

Assignment

❖ Extra Lab Practice For Database Concepts

1. Introduction to SQL

Lab 3: Create a database called library_db and a table books with columns: book_id, title, author, publisher, year_of_publication, and price. Insert five records into the table.

Answer:

- Create database and books table

Step 1: Create database

```
CREATE DATABASE library_db;
```

Step 2: Use the database

```
USE library_db;
```

Step 3: Create books table

```
CREATE TABLE books (
    book_id INT PRIMARY KEY,
    title VARCHAR(100),
    author VARCHAR(50),
    publisher VARCHAR(50),
    year_of_publication YEAR,
    price DECIMAL(6,2)
);
```

Step 4: Insert 5 records into books table

```
INSERT INTO books (book_id, title, author, publisher, year_of_publication, price)
VALUES
(1, 'Harry Potter', 'J.K. Rowling', 'Bloomsbury', 2001, 500.00),
(2, 'Lord of Rings', 'J.R.R. Tolkien', 'Allen & Unwin', 1954, 600.00),
(3, 'The Alchemist', 'Paulo Coelho', 'HarperCollins', 1988, 300.00),
```

(4, 'Pride & Prejudice', 'Jane Austen', 'T. Egerton', 2020, 250.00),

(5, 'Inferno', 'Dan Brown', 'Doubleday', 2013, 450.00);

Step 5: View records

SELECT * FROM books;

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** 127.0.0.1
- Database:** library_db
- Table:** books
- SQL Query:** SELECT * FROM `books`
- Results:** A grid of 5 rows representing book data.

	book_id	title	author	publisher	year_of_publication	price
<input type="checkbox"/>	1	Harry Potter	J.K. Rowling	Bloomsbury	2001	500.00
<input type="checkbox"/>	2	Lord of Rings	J.R.R. Tolkien	Allen & Unwin	1954	600.00
<input type="checkbox"/>	3	The Alchemist	Paulo Coelho	HarperCollins	1988	300.00
<input type="checkbox"/>	4	Pride & Prejudice	Jane Austen	T. Egerton	2020	250.00
<input type="checkbox"/>	5	Inferno	Dan Brown	Doubleday	2013	450.00

Lab 4: Create a table members in library_db with columns: member_id, member_name, date_of_membership, and email. Insert five records into this table.

Answer:

- **Create members table**

Step 1: Create members table

```
CREATE TABLE members (
    member_id INT PRIMARY KEY,
    member_name VARCHAR(50),
    date_of_membership DATE,
    email VARCHAR(50)
);
```

Step 2: Insert 5 records into members table

```
INSERT INTO members (member_id, member_name, date_of_membership, email)
VALUES
(1, 'Alice', '2025-01-10', 'alice@example.com'),
(2, 'Bob', '2025-02-15', 'bob@example.com'),
(3, 'Charlie', '2025-03-20', 'charlie@example.com'),
(4, 'David', '2025-04-25', 'david@example.com'),
(5, 'Eva', '2025-05-30', 'eva@example.com');
```

Step 3: View records

```
SELECT * FROM members;
```

The screenshot shows the phpMyAdmin interface with the 'library_db' database selected. The 'members' table is displayed in the main pane. The table structure includes columns: member_id, member_name, date_of_membership, and email. There are 5 records listed:

	member_id	member_name	date_of_membership	email
<input type="checkbox"/>	1	Alice	2025-01-10	alice@example.com
<input type="checkbox"/>	2	Bob	2025-02-15	bob@example.com
<input type="checkbox"/>	3	Charlie	2025-03-20	charlie@example.com
<input type="checkbox"/>	4	David	2025-04-25	david@example.com
<input type="checkbox"/>	5	Eva	2025-05-30	eva@example.com

2. SQL Syntax

Lab 3: Retrieve all members who joined the library before 2022. Use appropriate SQL syntax with WHERE and ORDER BY.

Answer:

Query: Retrieve all members who joined the library before 2022

```
SELECT member_id, member_name, date_of_membership, email FROM members  
WHERE date_of_membership < '2022-01-01' ORDER BY date_of_membership ASC;
```

The screenshot shows the phpMyAdmin interface. On the left, the database structure is visible, including the 'library_db' database which contains tables like 'books' and 'members'. The main area shows a query result for the following SQL statement:

```
SELECT member_id, member_name, date_of_membership, email FROM members WHERE date_of_membership < '2022-01-01' ORDER BY date_of_membership ASC;
```

The result set is empty, as indicated by the message: "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)". Below the query, there are options for "Query results operations" such as "Create view" and "Bookmark this SQL query".

- These data have membership dates only in 2025, so no records – because no one joined before 2022.

Lab 4: Write SQL queries to display the titles of books published by a specific author. Sort the results by year_of_publication in descending order.

Answer:

Query: Display the titles of books published by a specific author (sorted DESC)

Example: Author – J.K. Rowling

```
SELECT title, author, year_of_publication
```

```
FROM books
```

```
WHERE author = 'J.K. Rowling'
```

```
ORDER BY year_of_publication DESC;
```

The screenshot shows the phpMyAdmin interface. On the left, the database structure is visible with databases like employee_information_table, information_schema, library_db, members, mysql, performance_schema, school_db, test, university_db, and world_data. The library_db database is selected. In the center, the 'Table: books' page is shown. The query results are displayed in a table:

	title	author	year_of_publication
<input type="checkbox"/>	Harry Potter	J.K. Rowling	2001

The query entered in the SQL tab is:

```
SELECT title, author, year_of_publication FROM books WHERE author = 'J.K. Rowling' ORDER BY year_of_publication DESC;
```

3. SQL Constraints

Lab 3: Add a CHECK constraint to ensure that the price of books in the books table is greater than 0.

Answer:

Query: Add a Check constraint to ensure price > 0

```
ALTER TABLE books ADD CONSTRAINT chk_price CHECK (price > 0);
```

The screenshot shows the phpMyAdmin interface. On the left, the database tree is visible with the 'library_db' database selected. The main area shows the SQL tab with the following content:

```
ALTER TABLE books ADD CONSTRAINT chk_price CHECK (price > 0);
```

Below the query, the status message reads: "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)". There are three buttons at the bottom of the query box: "Edit inline", "Edit", and "Create PHP code".

- This ensures price cannot be 0 or negative.

Lab 4: Modify the members table to add a UNIQUE constraint on the email column, ensuring that each member has a unique email address.

Answer:

Query: Add UNIQUE constraint on email column in members table

ALTER TABLE members ADD CONSTRAINT unique_email UNIQUE (email);

The screenshot shows the phpMyAdmin interface. On the left, the database tree is visible with the 'library_db' database selected. The main area shows a query result: 'MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)' Below this, the executed SQL command is shown: 'ALTER TABLE members ADD CONSTRAINT unique_email UNIQUE (email);'. There are three buttons at the bottom of the query box: 'Edit inline', 'Edit', and 'Create PHP code'.

- This ensures no two members can have the same email.

4. Main SQL Commands and Sub-commands (DDL)

Lab 3: Create a table authors with the following columns: author_id, first_name, last_name, and country. Set author_id as the primary key.

Answer:

Query: Create a Table

```
CREATE TABLE authors (
    author_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    country VARCHAR(50)
);
```



The screenshot shows the phpMyAdmin interface for a database named 'library_db'. The 'authors' table is selected. The 'Table structure' tab is active, displaying the following schema:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	author_id	int(11)	utf8mb4_general_ci		No	None			Change Drop More
2	first_name	varchar(50)	utf8mb4_general_ci		Yes	NULL			Change Drop More
3	last_name	varchar(50)	utf8mb4_general_ci		Yes	NULL			Change Drop More
4	country	varchar(50)	utf8mb4_general_ci		Yes	NULL			Change Drop More

Below the table structure, there are buttons for 'Check all', 'With selected:', and various table operations like 'Browse', 'Change', 'Drop', 'Primary', 'Unique', 'Index', 'Spatial', 'Fulltext', and 'Add to central columns'. The left sidebar shows other databases and tables in the 'library_db' schema.

Lab 4: Create a table publishers with columns: publisher_id, publisher_name, contact_number, and address. Set publisher_id as the primary key and contact_number as unique.

Answer:

Query: Create a Table publisher with UNIQUE contact_number

```
CREATE TABLE publishers (
    publisher_id INT PRIMARY KEY,
    publisher_name VARCHAR(100),
    contact_number VARCHAR(20) UNIQUE,
    address VARCHAR(100)
);
```

The screenshot shows the phpMyAdmin interface for creating a table named 'publisher'. The left sidebar lists databases and tables, with 'library_db' selected. The main area shows the 'Table structure' tab for the 'publisher' table. The table has four columns: 'publisher_id' (int(11), primary key, not null), 'publisher_name' (varchar(100), general_ci), 'contact_number' (varchar(20), general_ci), and 'address' (varchar(100), general_ci). Below the table structure, the 'Indexes' section shows two indexes: 'PRIMARY' (on publisher_id, BTREE, unique, no) and 'contact_number' (on contact_number, BTREE, yes, no).

