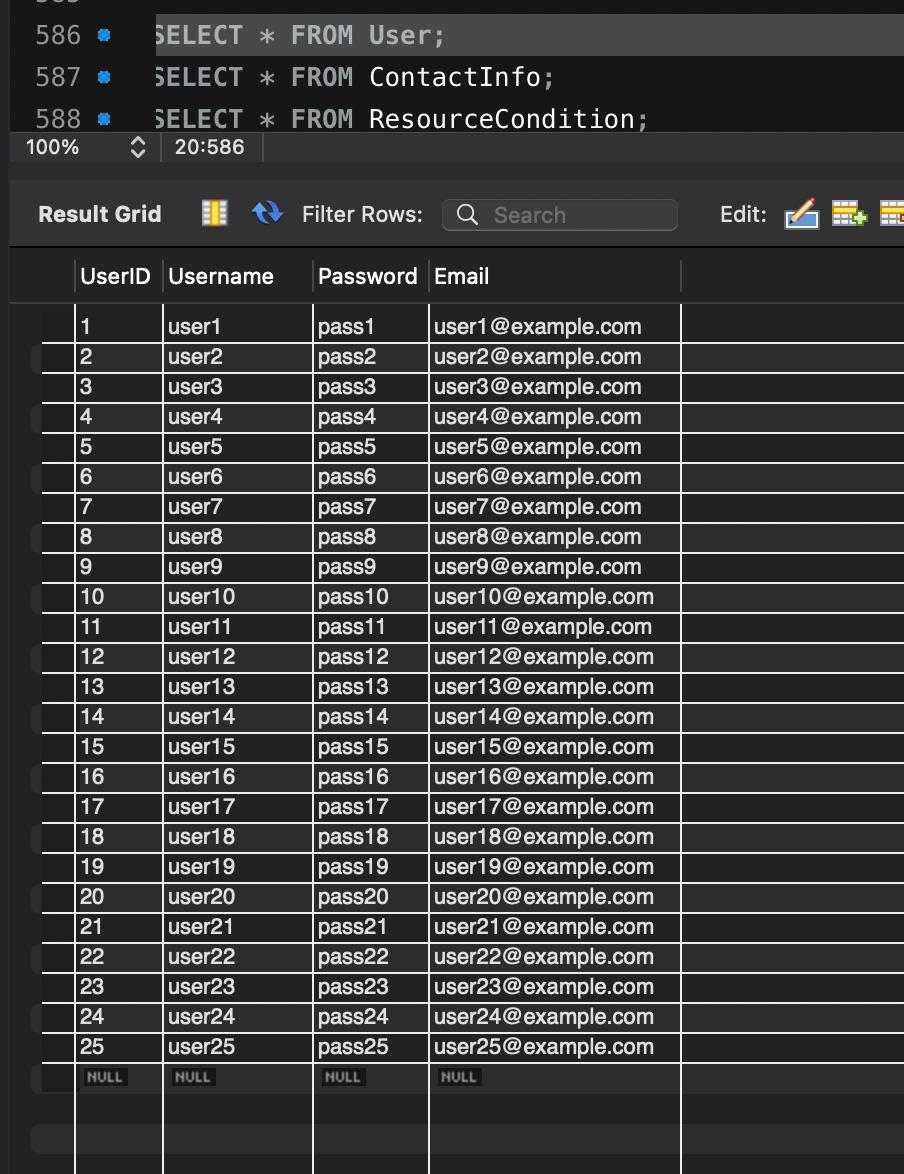
SELECT \* FROM Review;

