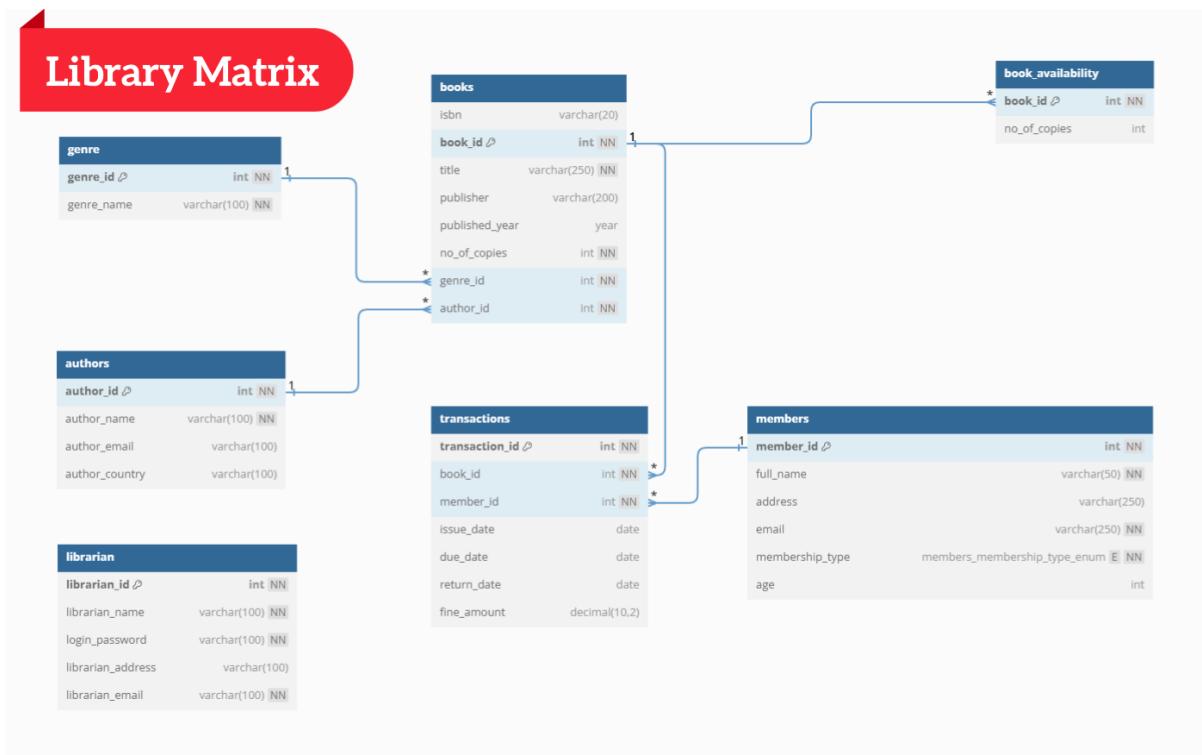


LIBRARY MATRIX

LibraryMatrix is a comprehensive database designed to streamline and enhance the management of library systems. This robust platform serves as a centralized repository for all information related to the library, ensuring efficient organization and easy access to critical data.



ER Diagram

Figure 1 1

The screenshot shows the MySQL Workbench interface. In the Database editor tab, a SQL script is being run:

```
1 -- Create a new database named 'libsis_db'
2 -- This database will be used to store all information related to the library system,
3 -- including books, authors, members, and transactions.
4
5 • CREATE DATABASE libsys_db;
6
7 -- Switch to the 'libsis_db' database
8 -- This command sets the context to the 'libsys_db' database,
9 -- allowing subsequent SQL commands to operate within this database.
10
11 • USE libsys_db;
```

In the Output tab, the results of the command are displayed:

Action	Time	Message	Duration / Fetch
CREATE DATABASE libsys_db	17:15:10	1 row(s) affected	0.015 sec

Figure 2

The screenshot shows the MySQL Workbench interface. In the Database editor tab, a SQL script is being run:

```
1 -- Create a new database named 'libsis_db'
2 -- This database will be used to store all information related to the library system,
3 -- including books, authors, members, and transactions.
4
5 • CREATE DATABASE libsys_db;
6
7 -- Switch to the 'libsis_db' database
8 -- This command sets the context to the 'libsys_db' database,
9 -- allowing subsequent SQL commands to operate within this database.
10
11 • USE libsys_db;
```

In the Output tab, the results of the command are displayed:

Action	Time	Message	Duration / Fetch
CREATE DATABASE libsys_db	17:15:10	1 row(s) affected	0.015 sec
USE libsys_db	17:15:48	0 row(s) affected	0.000 sec

Figure 3

The screenshot shows the MySQL Workbench interface with the 'Database' tab selected. In the main pane, a SQL script is being run to create a table named 'genre'. The script includes comments explaining the fields: 'genre_id' is the primary key, auto-incremented, and unique; 'genre_name' is a required field. The output pane shows the command executed at 17:19:09 and completed successfully with 0 rows affected.

```
11
12
13 -- Create a table named 'genre' to store information about book genres
14
15 • CREATE TABLE genre (
16     genre_id int NOT NULL AUTO_INCREMENT, -- Unique identifier for each genre, auto-incremented
17     genre_name varchar(100) NOT NULL, -- Name of the genre, required field
18     PRIMARY KEY (genre_id), -- Set 'genre_id' as the primary key for the table
19     UNIQUE KEY genre_name (genre_name) -- Ensure that each genre name is unique
20 );
21
22
23
24
25
```

