A REPORT

**ON**

**DBMS – PROJECT ON LIBRARY MANAGEMENT SYSTEM**

BY

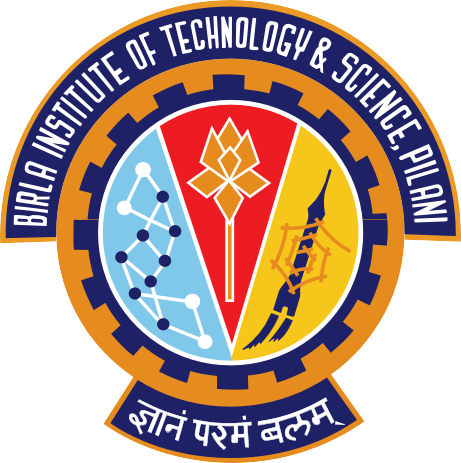
SATVIK JAIN 2020B3A70791P

SIDARTHA SANKARA PATI 2020B2A70687P

AT

IN PARTIAL FULLFILLMENT OF REQUIREMENTS OF

**CS – F212**

****

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, PILANI CAMPUS**

APRIL 2023

**ACKNOWLEDGEMENTS**

We would like to express my sincere gratitude to Amit Dua, the instructor in charge of CS-F212 course, for providing us with the opportunity to work on this project on database design of a library management system.

We would also like to thank the faculty members of the Computer Science department at BITS Pilani for imparting us with the knowledge and skills necessary for completing this project.

We are grateful to the library staff for their cooperation and support throughout the project, without which the completion of this project would not have been possible.

We would also like to acknowledge the contributions of our fellow team members who worked tirelessly and diligently to ensure the success of this project.

Finally, we would like to thank our family and friends for their constant support and encouragement throughout the duration of this project.

**LIST OF FIGURES**

Fig 1. ER Diagram

Fig 2. Relational Diagram

Fig 3. Sample data in each table

Fig 4. Displaying all books in the library

Fig 5. All books currently issued by users query output

Fig 6. Display of total fines of a user

**CHAPTER 1 :ER DIADGRAM AND FORMULATION**

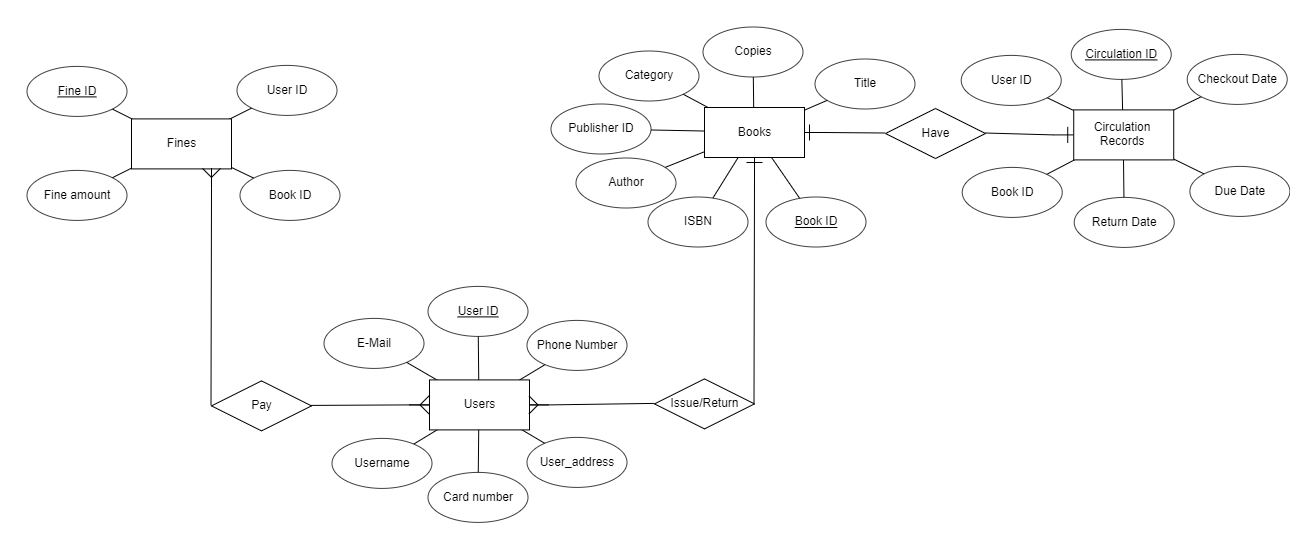


Fig 1.ER Diagram

The ER diagram of our database is shown as above. It has been designed and tweaked to fulfill the users’ needs and avoid additional irrelevant information. The diagram consists of four tables books, fines, circulation records, users and having three relations pay, issue/return and have. Each of these tables have different attributes and various constraints defined on them to ensure no information loss and easy retrieval. The details about various tables and their attributes are mentioned below:

* Fines – Is related to users table by the relation pays. It consists of four attributes User\_ID , Fine\_ID , Fine\_amount and Book\_id. The fine\_id acts as a primary key for this table and user\_id acts as a foreign key in this table. According to user requirements user\_id ,book\_id and fine\_id are not null and fine amount is fixed to greater than 0.
* Circulation records – Is related to books table by the relation have. It consists of six attributes User\_ID , circulation ID , checkout date, due date , return date and Book\_id. The circulation id acts as a primary key for this table and book\_id acts as a foreign key in this table. According to user requirements user\_id , book\_id , circulation\_id checkout date, due date and return date are not null.
* Books – It is the most important table and would be found in almost all real-world databases related to library management systems. Is related to users table by the relation issue/return. It consists of seven attributes title, copies, category, publisher id, author , ISBN and Book\_id. The book id acts as a primary key for this table. According to user requirements all attributes are not null.
* Users –. It is related to books table by the relation issue/return and related to fines by pay. It consists of six attributes phone number, user id, e-mail, card number, username and user\_address. The user id acts as a primary key for this table. According to user requirements all attributes are not null.

**CHAPTER 2 : RELATIONAL DIAGRAM CONVERSION**

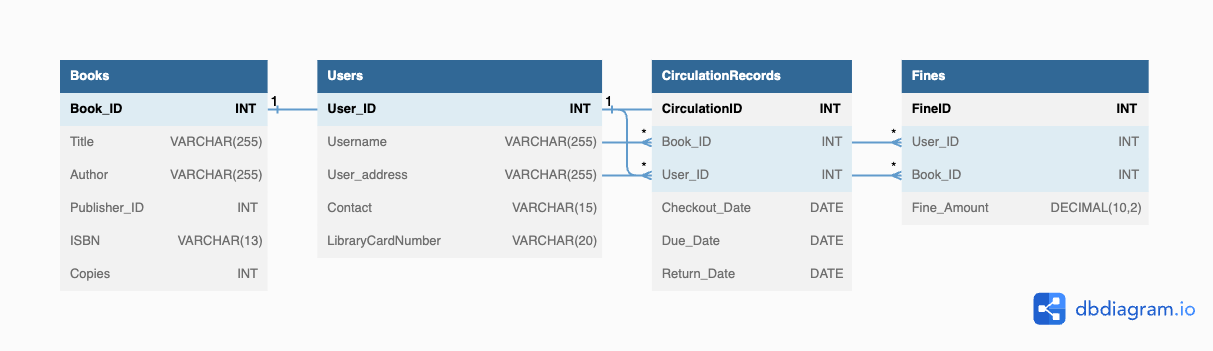


Fig 2. Relational diagram

The diagram shows the tables used in the database, along with their primary keys and foreign keys. The Books table is connected to the CirculationRecords and Fines tables via the Book\_ID field, and the Users table is connected to the CirculationRecords and Fines tables via the User\_ID field. The CirculationRecords table tracks the borrowing history of users, while the Fines table tracks any fines incurred by users. The Books and Users tables contain information about the books and users in the library, respectively. The CirculationRecords table has check constraints to ensure that the required fields are not null, and the Fines table has a check constraint to ensure that the fine amount is not negative. The Books and Users tables also have check constraints to ensure that certain fields are not null.

**CHAPTER 3 : NORMALIZATION**

Initially our design was not in any of the normal forms. Normalization is a process of organizing data in a database to eliminate redundancy and maintain data consistency. It is important in DBMS for the following reasons:

* Reducing data redundancy: Normalization helps to eliminate data redundancy by dividing data into smaller, logical units. This saves storage space and reduces the chances of data inconsistencies.
* Improving data consistency: Normalization ensures that data is consistent and accurate by preventing duplication and reducing the chances of conflicting data.
* Enforcing data integrity: Normalization ensures that each table in a database contains only related data, which makes it easier to enforce data integrity rules.
* Simplifying database design: Normalization simplifies the database design process by breaking down complex data into simpler, more manageable units.

