EECS 447 Project Part 5

Team SQLibrary

April 27, 2025

1 Project Overview

The Library Management System (LMS) will manage various types of library materials, including physical and digital books, magazines, and user accounts. The system will enforce borrowing policies, track overdue items, regulate borrowing limits according to membership categories, enable reservations, and generate reports. In this phase, the focus is on mapping the conceptual ER model to a relational schema, defining attributes, primary keys, foreign keys, and establishing functional dependencies. The system will maintain strict normalization principles to minimize redundancy and ensure efficiency. However, it will not incorporate physical checkout mechanisms like barcode scanners or RFID tags, nor will it integrate with external content providers.

2 Scope

The scope of this project remains consistent with previous phases. The LMS manages various physical and digital library materials, member accounts, borrowing policies, overdue tracking, fine generation, and item reservations. The system enforces membership category restrictions, maintains item availability, and monitors due dates. Physical device integrations (e.g., barcode scanning, RFID tagging) and third-party content integration remain outside the scope.

For this phase, we focus specifically on:

- Creation of the physical schema via SQL DDL scripts.
- Loading realistic sample data into all tables.
- Verifying referential integrity and operational functionality.

3 Glossary

- **DDL (Data Definition Language)**: A set of SQL commands used to define and manage database schema objects like tables and constraints.
- **Physical Schema**: The actual SQL-based implementation of the database structure.
- **Fabricate**: A tool provided by Mockaroo that enables generation of realistic random data for database population.
- **DataGrip**: A database management environment used for manipulating, querying, and managing the LMS database.
- **Referential Integrity**: Ensures that relationships between tables remain consistent through enforced primary and foreign key constraints.
- **3NF (Third Normal Form)**: A normalization form ensuring that all attributes are functionally dependent only on the primary key and not on any non-key attributes.

- **DM**: Shorthand for Digital Media.
- **Foreign Key**: A field in one table that references the primary key of another table, ensuring referential integrity.
- **Functional Dependency**: A constraint that describes the relationship between attributes in a relation.
- **ISBN**: International Standard Book Number; a unique identifier assigned to books and other published materials.
- **Library Management System (LMS)**: An architecture created to manage library assets, member enrollments, and transaction operations.
- **Membership Type**: A classification of customers that identifies the borrower ability and charge terms for these customers (e.g., normal, student, senior).
- **Normalization**: The process of organizing a database to reduce redundancy and improve data integrity.
- **Overdue Monitoring**: A device that tracks and records overdue books and calculates corresponding fees.
- **Primary Key**: A unique identifier for each record in a table.
- **Reservation**: A feature allowing clients to place holds on borrowed items.
- **Notifications**: Automated alerts for due dates, overdue items, and reserved book availability.
- **Relational Schema**: The structured representation of a database that defines relations (tables), attributes, and constraints.
- **Item**: A generalized category for the objects the library contains, categorized by the upper-case "I" to represent the entity that Digital Media, Magazine, and Book inherit from.

4 Platform Choice

We chose **MariaDB 11.7.2** as the database platform, hosted and manipulated using **DataGrip**. Our selection was influenced by:

- Familiarity with MariaDB's syntax and behavior, which is very close to MySQL, and was utilized in previous courses under Professor Saiedian.
- DataGrip's intuitive interface for database design, data population, and query testing, but primarily the free student license.
- MariaDB's support for standard SQL features and easy foreign key enforcement.
- Flexibility in running the database locally for faster iteration and easier backup management, with our setup we can easier view and manipulate data.

5 Create Database

The physical LMS database was created based on the logical design developed in Part 4. Tables were defined with primary keys, foreign keys, and necessary data constraints. The creation process involved writing DDL scripts manually and executing them sequentially in DataGrip to ensure proper table creation and constraint satisfaction. Proof of our SQL DDL statements and code format can be found in the SQL folder of this repo. We loaded our data from an online tool: `Fabricate by Mockaroo`. Fabricate allowed us to create detailed tables perfectly matching our design, but limited us to 100 rows for any given table, due to a free account.

This, however, was more than enough and the data we created can be found in the Data folder in the repo. Through DataGrip we were then able to simply click on our tables created by the DDL statements, and import the CSV files directly. From there we were able to begin confirming the correctness of our database.