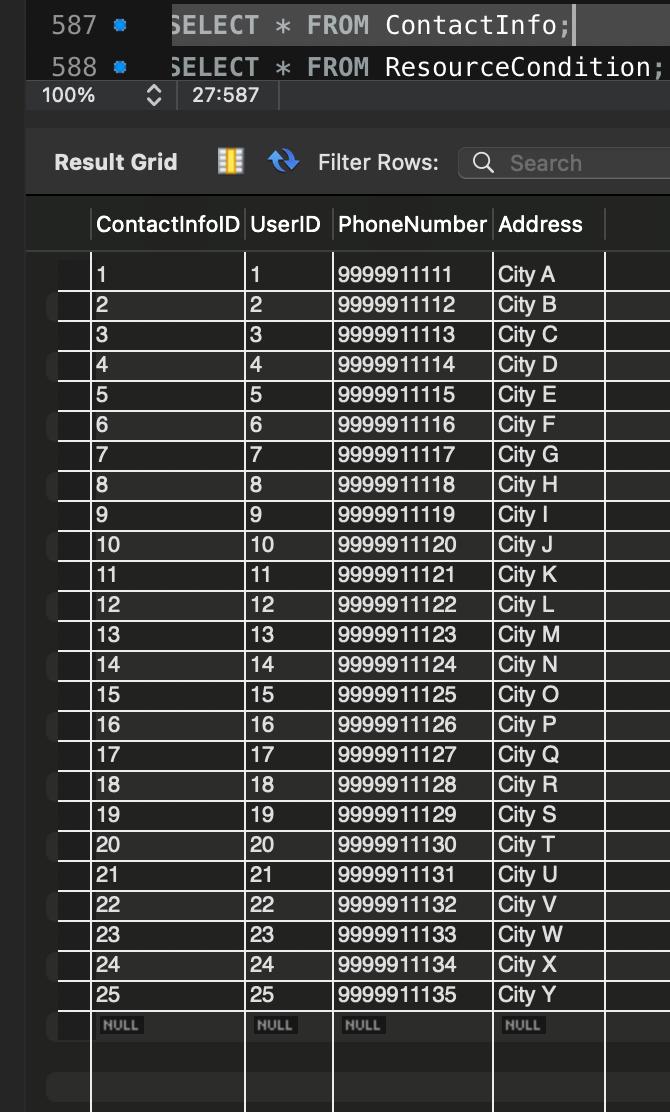
**OUTPUT:**

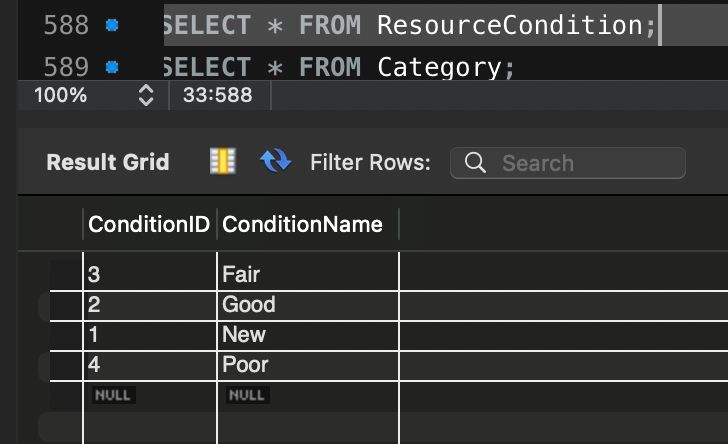
****

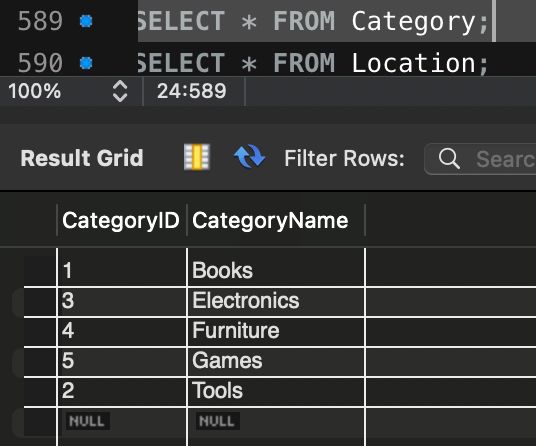
****

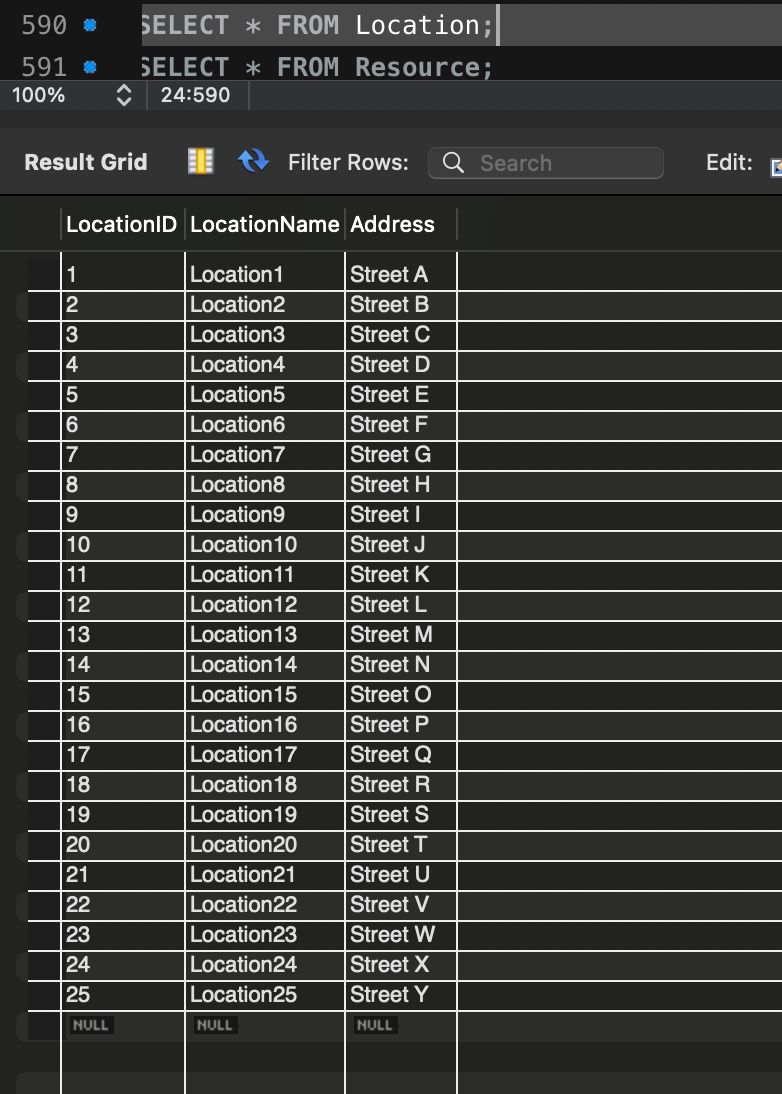


