Library Database Management System
Database Implementation and Testing

Version 1.0

Library DBMS	Version: 1.0
Database Implementation and Testing	Date: 11/30/2024
EECS447_DBMS_Implement_Test	

Table of Contents

1. Introduction

    1.1 Project Overview

    1.2 Scope

    1.3 Glossary

2. Platform Selection
3. Data Population
4. Functionality Testing
5. Testing and Validation
6. Documentation

Database Implementation and Testing

1. Introduction

1.1 Project Overview

The Library Database Management System project will deliver an efficient, user-friendly, and secure database system to support small library operations. The DBMS will enable seamless management of loanable items (books, digital media, magazines), enforce borrowing rules based on diverse membership categories, and generate insightful reports. By integrating modern database design principles, the system will streamline item tracking, membership management, and financial oversight.

1.2 Scope

The Library Database Management System (LMS) project will design and implement a relational database for a small library to manage loanable items, memberships, borrowing rules, and generate reports. It will model entities like books, digital media, magazines, and clients, and enforce constraints like borrowing limits and fees based on membership type. The system will include features for managing loans, returns, and client accounts, with user interfaces for both staff and clients. Advanced queries will generate financial and activity reports, while concurrency and transaction management will ensure seamless multi-user operations. The LMS will be developed through domain modeling, database design, and implementation phases, ensuring functionality and data integrity.

1.3 Glossary

LMS - Library Management System
DBMS - Database Management System

2. Platform Selection

The Platform we have chosen to create our database on is **mySQL**. MySQL was chosen for the library DBMS due to its reliability, performance, and cost-effectiveness as a free, open-source system. It is beginner-friendly, well-documented, and widely used in educational and professional settings, making it ideal for those with little prior experience. MySQL supports SQL standards and data integrity through constraints which are crucial for a structured library system.

Along with using MySQL, we also used the IDE (Integrated Development Environment) **DataGrip**, integrated with MySQL, to create and manage the database. DataGrip offers a user-friendly interface and robust features like intelligent code completion, error detection, and powerful debugging tools, making it an ideal choice for database development. Its seamless integration with GitHub allows for efficient version control and collaboration on SQL scripts. The platform supports features like data visualization, schema management, and easy query execution, which streamline database design and testing.

3. Data Population

The data for our database was generated using an online tool called **Fabricate by Mockaroo**, available here. This tool leverages AI to create realistic data based on a provided schema or DDL statements. For our project, we input our Library Database DDL statements into the tool, which generated 100 fake data points for each table (excluding tables with limited domains as keys). While the initial data required some adjustments, we fine-tuned it to produce accurate `.csv` files for each relation in our database.

4. Functionality Testing

For our Functionality Testing we decided on 18 unique queries to show the capabilities and accuracy of our database. These test cases, including their output, are all shown below.

Query 1: Insert data into the Item Relation

- This query shows that the library can add new items to the database when they are received.
- The output shows that the items specified in the query were added to the Item Relation.

```
INSERT INTO Item (Item_ID, Price, Year, Availability_Status, Item_Type)
VALUES
    ( Item_ID '101', Price 500, Year 2021, Availability_Status 'Available', Item_Type 'Book'),
    ( Item_ID '102', Price 300, Year 2020, Availability_Status 'Checked Out', Item_Type 'Magazine'),
    ( Item_ID '103', Price 150, Year 2022, Availability_Status 'Reserved', Item_Type 'DVD');
```

96	12	24	2009	Purchased	Music
97	11	49	2006	Purchased	Music
98	103	150	2022	Reserved	DVD
99	102	300	2020	Checked Out	Magazine
100	101	500	2021	Available	Book
101	100	19	2005	Checked Out	Magazine
102	10	34	2024	Checked Out	Music

Query 2: Inserting data into the Book Relation

- This query shows that the library can add new items to the database when they are received.
- The output shows that the item specified in the query was added to the Book Relation

```
INSERT INTO Book (Item_ID, ISBN, Title, Subject)
VALUES
    ( Item_ID '101', ISBN '9780131103627', Title 'Introduction to Algorithms', Subject 'Computer Science');
```

1	100	7512933088888	Legends Rising	Thriller
2	101	9780131103627	Introduction to Alg…	Computer Science
3	11	9751378040098	Dreams of Infinity	Young Adult

Query 3: Inserting data into Magazine Relation

- This query shows that the library can add new items to the database when they are received.
- The output shows that the item specified in the query was added to the Magazine Relation.