6 Print Physical Schema

The following DDL scripts were used to create the LMS schema:

```
CREATE TABLE MembershipType (
    TypeID CHAR(1) NOT NULL (PK),
    TypeName VARCHAR(20) DEFAULT NULL,
    BorrowLimit INT(11) DEFAULT NULL,
    LateFeeRate INT(11) DEFAULT NULL
);

CREATE TABLE Member (
    MemberID VARCHAR(36) NOT NULL (PK),
    Name VARCHAR(100) DEFAULT NULL,
    Contact VARCHAR(100) DEFAULT NULL,
    TypeID CHAR(1) NOT NULL,
    AccountStatus ENUM('Active', 'Suspended', 'Closed') DEFAULT NULL,
    FOREIGN KEY (TypeID) REFERENCES MembershipType(TypeID)
);

CREATE TABLE Item (
    ItemID BIGINT(20) NOT NULL (PK),
    Title VARCHAR(200) DEFAULT NULL,
    Availability ENUM('Available', 'Reserved') NOT NULL,
    ItemType ENUM('Book', 'Magazine', 'Digital Media') NOT NULL
);

CREATE TABLE Book (
    ItemID BIGINT(20) NOT NULL (PK),
    Author VARCHAR(100) DEFAULT NULL,
    Genre VARCHAR(50) DEFAULT NULL,
    PubYear YEAR DEFAULT NULL,
    FOREIGN KEY (ItemID) REFERENCES Item(ItemID)
);

CREATE TABLE DigitalMedia (
    ItemID BIGINT(20) NOT NULL (PK),
    MediaType VARCHAR(50) DEFAULT NULL,
    Creator VARCHAR(100) DEFAULT NULL,
    FOREIGN KEY (ItemID) REFERENCES Item(ItemID)
);

CREATE TABLE Magazine (
    ItemID BIGINT(20) NOT NULL (PK),
    IssueNumber INT(11) DEFAULT NULL,
    PublicationDate YEAR DEFAULT NULL,
    FOREIGN KEY (ItemID) REFERENCES Item(ItemID)
);
```

```
CREATE TABLE Librarian (
    LibrarianID INT(11) AUTO_INCREMENT (PK),
    Name VARCHAR(100) DEFAULT NULL,
    Contact VARCHAR(100) DEFAULT NULL,
    Role ENUM('Admin', 'Staff') DEFAULT NULL
);

CREATE TABLE Fine (
    FineID BIGINT(20) NOT NULL (PK),
    Amount DECIMAL(10, 2) NOT NULL,
    PaymentStatus ENUM('Paid', 'Unpaid') DEFAULT NULL,
    MemberID VARCHAR(36) NOT NULL,
    MemberType CHAR(1) NOT NULL,
    FOREIGN KEY (MemberID) REFERENCES Member(MemberID),
    FOREIGN KEY (MemberType) REFERENCES MembershipType(TypeID)
);

CREATE TABLE Loan (
    LoanID BIGINT(20) NOT NULL (PK),
    MemberID VARCHAR(36) NOT NULL,
    ItemID BIGINT(20) NOT NULL,
    LoanDate DATETIME DEFAULT NULL,
    DueDate DATETIME DEFAULT NULL,
    ReturnDate DATETIME DEFAULT NULL,
    FineID BIGINT(20) DEFAULT NULL,
    LibrarianID INT(11) DEFAULT NULL,
    LateReturn ENUM('Yes', 'No') NOT NULL,
    FOREIGN KEY (MemberID) REFERENCES Member(MemberID),
    FOREIGN KEY (ItemID) REFERENCES Item(ItemID),
    FOREIGN KEY (FineID) REFERENCES Fine(FineID),
    FOREIGN KEY (LibrarianID) REFERENCES Librarian(LibrarianID)
);

CREATE TABLE Reservation (
    ReservationID BIGINT(20) NOT NULL (PK),
    MemberID VARCHAR(36) NOT NULL,
    ItemID BIGINT(20) NOT NULL,
    RequestDate DATETIME DEFAULT NULL,
    FOREIGN KEY (MemberID) REFERENCES Member(MemberID),
    FOREIGN KEY (ItemID) REFERENCES Item(ItemID)
);
```

7 Data Population

Data for the LMS was generated using **Fabricate by Mockaroo**. Each table was populated with realistic sample entries, and was validated through Fabrication's means of table validation, as well as sample SQL queries to ensure cross-table relationships were correct. In total, the values we generated were:

- 100 Items, categorized across Book, Digital Media, and Magazine.
- 50 Book records, 30 Digital Media records, 20 Magazine records.
- 100 Members associated with 4 Membership Types.
- 15 Librarians.

- 100 Loan records.
- 24 Fine records.
- 20 Reservations.

Again, all of this data can be viewed in the Repo's Data folder. In some cases, manual adjustments were made to ensure data integrity, particularly:

- Correct alignment of ItemType categories.
- Ensuring Reservations corresponded to Items marked as Reserved.
- Matching MemberIDs between Loan and Fine correctly.

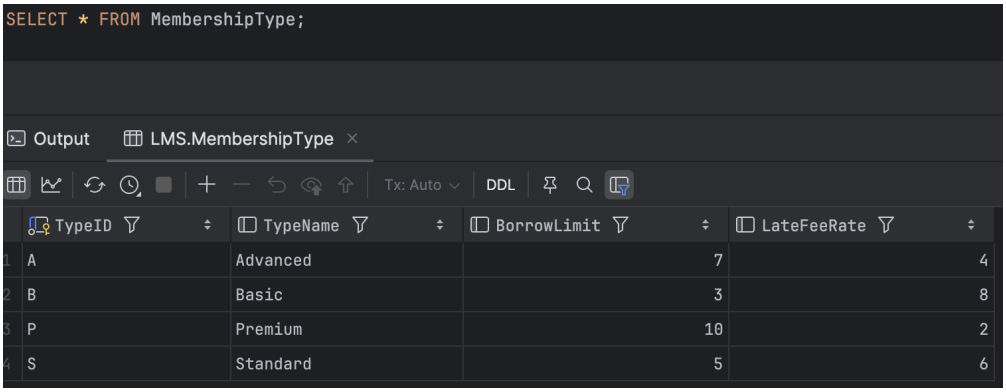
8 Printing Table Contents

The contents of each table were verified using the following command:

```
SELECT * FROM TableName;
```

Below are the screenshots showing the contents of each table after data population.

MembershipType Table



Member Table

```
SELECT * FROM Member;
```

Output | LMS.Member

	MemberID	Name	Contact	TypeID	AccountStatus
1	0b549e0b-7ac6-49c2-839b-fcbda42895cf	Kacey Vermer	kacey.vermer@hotmail.es	A	Suspended
2	0df39b29-b8bb-4c2e-92a2-4aea0f6788c8	Ruby Ashingden	ruby.ashingden@gmail.com	A	Closed
3	10491a05-44fe-4b8c-9a77-267476517583	Raddy Huet	raddy.huet@wanadoo.fr	A	Active
4	10ddbbda-22f2-4ad8-a096-f4400456479a	Myrtie Bridgwood	myrtie.bridgwood@hotmail.com	A	Suspended
5	114ff942-62cb-46dc-91e8-a75fbcbbdb72	Sonny Warrilow	sonny.warrilow@gmail.com	A	Closed
6	12d30795-d3a2-4ffb-9edc-8b722055de48	William Mergue	william.mergue@aol.com	S	Closed
7	1656acb2-bbbd-4036-ba69-3d014e242cf9	Lauri Blezard	lauri.blezard@gmail.com	B	Active
8	1856f17b-086d-4981-b031-bc851c5b1403	Andros Yorke	andros.yorke@yahoo.co.uk	S	Active
9	1b563291-44ae-4521-8129-233f73cdf7af	Lawrence Hurll	lawrence.hurll@gmail.com	B	Suspended
10	2057974d-2e05-4496-aa0a-32a6269d9c72	Ariana Tuffey	ariana.tuffey@yahoo.com.br	B	Closed
11	2304291e-5d72-4e1e-bb81-8d0fa417f773	Shandra Drable	shandra.drable@gmail.com	B	Suspended
12	244bc2f5-ea58-434f-8dfa-70477485884e	Gayleen Lared	gayleen.lared@gmail.com	S	Closed
13	264f4dc5-0f66-4275-a04b-2cdf9b97536d	Onfre Saunper	onfre.saunper@yahoo.co.uk	P	Closed
14	2fcef34b-a11f-4f94-bba2-efdc7b582464	Clarisse Water	clarisse.water@libero.it	S	Closed
15	3035facb-5455-4f37-a89f-e54c4c7c9937	Jerrome Pierse	jerrome.pierse@gmail.com	S	Active
16	31816ee1-3995-4fdb-8e78-5b5339f31ef2	Laurie Janku	laurie.janku@yahoo.com	A	Closed
17	34ab3752-efce-4af4-856a-46ecb60fac2b	Prudy Plester	prudy.plester@yahoo.com	A	Closed
18	371a84a8-14bf-4a59-999e-969ca93c7092	Madella Veck	madella.veck@hotmail.com	S	Active
19	3d863bbf-e6bf-4d99-84c8-66d14b726662	Biron Thexton	biron.thexton@yahoo.com	S	Closed
20	40cf5577-fd8d-4d9e-bef0-61e5bc601387	Henderson Pendock	henderson.pendock@hotmail.com	S	Closed
21	42051088-c385-49ea-ba68-e25d7f8baa5a	Babbette McVey	[illegible]@live.co.uk	P	Closed
22	4b040e89-7b90-4dd5-b7ba-04a507e14c7b	Nessi Hann	[illegible]oo.com	P	Suspended