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit	PRIMARY	BTREE	Yes	No	publisher_id	0	A	No	
Edit	contact_number	BTREE	Yes	No	contact_number	0	A	Yes	

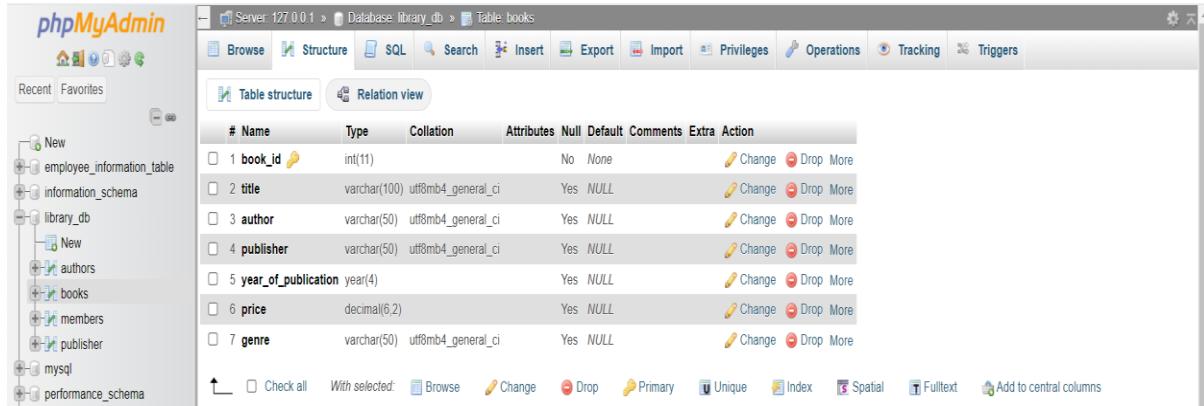
5. ALTER Command

Lab 3: Add a new column genre to the books table. Update the genre for all existing records.

Answer:

Query 1: Add a new column genre to the books table

```
ALTER TABLE books ADD genre VARCHAR(50);
```



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	book_id	int(11)			No	None			More
2	title	varchar(100)	utf8mb4_general_ci		Yes	NULL			More
3	author	varchar(50)	utf8mb4_general_ci		Yes	NULL			More
4	publisher	varchar(50)	utf8mb4_general_ci		Yes	NULL			More
5	year_of_publication	year(4)			Yes	NULL			More
6	price	decimal(6,2)			Yes	NULL			More
7	genre	varchar(50)	utf8mb4_general_ci		Yes	NULL			More

Query 2: Update genre for all existing books

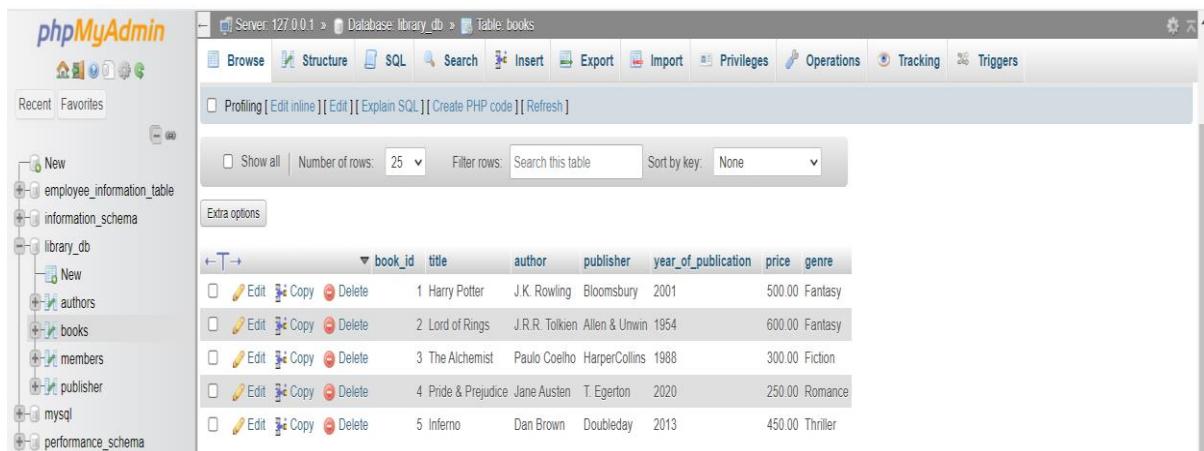
```
UPDATE books SET genre = 'Fantasy' WHERE book_id = 1;
```

```
UPDATE books SET genre = 'Fantasy' WHERE book_id = 2;
```

```
UPDATE books SET genre = 'Fiction' WHERE book_id = 3;
```

```
UPDATE books SET genre = 'Romance' WHERE book_id = 4;
```

```
UPDATE books SET genre = 'Thriller' WHERE book_id = 5;
```



	book_id	title	author	publisher	year_of_publication	price	genre
1	1	Harry Potter	J.K. Rowling	Bloomsbury	2001	500.00	Fantasy
2	2	Lord of Rings	J.R.R. Tolkien	Allen & Unwin	1954	600.00	Fantasy
3	3	The Alchemist	Paulo Coelho	HarperCollins	1988	300.00	Fiction
4	4	Pride & Prejudice	Jane Austen	T. Egerton	2020	250.00	Romance
5	5	Inferno	Dan Brown	Doubleday	2013	450.00	Thriller

Lab 4: Modify the members table to increase the length of the email column to 100 characters.

Answer:

Query: Modify email length in members table to 100 characters

```
ALTER TABLE members MODIFY email VARCHAR(100);
```

The screenshot shows the phpMyAdmin interface for a MySQL database named 'library_db'. The 'members' table is selected. The table structure is displayed with the following columns:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	member_id	int(11)			No	None			Change Drop More
2	member_name	varchar(50)	utf8mb4_general_ci		Yes	NULL			Change Drop More
3	date_of_membership	date			Yes	NULL			Change Drop More
4	email	varchar(100)	utf8mb4_general_ci		Yes	NULL			Change Drop More

At the bottom of the table structure view, there are several buttons: 'Check all', 'With selected:', 'Browse', 'Change', 'Drop', 'Primary', 'Unique', 'Index', 'Spatial', 'Fulltext', and 'Add to central columns'. There is also a link 'Remove from central columns'.

6. DROP Command

Lab 3: Drop the publishers table from the database after verifying its structure.

Answer:

Query 1: Verify table structure

DESCRIBE publisher;

The screenshot shows the phpMyAdmin interface for the 'library_db' database. The left sidebar lists various tables and databases. The main area shows the structure of the 'publisher' table with the following columns:

Field	Type	Null	Key	Default	Extra
publisher_id	int(11)	NO	PRI	NULL	
publisher_name	varchar(100)	YES		NULL	
contact_number	varchar(20)	YES	UNI	NULL	
address	varchar(100)	YES		NULL	

Query 2: Drop the table

DROP TABLE publisher;

The screenshot shows the phpMyAdmin interface for the 'library_db' database. The left sidebar lists various tables and databases. The main area displays a message: "MySQL returned an empty result set (i.e. zero rows). (Query took 0.0020 seconds.)". This indicates that the table has been successfully dropped.

Lab 4: Create a backup of the members table and then drop the original members table.

Answer:

Query 1: Create backup table

```
CREATE TABLE members_backup AS SELECT * FROM members;
```

The screenshot shows the phpMyAdmin interface for the 'library_db' database. In the left sidebar, the 'members' table is selected under the 'library_db' schema. The main area displays the results of the query 'SELECT * FROM members_backup'. The results show five rows of data with columns: member_id, member_name, date_of_membership, and email. The data is as follows:

member_id	member_name	date_of_membership	email
1	Alice	2025-01-10	alice@example.com
2	Bob	2025-02-15	bob@example.com
3	Charlie	2025-03-20	charlie@example.com
4	David	2025-04-25	david@example.com
5	Eva	2025-05-30	eva@example.com

Query 2: Drop the Original Members table

```
DROP TABLE members;
```

The screenshot shows the phpMyAdmin interface for the 'library_db' database. In the left sidebar, the 'members' table is selected under the 'library_db' schema. The main area displays the results of the query 'DROP TABLE members;'. The message in the results area is: 'MySQL returned an empty result set (i.e. zero rows). (Query took 0.0011 seconds.)'.

Query 3: View backup table (optional)

```
SELECT * FROM members_backup;
```

The screenshot shows the phpMyAdmin interface for the 'library_db' database. In the left sidebar, the 'members_backup' table is selected under the 'library_db' schema. The main area displays the results of the query 'SELECT * FROM members_backup'. The results show the same five rows of data as the original members table, with columns: member_id, member_name, date_of_membership, and email. The data is as follows:

member_id	member_name	date_of_membership	email
1	Alice	2025-01-10	alice@example.com
2	Bob	2025-02-15	bob@example.com
3	Charlie	2025-03-20	charlie@example.com
4	David	2025-04-25	david@example.com
5	Eva	2025-05-30	eva@example.com

7. Data Manipulation Language (DML)

Lab 4: Insert three new authors into the authors table, then update the last name of one of the authors.