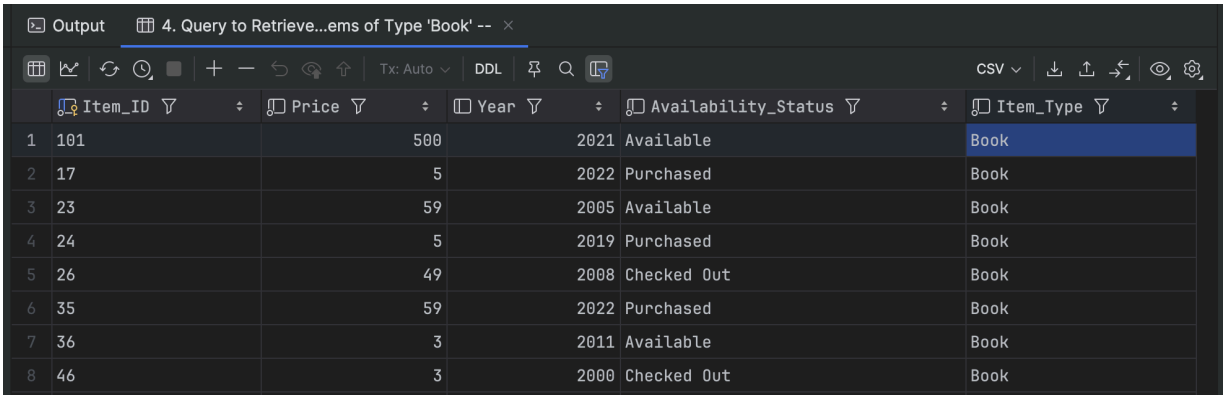
```
INSERT INTO Magazine (Item_ID, ISSN, Name, Edition, Publish_Date)
VALUES
    ( Item_ID '102', ISSN '1234567890123', Name 'National Geographic', Edition 45, Publish_Date '2020-05-01');
```

1	102	1234567890123	National Geographic	45	2020-05-01
2	12	2675-2913	Kid's Corner	65	2019-04-10
3	15	4960-4888	Digital Times	78	1996-04-02

Query 4: Retrieve all items of type ‘Book’

- This query shows that the library can retrieve specific selections of data from the database.
- The output shows all items within the database of type ‘Book’.

```
SELECT *
FROM Item
WHERE Item_Type = 'Book';
```



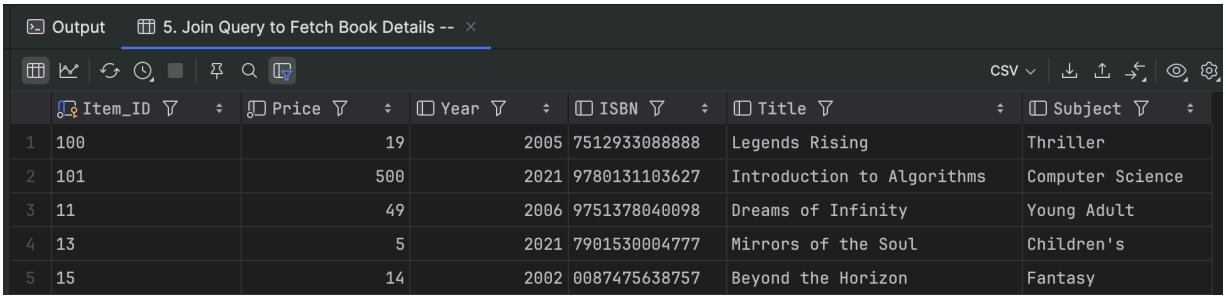
Output | 4. Query to Retrieve...ems of Type 'Book' --

	Item_ID	Price	Year	Availability_Status	Item_Type
1	101	500	2021	Available	Book
2	17	5	2022	Purchased	Book
3	23	59	2005	Available	Book
4	24	5	2019	Purchased	Book
5	26	49	2008	Checked Out	Book
6	35	59	2022	Purchased	Book
7	36	3	2011	Available	Book
8	46	3	2000	Checked Out	Book