100 rows

Item Table

```
SELECT * FROM Item;
```

Output | LMS.Item

	ItemID	Title	Availability	ItemType
1	4005383372515	The Twilight Saga	Available	Book
2	4075482286132	Beneath the Blood Moon	Reserved	Book
3	4177629731163	The Media Revolution	Reserved	Digital Media
4	4271224941574	Beyond the Veil	Available	Book
5	4318801097754	In the Shadow of the Mountain	Reserved	Book
6	4572978567440	A World Apart	Available	Book
7	4607262778345	Style Chronicles	Available	Magazine
8	4677199637782	Whispers of the Wind	Available	Book
9	4707350781583	World of Tech	Available	Digital Media
10	4970217494564	Time Out	Available	Magazine
11	340108386021429	Dreams of Tomorrow	Available	Book
12	340650926418121	Tales of the Forgotten	Available	Book
13	341276731963019	World Affairs	Available	Magazine
14	341316749851802	The Soundscape	Reserved	Digital Media
15	341348913643079	The Gourmet Guide	Available	Magazine
16	341649294642295	Lost and Found	Available	Book
17	341888613444090	Digital Horizons	Reserved	Digital Media
18	342212465113615	The Hidden Truth	Available	Book
19	342595629768639	Virtual Escape	Reserved	Digital Media
20	344049291053328	The Hidden Garden	Reserved	Book
21	344506483047594	The Digital Awakening	Avail[illegible]	Digital Media
22	344679237233875	The Ivory Tower	Reser[illegible]	Book

100 rows

Book Table

```
SELECT * FROM Book;
```

Output | LMS.Book

Tx: Auto | DDL

	ItemID	Author	Genre	PubYear
1	4005383372515	Cecily Bemlott	Dystopian	2001
2	4075482286132	Mill Lots	Paranormal	2011
3	4271224941574	Webb Lawlie	Literary Fiction	2008
4	4318801097754	Kaela Igo	Mystery	2007
5	4572978567440	Nelie Bleakman	Fantasy	2010
6	4677199637782	Random Author 1	Fiction	2021
7	340108386021429	Random Author 2	Mystery	2020
8	340650926418121	Alex Selman	Mystery	2012
9	341649294642295	Random Author 3	Science Fiction	2019
10	342212465113615	Noelani Cluer	Dystopian	1995
11	344049291053328	Random Author 4	Historical	2022
12	344679237233875	Johny Getsham	Graphic Novel	1998
13	345768079942541	Sly Soans	Horror	2011
14	348387027107323	Nady Shorland	Young Adult	2004
15	370567932669295	Random Author 5	Fantasy	2021
16	373377531281616	Brooke Glazyer	Graphic Novel	2001
17	373987461974889	Lorelei Piwell	Historical Fiction	2000
18	375579485277257	Melessa Stotherfield	Classic	1990
19	376926633861699	Georgina Sindall	Romance	1992
20	378935911713863	Markos Rabbe	Graphic Novel	1998
21	4066026457822615	Random Author 6	Thriller	2020
22	4157552767062824	Tory Nowell	Young Adult	2008

50 rows

DigitalMedia Table

```
SELECT * FROM DigitalMedia;
```

Output | LMS.DigitalMedia

Tx: Auto | DDL

	ItemID	MediaType	Creator
1	4177629731163	CD	Alex Johnson
2	4707350781583	DVD	Taylor Brooks
3	341316749851802	Movie	Jordan Rivera
4	341888613444090	Video Game	Casey Morgan
5	342595629768639	Audiobook	Riley Bennett
6	344506483047594	DVD	Federico Starkie
7	374942930505785	CD	Cameron Price
8	375206806454615	CD	Dewie Goddman
9	377434522929599	DVD	Hayden Scott
10	379123436856840	Audiobook	Minne Brech
11	4027806687352030	Movie	Peyton Carter
12	4369865887524570	Video Game	Quinn Simmons
13	4544566281300371	Audiobook	Reese Armstrong
14	4585173843216201	CD	Jordan Nguyen
15	4744868971409142	DVD	De Erington
16	5120085085213978	Movie	Pen Mowles
17	5132701528932166	DVD	Morgan Blake
18	5151357605498005	Movie	Rufus Laingmaid
19	5253437748129191	DVD	Kai Magnay
20	5314976393902987	Movie	Logan Pierce
21	5319954808750148	Video Game	Parker Sullivan
22	5339797658943966	CD	Glynda Rentcome

30 rows

Magazine Table

```
SELECT * FROM Magazine;
```

Output | LMS.Magazine

Tx: Auto | DDL

	ItemID	IssueNumber	PublicationDate
1	4607262778345	45	2019
2	4970217494564	62	2022
3	341276731963019	18	2018
4	341348913643079	73	2020
5	375089867501319	29	2021
6	4257599750999388	56	2023
7	4285672507316051	38	2020
8	4405479546156455	81	2024
9	4809205974640948	25	2017
10	5157500757411409	67	2021
11	5180118089284724	49	2016
12	5279567886786202	32	2019
13	5280952728249576	77	2015
14	5518194732932400	2	2003
15	5524266640367689	6	1975
16	602389821564514456	7	1998
17	604993258452733407	21	2022
18	615275056487440492	88	2023
19	617151907212232460	34	2010
20	619950363365007664	43	2006