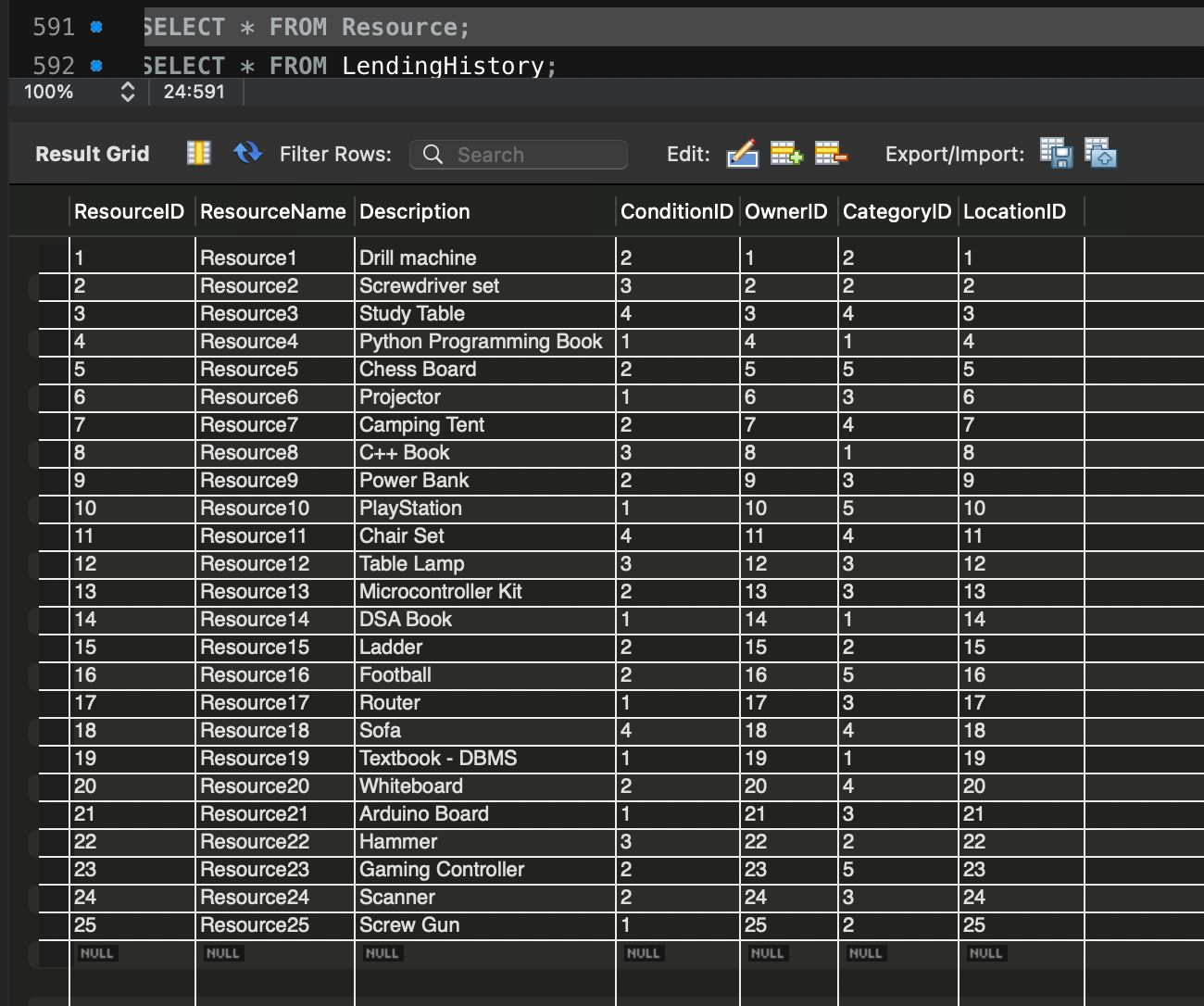
****

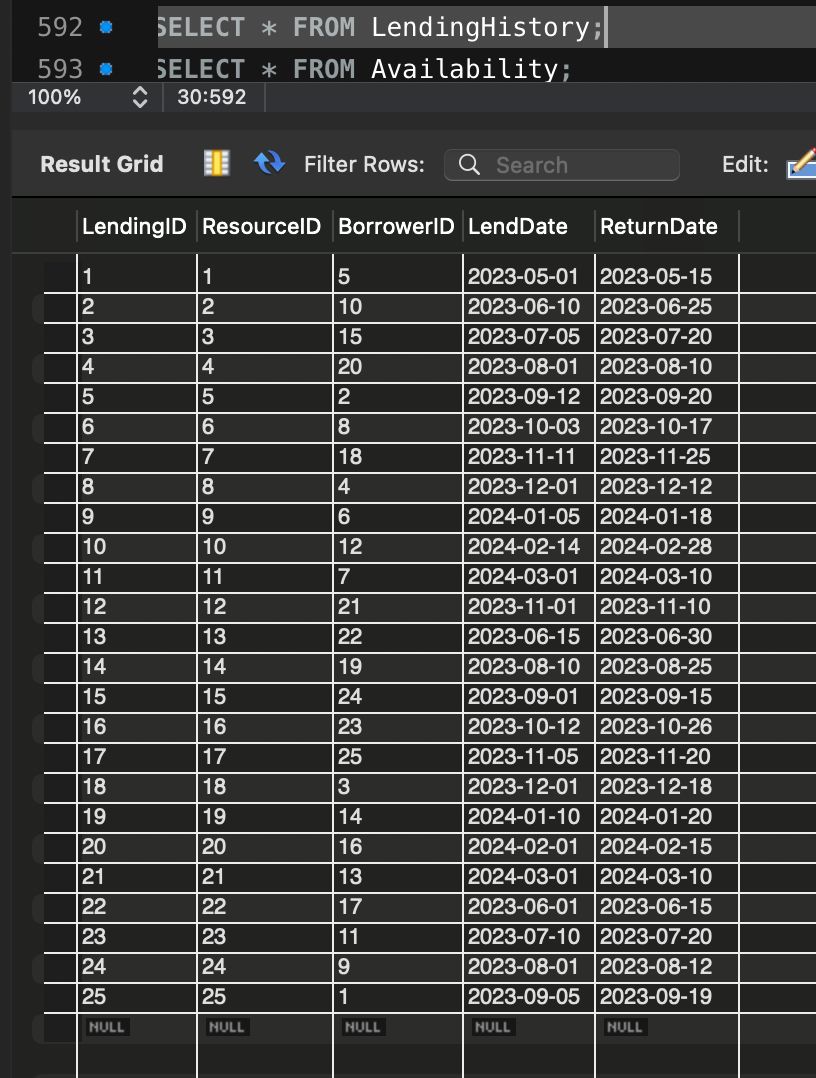
****

****

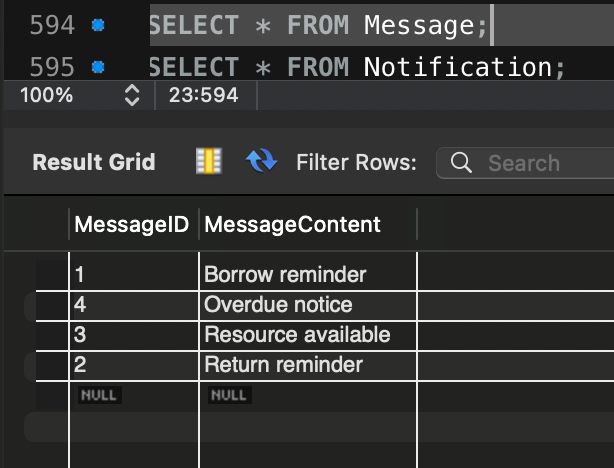
****

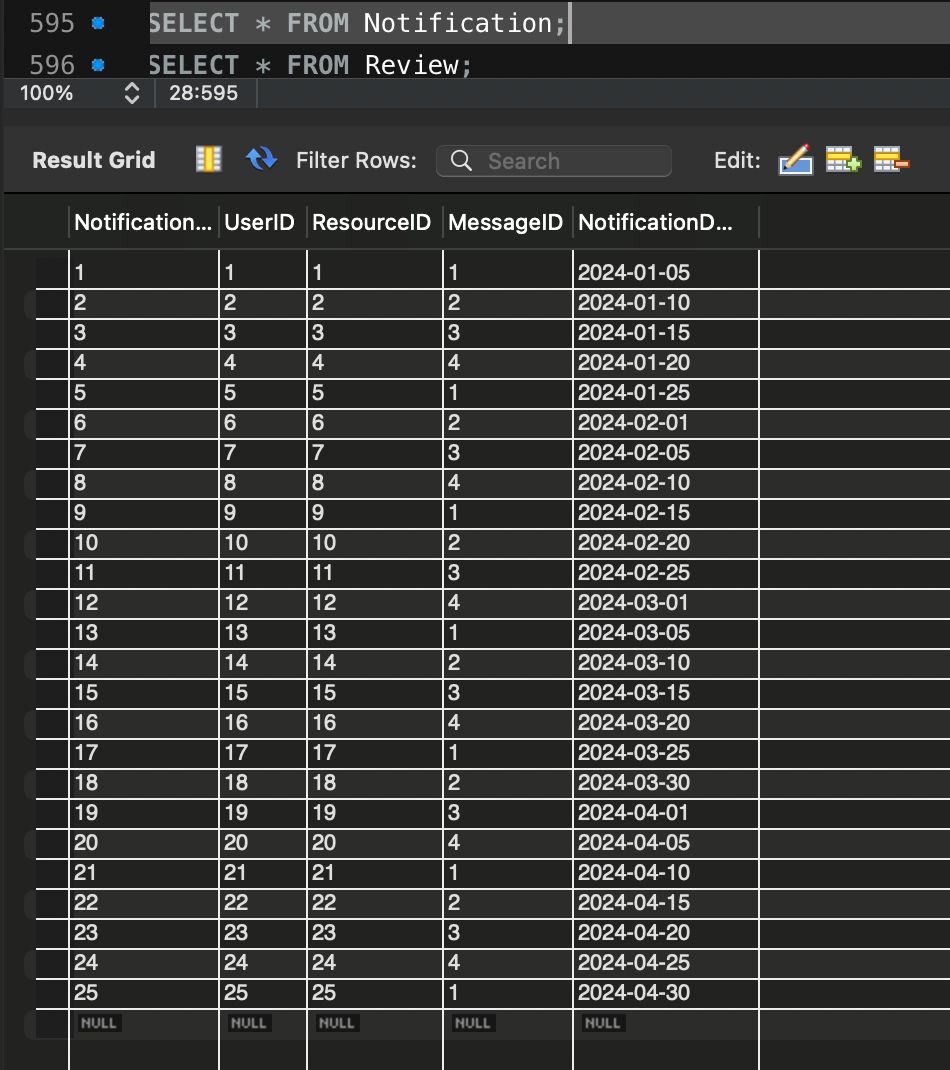