Query 5: Using Join to retrieve Book details

- This query shows that the library can retrieve specific selections of data using join.
- The output shows all attributes related to a specific item 'Book'

```
SELECT i.Item_ID, i.Price, i.Year, b.ISBN, b.Title, b.Subject
FROM Item i
JOIN Book b ON i.Item_ID = b.Item_ID;
```



Output | 5. Join Query to Fetch Book Details --

	Item_ID	Price	Year	ISBN	Title	Subject
1	100	19	2005	7512933088888	Legends Rising	Thriller
2	101	500	2021	9780131103627	Introduction to Algorithms	Computer Science
3	11	49	2006	9751378040098	Dreams of Infinity	Young Adult
4	13	5	2021	7901530004777	Mirrors of the Soul	Children's
5	15	14	2002	0087475638757	Beyond the Horizon	Fantasy

Query 6: Retrieve all items marked 'Available'

- This query shows that the library can retrieve specific selections of data.
- The output shows all items where the availability status is 'Available'.

```sql
SELECT Item_ID, Item_Type, Price, Availability_Status
FROM Item
WHERE Availability_Status = 'Available';
```

Output | 6. Query to Find Available Items --

	Item_ID	Item_Type	Price	Availability_Status
1	101	Book	500	Available
2	18	Digital Media	24	Available
3	21	Magazine	15	Available
4	22	Digital Media	49	Available
5	23	Book	59	Available
6	31	DVD	22	Available

Query 7: Count Items by Type

- This query shows that the library can display specific information about the data.
- The output shows the number of each item based on the item type.

```sql
SELECT Item_Type, COUNT(*) AS Item_Count
FROM Item
GROUP BY Item_Type;
```

Output | 7. Query to Count Items by Type --

	Item_Type	Item_Count
1	DVD	22
2	Music	13
3	Magazine	22
4	Book	17
5	Digital Media	29

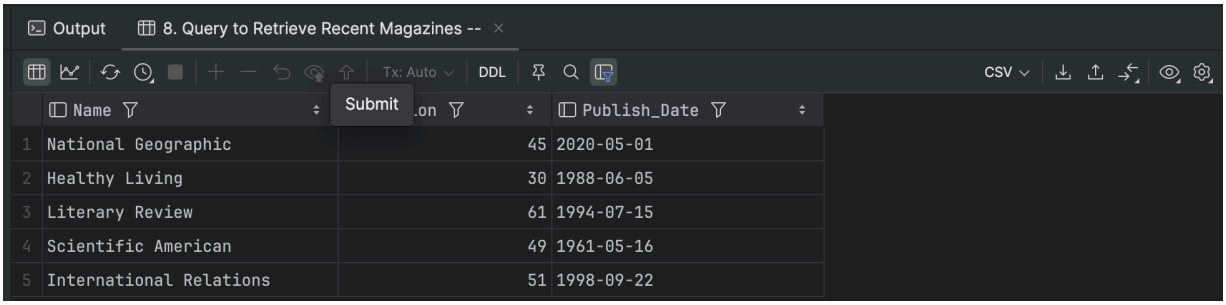
Item_Type: char(13)

Query 8: Retrieve recently added magazines

- This query shows that the library can retrieve specific selections of data.

- The output shows recently published magazines based on input specified.

```
SELECT m.Name, m.Edition, m.Publish_Date
FROM Magazine m
JOIN Item i ON m.Item_ID = i.Item_ID
WHERE i.Year >= 2020;
```



	Name	Edition	Publish_Date
1	National Geographic	45	2020-05-01
2	Healthy Living	30	1988-06-05
3	Literary Review	61	1994-07-15
4	Scientific American	49	1961-05-16
5	International Relations	51	1998-09-22

Query 9: Update a relation

- This query shows that the library can update relations.
- The output shows the updated status of the item in the relation.

```
UPDATE Item
SET Availability_Status = 'Purchased'
WHERE Item_ID = '103';
```

97	11	49	2006	Purchased	Music
98	103	150	2022	Purchased	DVD
99	102	300	2020	Checked Out	Magazine

Query 10: Delete data from a relation

- This query shows that the library can delete data from a specified relation.
- The output shows the updated relation with the specified information deleted.

```
DELETE FROM Has_Membership
WHERE User_ID = 100;
```

	User_ID	Membership_Type
1	99	Super
2	97	Standard
3	88	Super
4	85	Basic
5	84	Standard
6	77	Advanced
7	74	Advanced