20 rows

Librarian Table

```
SELECT * FROM Librarian;
```

Output | LMS.Librarian

Tx: Auto | DDL

	LibrarianID	Name	Contact	Role
1	1000	Arielle Salamon	arielle.salamon@free.fr	Staff
2	1001	Kip Pittwood	kip.pittwood@bol.com.br	Admin
3	1002	Joellen Wort	joellen.wort@gmail.com	Staff
4	1003	Nat Rainard	nat.rainard@yahoo.co.uk	Staff
5	1004	Zola Rowlings	zola.rowlings@yahoo.com	Admin
6	1005	Priscella Tideswell	priscella.tideswell@gmail.com	Staff
7	1006	Port Brogini	port.brogini@hotmail.com	Staff
8	1007	Roscoe Yewdall	roscoe.yewdall@gmail.com	Staff
9	1008	Denver Langley	denver.langley@yahoo.com	Staff
10	1009	Evvie Cutbirth	evvie.cutbirth@hotmail.com	Staff
11	1010	Mikel Whittock	mikel.whittock@hotmail.com	Staff
12	1011	Ania Locke	ania.locke@yahoo.com	Staff
13	1012	Stacee Foxten	stacee.foxten@orange.fr	Staff
14	1013	Clarissa Glavin	clarissa.glavin@hotmail.co.uk	Staff
15	1014	Moises Pol	moises.pol@gmail.com	Admin

Loan Table

```
SELECT * FROM Loan;
```

Output | LMS.Loan

Tx: Auto | DDL | CSV

	LoanID	MemberID	ItemID	LoanDate	DueDate	ReturnDate	FineID
1	4802221843753	5a564bb8-76eb-4f40-a0e9-25586b1cdfc0	6124133888259871 67	2024-04-30 13:47:06	2024-05-07 12:00:00	2024-05-01 14:56:45	<nu
2	4151938059386	42051088-c385-49ea-ba68-e25d7f8baa5a	3733775312816 16	2024-04-29 16:21:11	2024-05-06 12:00:00	2024-04-30 11:45:26	<nu
3	4412868793942	8df39b29-b8bb-4c2e-92a2-4aea0f6788c8	460726278345	2024-04-30 11:14:36	2024-05-07 12:00:00	2024-05-01 19:44:51	<nu
4	4436701801901	e1cea610-8f65-4e82-8c62-f21a3483d7a4	5473724526011480	2024-04-28 06:41:00	2024-05-05 12:00:00	2024-04-29 15:24:01	<nu
5	4462924047786	c0084401-4ed6-4752-8c7a-c54dc09c745c	365768079942541	2024-04-28 19:05:04	2024-05-05 12:00:00	2024-05-05 18:09:14	6608718
6	4743486368811	fa9ee51b-938d-4c95-aaaa-bc46cc457cf2	467719963778 2	2024-04-27 20:51:11	2024-05-04 12:00:00	2024-04-28 22:31:22	<nu
7	4857414731835	8e6afdfb-4ab0-4347-858b-6d13b2dd381d	6199503633650076 64	2024-04-27 22:34:45	2024-05-04 12:00:00	2024-04-28 19:58:26	<nu
8	4903869729568	3d863bbf-e6bf-4d99-84c8-66d14b726662	378935911713863	2024-04-28 03:55:17	2024-05-05 12:00:00	2024-04-29 22:18:16	<nu
9	4913766086216	f5ac1896-be84-4739-bc4f-611c4deebe0a	6138562745539669 18	2024-04-28 19:54:27	2024-05-05 12:00:00	2024-04-29 08:18:43	<nu
10	4916317064024	6238af86-38d7-4faf-b933-262fb4a60c60	531995480875014 8	2024-04-28 14:08:43	2024-05-05 12:00:00	2024-04-29 03:59:35	<nu
11	3401516851017 71	d1fd582e-8d87-4f65-bf80-7756e587aaa7	497021749456 4	2024-04-28 12:36:28	2024-05-05 12:00:00	2024-04-29 19:51:06	<nu
12	3412841012755 80	5d0d7bcb-b3f8-4a2a-b7d9-3128860bdb48	528095272824957 6	2024-04-30 11:48:07	2024-05-07 12:00:00	2024-05-01 19:26:55	<nu
13	3424433117709 38	9b545725-d38e-492f-ac35-19b6f5117d7f	515135760549880 5	2024-04-27 22:16:08	2024-05-04 12:00:00	2024-05-04 22:06:04	6512424
14	3425321885462 26	bcf65f4b-6cc6-4c51-9348-be5d8d9f2125	375579485277257	2024-04-28 14:56:51	2024-05-05 12:00:00	2024-04-29 07:37:18	<nu
15	3429695266312 39	5c3fd376-11a6-404f-9f61-d7320f581bab	531497639398298 7	2024-04-28 12:48:17	2024-05-05 12:00:00	2024-04-29 02:43:16	<nu
16	3448841096201 64	aa18715d-c041-4e2d-a528-bf0ea09f6c8b	616507995078888 106	2024-04-27 18:20:33	2024-05-04 12:00:00	2024-05-04 16:47:18	2484302
17	3446088345533188	fc928210-fa26-4458-9fa7-bd59e3afdb37	615275856487440 492	2024-04-28 21:30:26	2024-05-05 12:00:00	2024-04-29 18:30:32	<nu
18	3466772995268 78	e40bf7be-dc5a-4d45-be28-58ee48969b3a	48092059746409 48	2024-04-29 12:17:17	2024-05-06 12:00:00	2024-04-30 15:34:06	<nu
19	3468379884872 64	9e68927b-dd54-432e-a16c-217806e8872e	5109157297917434	2024-04-28 01:08:15	2024-05-05 12:00:00	2024-04-29 11:35:53	<nu
20	3480728078689 60	84777f33-79a5-4fb6-9508-bcc6cb191efa	5524266640367689	2024-04-29 10:17:58	2024-05-06 12:00:00	2024-05-06 23:13:58	3807020
21	3483925157441 50	63fc8e62-1696-4b27-a951-f8c85b2d27be	344679237233875	2024-04-27 16:43:59	2024-05-04 12:00:00	2024-04-28 02:08:04	<nu
22	3719828164464 50	84577ba9-4678-4c66-a8ca-6a75f57bbe2d	341316749851802	2024-04-29 02:18:14	2024-05-06 12:00:00	2024-04-30 09:48:02	<nu
23	3728044089427 11	2fcef34b-a11f-4f94-bba2-efdc7b582464	6136847055832489 99	2024-04-28 08:36:19	2024-05-05 12:00:00	2024-05-05 16:24:38	4214103
24	3736551835843 47	87b7d558-5f8f-45ba-92dd-259facf46687	527956788678628 2	2024-04-28 18:42:46	2024-05-05 12:00:00	2024-05-05 03:41:23	<nu
25	3764672807738 51	244bc2f5-ea58-434f-8dfa-70477485884e	513270152893216 6	2024-04-26 14:58:43	2024-05-03 12:00:00	2024-04-27 08:44:25	<nu
26	3766533152665 474	31816ee1-3995-4fdb-8e78-5b5339f31ef2	6185897111245475 52	2024-04-28 01:15:46	2024-05-05 12:00:00	2024-05-05 01:58:58	<nu
27	3774864615537 317	10491a05-44fe-4b8c-9a77-267476517583	614[illegible]	2024-04-29 10:08:14	2024-05-06 12:00:00	2024-04-30 06:39:38	<nu
28	3778495583306 20	78af5fbb-2314-43a7-9cdf-2d7958a84932	61 100 rows	-29 21:22:49	2024-05-06 12:00:00	2024-05-06 16:48:43	2713434
29	3781979491115658	d7d9f14f-dba7-4935-b3be-9d5ac73c336e	4655711671912550	2024-04-27 15:13:36	2024-05-04 12:00:00	2024-05-04 08:57:52	<nu