Answer:

Query 1: Insert 3 authors

```
INSERT INTO authors (author_id, first_name, last_name, country)
```

```
VALUES
```

```
(1, 'George', 'Orwell', 'United Kingdom'),  
(2, 'Mark', 'Twain', 'United States'),  
(3, 'Agatha', 'Christie', 'United Kingdom');
```

The screenshot shows the phpMyAdmin interface for the 'library_db' database. The 'authors' table is selected. The table structure is shown with columns: author_id, first_name, last_name, and country. There are three rows of data: (1, George, Orwell, United Kingdom), (2, Mark, Twain, United States), and (3, Agatha, Christie, United Kingdom). Each row has edit, copy, delete, and export options.

Query 2: Update last name of one author

Example: Change Mark Twain -> Mark Johnson

```
UPDATE authors SET last_name = 'Johnson' WHERE author_id = 2;
```

The screenshot shows the phpMyAdmin interface for the 'library_db' database. The 'authors' table is selected. The table structure is shown with columns: author_id, first_name, last_name, and country. There are three rows of data: (1, George, Orwell, United Kingdom), (2, Mark, Johnson, United States), and (3, Agatha, Christie, United Kingdom). Each row has edit, copy, delete, and export options. The 'last_name' column for the second row has been updated from 'Twain' to 'Johnson'.

Lab 5: Delete a book from the books table where the price is higher than \$100.

Answer:

Query: Delete a book where price > 100

DELETE FROM books WHERE price > 100 LIMIT 1;

- Based on data, all books have price > 100, so deleting any one book is valid.

book_id	title	author	publisher	year_of_publication	price	genre
2	Lord of Rings	J.R.R. Tolkien	Allen & Unwin	1954	600.00	Fantasy
3	The Alchemist	Paulo Coelho	HarperCollins	1988	300.00	Fiction
4	Pride & Prejudice	Jane Austen	T. Egerton	2020	250.00	Romance
5	Inferno	Dan Brown	Doubleday	2013	450.00	Thriller

- Deletes only one book, even though many match.

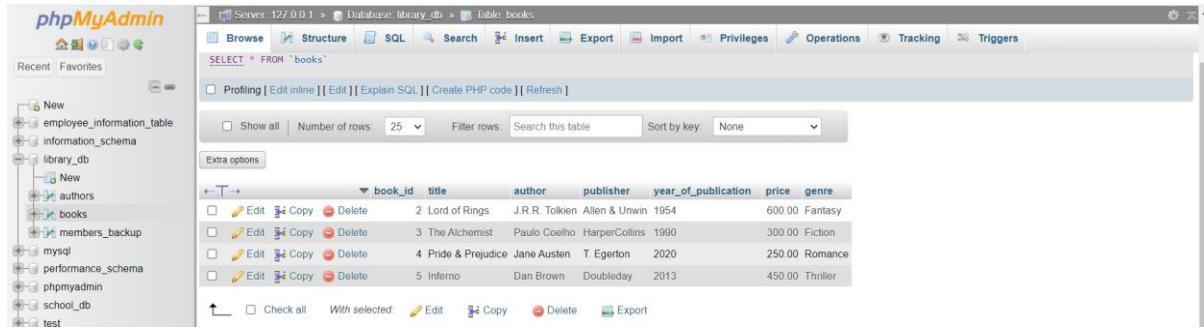
8. UPDATE Command

Lab 3: Update the year_of_publication of a book with a specific book_id.

Answer:

Query: Update the year of publication of book_id = 3 (The Alchemist)

UPDATE books SET year_of_publication = 1990 WHERE book_id = 3;



The screenshot shows the phpMyAdmin interface for the 'library_db' database. The left sidebar lists databases and tables, including 'books'. The main area displays the 'books' table with the following data:

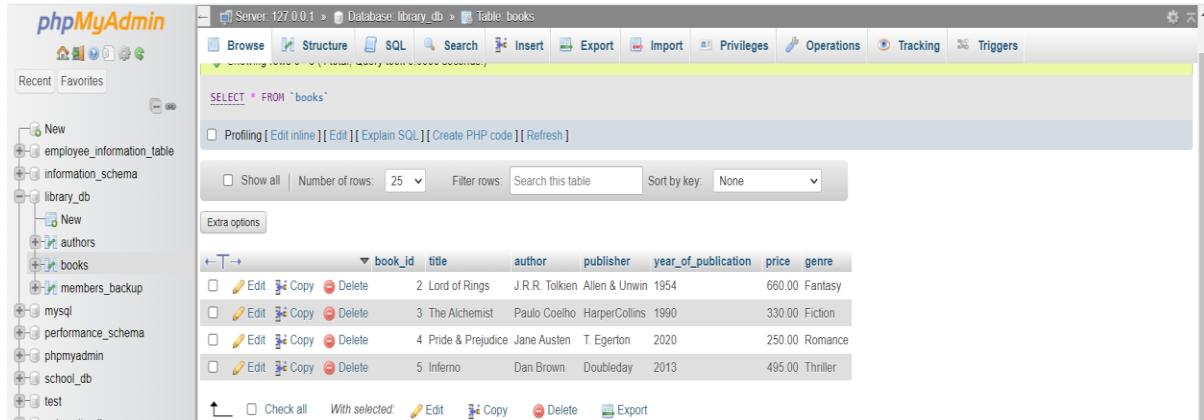
book_id	title	author	publisher	year_of_publication	price	genre
2	Lord of Rings	J.R.R. Tolkien	Allen & Unwin	1954	600.00	Fantasy
3	The Alchemist	Paulo Coelho	HarperCollins	1990	300.00	Fiction
4	Pride & Prejudice	Jane Austen	T. Egerton	2020	250.00	Romance
5	Inferno	Dan Brown	Doubleday	2013	450.00	Thriller

Lab 4: Increase the price of all books published before 2015 by 10%.

Answer:

Query: increase price of all books

UPDATE books SET price = price * 1.10 WHERE year_of_publication < 2015;



The screenshot shows the phpMyAdmin interface for the 'library_db' database. The left sidebar lists databases and tables, including 'books'. The main area displays the 'books' table with the following data:

book_id	title	author	publisher	year_of_publication	price	genre
2	Lord of Rings	J.R.R. Tolkien	Allen & Unwin	1954	660.00	Fantasy
3	The Alchemist	Paulo Coelho	HarperCollins	1990	330.00	Fiction
4	Pride & Prejudice	Jane Austen	T. Egerton	2020	250.00	Romance
5	Inferno	Dan Brown	Doubleday	2013	495.00	Thriller

9. DELETE Command

Lab 3: Remove all members who joined before 2020 from the members table.

Answer:

Query: Remove all members who joined before 2020

DELETE FROM members_backup WHERE date_of_membership < '2020-01-01';

The screenshot shows the phpMyAdmin interface for the 'library_db' database. The 'members_backup' table is selected. The table has columns: member_id, member_name, date_of_membership, and email. The data is as follows:

member_id	member_name	date_of_membership	email
1	Alice	2025-01-10	alice@example.com
2	Bob	2025-02-15	bob@example.com
3	Charlie	2025-03-20	charlie@example.com
4	David	2025-04-25	david@example.com
5	Eva	2025-05-30	eva@example.com

- In members table, all members are joined on 2025.
- All dates are after 2020, so no record will be deleted.

Lab 4: Delete all books that have a NULL value in the author column.

Answer:

Query: Delete all books with NULL author

DELETE FROM books WHERE author IS NULL;

The screenshot shows the phpMyAdmin interface for the 'library_db' database. The 'books' table is selected. The table has columns: book_id, title, author, publisher, year_of_publication, price, and genre. The data is as follows:

book_id	title	author	publisher	year_of_publication	price	genre
2	Lord of Rings	J.R.R. Tolkien	Allen & Unwin	1954	660.00	Fantasy
3	The Alchemist	Paulo Coelho	HarperCollins	1990	330.00	Fiction
4	Pride & Prejudice	Jane Austen	T. Egerton	2020	250.00	Romance
5	Inferno	Dan Brown	Doubleday	2013	495.00	Thriller

- In books table, all books have authors so no rows are deleted.

10. Data Query Language (DQL)

Lab 4: Write a query to retrieve all books with price between \$50 and \$100.

Answer:

Query: retrieve all books with price between \$50 and \$100

SELECT * FROM books WHERE price BETWEEN 50 AND 100;

The screenshot shows the phpMyAdmin interface. On the left, the database structure is visible with the 'books' table selected. The main area contains the following SQL query and its execution results:

```
SELECT * FROM books WHERE price BETWEEN 50 AND 100;
```

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0082 seconds.)

- None are between 50 and 100, so result is empty.

Lab 5: Retrieve the list of books sorted by author in ascending order and limit the results to the top 3 entries.