Output

Action	Time	Message	Duration / Fetch
CREATE TABLE genre (genre_id int NOT NULL AUTO_INCREMENT, - Unique identifier for each genre, auto-incremented genre_name varchar(100) NOT NULL, - Name of the genre, required field PRIMARY KEY (genre_id), - Set 'genre_id' as the primary key for the table UNIQUE KEY genre_name (genre_name) - Ensure that each genre name is unique);	17:19:09	0 row(s) affected	0.015 sec

Figure 4

The screenshot shows the MySQL Workbench interface with the 'Database' tab selected. A SQL script is being run to create a table named 'authors'. The script includes comments for each field: 'author_id' is the primary key, auto-incremented; 'author_name' is a required field; 'author_email' is an optional field with a default value of NULL; 'author_country' is another optional field with a default value of NULL. The output pane shows the command executed at 17:17:16 and completed successfully with 0 rows affected.

```
1
2
3
4 -- Create a table named 'authors' to store information about authors in the library
5
6 • CREATE TABLE authors (
7     author_id int NOT NULL AUTO_INCREMENT, -- Unique identifier for each author, auto-incremented
8     author_name varchar(100) NOT NULL, -- Name of the author, required field
9     author_email varchar(100) DEFAULT NULL, -- Email address of the author, optional field
10    author_country varchar(100) DEFAULT NULL, -- Country of the author, optional field
11    PRIMARY KEY (author_id) -- Set 'author_id' as the primary key for the table
12 );
13
14
15
```

Output

Action	Time	Message	Duration / Fetch
CREATE TABLE authors (author_id int NOT NULL AUTO_INCREMENT, - Unique identifier for each author, auto-incremented auth... author_name varchar(100) NOT NULL, - Name of the author, required field author_email varchar(100) DEFAULT NULL, - Email address of the author, optional field author_country varchar(100) DEFAULT NULL, - Country of the author, optional field PRIMARY KEY (author_id) - Set 'author_id' as the primary key for the table);	17:17:16	0 row(s) affected	0.015 sec

Figure 5

The screenshot shows the MySQL Workbench interface with the 'Tables' tab selected. In the main pane, SQL code is being entered to create a table named 'book_availability'. The code includes comments explaining the fields and constraints. The 'Output' pane at the bottom shows the execution log, indicating the table was created successfully with 0 rows affected in 0.016 seconds.

```
44
45
46
47 -- Create a table named 'book_availability' to track the availability of books in the library
48
49 • CREATE TABLE book_availability (
50     book_id int NOT NULL, -- Foreign key referencing the unique identifier of the book
51     no_of_copies int DEFAULT NULL, -- Number of copies available for the book, optional field
52     PRIMARY KEY (book_id), -- Set 'book_id' as the primary key for the table
53     CONSTRAINT fk_book_id FOREIGN KEY (book_id) REFERENCES books (book_id) -- Foreign key constraint linking to 'books' table
54     ON DELETE CASCADE -- If a book is deleted, its availability record will also be deleted
55     ON UPDATE CASCADE -- If the book ID is updated, the change will be reflected in this table
56 );
57
58
59
60
```

#	Time	Action	Message	Duration / Fetch
1	17:22:01	CREATE TABLE book_availability (book_id int NOT NULL, -- Foreign key referencing the unique identifier of the book no_of_copies... 0 row(s) affected		0.016 sec

Figure 6

The screenshot shows the MySQL Workbench interface with the 'Tables' tab selected. In the main pane, SQL code is being entered to create a table named 'books'. The code defines various fields with their data types and constraints, including foreign keys linking to other tables like 'authors' and 'genre'. The 'Output' pane at the bottom shows the execution log, indicating the table was created successfully with 0 rows affected in 0.031 seconds.

```
26 • CREATE TABLE books (
27     isbn varchar(20) DEFAULT NULL, -- ISBN number of the book, optional field
28     book_id int NOT NULL AUTO_INCREMENT, -- Unique identifier for each book, auto-incremented
29     title varchar(250) NOT NULL, -- Title of the book, required field
30     publisher varchar(200) DEFAULT NULL, -- Publisher of the book, optional field
31     published_year year NOT NULL, -- Year the book was published, required field
32     genre varchar(150) DEFAULT NULL, -- Genre of the book, optional field
33     no_of_copies int NOT NULL, -- Number of copies available in the library, required field
34     genre_id int NOT NULL, -- Foreign key referencing the genre of the book
35     author_id int NOT NULL, -- Foreign key referencing the author of the book
36     PRIMARY KEY (book_id), -- Set 'book_id' as the primary key for the table
37     KEY fk_author_id (author_id), -- Index for the 'author_id' foreign key
38     KEY fk_genre_id (genre_id), -- Index for the 'genre_id' foreign key
39     CONSTRAINT fk_author_id FOREIGN KEY (author_id) REFERENCES authors (author_id), -- Foreign key constraint linking to 'authors' table
40     CONSTRAINT fk_genre_id FOREIGN KEY (genre_id) REFERENCES genre (genre_id) -- Foreign key constraint linking to 'genre' table
41 );
42
```

#	Time	Action	Message	Duration / Fetch
1	17:20:37	CREATE TABLE books (isbn varchar(20) DEFAULT NULL, -- ISBN number of the book, optional field book_id int NOT NULL AUT... 0 row(s) affected		0.031 sec