Fine Table

```
SELECT * FROM Fine;
```

Output | LMS.Fine

	FineID	Amount	PaymentStatus	MemberID	MemberType
1	2484302526	6.20	Unpaid	aa18715d-c041-4e2d-a528-bf0ea09f6c8b	S
2	2646818843	8.95	Unpaid	f7fd4372-186f-4d0a-9414-020901c67302	B
3	2713416882	5.80	Paid	78af5fbb-2314-43a7-9cdf-2d7958a84932	A
4	2730987611	5.31	Unpaid	97c15564-1a62-48f4-9d9a-f6475042696b	A
5	2806700209	5.04	Unpaid	9069871e-073c-4926-acfa-bb6a96a8b23f	P
6	3807020199	6.87	Paid	84777f33-79a5-4fb6-9508-bcc6cb191efa	A
7	3845557247	5.96	Unpaid	800aff3c-7dc3-41fe-8569-9b679bef6c3b	A
8	4054583119	6.99	Paid	3035facb-5455-4f37-a89f-e54c4c7c9937	S
9	4214101957	6.10	Unpaid	2fcef34b-a11f-4f94-bba2-efdc7b582464	S
10	4380739627	6.58	Unpaid	4e782040-ddc6-41a3-9ed1-bc2b54157763	S
11	4387427245	6.19	Unpaid	eb35a2d8-196a-48a9-bd97-4dd7bb8da4c1	B
12	5541239776	5.92	Unpaid	ff53f69f-1ed0-4f6b-ae69-5565dbf41a7f	P
13	6512426636	7.53	Unpaid	9b545725-d30e-492f-ac35-19b6f5117d7f	S
14	6608718040	6.03	Unpaid	c00844d1-4ed6-4752-8c7a-c54dc09c745c	A
15	7602071153	5.32	Unpaid	ae13419f-ce14-45bb-b49c-cc144cc19f82	A
16	7687593947	6.05	Unpaid	a59cb134-154d-42d0-85db-2c551387e40e	A
17	7722152460	7.15	Unpaid	c39b445a-bbf4-46e3-bceb-f3bb9c02a5fa	S
18	7888817810	5.54	Unpaid	e8c2144f-e6c0-4c00-b0a0-19d10fd0563b	A
19	8530919163	6.10	Paid	1656acb2-bbbd-4036-ba69-3d014e242cf9	B
20	8885478189	7.15	Unpaid	40cf5577-fd8d-4d9e-bef0-61e5bc601387	S
21	9352984456	5.84	Paid	[illegible]8-14bf-4a59-999e-969ca93c7092	S
22	9785202701	5.38	Paid	[illegible]3-a370-40c3-818d-c024008bd47e	P

24 rows

Reservation Table

```
SELECT * FROM Reservation;
```

Output | LMS.Reservation

	ReservationID	MemberID	ItemID	RequestDate
1	4403377272730	cb4a015c-bfed-4931-9eee-809…	344049291053328	2024-05-09 03:48:57
2	4546168780255	42051088-c385-49ea-ba68-e25…	4285672507316051	2024-05-09 06:23:37
3	4814421504784	78e42022-ded0-49dd-93ab-777…	616491866622548603	2024-05-07 05:53:21
4	345620425206242	6d7e2fc7-4f38-466b-8d86-d69…	4177629731163	2024-05-08 12:41:47
5	346538463139413	87eda8fb-c168-432d-90c6-bd5…	342595629768639	2024-05-09 14:39:51
6	370185766950242	9b91d8b5-613a-467c-b7b1-4b1…	5473724526011400	2024-05-09 04:56:59
7	375286436714433	6b023a74-a9b6-4357-b739-e48…	610589711124547552	2024-05-10 00:02:34
8	378705557985462	84777f33-79a5-4fb6-9508-bcc…	5304561818991187	2024-05-08 16:56:24
9	379278718954701	e575cdaf-f89c-40a6-bec2-2fd…	373987461974889	2024-05-09 22:42:26
10	4056056790163247	c39b445a-bbf4-46e3-bceb-f3b…	4027806687352030	2024-05-08 00:04:06
11	4211751435138628	800aff3c-7dc3-41fe-8569-9b6…	615865578099736492	2024-05-08 03:19:11
12	4846657615642516	79e33b0c-d033-47d0-954f-4cf…	370567932669295	2024-05-09 03:28:11
13	5357241495974642	8e6afdfb-4ab0-4347-858b-6d1…	617151907212232460	2024-05-08 15:27:37
14	5404507502493682	5037d41b-2842-4e92-85a0-8ea…	341316749851802	2024-05-09 05:53:37
15	5427777787895918	f5d71fc8-9554-4394-9b85-9fd…	5518194732932400	2024-05-07 05:00:00
16	5495369330248634	8058c92b-239a-4852-bb9c-361…	348387027107323	2024-05-10 08:40:21
17	606659474323026208	d123be72-f4f3-4b46-a6fe-1d7…	612413388825907167	2024-05-10 19:55:17
18	612410076994374334	93fb0f82-2afd-496b-b5a0-02f…	4075482286132	2024-05-08 19:01:31
19	618544953733142030	fc928210-fa26-4458-9fa7-bd5…	607815377153601793	2024-05-10 16:37:42
20	619669506149084761	4ce2c7fc-9d4a-43a9-8f5c-6bf…	602389821564514456	2024-05-10 23:02:59