To make use of all these advantages we tried to convert our design to 3rd Normal Form. To do this we also required knowledge of all the functional dependencies in the database(these are stated below). We took the following steps ensuring our database design leading to 3rd NF and ensuring our database performed better:

* Conversion to 1st NF - To make a database conform to the first normal form (1NF), we needed to ensure that each table has a primary key and that each attribute in the table contains only atomic (indivisible) values. This meant that we avoided storing multiple values in a single attribute or column. For eg- we removed multi-valued attributes such as name and phone-number from the user table. If a table had repeating groups, we removed them by creating a new table with a foreign key relationship to the original table. This way we ensured our database was in 1st NF.
* Conversion from 1st to 2nd NF - To convert a database from 1st normal form to 2nd normal form, we need to ensure that all non-key attributes in each table are fully functionally dependent on the primary key. If any non-key attribute is dependent on only a part of the primary key, it should be moved to a separate table with its own primary key. To make all our relations to 2nd NF we kept the primary key of all tables a single attribute, for eg in table users a single attribute users\_id was kept as a primary key. This ensured fully functional dependency of all attributes on the primary key in all tables. This made our database design in 2nd NF.
* Conversion from 2nd to 3rd NF - To convert a database from 2nd normal form to 3rd normal form, we need to identify and remove transitive dependencies. This means that any non-key attribute should not depend on another non-key attribute. To achieve this, we created separate tables for the attributes with transitive dependencies and used foreign keys to link them to the original table(for eg fines was initially attribute of Users as each user will have a particular amount of fine but this created transitive dependencies so we created another table fines with user id as a foreign key). Now by doing this and according to our functional dependencies our database was in 3rd NF.

**CHAPTER 4: FUNCTIONAL DEPENDENCIES**

* Fine id-> fine amount
* User id->Phone number
* User id->Email
* User id->Username
* User id->Card\_Number
* User id->User address
* Book id->ISBN
* Book id->Author
* Book id->Copies
* Book id->Title
* Book id->Publisher id
* Book id-> Category
* Circulation id->Return date
* Circulation id->Due date
* Circulation id->checkout date

**Aside from these all trivial dependencies hold( ie A->A)**

**CHAPTER 5 : QUERIES AND OUTPUTS**

After creation of our relational diagram, we created our source code for the SQL database. The source code was kept as simple as possible and is attached below:

-- Create Books table

CREATE TABLE Books (

Book\_ID INT NOT NULL PRIMARY KEY,

Title VARCHAR(255) NOT NULL,

Author VARCHAR(255) NOT NULL,

Publisher\_ID INT NOT NULL,

ISBN VARCHAR(13) NOT NULL,

Copies INT NOT NULL

);

-- Create Users table

CREATE TABLE Users (

User\_ID INT NOT NULL PRIMARY KEY,

Username VARCHAR(255) NOT NULL,

User\_address VARCHAR(255) NOT NULL,

Contact VARCHAR(15) NOT NULL,

LibraryCardNumber VARCHAR(20) NOT NULL

);

-- Create CirculationRecords table

CREATE TABLE CirculationRecords (

CirculationID INT NOT NULL PRIMARY KEY,

Book\_ID INT NOT NULL,

User\_ID INT NOT NULL,

Checkout\_Date DATE NOT NULL,

Due\_Date DATE NOT NULL,

Return\_Date DATE DEFAULT '1000-01-01',

FOREIGN KEY (Book\_ID) REFERENCES Books(Book\_ID),

FOREIGN KEY (User\_ID) REFERENCES Users(User\_ID),

CHECK (Book\_ID IS NOT NULL),

CHECK (User\_ID IS NOT NULL),

CHECK (Checkout\_Date IS NOT NULL),

CHECK (Due\_Date IS NOT NULL),

CHECK (Return\_Date IS NOT NULL)

);

-- Create Fines table

CREATE TABLE Fines (

FineID INT NOT NULL PRIMARY KEY,

User\_ID INT NOT NULL,

Book\_ID INT NOT NULL,

Fine\_Amount DECIMAL(10,2) NOT NULL,

FOREIGN KEY (User\_ID) REFERENCES Users(User\_ID),

FOREIGN KEY (Book\_ID) REFERENCES Books(Book\_ID),

CHECK (User\_ID IS NOT NULL),

CHECK (Book\_ID IS NOT NULL),

CHECK (Fine\_Amount >= 0)

);

-- Add check constraints for Books table

ALTER TABLE Books ADD CHECK (Title IS NOT NULL);