Query 11: Retrieve all books published after 2015

- This query shows that the library can retrieve data from a specified relation.
- This output shows the books published after 2015.

```
SELECT b.Title, b.ISBN, i.Year
FROM Book b
JOIN Item i ON b.Item_ID = i.Item_ID
WHERE i.Year > 2015;
```

	Title	ISBN	Year
1	Introduction to Algorithms	9780131103627	2021
2	Mirrors of the Soul	7901530004777	2021
3	Mirrors of the Soul	6406649445322	2024
4	The Forgotten Path	0798326522385	2019
5	Revelations in Darkness	7860567199440	2017
6	Beyond the Horizon	9107131360447	2020
7	Adventures of the Mind	2566138484731	2020
8	Whispers in the Wind	8735674773691	2023
9	Adventures of the Mind	9347479839542	2022

9 rows

Query 12: List all items marked ‘Reserved’

- This query shows that the library can retrieve specific selections of data.
- The output shows all items where the availability status is ‘Reserved’.

```
SELECT Item_ID, Item_Type, Price, Year, Availability_Status
FROM Item
WHERE Availability_Status = 'Reserved';
```

	Item_ID	Item_Type	Price	Year	Availability_Status
1	1	DVD	12	2002	Reserved
2	13	Magazine	5	2021	Reserved
3	14	Digital Media	3	2009	Reserved
4	15	DVD	14	2002	Reserved
5	2	DVD	39	2006	Reserved
6	3	DVD	29	2019	Reserved
7	39	Digital Media	24	2021	Reserved
8	41	Digital Media	3	2001	Reserved

Query 13: Find the most expensive item

- This query shows that the library can retrieve data from specified relations.
- The output shows the items listed by price in descending order.

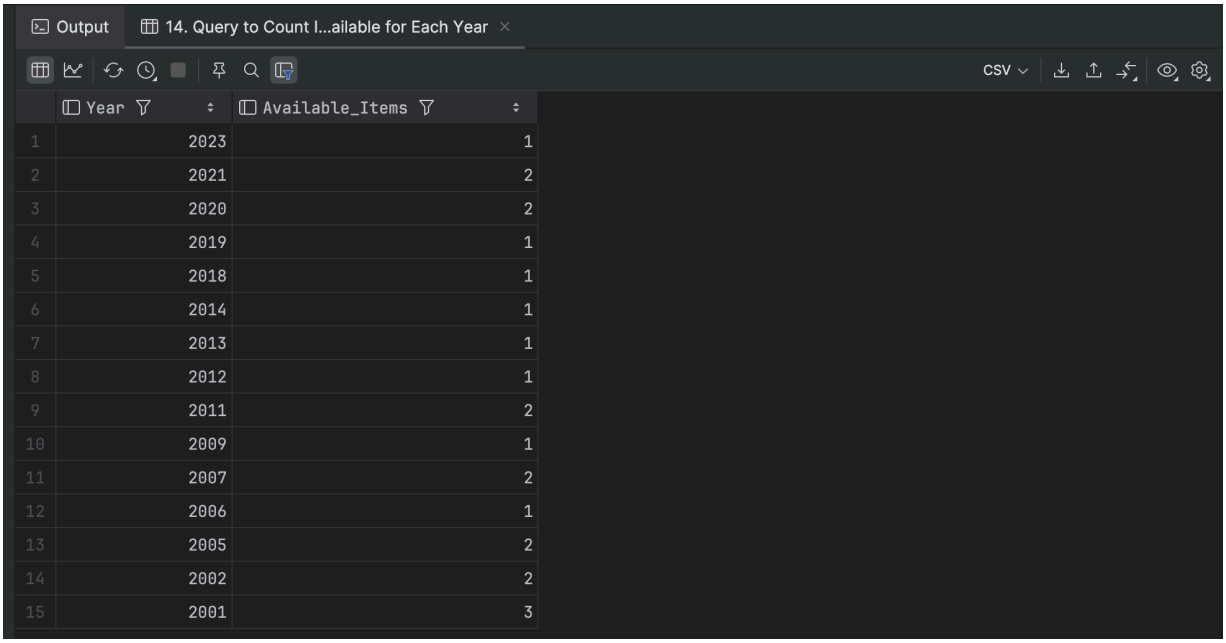
```
SELECT Item_ID, Item_Type, Price
FROM Item
ORDER BY Price DESC
LIMIT 1;
```