Answer:

Query: Retrieve books sorted by author (ASC) and show top 3

SELECT * FROM books ORDER BY author ASC LIMIT 3;

The screenshot shows the phpMyAdmin interface. On the left, the database structure is visible with the 'books' table selected. The main area contains the following SQL query and its execution results:

```
SELECT * FROM books ORDER BY author ASC LIMIT 3;
```

Showing rows 0 - 2 (3 total, Query took 0.0006 seconds.) [author: DAN BROWN... - JANE AUSTEN...]

book_id	title	author	publisher	year_of_publication	price	genre
5	Inferno	Dan Brown	Doubleday	2013	495.00	Thriller
2	Lord of Rings	J.R.R. Tolkien	Allen & Unwin	1954	660.00	Fantasy
4	Pride & Prejudice	Jane Austen	T. Egerton	2020	250.00	Romance

11. Data Control Language (DCL)

Lab 3: Grant SELECT permission to a user named librarian on the books table.

Answer:

Query: Grant SELECT permission to user librarian on books table

GRANT SELECT ON library_db.books TO 'librarian'@'localhost';

The screenshot shows the phpMyAdmin interface with the following details:

- Left sidebar:** Shows the database structure with databases like employee_information_table, information_schema, library_db, mysql, performance_schema, phpmyadmin, school_db, test, university_db, and world_data.
- Top menu:** Includes Databases, SQL, Status, User accounts, Export, Import, Settings, Replication, Variables,Charsets, Engines, and Plugins.
- Main area:** A message box says "Your SQL query has been executed successfully." Below it, the query "SHOW GRANTS FOR 'librarian'@'localhost';" is run, and the result shows "GRANT USAGE ON *.* TO 'librarian'@'localhost'" and "GRANT SELECT ON 'library_db'.books TO 'librarian'".
- Bottom buttons:** Print, Copy to clipboard, Create view.

Lab 4: Grant INSERT and UPDATE permissions to the user admin on the members table.

Answer:

Query: Grant INSERT and UPDATE permissions to user admin on members table

GRANT INSERT, UPDATE ON library_db.members_backup TO 'admin'@'localhost';

The screenshot shows the phpMyAdmin interface with the following details:

- Left sidebar:** Shows the database structure with databases like employee_information_table, information_schema, library_db, mysql, performance_schema, phpmyadmin, school_db, test, university_db, and world_data.
- Top menu:** Includes Databases, SQL, Status, User accounts, Export, Import, Settings, Replication, Variables,Charsets, Engines, and Plugins.
- Main area:** A message box says "Your SQL query has been executed successfully." Below it, the query "SHOW GRANTS FOR 'admin'@'localhost';" is run, and the result shows "GRANT USAGE ON *.* TO 'admin'@'localhost'" and "GRANT INSERT, UPDATE ON 'library_db'.members_back...".
- Bottom buttons:** Print, Copy to clipboard, Create view.

12. REVOKE Command

Lab 3: Revoke the INSERT privilege from the user librarian on the books table.

Answer:

Query: 1

```
REVOKE INSERT ON library_db.books FROM 'librarian'@'localhost';
```

Query 2: run

```
SHOW GRANTS FOR 'librarian'@'localhost';
```

The screenshot shows the phpMyAdmin interface with the 'Server 127.0.0.1' connection selected. In the left sidebar, the 'library_db' database is expanded, showing tables like 'authors', 'books', and 'members_backup'. The main area displays the results of a SQL query: 'SHOW GRANTS FOR 'librarian'@'localhost''.

Your SQL query has been executed successfully

```
SHOW GRANTS FOR 'librarian'@'localhost';
GRANT USAGE ON *.* TO 'librarian'@'localhost';
GRANT SELECT ON 'library_db'.'books' TO 'librarian';
```

Grants for librarian@localhost

```
GRANT USAGE ON *.* TO 'librarian'@'localhost';
GRANT SELECT ON 'library_db'.'books' TO 'librarian';
```

Query results operations

Print Copy to clipboard Create view

- You will no longer see INSERT privilege for library_db.book.

Lab 4: Revoke all permissions from user admin on the members table.

Answer:

Query: 1

```
REVOKE ALL PRIVILEGES ON library_db.members_backup FROM 'admin'@'localhost';
```

Query 2: run

```
SHOW GRANTS FOR 'admin'@'localhost';
```

The screenshot shows the phpMyAdmin interface with the 'Server 127.0.0.1' connection selected. In the left sidebar, the 'library_db' database is expanded, showing tables like 'authors', 'books', and 'members_backup'. The main area displays the results of a SQL query: 'SHOW GRANTS FOR 'admin'@'localhost''.

Your SQL query has been executed successfully

```
SHOW GRANTS FOR 'admin'@'localhost';
GRANT USAGE ON *.* TO 'admin'@'localhost';
```

Grants for admin@localhost

```
GRANT USAGE ON *.* TO 'admin'@'localhost';
```

Query results operations

Print Copy to clipboard Create view

- You should see no privileges for the members_backup table.

13. Transaction Control Language (TCL)

Lab 3: Use COMMIT after inserting multiple records into the books table, then make another insertion and perform a ROLLBACK.

Answer:

Query:

Step 1: Start transaction

START TRANSACTION;

Step 2: Insert multiple records

INSERT INTO books (book_id, title, author, publisher, year_of_publication, price)

VALUES

(201, 'Book A', 'Author A', 'Publisher A', 2018, 250),

(202, 'Book B', 'Author B', 'Publisher B', 2020, 300);

Step 3: Commit the transaction

COMMIT;

Step 4: Insert another record

INSERT INTO books (book_id, title, author, publisher, year_of_publication, price)

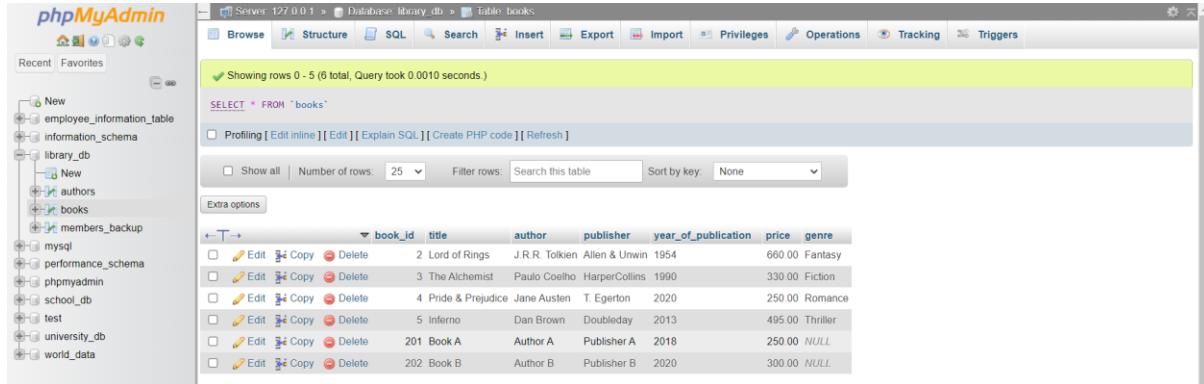
VALUES (203, 'Book C', 'Author C', 'Publisher C', 2021, 350);

The screenshot shows the phpMyAdmin interface for a MySQL database named 'library_db'. The left sidebar lists various databases and tables. The 'books' table is selected, showing 7 rows of data. The columns are book_id, title, author, publisher, year_of_publication, price, and genre. The data includes entries for 'Lord of Rings', 'The Alchemist', 'Pride & Prejudice', 'Inferno', 'Book A', 'Book B', and 'Book C'. The 'genre' column for 'Book A' and 'Book B' is 'NULL', while for 'Book C' it is 'NULL'.

book_id	title	author	publisher	year_of_publication	price	genre
2	Lord of Rings	J.R.R. Tolkien	Allen & Unwin	1954	660.00	Fantasy
3	The Alchemist	Paulo Coelho	HarperCollins	1990	330.00	Fiction
4	Pride & Prejudice	Jane Austen	T. Egerton	2020	250.00	Romance
5	Inferno	Dan Brown	Doubleday	2013	495.00	Thriller
201	Book A	Author A	Publisher A	2018	250.00	NULL
202	Book B	Author B	Publisher B	2020	300.00	NULL
203	Book C	Author C	Publisher C	2021	350.00	NULL

Step 5: Rollback

ROLLBACK;



The screenshot shows the phpMyAdmin interface for a MySQL database named 'library_db'. The left sidebar lists various databases and tables, including 'employee_information_table', 'information_schema', 'library_db' (which contains 'authors', 'books', and 'members_backup'), 'mysql', 'performance_schema', 'phpmyadmin', 'school_db', 'test', 'university_db', and 'world_data'. The main area is titled 'table books' and displays a list of 5 rows from the 'books' table. The columns are: book_id, title, author, publisher, year_of_publication, price, and genre. The data is as follows:

book_id	title	author	publisher	year_of_publication	price	genre
2	Lord of Rings	J.R.R. Tolkien	Allen & Unwin	1954	660.00	Fantasy
3	The Alchemist	Paulo Coelho	HarperCollins	1990	330.00	Fiction
4	Pride & Prejudice	Jane Austen	T. Egerton	2020	250.00	Romance
5	Inferno	Dan Brown	Doubleday	2013	495.00	Thriller
201	Book A	Author A	Publisher A	2018	250.00	NULL
202	Book B	Author B	Publisher B	2020	300.00	NULL

- Records 201 and 202 -> saved permanently.
- Records 203 -> not saved (because of rollback).