****

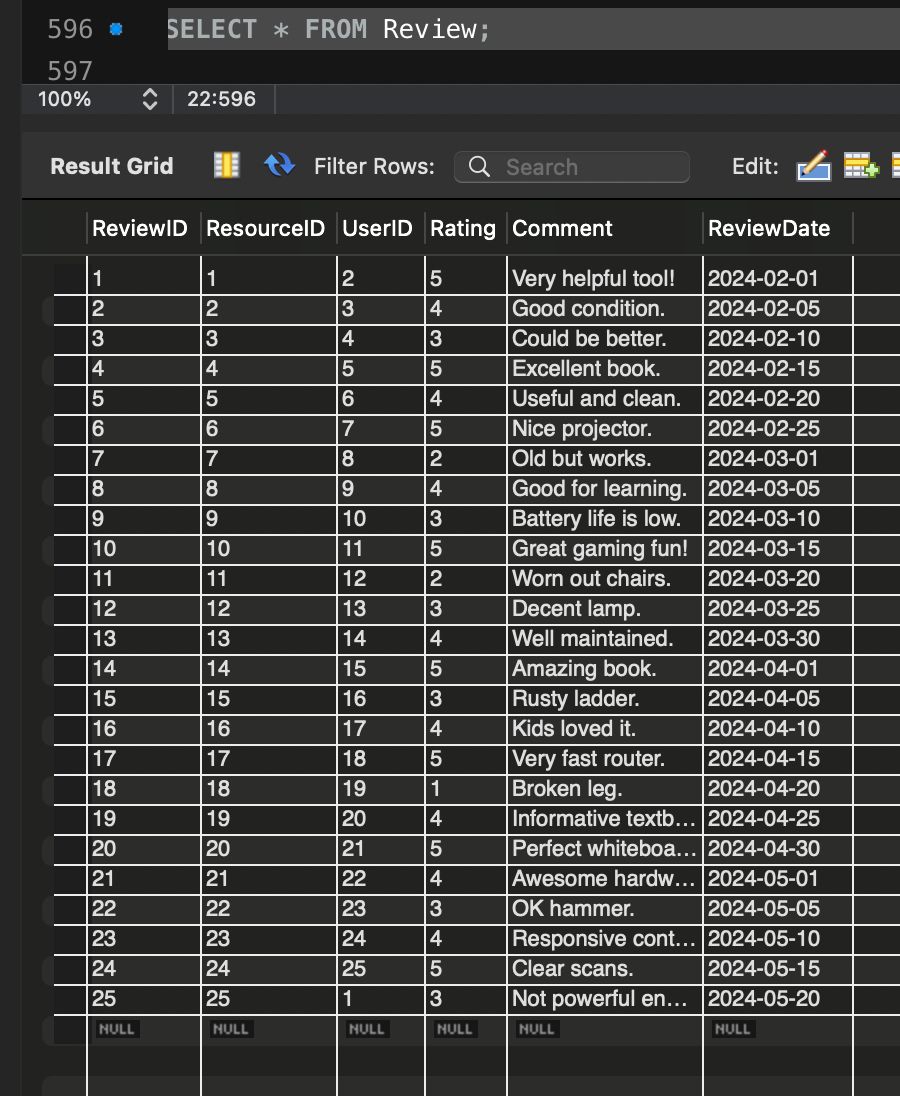
****

****

****

****

****

****

**VI. Project demonstration**

* Tools/software/ libraries used
* Screenshot and Description of the Demonstration of project ( If GUI is made)

**VII. Self -Learning beyond classroom**

:

* What new aspects did you learn on your own ? You have to mention learning beyond the classroom

**VIII. Learning from the Project**

Include learning from the project:

* How this project helped you?

**IX. Challenges Faced**

**X. Conclusion**

* What are the key takeaways from the project?