20 rows

As should be evident, the data loaded correctly and the DDL statements worked. Although we couldn't show all our data in one screenshot for each table, at the bottom, you can see that the number we're getting as a result of the query matches up with the data we populated. Next, we have provided several examples (with descriptions and proof) of testing we have done to ensure the database is correct in it's integrity.

9 Functionality Testing

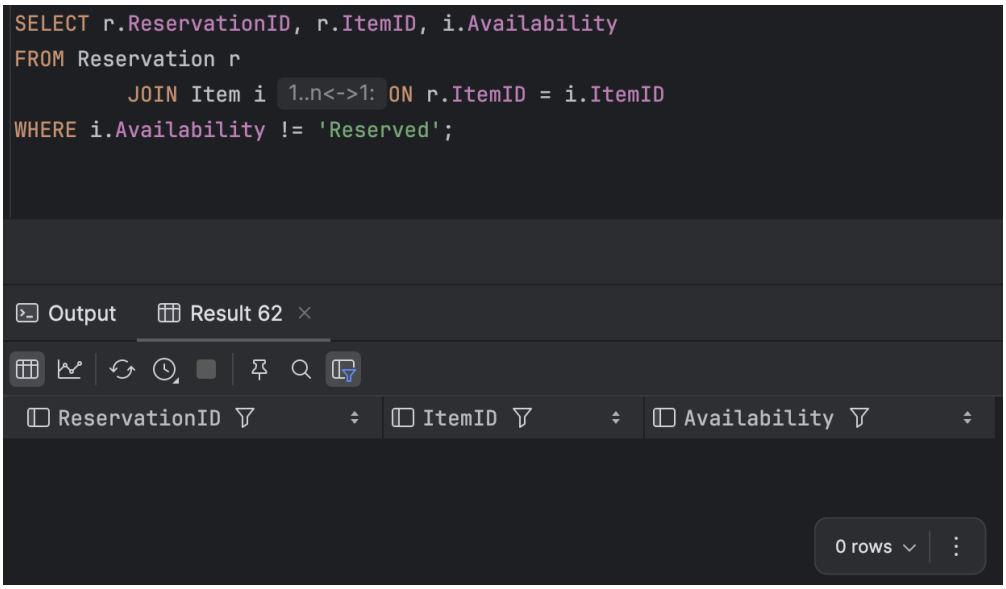
The following queries were executed to validate the functionality and referential integrity of the LMS database. Each subsection describes the goal of the test, the expected outcome, and the observed results, with screenshots provided for evidence.

Test 1: Cross-table MemberID Match

Description: Verify that for every Loan with a FineID, the MemberID in Loan matches the MemberID in the associated Fine record, this primarily showcases the consistency of our DB.
Expected Behavior: The query should return zero rows, indicating there are no mismatches.
Actual Result: Zero rows were returned. This confirms that every Fine correctly matches its originating Loan's MemberID. This proves that a member will consistency maintain associated records throughout multiple tables.

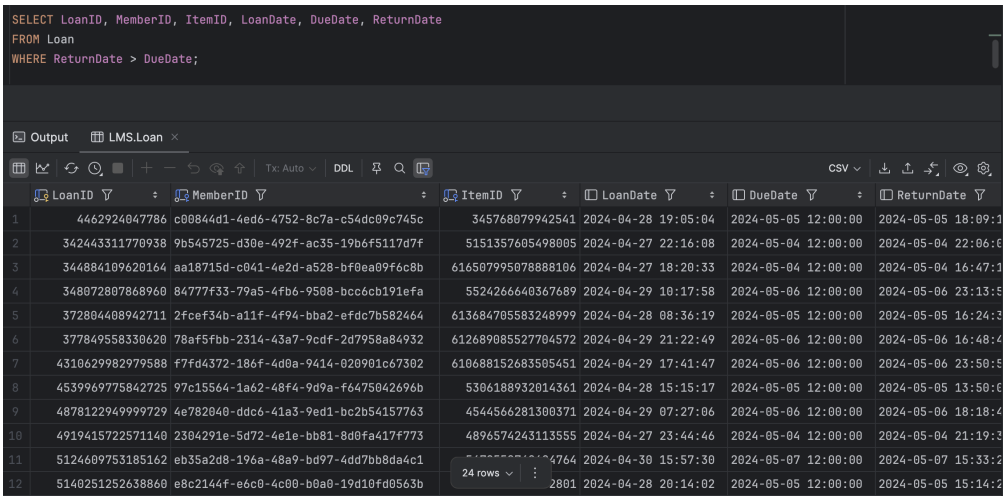
Test 2: Reservation Availability Match

Description: Ensure that every ItemID associated with a Reservation has an Availability status of 'Reserved' in the Item table. This establishes consistency at our most base level- Items.
Expected Behavior: The query should return zero rows, indicating all reserved items are properly marked.
Actual Result: Zero rows were returned. This validates correct synchronization between Reservations and Item availability statuses.



Test 3: Overdue Loans Detection

Description: Identify loans that were returned after their due dates by comparing ReturnDate and DueDate. This means, these loans should incur a Fine.
Expected Behavior: The query should return any loans where ReturnDate is greater than DueDate, indicating they were overdue and in the Fine table. Since our Fine table has 24 values initially, we should expect 24 loans.
Actual Result: 24 rows were returned, and they can be seen to match all of the rows in the Fine table, proving synchronization between the Loan and Fine tables.