Figure 7

The screenshot shows the MySQL Workbench interface with the 'Tables' tab selected. In the main pane, the SQL code for creating the 'transactions' table is displayed:

```
-- Create a table named 'transactions' to store information about book transactions
CREATE TABLE transactions (
    transaction_id int NOT NULL AUTO_INCREMENT, -- Unique identifier for each transaction, auto-incremented
    book_id int NOT NULL, -- Foreign key referencing the unique identifier of the book being borrowed
    member_id int NOT NULL, -- Foreign key referencing the unique identifier of the member borrowing the book
    issue_date date DEFAULT NULL, -- Date when the book was issued, optional field
    due_date date DEFAULT NULL, -- Date when the book is due to be returned, optional field
    return_date date DEFAULT NULL, -- Date when the book was returned, optional field
    fine_amount decimal(10,2) DEFAULT '0.00', -- Fine amount for overdue books, default is 0.00
    PRIMARY KEY (transaction_id), -- Set 'transaction_id' as the primary key for the table
    KEY book_id (book_id), -- Index on 'book_id' for faster lookups
    KEY member_id (member_id), -- Index on 'member_id' for faster lookups
    CONSTRAINT fk_transaction_book_id FOREIGN KEY (book_id) REFERENCES books (book_id) ON DELETE CASCADE, -- Foreign key constraint for book
    CONSTRAINT fk_transaction_member_id FOREIGN KEY (member_id) REFERENCES members (member_id) ON DELETE CASCADE -- Foreign key constraint for member
);
```

In the bottom pane, the 'Output' tab shows the execution results:

Action	Time	Message	Duration / Fetch
CREATE TABLE transactions (transaction_id int NOT NULL AUTO_INCREMENT, -- Unique identifier for each transaction, auto-incre...)	17:32:44	0 row(s) affected	0.047 sec

Figure 8

The screenshot shows the MySQL Workbench interface with the 'Tables' tab selected. In the main pane, the SQL code for creating the 'members' table is displayed:

```
-- Create a table named 'members' to store information about library members
CREATE TABLE members (
    member_id int NOT NULL AUTO_INCREMENT, -- Unique identifier for each member, auto-incremented
    full_name varchar(50) NOT NULL, -- Full name of the member, required field
    address varchar(250) DEFAULT NULL, -- Address of the member, optional field
    email varchar(250) NOT NULL, -- Email address of the member, required field
    membership_type enum('Student','General','Temporary','Senior citizen','Faculty') NOT NULL, -- Type of membership, required field
    age int DEFAULT NULL, -- Age of the member, optional field
    PRIMARY KEY (member_id), -- Set 'member_id' as the primary key for the table
    UNIQUE KEY email (email) -- Ensure that each email address is unique among members
);
```

In the bottom pane, the 'Output' tab shows the execution results:

Action	Time	Message	Duration / Fetch
CREATE TABLE members (member_id int NOT NULL AUTO_INCREMENT, -- Unique identifier for each member, auto-incre...)	17:22:56	0 row(s) affected	0.015 sec

Figure 9

The screenshot shows the MySQL Workbench interface with the 'Triggers' tab selected. The SQL editor contains the following code:

```
1 -- Create a trigger to replicate the number of copies of a book into the book_availability table after a new book is inserted
2
3 DELIMITER //
4 • CREATE TRIGGER tr_replicate_books_no_of_copies_on_insert
5 AFTER INSERT ON books
6 FOR EACH ROW
7 BEGIN
8     -- Insert the new book's ID and number of copies into the book_availability table
9     INSERT INTO book_availability(book_id, no_of_copies)
10    VALUES (NEW.book_id, NEW.no_of_copies);
11 END //
12
13 DELIMITER ;
```

The output window shows the execution log:

Action	Time	Action	Message	Duration / Fetch
CREATE TRIGGER	17:38:09	tr_replicate_books_no_of_copies_on_insert	AFTER INSERT ON books FOR EACH ROW BEGIN -- Insert the new book's ID and number of copies into the book_availability table	0 row(s) affected 0.015 sec

Figure 10

The screenshot shows the MySQL Workbench interface with the 'Tables' tab selected. The SQL editor contains the following code:

```
92
93
94 -- Create a table named 'librarian' to store information about library staff
95
96 • CREATE TABLE librarian (
97     librarian_id int NOT NULL AUTO_INCREMENT, -- Unique identifier for each librarian, auto-incremented
98     librarian_name varchar(100) DEFAULT NULL, -- Name of the librarian, optional field
99     login_password varchar(100) DEFAULT NULL, -- Password for librarian login, optional field
100    librarian_address varchar(100) DEFAULT NULL, -- Address of the librarian, optional field
101    librarian_email varchar(100) NOT NULL, -- Email address of the librarian, required field
102    PRIMARY KEY (librarian_id), -- Set 'librarian_id' as the primary key for the table
103    UNIQUE KEY librarian_email_UNIQUE (librarian_email) -- Ensure that each email address is unique among librarians
104 );
```

The output window shows the execution log:

Action	Time	Action	Message	Duration / Fetch
CREATE TABLE	17:33:50	librarian	(librarian_id int NOT NULL AUTO_INCREMENT, -- Unique identifier for each librarian, auto-incremented lib... 0 rows) affected	0.031 sec