	Item_ID	Item_Type	Price
1	101	Book	500

Query 14: Count the items available for each year

- This query shows that the library can retrieve data and count it from specified relations.
- The output shows the number of items available for each year in descending order.

```
SELECT Year, COUNT(*) AS Available_Items
FROM Item
WHERE Availability_Status = 'Available'
GROUP BY Year
ORDER BY Year DESC;
```



Output | 14. Query to Count I...ailable for Each Year

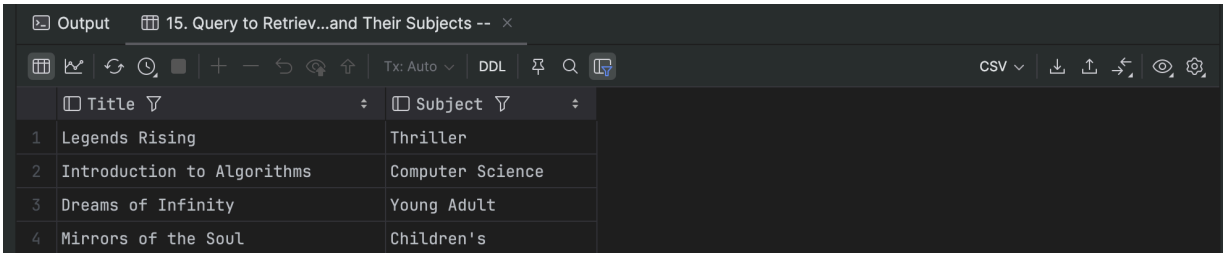
CSV

	Year	Available_Items
1	2023	1
2	2021	2
3	2020	2
4	2019	1
5	2018	1
6	2014	1
7	2013	1
8	2012	1
9	2011	2
10	2009	1
11	2007	2
12	2006	1
13	2005	2
14	2002	2
15	2001	3

Query 15: Retrieve book titles and their subjects

- This query shows that the library can retrieve specified attributes of a relation.
- The output shows a list of all book titles and their subjects.

```
SELECT b.Title, b.Subject
FROM Book b;
```



Output | 15. Query to Retriev...and Their Subjects --

Tx: Auto | DDL | CSV

	Title	Subject
1	Legends Rising	Thriller
2	Introduction to Algorithms	Computer Science
3	Dreams of Infinity	Young Adult
4	Mirrors of the Soul	Children's

Query 16: Update the price of DVDs released before 2020

- This query shows that the library can update specified data within a relation.
- The output shows the updated prices of DVDs released before 2020.

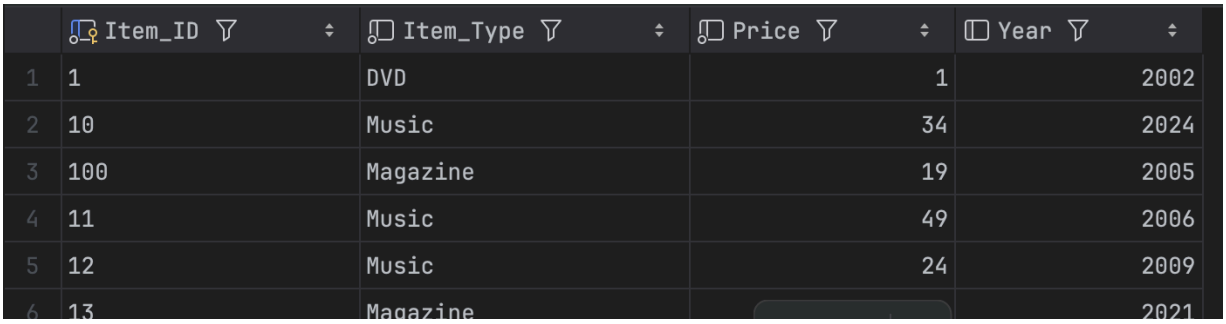
```
UPDATE Item
SET Price = Price * .1
WHERE Item_Type = 'DVD' AND Year < 2020;
```

	Item_ID	Price	Year	Availability_Status	Item_Type
70	103	150	2022	Purchased	DVD
71	15	2	2002	Reserved	DVD
72	2	4	2006	Reserved	DVD
73	25	49	2023	Checked Out	DVD
74	3	3	2019	Reserved	DVD
75	31	2	2001	Available	DVD
76	37	2	2013	Checked Out	DVD
77	44	0	2014	Purchased	DVD
78	45	2	2006	Available	DVD
79	48	49	2020	Available	DVD