Test 4: Unpaid Fines Query

Description: List all fines where PaymentStatus is 'Unpaid'.
Expected Behavior: The query should return all unpaid fines.
Actual Result: 17 rows were returned, and if you count the Unpaid fines in the Fine table, you will see, visually, this result is correct. This shows that our table creation is working well enough to query specific attributes, and is a base-case test.

```sql
SELECT FineID, MemberID, Amount
FROM Fine
WHERE PaymentStatus = 'Unpaid';
```

Output | LMS.Fine

	FineID	MemberID	Amount
1	2484302526	aa18715d-c041-4e2d-a528-bf0ea09f6c8b	6.20
2	2646818843	f7fd4372-186f-4d0a-9414-020901c67302	8.95
3	2730987611	97c15564-1a62-48f4-9d9a-f6475042696b	5.31
4	2806700209	9069871e-073c-4926-acfa-bb6a96a8b23f	5.04
5	3845557247	800aff3c-7dc3-41fe-8569-9b679bef6c3b	5.96
6	4214101957	2fcef34b-a11f-4f94-bba2-efdc7b582464	6.10
7	4380739627	4e782040-ddc6-41a3-9ed1-bc2b54157763	6.58
8	4387427245	eb35a2d8-196a-48a9-bd97-4dd7bb8da4c1	6.19
9	5541239776	ff53f69f-1ed0-4f6b-ae69-5565dbf41a7f	5.92
10	6512426636	9b545725-d30e-492f-ac35-19b6f5117d7f	7.53
11	6608718040	c00844d1-4ed6-4752-8c7a-c54dc09c745c	6.03
12	7602071153	ae13419f-ce14-45bb-b49c-cc144cc19f82	5.32

17 rows

Test 5: Active Reservations Query

Description: Display all active reservations sorted by most recent RequestDate.
Expected Behavior: The query should return all current reservations, and they should be sorted accurately.
Actual Result: All 20 initial reservations were found, and since they're all active, this can be seen to be correct. This once again demonstrates the stability of our DB, and how base-case queries are successful.

```sql
SELECT ReservationID, MemberID, ItemID, RequestDate
FROM Reservation
ORDER BY RequestDate DESC;
```

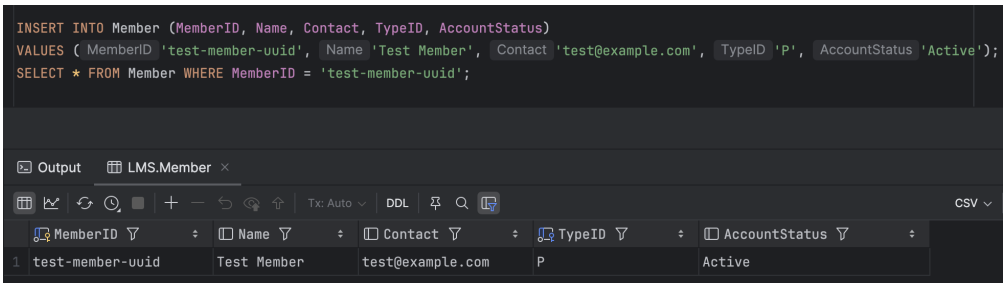
Output | LMS.Reservation

	ReservationID	MemberID	ItemID	RequestDate
1	619669506149084761	4ce2c7fc-9d4a-43a9-8f5c-6bfcfd00c4f7	602389821564514456	2024-05-10 23:02:59
2	606659474323026208	d123be72-f4f3-4b46-a6fe-1d7eb605d867	612413388825907167	2024-05-10 19:55:17
3	618544953733142030	fc928210-fa26-4458-9fa7-bd59e3afdb37	607815377153601793	2024-05-10 16:37:42
4	5495369330248634	8058c92b-239a-4852-bb9c-3617956299a5	348387027107323	2024-05-10 08:40:21
5	375286436714433	6b023a74-a9b6-4357-b739-e4873a199eb0	610589711124547552	2024-05-10 00:02:34
6	379278718954701	e575cdaf-f89c-40a6-bec2-2fdeeb66a296	373987461974889	2024-05-09 22:42:26
7	346538463139413	87eda8fb-c168-432d-90c6-bd575b102aba	342595629768639	2024-05-09 14:39:51
8	4546168780255	42051088-c385-49ea-ba68-e25d7f8baa5a	4285672507316051	2024-05-09 06:23:37
9	5404507502493682	5037d41b-2842-4e92-85a0-8eaf4ada8c79	341316749851802	2024-05-09 05:53:37
10	370185766950242	9b91d8b5-613a-467c-b7b1-4b1dfc5b2d51	5473724526011400	2024-05-09 04:56:59
11	4403377272730	cb4a015c-bfed-4931-9eee-8096079	344049291053328	2024-05-09 03:48:57
12	4846657615642516	79e33b0c-d033-47d0-954f-4cf0cdd	370567932669295	2024-05-09 03:28:11