Figure 11

The screenshot shows the MySQL Workbench interface with the 'Triggers' tab selected. A trigger named 'set_issue_and_due_date' is being created. The code defines a trigger that runs before an insert operation on the 'transactions' table. It sets the 'issue_date' field to the current date and calculates the 'due_date' as 14 days from the 'issue_date'. The output pane shows the execution log with a single entry indicating the successful creation of the trigger.

```
30
31 -- Create a trigger to set the issue date and due date before inserting a new transaction
32
33 DELIMITER //
34 • CREATE TRIGGER set_issue_and_due_date
35 BEFORE INSERT ON transactions
36 FOR EACH ROW
37 BEGIN
38     -- Set the issue date to the current date
39     SET NEW.issue_date = CURRENT_DATE();
40
41     -- Set the due date to 14 days after the issue date
42     SET NEW.due_date = DATE_ADD(CURRENT_DATE(), INTERVAL 14 DAY);
43 END //
44 DELIMITER ;
45
```

Action	Time	Message	Duration / Fetch
CREATE TRIGGER set_issue_and_due_date BEFORE INSERT ON transactions FOR EACH ROW BEGIN -- Set the issue date to t...	17:42:06	0 row(s) affected	0.000 sec

Figure 12

The screenshot shows the MySQL Workbench interface with the 'Triggers' tab selected. A trigger named 'tr_replicate_books_no_of_copies_on_update' is being created. The trigger runs after an update operation on the 'books' table. It checks if the number of copies has changed. If it has, it updates the 'book_availability' table by setting the 'no_of_copies' field to the new value. The output pane shows the execution log with a single entry indicating the successful creation of the trigger.

```
15 -- Create a trigger to update the number of copies of a book in the book_availability table after a book is updated
16 DELIMITER //
17 • CREATE TRIGGER tr_replicate_books_no_of_copies_on_update
18 AFTER UPDATE ON books
19 FOR EACH ROW
20 BEGIN
21     -- Check if the number of copies has changed
22     IF OLD.no_of_copies <> NEW.no_of_copies THEN
23         -- Update the number of copies in the book_availability table
24         UPDATE book_availability |
25             SET no_of_copies = NEW.no_of_copies
26             WHERE book_id = NEW.book_id;
27     END IF;
28 END //
29 DELIMITER ;
30
```

Action	Time	Message	Duration / Fetch
CREATE TRIGGER tr_replicate_books_no_of_copies_on_update AFTER UPDATE ON books FOR EACH ROW BEGIN -- Check if... 0 row(s) affected	17:41:05	0 row(s) affected	0.000 sec

Figure 13

The screenshot shows the MySQL Workbench interface with the 'Triggers' tab selected. The SQL editor contains the following code:

```
62 -- Create a trigger to update the number of copies available after a new transaction is inserted
63 DELIMITER //
64 • CREATE TRIGGER tr_update_books_no_of_copies_after_book_returned
65 AFTER UPDATE ON transactions
66 FOR EACH ROW
67 BEGIN
68     -- Check if the book was just marked as returned
69     IF OLD.return_date IS NULL AND NEW.return_date IS NOT NULL THEN
70         -- Update the number of available copies for the book
71         UPDATE book_availability
72             SET no_of_copies = no_of_copies + 1
73             WHERE book_id = NEW.book_id;
74     END IF;
75 END //
76 DELIMITER ;
```

The output window shows the execution log:

Action	Time	Message	Duration / Fetch
CREATE TRIGGER	19:45:39	CREATE TRIGGER tr_update_books_no_of_copies_after_book_returned AFTER UPDATE ON transactions FOR EACH ROW BEGIN... 0 row(s) affected	0.016 sec

Query Completed

Figure 14

The screenshot shows the MySQL Workbench interface with the 'Triggers' tab selected. The SQL editor contains the following code:

```
45
46 -- Create a trigger to update the number of copies available after a new transaction is inserted
47
48 DELIMITER //
49 • CREATE TRIGGER tr_update_books_no_of_copies_issued
50 AFTER INSERT ON transactions
51 FOR EACH ROW
52 BEGIN
53     -- Update the number of copies in the book_availability table
54     UPDATE book_availability
55         SET no_of_copies = no_of_copies - 1
56         WHERE book_id = NEW.book_id AND no_of_copies > 0; -- Ensure no negative copies
57 END //
58
59 DELIMITER ;
```

The output window shows the execution log:

Action	Time	Message	Duration / Fetch
CREATE TRIGGER	17:42:37	CREATE TRIGGER tr_update_books_no_of_copies_issued AFTER INSERT ON transactions FOR EACH ROW BEGIN... 0 row(s) affected	0.016 sec

Figure 15

The screenshot shows the MySQL Workbench interface. In the top-left pane, a script editor displays the following SQL code:

```
26
27 • INSERT INTO genre (genre_name) VALUES
28     ('Fiction'), ('Non-Fiction'), ('Mystery'), ('Thriller'), ('Fantasy'), ('Science Fiction'), ('Biography'), ('Romance'),
29     ('Historical Fiction'), ('Self-Help'), ('Horror'), ('Adventure'), ('Poetry'), ('Graphic Novel'), ('Young Adult'), ('Technology'),
30     ('Classic'), ('Dystopian'), ('Coming-of-age');
31
32 • SELECT * FROM genre;
33
```

In the bottom-left pane, the "Result Grid" shows the data inserted into the "genre" table:

genre_id	genre_name
8	Romance
6	Science Fiction
10	Self-Help
16	Technology
4	Thriller
15	Young Adult

The bottom-right pane, titled "Output", displays the execution log:

#	Time	Action	Message	Duration / Fetch
1	17:45:49	INSERT INTO genre (genre_name) VALUES ('Fiction'), ('Non-Fiction'), ('Mystery'), ('Thriller'), ('Fantasy'), ('Science Fiction'), ('Biography'), ('Romance'), ('Historical Fiction'), ('Self-Help'), ('Horror'), ('Adventure'), ('Poetry'), ('Graphic Novel'), ('Young Adult'), ('Technology'), ('Classic'), ('Dystopian'), ('Coming-of-age');	19 row(s) affected Records: 19 Duplicates: 0 Warnings: 0	0.000 sec
2	17:47:09	SELECT * FROM genre LIMIT 0, 1000	19 row(s) returned	0.000 sec / 0.000 sec

Figure 16

The screenshot shows the MySQL Workbench interface. In the top-left pane, a script editor displays the following SQL code with explanatory comments:

```
16 -- 12. Adventure: A genre that involves exciting and often dangerous journeys or quests.
17 -- 13. Poetry: A literary genre that expresses ideas and emotions through rhythmic and metaphorical language.
18 -- 14. Graphic Novel: A genre that combines visual art with narrative storytelling, often in comic book format.
19 -- 15. Young Adult: A genre targeted towards teenage readers, often dealing with themes relevant to young adults.
20 -- 16. Technology: A genre that explores themes related to technology and its impact on society.
21 -- 17. Classic: A genre that includes works recognized for their literary merit and lasting significance.
22 -- 18. Dystopian: A genre that explores social and political structures in a dark, nightmare world.
23 -- 19. Coming-of-age: A genre that focuses on the growth and development of a protagonist from youth to adulthood.
24 --
25 -- The following SQL statement inserts these genres into the 'genre' table:
26
27 • INSERT INTO genre (genre_name) VALUES
28     ('Fiction'), ('Non-Fiction'), ('Mystery'), ('Thriller'), ('Fantasy'), ('Science Fiction'), ('Biography'), ('Romance'),
29     ('Historical Fiction'), ('Self-Help'), ('Horror'), ('Adventure'), ('Poetry'), ('Graphic Novel'), ('Young Adult'), ('Technology'),
30     ('Classic'), ('Dystopian'), ('Coming-of-age');
31
32
```

The bottom-right pane, titled "Output", displays the execution log:

#	Time	Action	Message	Duration / Fetch
1	17:45:49	INSERT INTO genre (genre_name) VALUES ('Fiction'), ('Non-Fiction'), ('Mystery'), ('Thriller'), ('Fantasy'), ('Science Fiction'), ('Biography'), ('Romance'), ('Historical Fiction'), ('Self-Help'), ('Horror'), ('Adventure'), ('Poetry'), ('Graphic Novel'), ('Young Adult'), ('Technology'), ('Classic'), ('Dystopian'), ('Coming-of-age');	19 row(s) affected Records: 19 Duplicates: 0 Warnings: 0	0.000 sec

Figure 17

```

51
52 -- Inserting demo data into the 'books' table to populate it with sample book information, including ISBN, title, publisher, published year, n
53
54 • INSERT INTO books (isbn, title, publisher, published_year, no_of_copies, genre_id, author_id) VALUES
55 ('978-3-16-148410-0', 'Harry Potter and the Philosophers Stone', 'Bloomsbury', 1997, 10, 1, 1),
56 ('978-0-7432-7356-5', 'A Game of Thrones', 'Bantam Books', 1996, 5, 2, 2),
57 ('978-0-06-112008-4', 'To Kill a Mockingbird', 'J.B. Lippincott & Co.', 1960, 8, 3, 3),
58 ('978-0-452-28423-4', 'The Great Gatsby', 'Charles Scribners Sons', 1925, 6, 3, 4),
59 ('978-1-56619-909-4', 'Norwegian Wood', 'Harvill Secker', 1987, 7, 3, 5),
60 ('978-0-14-028333-4', 'Things Fall Apart', 'Heinemann', 1958, 4, 3, 6),
61 ('978-0-19-953556-9', 'Pride and Prejudice', 'T. Egerton', 1813, 9, 4, 7),
62 ('978-1-5011-2630-0', 'The Shining', 'Doubleday', 1977, 12, 5, 8),
63 ('978-0-06-231500-7', 'The House of the Spirits', 'Alfred A. Knopf', 1982, 3, 6, 9),
64 ('978-0-06-088328-7', 'One Hundred Years of Solitude', 'Harper & Row', 1967, 2, 6, 10);
65
66 • SELECT * FROM books;

```

Output

#	Time	Action	Message	Duration / Fetch
1	17:48:42	INSERT INTO authors (author_name, author_email, author_country) VALUES ('J.K. Rowling', 'jk.rowling@example.com', 'United Kingdom')	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.000 sec
2	17:49:08	SELECT * FROM authors LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

Figure 18

```

42 ('Haruki Murakami', 'haruki.murakami@example.com', 'Japan'),
43 ('Chinua Achebe', 'chinua.achebe@example.com', 'Nigeria'),
44 ('Jane Austen', 'jane.austen@example.com', 'United Kingdom'),
45 ('Stephen King', 'stephen.king@example.com', 'United States'),
46 ('Isabel Allende', 'isabel.allende@example.com', 'Chile'),
47 ('Gabriel García Márquez', 'gabriel.garcia@example.com', 'Colombia');
48
49 • SELECT * FROM authors;

```

Result Grid

author_id	author_name	author_email	author_country
1	J.K. Rowling	jk.rowling@example.com	United Kingdom
2	George R.R. Martin	george.martin@example.c...	United States
3	Agatha Christie	agatha.christie@example....	United Kingdom
4	Mark Twain	mark.twain@example.com	United States
5	Haruki Murakami	haruki.murakami@exampl...	Japan
6	Chinua Achebe	chinua.achebe@example....	Nigeria