ALTER TABLE Books ADD CHECK (Author IS NOT NULL);

-- Add check constraints for Users table

ALTER TABLE Users ADD CHECK (Username IS NOT NULL);

ALTER TABLE Users ADD CHECK (User\_address IS NOT NULL);

After creation of the structure of these tables we populated the database with some sample data as shown in the figure below:

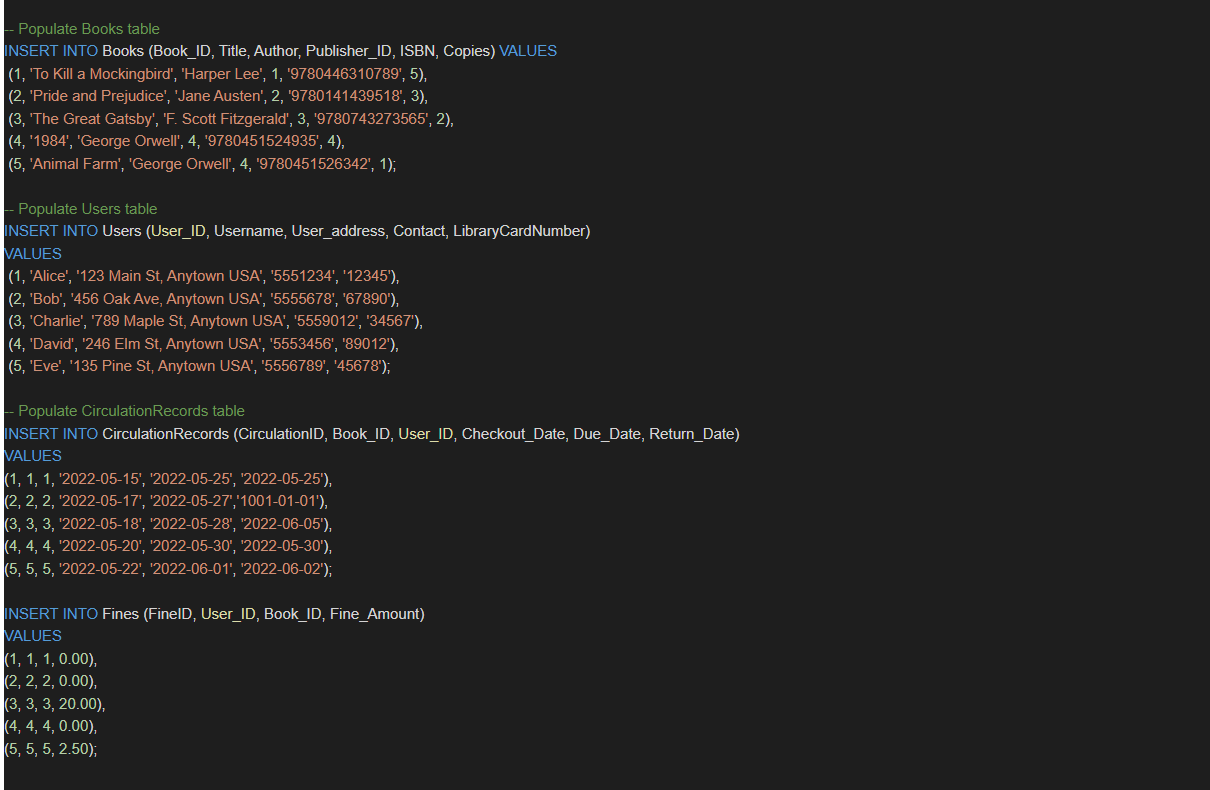


Fig 3. Sample data in each table

After populating the tables, we ran a few queries as requested by the user:

1. Display all books available in library:

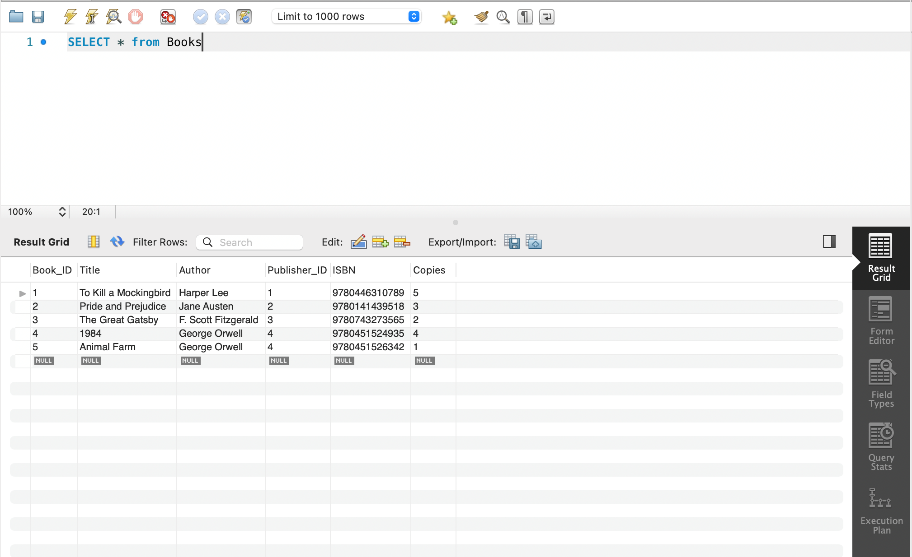


Fig 4. Displaying all books in the library

1. Display all the books currently issued by users. The query for this is:

SELECT Books.\*

FROM Books

INNER JOIN CirculationRecords ON Books.Book\_ID = CirculationRecords.Book\_ID

WHERE CirculationRecords.Return\_Date = '1001-01-01';

Here's a breakdown of the query:

* SELECT Books.\*: This statement selects all columns (denoted by the asterisk) from the "Books" table.
* FROM Books: This specifies that the data is being pulled from the "Books" table.
* INNER JOIN CirculationRecords ON Books.Book\_ID = CirculationRecords.Book\_ID: This specifies that the "Books" table is being joined with the "CirculationRecords" table using the "Book\_ID" column as the join condition.
* WHERE CirculationRecords.Return\_Date = '1001-01-01': This filters the results by only selecting rows from the joined tables where the "Return\_Date" column in the "CirculationRecords" table is equal to '1001-01-01'.
* Therefore, this query will return all columns from the "Books" table for books that have a corresponding record in the "CirculationRecords" table where the return date is equal to '1001-01-01'. By default, all the books that are issued but are not returned yet have a return date vale “1001-01-01”. So we would get all books currently issued by the users.

A figure of the output of the following query is shown below:

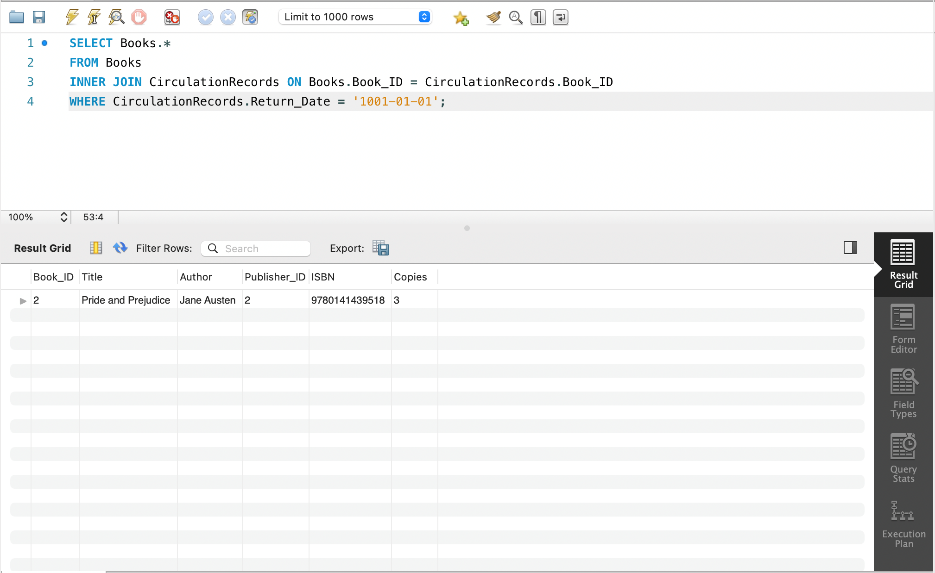


Fig 5. All books currently issued by users query output

1. Query to calculate all the total fines of a user. The query for this is:

SELECT SUM(Fines.Fine\_Amount) AS TotalFines

FROM Fines

INNER JOIN CirculationRecords ON Fines.Book\_ID = CirculationRecords.Book\_ID

AND Fines.User\_ID = CirculationRecords.User\_ID

INNER JOIN Users ON CirculationRecords.User\_ID = Users.User\_ID

The idea behind this query is:

* The FROM clause specifies that we're querying the Fines table, which presumably contains information about each fine, including the amount of the fine (Fine\_Amount).
* The INNER JOIN clauses join the Fines table with the CirculationRecords and Users tables.
* The ON clause of the first INNER JOIN specifies that we want to join the Fines and CirculationRecords tables on two conditions: that the Book\_ID in Fines matches the Book\_ID in CirculationRecords, and that the User\_ID in Fines matches the User\_ID in CirculationRecords. This is to ensure that we're only looking at fines that were incurred by users who checked out books from the library.
* The ON clause of the second INNER JOIN specifies that we want to join the CirculationRecords and Users tables on the User\_ID column. This is so we can get information about the users who incurred the fines (e.g., their names, addresses, etc.).
* The SELECT clause specifies that we want to select the sum of all Fine\_Amount values in the resulting joined table, and give it an alias of TotalFines.
* Putting it all together, the query will return a single row with a single column, which contains the total amount of fines paid by users for overdue books, calculated by summing up the Fine\_Amount values in the Fines table that match the specified conditions in the INNER JOIN clauses.

A sample run of this query with output is attached as shown below:

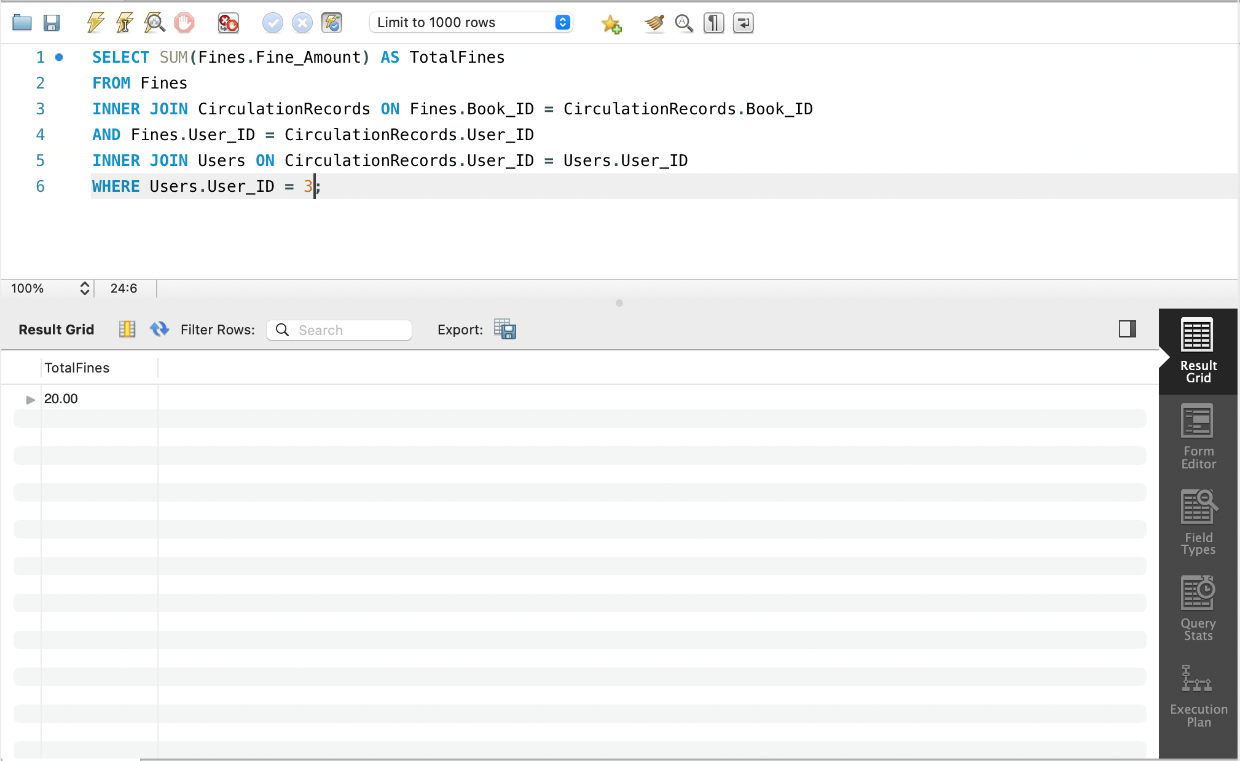


Fig 6. Display of total fines of a user

LINK FOR VIDEO RECORDINGS: <https://drive.google.com/drive/folders/1h8z3gPx9A4Mo9gmpN3UVkMzcpOI7_gq3>