Query 17: Find all items with price below 50

- This query shows that the library can retrieve specified information across all relations.
- The output shows all items with a price of 50 or less.

```
SELECT Item_ID, Item_Type, Price, Year
FROM Item
WHERE Price < 50;
```

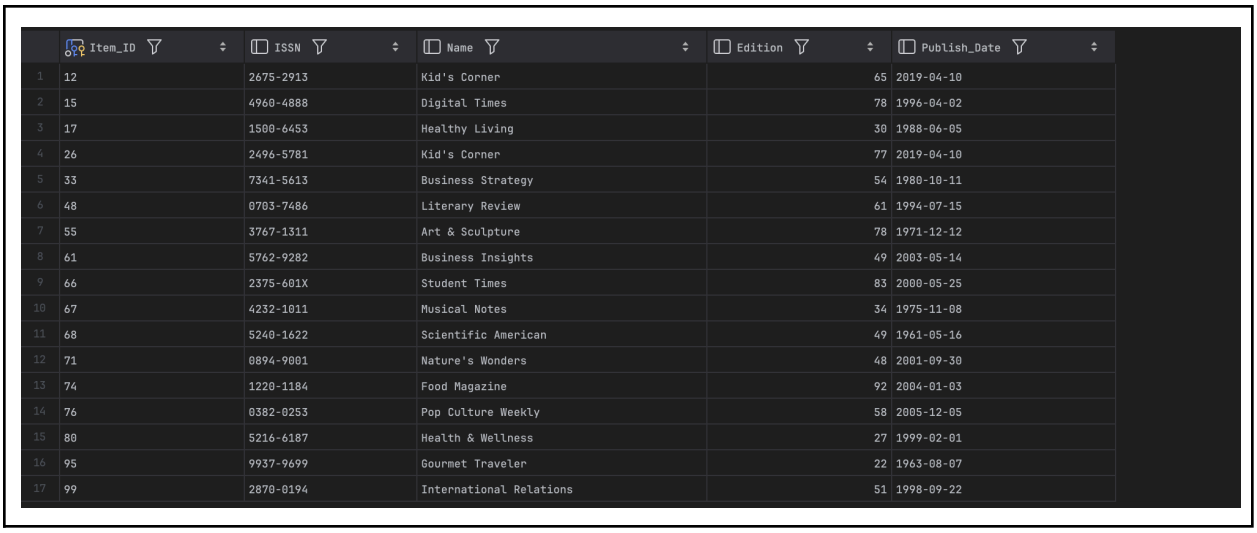


	Item_ID	Item_Type	Price	Year
1	1	DVD	1	2002
2	10	Music	34	2024
3	100	Magazine	19	2005
4	11	Music	49	2006
5	12	Music	24	2009
6	13	Magazine		2021

Query 18: Delete all records of a specific magazine

- This query shows that the library can delete information from a specified relation.
- The output shows the magazine relation with the specified information deleted.

```
DELETE FROM Magazine
WHERE Name = 'National Geographic' AND Edition = 45;
```



	Item_ID	ISSN	Name	Edition	Publish_Date
1	12	2675-2913	Kid's Corner	65	2019-04-10
2	15	4960-4888	Digital Times	78	1996-04-02
3	17	1500-6453	Healthy Living	30	1988-06-05
4	26	2496-5781	Kid's Corner	77	2019-04-10
5	33	7341-5613	Business Strategy	54	1980-10-11
6	48	0703-7486	Literary Review	61	1994-07-15
7	55	3767-1311	Art & Sculpture	78	1971-12-12
8	61	5762-9282	Business Insights	49	2003-05-14
9	66	2375-601X	Student Times	83	2000-05-25
10	67	4232-1011	Musical Notes	34	1975-11-08
11	68	5240-1622	Scientific American	49	1961-05-16
12	71	0894-9001	Nature's Wonders	48	2001-09-30
13	74	1220-1184	Food Magazine	92	2004-01-03
14	76	0382-0253	Pop Culture Weekly	58	2005-12-05
15	80	5216-6187	Health & Wellness	27	1999-02-01
16	95	9937-9699	Gourmet Traveler	22	1963-08-07
17	99	2870-0194	International Relations	51	1998-09-22

7. Test and validate:

- Your queries and reports produce the correct results.
  - Our queries and reports are all tested with correct results in: Section 4: Functionality Testing
- Your database handles unexpected input or edge cases without crashing.
  - Here are a few queries written to throw specific errors.