Output

#	Time	Action	Message	Duration / Fetch
1	17:48:42	INSERT INTO authors (author_name, author_email, author_country) VALUES ('J.K. Rowling', 'jk.rowling@example.com', 'United Kingdom')	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.000 sec
2	17:49:08	SELECT * FROM authors LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

Figure 19

The screenshot shows the MySQL Workbench interface. In the top-left pane, there is a SQL editor window containing the following code:

```

62 ('978-0-06-231500-7', 'The House of the Spirits', 'Alfred A. Knopf', 1982, 3, 6, 9),
63 ('978-0-06-088328-7', 'One Hundred Years of Solitude', 'Harper & Row', 1967, 2, 6, 10);
64
65 • SELECT * FROM books;
66
67
68 • SELECT * FROM book_availability;
69

```

Below the SQL editor is a Results Grid showing the output of the last query:

book_id	no_of_copies
21	10
22	5
23	8
24	6
25	7
26	4

At the bottom of the interface, the Output panel displays the execution log:

#	Time	Action	Message	Duration / Fetch
1	18:13:24	INSERT INTO books (isbn, title, publisher, published_year, no_of_copies, genre_id, author_id) VALUES ('978-3-16-148410-0', 'Harry P...', 10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0)		0.000 sec
2	18:13:31	SELECT * FROM books LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
3	18:14:45	SELECT * FROM book_availability LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

The status bar at the bottom right indicates "Query Completed".

Figure 20

The screenshot shows the MySQL Workbench interface. In the top-left pane, there is a SQL editor window containing the following code:

```

59 ('978-0-14-028333-4', 'Things Fall Apart', 'Heinemann', 1958, 4, 3, 6),
60 ('978-0-19-953556-9', 'Pride and Prejudice', 'T. Egerton', 1913, 9, 4, 7),
61 ('978-1-5011-2630-0', 'The Shining', 'Doubleday', 1977, 12, 5, 8),
62 ('978-0-06-231500-7', 'The House of the Spirits', 'Alfred A. Knopf', 1982, 3, 6, 9),
63 ('978-0-06-088328-7', 'One Hundred Years of Solitude', 'Harper & Row', 1967, 2, 6, 10);
64
65 • SELECT * FROM books;
66

```

Below the SQL editor is a Results Grid showing the output of the last query:

isbn	book_id	title	publisher	published_year	no_of_copies	genre_id	author_id
978-3-16-148410-0	21	Harry Potter and the Philo...	Bloomsbury	1997	10	1	1
978-0-7432-7356-5	22	A Game of Thrones	Bantam Books	1996	5	2	2
978-0-06-112008-4	23	To Kill a Mockingbird	J.B. Lippincott & Co.	1960	8	3	3
978-0-452-28423-4	24	The Great Gatsby	Charles Scribners Sons	1925	6	3	4
978-1-56619-909-4	25	Norwegian Wood	Harvill Secker	1987	7	3	5
978-0-14-028333-4	26	Things Fall Apart	Heinemann	1958	4	2	6

At the bottom of the interface, the Output panel displays the execution log:

#	Time	Action	Message	Duration / Fetch
1	18:13:24	INSERT INTO books (isbn, title, publisher, published_year, no_of_copies, genre_id, author_id) VALUES ('978-3-16-148410-0', 'Harry P...', 10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0)		0.000 sec
2	18:13:31	SELECT * FROM books LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

The status bar at the bottom right indicates "SQL Editor closed".

Figure 21

The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Displays the SQL query:


```
97  (22, 6, '2024-02-10', '2024-03-10', NULL, 0.00),
98  (25, 8, '2023-11-28', '2023-12-28', '2024-01-02', 20.00),
99  (21, 4, '2023-10-15', '2023-11-15', '2023-11-10', 0.00),
100 (21, 7, '2024-01-20', '2024-02-20', NULL, 0.00),
101 (29, 7, '2023-12-12', '2024-01-12', '2024-01-10', 0.00);
102
103 • SELECT * FROM transactions;
```
- Result Grid:** Shows the results of the query, displaying 18 rows from the transactions table.
- Output Panel:** Displays the execution log with two entries:

#	Time	Action	Message	Duration / Fetch
1	18:26:38	INSERT INTO transactions (book_id, member_id, issue_date, due_date, return_date, fine_amount) VALUES (25, 1, '2023-11-15', '2024-12-08', '2023-12-10', 0.00)	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.078 sec
2	18:26:41	SELECT * FROM transactions LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

Figure 22

The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Displays the SQL query:


```
79  ('Ethan Hunt', '654 Cedar St, Springfield', 'ethan.hunt@example.com', 'Faculty', 45),
80  ('Fiona Gallagher', '987 Birch St, Springfield', 'fiona.gallagher@example.com', 'Student', 22),
81  ('George Costanza', '135 Willow St, Springfield', 'george.costanza@example.com', 'General', 40),
82  ('Hannah Baker', '246 Spruce St, Springfield', 'hannah.baker@example.com', 'Temporary', 30);
83
84
85 • SELECT * FROM members;
```
- Result Grid:** Shows the results of the query, displaying 13 rows from the members table.
- Output Panel:** Displays the execution log with two entries:

#	Time	Action	Message	Duration / Fetch
1	18:15:41	INSERT INTO members (full_name, address, email, membership_type, age) VALUES ('Alice Johnson', '123 Maple St, Springfield', 'alice.johnson@example.com', 'Senior citizen', 65)	8 row(s) affected Records: 8 Duplicates: 0 Warnings: 0	0.016 sec
2	18:15:44	SELECT * FROM members LIMIT 0, 1000	8 row(s) returned	0.000 sec / 0.000 sec

Figure 23

The screenshot shows the MySQL Workbench interface with the 'Tools' tab selected. A SQL editor window displays the following code:

```
1 -- Create or replace a view named 'get_book_details' to retrieve detailed information about books
2 • CREATE OR REPLACE VIEW get_book_details AS
3   SELECT
4     b.book_id AS book_id,          -- Unique identifier for the book
5     b.title AS title,             -- Title of the book
6     b.author_id AS author_id,      -- Unique identifier for the author
7     a.author_name AS author_name,  -- Name of the author
8     b.genre_id AS genre_id,        -- Unique identifier for the genre
9     g.genre_name AS genre_name,    -- Name of the genre
10    b.publisher AS publisher,      -- Publisher of the book
11    b.published_year AS published_year, -- Year the book was published
12    b.isbn AS isbn,               -- ISBN number of the book
13    b.no_of_copies AS no_of_copies, -- Number of copies available
14   FROM books b
15  JOIN authors a ON b.author_id = a.author_id -- Join with authors table
16  JOIN genre g ON b.genre_id = g.genre_id -- Join with genre table
17 ORDER BY b.book_id DESC;           -- Order results by book_id in descending order
```

The output pane shows the execution log:

Action	Time	Action	Message	Duration / Fetch
CREATE OR REPLACE VIEW get_book_details AS SELECT b.book_id AS book_id, -- Unique identifier for the book b... 0 row(s) affected	1 18:29:58			0.000 sec

Query Completed

Figure 24

The screenshot shows the MySQL Workbench interface with the 'Tools' tab selected. A SQL editor window displays the following code:

```
114
115 -- Inserting demo data into the 'librarian' table to populate it with sample librarian information, including names, addresses, emails, and so on
116
117 • INSERT INTO librarian (librarian_name, login_password, librarian_address, librarian_email) VALUES
118 ('John Doe', 'Nimda@36', '123 Library Lane, Springfield', 'john.doe@example.com');
119
120 • SELECT * FROM librarian;
```

The result grid shows the inserted data:

librarian_id	librarian_name	login_password	librarian_address	librarian_email
1	John Doe	Nimda@36	123 Library Lane, Springfield	john.doe@example.com

The output pane shows the execution log:

Action	Time	Action	Message	Duration / Fetch
INSERT INTO librarian (librarian_name, login_password, librarian_address, librarian_email) VALUES ('John Doe', 'Nimda@36', '123 Libr...', 1 row(s) affected	1 18:28:06			0.047 sec
SELECT * FROM librarian LIMIT 0,1000	2 18:28:09		1 row(s) returned	0.000 sec / 0.000 sec

Query Completed

Figure 25

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Database Tables Triggers Sample Data Views*
transaction_details2
34     t.return_date AS return_date,          -- Date the book was returned
35     t.fine_amount AS fine_amount          -- Fine amount for late returns
36  FROM transactions t
37  JOIN books b ON t.book_id = b.book_id   -- Join with books table
38  JOIN members m ON t.member_id = m.member_id; -- Join with members table;
39
40 • SELECT * FROM transaction_details;

```

Result Grid | Filter Rows | Export | Wrap Cell Content | Result Grid | Form Editor | Field Types | Read Only

transaction_id	full_name	book_id	title	issue_date	due_date	return_date	fine_amount
35	Bob Smith	27	Pride and Prejudice	2024-11-24	2024-12-08	2024-01-05	5.00
38	Diana Prince	21	Harry Potter and the Philo...	2024-11-24	2024-12-08	2023-11-10	0.00
32	Ethan Hunt	22	A Game of Thrones	2024-11-24	2024-12-08	2023-11-25	15.00
34	Ethan Hunt	28	The Shining	2024-11-24	2024-12-08	2023-10-20	0.00
36	Fiona Gallagher	22	A Game of Thrones	2024-11-24	2024-12-08	2023-11-25	0.00
39	George Costanza	21	Harry Potter and the Philo...	2024-11-24	2024-12-08	2023-11-25	0.00
40	George Costanza	29	The House of the Spirits	2024-11-24	2024-12-08	2024-01-10	0.00

transaction_details2 x

Action Output

#	Time	Action	Message	Duration / Fetch
1	18:33:41	CREATE OR REPLACE VIEW transaction_details AS SELECT t.transaction_id AS transaction_id, -- Unique identifier for each transaction ... 0 row(s) affected		0.016 sec
2	18:33:44	SELECT * FROM transaction_details LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

Query Completed

Figure 26

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Database Tables Triggers Sample Data Views*
book_details1
13     b.no_of_copies AS no_of_copies      -- Number of copies available
14  FROM books b                         -- Books table
15  JOIN authors a ON b.author_id = a.author_id -- Join with authors table
16  JOIN genre g ON b.genre_id = g.genre_id    -- Join with genre table
17  ORDER BY b.book_id DESC;                -- Order results by book_id in descending order
18
19 • SELECT * FROM get_book_details;
20