Lab 4: Set a SAVEPOINT before making updates to the members table, perform some updates, and then roll back to the SAVEPOINT.

Answer:

Query:

Step 1: Start transaction

```
START TRANSACTION;
```

Step 2: Set a SAVEPOINT

```
SAVEPOINT sp_before_update;
```

Step 3: Perform updates

```
UPDATE members SET email = 'newemail1@example.com' WHERE member_id = 1;
```

```
UPDATE members SET member_name = 'Updated Name' WHERE member_id = 2;
```

Step 4: Roll back to SAVEPOINT

```
ROLLBACK TO sp_before_update;
```

Step 5: Commit final state

```
COMMIT;
```

The screenshot shows the phpMyAdmin interface. On the left, the database structure is visible with the library_db database selected. Inside library_db, there are several tables: New, employee_information_table, information_schema, authors, books, members, and members_backup. The members table is currently selected. At the top, a navigation bar includes links for Browse, Structure, SQL, Search, Insert, Export, Import, Privileges, Operations, Tracking, and Triggers. Below the navigation bar, a message indicates "Showing rows 0 - 4 (total, Query took 0.0006 seconds)." A SQL query is displayed: "SELECT * FROM `members`". Below the query, there are buttons for Profiling, Edit inline, Edit, Explain SQL, Create PHP code, and Refresh. A search bar and filter options are also present. The main area shows the data in the members table:

	member_id	member_name	date_of_membership	email
<input type="checkbox"/>	1	Alice	2025-01-10	alice@example.com
<input type="checkbox"/>	2	Bob	2025-02-15	bob@example.com
<input type="checkbox"/>	3	Charlie	2025-03-20	charlie@example.com
<input type="checkbox"/>	4	David	2025-04-25	david@example.com
<input type="checkbox"/>	5	Eva	2025-05-30	eva@example.com

At the bottom of the table view, there are buttons for Check all, With selected, Edit, Copy, Delete, and Export.

- Updates on member 1 & 2 -> Undone
- Database returns to the states at SAVEPOINT sp_before_update

14. SQL Joins

Lab 3: Perform an INNER JOIN between books and authors tables to display the title of books and their respective authors' names.

Answer:

Query:

```
SELECT b.title, a.first_name, a.last_name FROM books b INNER JOIN authors a ON b.author_id = a.author_id;
```

The screenshot shows the phpMyAdmin interface. On the left, there's a sidebar with a tree view of databases and tables. The main area shows a query result for the 'books' table joined with the 'authors' table. The results are as follows:

title	first_name	last_name
Lord of Rings	Mark	Johnson
The Alchemist	Agatha	Christie
Pride & Prejudice	George	Orwell
Inferno	Mark	Johnson

Lab 4: Use a FULL OUTER JOIN to retrieve all records from the books and authors tables, including those with no matching entries in the other table.

Answer:

Query:

```
SELECT b.title, a.first_name, a.last_name FROM books b LEFT JOIN authors a ON b.author_id = a.author_id
```

UNION

```
SELECT b.title, a.first_name, a.last_name FROM books b RIGHT JOIN authors a ON b.author_id = a.author_id;
```

phpMyAdmin

Server: 127.0.0.1 » Database: library.db

Structure SQL Search Query Export Import Operations Privileges Routines Events Triggers Tracking Designer

Show query box

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 5 (6 total, Query took 0.0099 seconds)

```
SELECT b.title, a.first_name, a.last_name FROM books b LEFT JOIN authors a ON b.author_id = a.author_id UNION SELECT b.title, a.first_name, a.last_name FROM books b RIGHT JOIN authors a ON b.author_id = a.author_id;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all Number of rows: 25 Filter rows: Search this table

Extra options

title	first_name	last_name
Lord of Rings	Mark	Johnson
The Alchemist	Agatha	Christie
Pride & Prejudice	George	Orwell
Inferno	Mark	Johnson
Book A	NULL	NULL
Book B	NULL	NULL

Show all Number of rows: 25 Filter rows: Search this table

15. SQL Group By

Lab 3: Group books by genre and display the total number of books in each genre.

Answer:

Query:

SELECT

genre,

COUNT(*) AS total_books

FROM books

GROUP BY genre;

The screenshot shows the phpMyAdmin interface. On the left is a tree view of databases and tables. The main area contains a query editor with the following content:

```
Showing rows 0 - 4 (total, Query took 0.0085 seconds)

SELECT genre, COUNT(*) AS total_books FROM books GROUP BY genre;
```

Below the query editor is a results table:

genre	total_books
NULL	2
Fantasy	1
Fiction	1
Romance	1
Thriller	1

Lab 4: Group members by the year they joined and find the number of members who joined each year.

Answer:

Query:

SELECT

YEAR(date_of_membership) AS join_year,

COUNT(*) AS total_members

FROM members

GROUP BY YEAR(date_of_membership);

The screenshot shows the phpMyAdmin interface. The left sidebar lists databases: New, employee_information_table, information_schema, library_db (with sub-tables New, authors, books, members, members_backup), mysql, performance_schema, phpmyadmin, school_db, test, university_db, and world_data. The main area shows a successful query execution:

```
SELECT YEAR(date_of_membership) AS join_year, COUNT(*) AS total_members FROM members GROUP BY YEAR(date_of_membership);
```

The results table shows one row:

join_year	total_members
2025	5

Below the results are operations: Print, Copy to clipboard, Export, Display chart, and Create view.

16. SQL Stored Procedure

Lab 3: Write a stored procedure to retrieve all books by a particular author.

Answer:

Query 1: Create the stored procedure

```
DELIMITER $$
```

```
CREATE PROCEDURE GetBooksByAuthor(IN authorName VARCHAR(100))
```

```
BEGIN
```

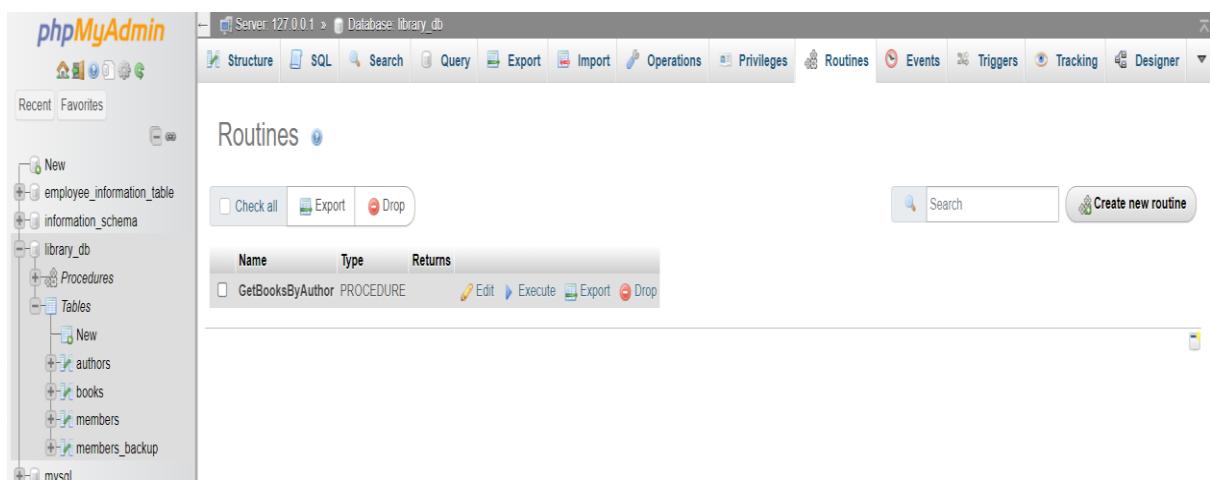
```
    SELECT book_id, title, author, publisher, year_of_publication, price FROM books  
    WHERE author = authorName;
```

```
END $$
```

```
DELIMITER ;
```

Query 2: call (run) the stored procedure

```
CALL GetBooksByAuthor('J.K. Rowling');
```



Lab 4: Write a stored procedure that takes book_id as an argument and returns the price of the book.

Answer:

Query 1: stored procedure – getbookprice

DELIMITER \$\$

```
CREATE PROCEDURE GetBookPrice(IN p_book_id INT)
```

```
BEGIN
```

```
    SELECT price
```

```
    FROM books
```

```
    WHERE book_id = p_book_id;
```

```
END $$
```

```
DELIMITER ;
```

Query 2: call (run) the stored procedure

```
CALL GetBookPrice(1);
```

The screenshot shows the phpMyAdmin interface for a MySQL database named 'library_db'. The left sidebar shows tables like 'employee_information_table', 'information_schema', 'library_db' (which contains 'Procedures', 'Tables', 'New', 'authors', 'books', 'members', and 'members_backup'), and other databases like 'employee_information_table' and 'information_schema'. The main panel is titled 'Routines' and lists two entries:

Name	Type	Returns
GetBookPrice	PROCEDURE	Edit Execute Export Drop
GetBooksByAuthor	PROCEDURE	Edit Execute Export Drop

17. SQL View

Lab 3: Create a view to show only the title, author, and price of books from the books table.

Answer:

Query 1: Create the View