20 rows

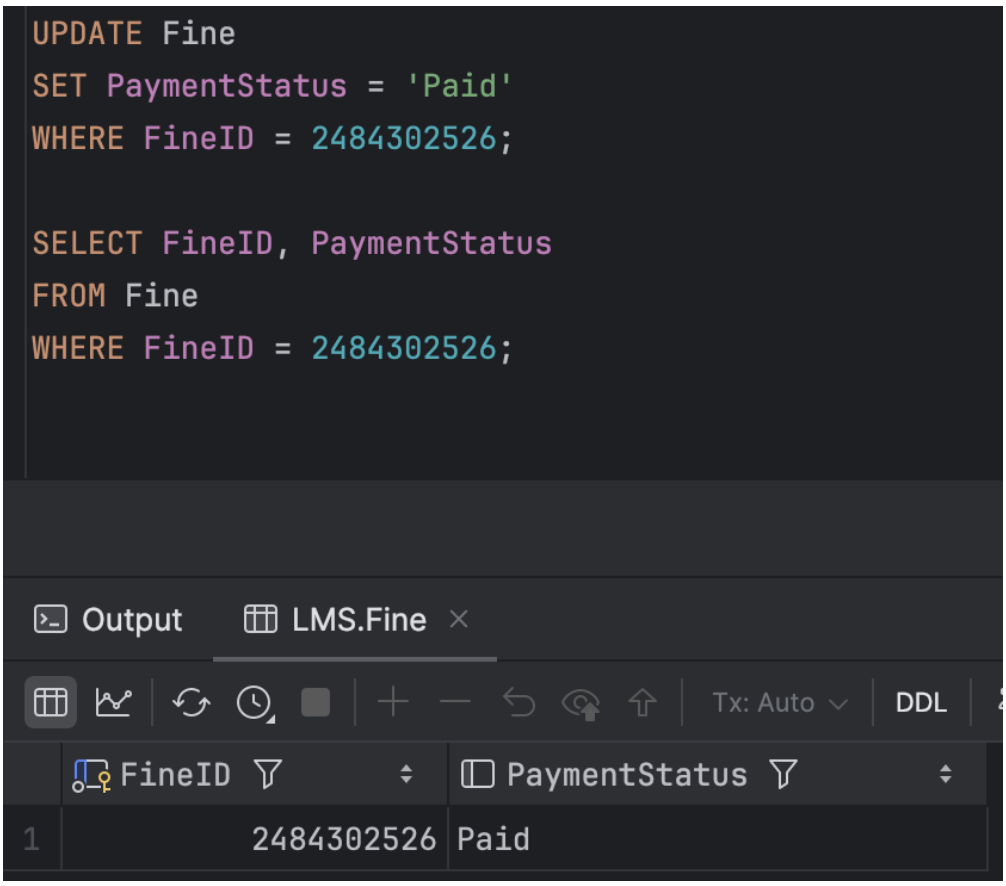
Test 6: Insert New Member

Description: Insert a new Member record into the database, validating that data storage (INSERT operations) works as expected.
Expected Behavior: The new Member should be inserted successfully and visible when queried.
Actual Result: The Member was inserted, and a SELECT query confirmed their existence. This demonstrates that the LMS database correctly stores new records.



Test 7: Update Fine Payment Status

Description: Manually update a fine's PaymentStatus from 'Unpaid' to 'Paid'. This tests the database's ability to perform controlled updates without errors. For this we chose FineID 2484302526, which can be seen in our Data directory to have a status of 'Unpaid'
Expected Behavior: The PaymentStatus for the selected Fine should update correctly.
Actual Result: After updating, the Fine's PaymentStatus reflected the new value. This shows that update operations are functioning properly in the LMS.



Test 8: Report - Total Number of Loans by Member

Description: Generate a report showing how many items each member has borrowed. This tests the database's ability to aggregate and group data efficiently.
Expected Behavior: The query should return a list of MemberIDs with their corresponding number of loans. Since we haven't modified any of the data, we are expecting each member to have one loan.
Actual Result: The system accurately reported the single loan per member, confirming correct grouping and aggregation. This also confirms data population integrity, since each member was assigned an individual loan.

```
SELECT MemberID, COUNT(*) AS NumberOfLoans
FROM Loan
GROUP BY MemberID
ORDER BY NumberOfLoans DESC;
```

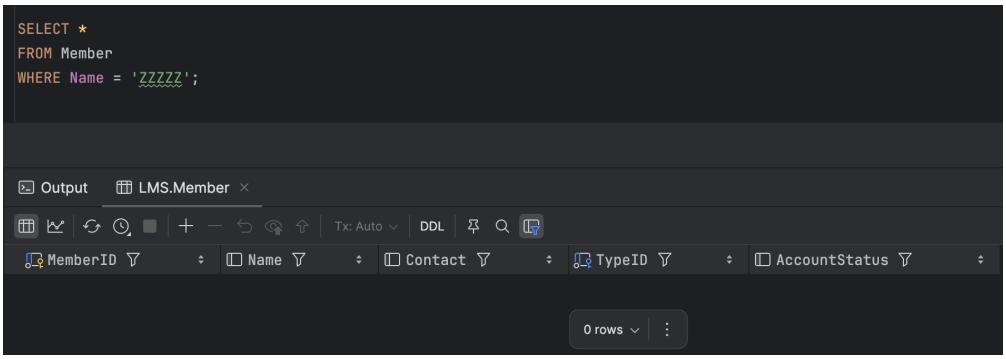
Output | Result 73

	MemberID	NumberOfLoans
1	0b549e0b-7ac6-49c2-839b-fcbda42895cf	1
2	1b563291-44ae-4521-8129-233f73cdf7af	1
3	34ab3752-efce-4af4-856a-46ecb60fac2b	1
4	4e782040-ddc6-41a3-9ed1-bc2b54157763	1
5	63fc8e62-1696-4b27-a951-f8c85b2d27be	1
6	8058c92b-239a-4852-bb9c-3617956299a5	1
7	8ab02011-fa2d-4dce-99b4-118bcb7905a9	1
8	97c15564-1a62-48f4-9d9a-f6475042696b	1
9	a59cb134-154d-42d0-85db-2c551387e40e	1
10	bc600d80-882e-42d6-9ae4-d86ac80571cf	1
11	d7d9f14f-dba7-4935-b3be-9d5ac73c336e	1
12	ead9b88f-b10f-4be6-9336-e43713709dbf	1

100 rows

Test 9: Edge Case - Search for Nonexistent Member

Description: Attempt to search for a member with a name unlikely to exist ('ZZZZZ'). This tests the system's behavior when no matching results are found, and fulfills the edge case testing requirement.
Expected Behavior: The query should return zero rows without crashing. A crash would indicate a critical failure that cannot occur in production.
Actual Result: No rows were returned, and the system handled the search gracefully without errors.



Test 10: Performance - Quick Loan Search by Date Range

Description: Search for all loans issued within a specific date range. This tests how quickly the database can retrieve filtered results based on dates.
Expected Behavior: The query should return all loans issued within the given date range without excessive delay, given that this is a small dataset.
Actual Result: 83 loans in the range were retrieved promptly, showing that the LMS database handles basic range queries efficiently. Considering the success of the tests thus far, we can assume this comparison was successful and doesn't require further validation other than visually. The retrieval, according to DataGrip, took 237 ms, but only 3 ms for execution, which is an optimal amount of time.