Query 1: Null Values in Non-Nullable Columns
Objective: Ensure NOT NULL constraints are enforced.

```
INSERT INTO Item (Item_ID, Price, Year, Availability_Status, Item_Type)
VALUES ( Item_ID '500', Price NULL, Year 2023, Availability_Status 'Available', Item_Type 'Book');
```

```
library> INSERT INTO Item (Item_ID, Price, Year, Availability_Status, Item_Type)
         VALUES ('500', NULL, 2023, 'Available', 'Book')
[2024-12-03 15:57:54] [23000][1048] Column 'Price' cannot be null
```

Query 2: Foreign Key Constraint Violation
Objective: Prevent inserting a Book that refers to a non-existent Item_ID.

```
INSERT INTO Book (Item_ID, ISBN, Title, Subject)
VALUES ( Item_ID 'I999', ISBN '9781234567890', Title 'Invisible Library', Subject 'Fantasy');
```

```
library> INSERT INTO Book (Item_ID, ISBN, Title, Subject)
         VALUES ('I999', '9781234567890', 'Invisible Library', 'Fantasy')
[2024-12-03 15:58:18] [23000][1452] Cannot add or update a child row: a foreign key constraint fails (`library`.`book`
```

Query 3: Invalid ENUM Values
Objective: Ensure that invalid ENUM values are rejected.

```
INSERT INTO Item (Item_ID, Price, Year, Availability_Status, Item_Type)
VALUES ( Item_ID '500', Price 500, Year 2022, Availability_Status 'Lost', Item_Type 'Book');
```

```
library> INSERT INTO Item (Item_ID, Price, Year, Availability_Status, Item_Type)
         VALUES ('500', 500, 2022, 'Lost', 'Book')
[2024-12-03 15:58:41] [01000][1265] Data truncated for column 'Availability_Status' at row 1
```

- Your database performs well (queries aren't too slow).
    - All of our queries took less than 500 ms to complete, which is very quick for our needs.
    - Here are a few specific examples from the DataGrip Terminal

        [2024-12-03 15:33:03] 9 rows retrieved starting from 1 in 149 ms
        [2024-12-03 15:38:12] 26 rows retrieved starting from 1 in 341 ms
        [2024-12-03 15:38:49] 26 rows retrieved starting from 1 in 183 ms
        [2024-12-03 15:40:26] 20 rows retrieved starting from 1 in 64 ms

6. Documentation

Downloading DataGrip and mySQL

- The first step in creating the database was downloading the required programs to run the server and create the database.
- The first program that needs to be downloaded is DataGrip, which can be done here (download for OS as needed).
- Then, once DataGrip is downloaded, mySQL must be downloaded as well. A good tutorial to get mySQL downloaded can be found here.

Starting the mySQL server

- Once mySQL has successfully been downloaded, it needs to be set up and then a server can be started. This was pretty confusing at first but doable with the tutorial here. Once you get a mySQL server running, then use DataGrip to create a mySQL database.

Creating the Database

- Now that a server is running, you can connect to it remotely with DataGrip. Open up DataGrip and start a new project. Then connect this GitHub repository to that new project. Once you have done that you will have to create a data source in the data explorer tab. This can, once again, be pretty confusing but a good tutorial can be found here. Once the data source is created and connected to the mySQL server, you will have to create a new schema for the Library Database. This new schema should be called "Library". Once this schema has been created, the DDL statements will have to be executed to create all of the tables within this new schema. The tables are located within the *src* directory in a file named *main.sql*. Once this has been ran, the database has been created.

Importing Data into Database

- Once again, importing the data is a bit tricky but should be possible to do with the tutorial here. All of the data is stored in the *Library Data* folder and can be imported into each of the tables with the import commands on DataGrip. Our data was gathered using **Fabricate by Mockaroo**, available here, which is a tool that leverages AI to create realistic data based on a provided schema or DDL statements. Once this data has been imported, all of the queries within our test cases should be executable. The test cases can be found in the *src* directory in a file named *testcase.sql*.