```
CREATE VIEW book_summary AS SELECT title, author, price FROM books;
```

Query 2: To see the view

```
SELECT * FROM book_summary;
```

The screenshot shows the phpMyAdmin interface for a MySQL database named 'library_db'. The left sidebar lists databases like 'employee_information_table', 'information_schema', 'library_db', 'members', and 'mysql'. Under 'library_db', there are 'Tables' (including 'authors', 'books', 'members') and 'Views' (including 'book_summary'). The main panel displays the contents of the 'book_summary' view, which contains the following data:

	title	author	price
<input type="checkbox"/>	Lord of Rings	J.R.R. Tolkien	660.00
<input type="checkbox"/>	The Alchemist	Paulo Coelho	330.00
<input type="checkbox"/>	Pride & Prejudice	Jane Austen	250.00
<input type="checkbox"/>	Inferno	Dan Brown	495.00
<input type="checkbox"/>	Book A	Author A	250.00
<input type="checkbox"/>	Book B	Author B	300.00

Below the table, there are buttons for 'Check all', 'With selected:', 'Edit', 'Copy', 'Delete', and 'Export'.

Lab 4: Create a view to display members who joined before 2020.

Answer:

Query 1: Create the View

```
CREATE VIEW members_before_2020 AS
```

```
SELECT member_id, member_name, date_of_membership, email FROM members  
WHERE YEAR(date_of_membership) < 2020;
```

Query 2: To see the view

```
SELECT * FROM members_before_2020;
```

The screenshot shows the phpMyAdmin interface for a database named 'library_db'. On the left, the database structure is visible with tables like 'employee_information_table', 'information_schema', and 'library_db'. Under 'library_db', there are 'Tables' (including 'authors', 'books', 'members', and 'members_backup') and 'Views' (including 'book_summary' and 'members_before_2020'). The main panel shows a query editor with the following content:

```
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0010 seconds.)  
SELECT * FROM `members_before_2020`  
member_id member_name date_of_membership email
```

Below the query editor, there are several buttons and options:

- Query results operations: 'Create view'
- Bookmark this SQL query: 'Label:' input field, 'Let every user access this bookmark' checkbox
- Bookmark this SQL query button

- No member joined before 2020 so rows are empty. All members joined on 2025.

18. SQL Trigger

Lab 3: Create a trigger to automatically update the last_modified timestamp of the books table whenever a record is updated.

Answer:

Query:

Step 1: Make sure books table has last_modified column

ALTER TABLE books

ADD COLUMN last_modified TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP;

- This ensures last_modified is updated automatically on UPDATE, but we can also do it via trigger if needed.

	book_id	title	author	publisher	year_of_publication	price	genre	author_id	last_modified
1	2	Lord of Rings	J.R.R. Tolkien	Allen & Unwin	1954	660.00	Fantasy	2	2025-11-20 17:19:55
2	3	The Alchemist	Paulo Coelho	HarperCollins	1990	330.00	Fiction	3	2025-11-20 17:19:55
3	4	Pride & Prejudice	Jane Austen	T. Egerton	2020	250.00	Romance	1	2025-11-20 17:19:55
4	5	Inferno	Dan Brown	Doubleday	2013	495.00	Thriller	2	2025-11-20 17:19:55
5	201	Book A	Author A	Publisher A	2018	250.00	NULL	NULL	2025-11-20 17:19:55
6	202	Book B	Author B	Publisher B	2020	300.00	NULL	NULL	2025-11-20 17:19:55

Step 2: Trigger to update last_modified on UPDATE

DELIMITER \$\$

CREATE TRIGGER update_books_timestamp

BEFORE UPDATE ON books

FOR EACH ROW

BEGIN

SET NEW.last_modified = NOW();

END\$\$

DELIMITER ;

The screenshot shows the phpMyAdmin interface. On the left, the database structure is visible, including the library_db database which contains tables like authors, books, and members. The main window is titled 'Triggers' and shows one trigger defined for the 'books' table:

Name	Time	Event
update_books_timestamp	BEFORE UPDATE	

Lab 4: Create a trigger that inserts a log entry into a log_changes table whenever a DELETE operation is performed on the books table.

Answer:

Query:

Step 1: Create log_changes table

```
CREATE TABLE log_changes (
    log_id INT AUTO_INCREMENT PRIMARY KEY,
    book_id INT,
    title VARCHAR(100),
    operation VARCHAR(10),
    change_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

The screenshot shows the phpMyAdmin interface. On the left, the database structure is visible, including the library_db database which contains the log_changes table. The main window is titled 'Table structure' for the log_changes table:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	log_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	book_id	int(11)			Yes	NULL		Change Drop More	
3	title	varchar(100)	utf8mb4_general_ci		Yes	NULL		Change Drop More	
4	operation	varchar(10)	utf8mb4_general_ci		Yes	NULL		Change Drop More	
5	change_time	timestamp			No	current_timestamp()		Change Drop More	

Step 2: Trigger to insert a log on DELETE

```
DELIMITER $$
```

```
CREATE TRIGGER log_books_deletion
```

```
AFTER DELETE ON books
```

```
FOR EACH ROW
```

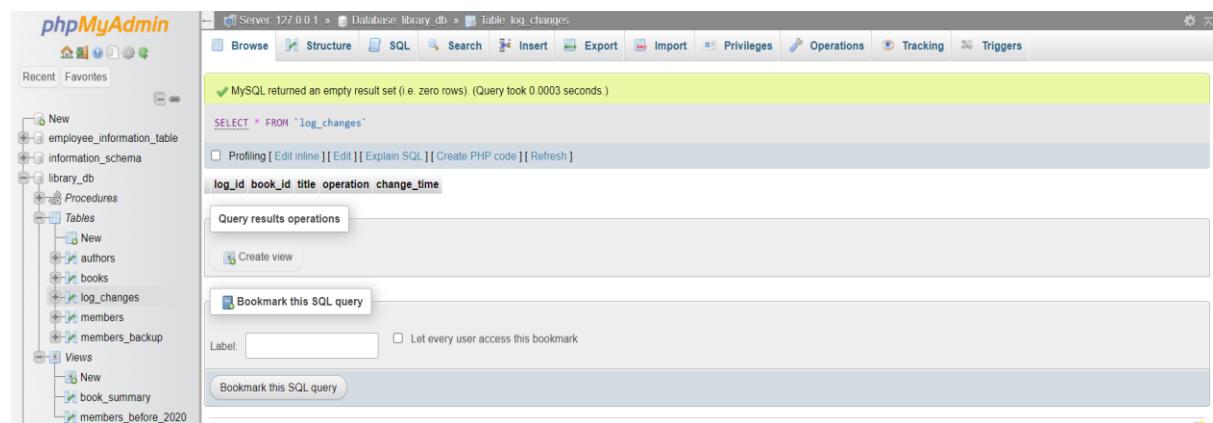
```
BEGIN
```

```
    INSERT INTO log_changes(book_id, title, operation)
```

```
        VALUES (OLD.book_id, OLD.title, 'DELETE');
```

```
END$$
```

```
DELIMITER ;
```



19. Introduction to PL/SQL

Lab 3: Write a PL/SQL block to insert a new book into the books table and display a confirmation message.

Answer:

Query: Insert a new book and display confirmation

```
DELIMITER $$
```

```
CREATE PROCEDURE insert_book()
```

```
BEGIN
```

```
    DECLARE v_book_id INT DEFAULT 6;
```

```
    DECLARE v_title VARCHAR(100) DEFAULT 'The Great Gatsby';
```

```
    DECLARE v_author VARCHAR(50) DEFAULT 'F. Scott Fitzgerald';
```

```
    DECLARE v_publisher VARCHAR(50) DEFAULT 'Scribner';
```

```
    DECLARE v_year INT DEFAULT 1925;
```

```
    DECLARE v_price DECIMAL(10,2) DEFAULT 450;
```

```
-- Insert the new book
```

```
    INSERT INTO books(book_id, title, author, publisher, year_of_publication, price)
```

```
        VALUES(v_book_id, v_title, v_author, v_publisher, v_year, v_price);
```

```
-- Display confirmation
```

```
    SELECT CONCAT('Book "', v_title, '" inserted successfully.') AS message;
```

```
END$$
```

```
DELIMITER ;
```

```
-- Call the procedure
```

```
CALL insert_book();
```

The screenshot shows the phpMyAdmin interface. On the left, the database structure for 'library_db' is visible, including tables like 'books', 'authors', and 'members'. The main area displays a query result for a stored procedure named 'insert_book'. The procedure creates a new book entry with title 'The Great Gatsby', author 'F. Scott Fitzgerald', publisher 'Scribner', year 1925, and price 450. A confirmation message is shown: 'Book "The Great Gatsby" inserted successfully.'.