```

Result Grid | Filter Rows | Export | Wrap Cell Content | Result Grid | Form Editor | Field Types | Read Only

book_id	title	author_id	author_name	genre_id	genre_name	publisher	published_y	isbn	no_of_copies
30	One Hundred Years of...	10	Gabriel García M...	6	Science Fiction	Harper & Row	1967	978-0-06-088328-7	2
29	The House of the Spirits	9	Isabel Allende	6	Science Fiction	Alfred A. Knopf	1982	978-0-06-231500-7	3
28	The Shining	8	Stephen King	5	Fantasy	Doubleday	1977	978-1-5011-2630-0	12
27	Pride and Prejudice	7	Jane Austen	4	Thriller	T. Egerton	1913	978-0-19-953556-9	9
26	Things Fall Apart	6	Chinua Achebe	3	Mystery	Heinemann	1958	978-0-14-028333-4	4
25	Norwegian Wood	5	Haruki Murakami	3	Mystery	Harvill Secker	1987	978-1-56619-909-4	7
24	The Great Gatsby	4	Mark Twain	3	Mystery	Charles Scribner's...	1925	978-0-452-28423-4	6

book_details1 x

Action Output

#	Time	Action	Message	Duration / Fetch
1	18:29:58	CREATE OR REPLACE VIEW get_book_details AS SELECT b.book_id AS book_id, -- Unique identifier for the book ... 0 row(s) affected		0.000 sec
2	18:31:06	SELECT * FROM get_book_details LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec

Query Completed

Figure 27

The screenshot shows the MySQL Workbench interface with the 'Stored Procedures' tab selected. The SQL editor contains the following code:

```
29 -- Create a stored procedure to retrieve author names by genre ID
30 DELIMITER //
31 • CREATE PROCEDURE getAuthorNameByGenreId(IN genre_id INT)
32 ◇ BEGIN
33     -- Select distinct author IDs and names for books that match the given genre ID
34     SELECT DISTINCT
35         a.author_id,          -- Unique identifier for the author
36         a.author_name        -- Name of the author
37     FROM |
38         books b            -- Books table
39     JOIN
40         authors a ON b.author_id = a.author_id -- Join with authors table
41     WHERE
42         b.genre_id = genre_id;    -- Filter by the provided genre ID
43     END //
44 DELIMITER ;
```

The output window shows the execution log:

#	Time	Action	Message	Duration / Fetch
1	18:47:36	CREATE PROCEDURE getAuthorNameByGenreId(IN genre_id INT) BEGIN -- Select distinct author IDs and names for books that m... 0 row(s) affected		0.000 sec

Query Completed

Figure 28

The screenshot shows the MySQL Workbench interface with the 'Stored Procedures' tab selected. The SQL editor contains the following code:

```
16     END //
17     DELIMITER ;
18
19
20 • SET @author_id = 0;
21 • CALL insert_author_name('J.K. Rowling', @author_id);
22 • SELECT @author_id; -- This will display the author ID
```

The result grid shows the value of the variable:

@author_id
1

The output window shows the execution log:

#	Time	Action	Message	Duration / Fetch
3	18:45:30	CALL insert_author_name('J.K. Rowling', @author_id)	1 row(s) affected	0.015 sec
4	18:45:34	SELECT @author_id LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

Query Completed

Figure 29

The screenshot shows the MySQL Workbench interface with the 'Stored Procedures' tab selected. A stored procedure named 'getAuthorNameByGenreId' is being created. The code includes a JOIN clause to link the 'authors' table to the 'books' table based on author IDs, a WHERE clause to filter by genre ID, and a call to the stored procedure.

```
39
40     authors a ON b.author_id = a.author_id -- Join with authors table
41
42     b.genre_id = genre_id; -- Filter by the provided genre ID
43 END //
44 DELIMITER ;
45
46 • CALL getAuthorNameByGenreId(3); -- Replace 1 with the desired genre ID
```

The results grid shows four rows of data:

author_id	author_name
3	Agatha Christie
4	Mark Twain
5	Haruki Murakami
6	Chinua Achebe

The output pane shows the execution log:

Action	Time	Message	Duration / Fetch
CREATE PROCEDURE	1 18:47:36	getAuthorNameByGenreId(N genre_id INT) BEGIN -- Select distinct author IDs and names for books that...	0.000 sec
CALL	2 18:49:25	getAuthorNameByGenreId()	0.000 sec / 0.000 sec
CALL	3 18:50:45	getAuthorNameByGenreId(3)	0.000 sec / 0.000 sec

Query Completed

Figure 30

The screenshot shows the MySQL Workbench interface with the 'Stored Procedures' tab selected. A stored procedure named 'getBookTitleByAuthorId' is being created. The code includes joins to link the 'books' table with the 'authors' and 'genre' tables, and a WHERE clause to filter by author ID.

```
57     FROM books b -- Books table
58     JOIN authors a ON b.author_id = a.author_id -- Join with authors table
59     JOIN genre g ON b.genre_id = g.genre_id -- Join with genre table (if needed)
60     WHERE a.author_id = author_id; -- Filter by the provided author ID
61 END //
62 DELIMITER ;
63
64 • CALL getBookTitleByAuthorId(3); -- Replace 1 with the desired author ID
```

The results grid shows one row of data:

book_id	title
23	To Kill a Mockingbird

The output pane shows the execution log:

Action	Time	Message	Duration / Fetch
CREATE PROCEDURE	1 18:52:49	getBookTitleByAuthorId(N author_id INT) BEGIN -- Select distinct book IDs and titles for books written by t...	0.000 sec
CALL	2 18:53:26	getBookTitleByAuthorId(3)	0.000 sec / 0.000 sec

Query Completed