Lab 4: Write a PL/SQL block to display the total number of books in the books table.

Answer:

Query: Display total number of books

DELIMITER \$\$

```
CREATE PROCEDURE total_books()
```

BEGIN

```
    DECLARE v_total_books INT;
```

```
    -- Count total books
```

```
    SELECT COUNT(*) INTO v_total_books FROM books;
```

```
    -- Display total
```

```
    SELECT CONCAT('Total number of books: ', v_total_books) AS message;
```

END\$\$

DELIMITER ;

```
-- Call the procedure
```

```
CALL total_books();
```

phpMyAdmin

Recent Favorites

Server: 127.0.0.1 » Database: library_db » table books

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Show query box

MySQL returned an empty result set (i.e. zero rows) (Query took 0.0042 seconds.)

```
CREATE PROCEDURE total_books() BEGIN DECLARE v_total_books INT; -- Count total books SELECT COUNT(*) INTO v_total_books FROM books; -- Display total SELECT CONCAT('Total number of books: ', v_total_books) AS message; END;
```

[Edit inline] [Edit] [Create PHP code]

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 0 (1 total, Query took 0.0018 seconds)

```
-- Call the procedure CALL total_books();
```

[Edit inline] [Edit] [Create PHP code]

Show all Number of rows: 25 Filter rows: Search this table

Extra options

message

Total number of books: 7

Show all Number of rows: 25 Filter rows: Search this table

The screenshot shows the phpMyAdmin interface for a MySQL server. On the left, the database structure is displayed under the 'library_db' database, including tables like 'books', 'authors', and 'log_changes'. A stored procedure named 'total_books' is shown in the 'Procedures' section. In the main pane, a query has been run to call this procedure, and the result is displayed: 'Total number of books: 7'. The interface includes various navigation and search tools at the top and bottom.

20. PL/SQL Syntax

Lab 3: Write a PL/SQL block to declare variables for book_id and price, assign values, and display the results.

Answer:

Query: Declare variables for book_id and price, assign values, display results

```
DELIMITER $$
```

```
CREATE PROCEDURE lab3_variables()
```

```
BEGIN
```

```
-- Declare variables
```

```
DECLARE v_book_id INT;
```

```
DECLARE v_price DECIMAL(10,2);
```

```
-- Assign values
```

```
SET v_book_id = 101;
```

```
SET v_price = 499.99;
```

```
-- Display results
```

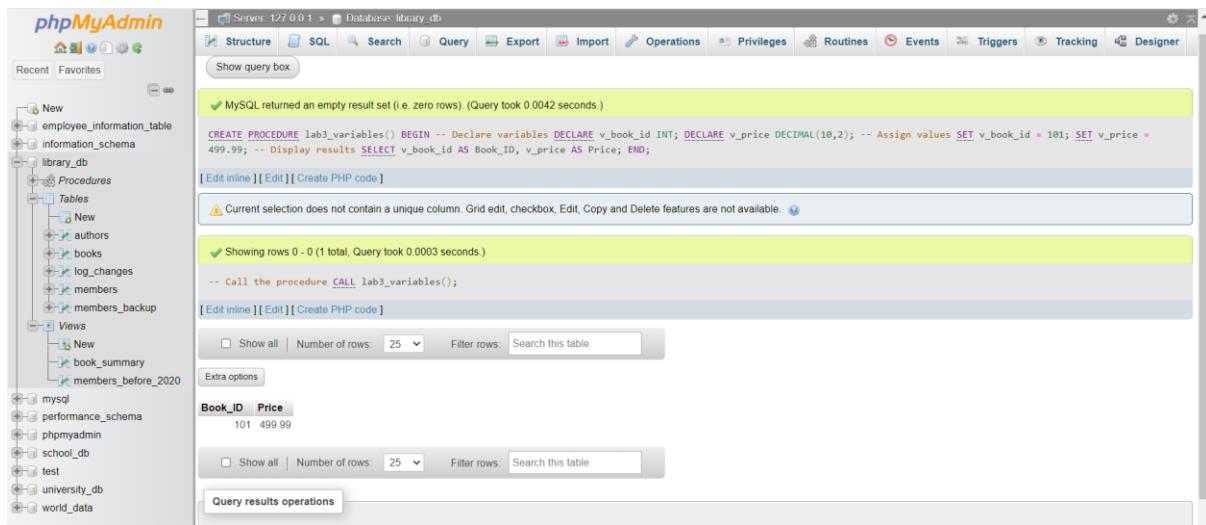
```
SELECT v_book_id AS Book_ID, v_price AS Price;
```

```
END$$
```

```
DELIMITER ;
```

```
-- Call the procedure
```

```
CALL lab3_variables();
```



Lab 4: Write a PL/SQL block using constants and perform arithmetic operations on book prices.

Answer:

Query: Using constants and arithmetic operations on book prices

DELIMITER \$\$

```
CREATE PROCEDURE lab4_constants()
```

BEGIN

-- Declare constants using user-defined variables (MySQL doesn't have true constants in procedures)

```
DECLARE price1 DECIMAL(10,2) DEFAULT 300;
```

```
DECLARE price2 DECIMAL(10,2) DEFAULT 450;
```

-- Perform arithmetic operations

```
DECLARE total_price DECIMAL(10,2);
```

```
DECLARE price_difference DECIMAL(10,2);
```

```
SET total_price = price1 + price2;
```

```
SET price_difference = price2 - price1;
```

-- Display results

```
SELECT price1 AS Price1, price2 AS Price2, total_price AS Total, price_difference AS Difference;
```

```
END$$
```

```
DELIMITER ;
```

-- Call the procedure

```
CALL lab4_constants();
```

The screenshot shows the phpMyAdmin interface with the following details:

- Left Panel (Database Structure):** Shows the database structure with the following schema:
 - library_db:** Contains **Tables** (authors, books, log_changes, members), **Procedures** (lab4_constants), and **Views**.
 - Other Databases:** mysql, performance_schema, phpmysqladmin, school_db, test, university_db, world_data.
- Top Bar:** Shows the server (127.0.0.1), database (library_db), and various navigation tabs like Structure, SQL, Search, Query, Export, Import, Operations, Privileges, Routines, Events, Triggers, Tracking, and Designer.
- Query Editor:** Displays the executed SQL code:

```
CREATE PROCEDURE lab4_constants() BEGIN -- Declare constants using user-defined variables (MySQL doesn't have true constants in procedures) DECLARE price1 DECIMAL(10,2) DEFAULT 300; DECLARE price2 DECIMAL(10,2) DEFAULT 450; -- Perform arithmetic operations DECLARE total_price DECIMAL(10,2); DECLARE price_difference DECIMAL(10,2); SET total_price = price1 + price2; SET price_difference = price2 - price1; -- Display results SELECT price1 AS Price1, price2 AS Price2, total_price AS Total, price_difference AS Difference; END;
```

Below the code, there are buttons for [Edit inline], [Edit], and [Create PHP code]. A warning message states: "Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available." A note also says: "Showing rows 0 - 0 (1 total, Query took 0.0010 seconds.)".

- Result Table:** A table showing the results of the procedure execution:

Price1	Price2	Total	Difference
300.00	450.00	750.00	150.00

21. PL/SQL Control Structures

Lab 3: Write a PL/SQL block using IF-THEN-ELSE to check if a book's price is above \$100 and print a message accordingly.

Answer:

Query: IF-THEN-ELSE to check book price

DELIMITER \$\$

```
CREATE PROCEDURE lab3_if_else(IN p_book_id INT)
BEGIN
    DECLARE v_price DECIMAL(10,2);

    -- Get the price of the book
    SELECT price INTO v_price FROM books WHERE book_id = p_book_id;

    -- IF-THEN-ELSE logic
    IF v_price > 100 THEN
        SELECT CONCAT('Book ID ', p_book_id, ' has a price above $100: $', v_price) AS message;
    ELSE
        SELECT CONCAT('Book ID ', p_book_id, ' has a price below or equal $100: $', v_price) AS message;
    END IF;
END$$

DELIMITER ;
```

-- Example call

```
CALL lab3_if_else(1);
```

The screenshot shows the phpMyAdmin interface for the library_db database. The left sidebar shows various databases and their structures. The main area has tabs for Structure, SQL, Search, Query, Export, Import, Operations, Privileges, Routines, Events, Triggers, Tracking, and Designer. The SQL tab contains the following code:

```
SET @p0='1'; CALL `lab3_if_else`(@p0);
```

The results of the execution are shown in a box:

Execution results of routine `lab3_if_else`

message
NULL

The Routines tab lists the following procedures:

Name	Type	Returns	Edit	Execute	Export	Drop
GetBookPrice	PROCEDURE		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
GetBooksByAuthor	PROCEDURE		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
insert_book	PROCEDURE		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
lab3_if_else	PROCEDURE		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
lab3_variables	PROCEDURE		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
lab4_constants	PROCEDURE		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
total_books	PROCEDURE		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Lab 4: Use a FOR LOOP in PL/SQL to display the details of all books one by one.

Answer:

Query: FOR LOOP to display all book details

DELIMITER \$\$

```
CREATE PROCEDURE lab4_all_books()
```

BEGIN

-- Simply select all book details

```
SELECT book_id AS Book_ID,
```

title AS Title,

author AS Author,

publisher AS Publisher,

year_of_publication AS Year,

price AS Price

```
FROM books;
```

END\$\$

DELIMITER ;

-- Call the procedure

```
CALL lab4_all_books();
```

The screenshot shows the phpMyAdmin interface for a MySQL database named 'library'. The left sidebar shows the database structure with tables like 'books', 'authors', and 'members'. The main area displays the results of a query:

```
CREATE PROCEDURE lab4_all_books() BEGIN -- simply select all book details SELECT book_id AS book_id, title AS title, author AS author, publisher AS publisher, year_of_publication AS Year, price AS Price FROM books; END;
```

Below the query, a message indicates: "Showing rows 0 - 6 (7 total, Query took 0.0004 seconds)". The results are displayed in a table:

Book_ID	Title	Author	Publisher	Year	Price
2	Lord of Rings	J R R. Tolkien	Allen & Unwin	1954	660.00
3	The Alchemist	Paulo Coelho	HarperCollins	1990	330.00
4	Pride & Prejudice	Jane Austen	T Egerton	2020	250.00
5	Inferno	Dan Brown	Doubleday	2013	495.00
6	The Great Gatsby	F Scott Fitzgerald	Scribner	1925	450.00
201	Book A	Author A	Publisher A	2018	250.00
202	Book B	Author B	Publisher B	2020	300.00

22. SQL Cursors

Lab 3: Write a PL/SQL block using an explicit cursor to fetch and display all records from the members table.

Answer:

Query: Fetch and display all members using a cursor

Step 1: Stored Procedure

DELIMITER \$\$

```
CREATE PROCEDURE lab3_members_cursor()
```

```
BEGIN
```

```
-- Variables to hold member details
```

```
DECLARE v_member_id INT;
```

```
DECLARE v_member_name VARCHAR(50);
```

```
DECLARE v_date_of_membership DATE;
```

```
DECLARE v_email VARCHAR(100);
```

```
DECLARE done INT DEFAULT 0;
```

```
-- Cursor for all members
```

```
DECLARE cur_members CURSOR FOR
```

```
SELECT member_id, member_name, date_of_membership, email FROM members;
```

```
-- Handler to exit loop
```

```
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
```

```
OPEN cur_members;
```

```
read_loop: LOOP
```

```
    FETCH cur_members INTO v_member_id, v_member_name, v_date_of_membership,  
    v_email;
```

```

IF done THEN
    LEAVE read_loop;
END IF;

-- Display member details
SELECT v_member_id AS Member_ID,
       v_member_name AS Name,
       v_date_of_membership AS Membership_Date,
       v_email AS Email;
END LOOP;

CLOSE cur_members;
END$$

DELIMITER ;

```

Step 2: call the procedure

```
CALL lab3_members_cursor();
```

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** 127.0.0.1 -> Database: library_db
- Structure, SQL, Search, Query, Export, Import, Operations, Privileges, Routines, Events, Triggers, Tracking, Designer** tabs are visible.
- Execution results of routine 'lab3_members_cursor':**

Member_ID	Name	Membership_Date	Email
1	Alice	2025-01-10	alice@example.com
2	Bob	2025-02-15	bob@example.com
3	Charlie	2025-03-20	charlie@example.com
4	David	2025-04-25	david@example.com
5	Eva	2025-05-30	eva@example.com
6	Robert Brown	2025-11-20	robert@example.com

- Routines** tab is also visible.

Lab 4: Create a cursor to retrieve books by a particular author and display their titles.

Answer:

Query: Retrieve books by a particular author using a cursor

Step 1: Stored Procedure

DELIMITER \$\$

```
CREATE PROCEDURE get_books_by_author(IN auth_name VARCHAR(100))
```

```
BEGIN
```

```
    DECLARE done INT DEFAULT 0;
```

```
    DECLARE b_title VARCHAR(100);
```

```
-- Cursor to select books by author
```

```
    DECLARE book_cursor CURSOR FOR
```

```
        SELECT title FROM books WHERE author = auth_name;
```

```
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
```

```
    OPEN book_cursor;
```

```
read_loop: LOOP
```

```
    FETCH book_cursor INTO b_title;
```

```
    IF done THEN
```

```
        LEAVE read_loop;
```

```
    END IF;
```

```
    SELECT b_title AS 'Book Title';
```

```
END LOOP;
```

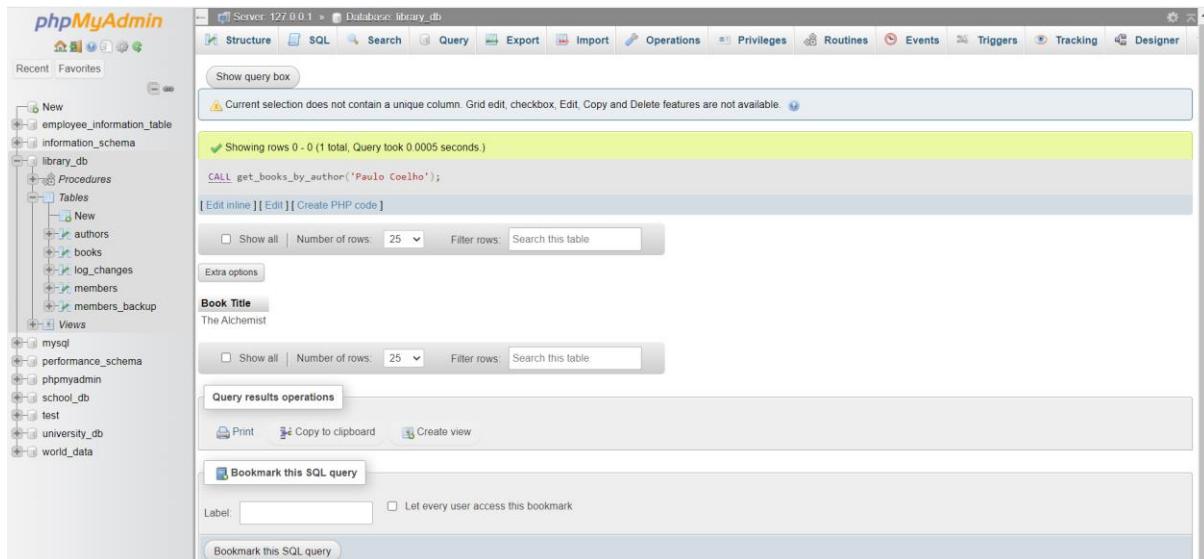
```
CLOSE book_cursor;
```

END\$\$

DELIMITER ;

Step 2: Call the procedure

CALL get_books_by_author('Paulo Coelho');



23. Rollback and Commit Savepoint

Lab 3: Perform a transaction that includes inserting a new member, setting a SAVEPOINT, and rolling back to the savepoint after making updates.

Answer:

Query: Transaction with SAVEPOINT and ROLLBACK

Suppose we have a members table with columns:

member_id, member_name, date_of_membership, email

-- Start the transaction

START TRANSACTION;

-- 1 Insert a new member

```
INSERT INTO members(member_id, member_name, date_of_membership, email)
VALUES (6, 'Robert Brown', '2025-11-20', 'robert@example.com');
```

-- 2 Set a SAVEPOINT

SAVEPOINT before_update;

-- 3 Update some records (example)

UPDATE members

SET member_name = 'Robert B.'

WHERE member_id = 6;

-- 4 Rollback to the savepoint

ROLLBACK TO SAVEPOINT before_update;

-- 5 Check the table state (SELECT)

SELECT * FROM members;

```
-- Optionally commit if everything is fine
```

```
COMMIT;
```

A screenshot of the phpMyAdmin interface. On the left, the database structure is shown with 'library_db' selected. Under 'Tables', there is a 'members' table. The main area shows the 'members' table with the following data:

member_id	member_name	date_of_membership	email
1	Alice	2025-01-10	alice@example.com
2	Bob	2025-02-15	bob@example.com
3	Charlie	2025-03-20	charlie@example.com
4	David	2025-04-25	david@example.com
5	Eva	2025-05-30	eva@example.com
6	Robert Brown	2025-11-20	robert@example.com

Lab 4: Use COMMIT after successfully inserting multiple books into the books table, then use ROLLBACK to undo a set of changes made after a savepoint.

Answer:

Query: COMMIT and ROLLBACK with multiple books

Suppose we have a books table:

```
book_id, title, author, publisher, year_of_publication, price
```

```
-- Start the transaction
```

```
START TRANSACTION;
```

```
-- 1 Insert multiple books
```

```
INSERT INTO books(book_id, title, author, publisher, year_of_publication, price)
```

```
VALUES
```

```
(7, 'Pride and Prejudice', 'Jane Austen', 'T. Egerton', 1813, 350),  
(8, '1984', 'George Orwell', 'Secker & Warburg', 1949, 400);
```

```
-- 2 Commit the transaction (permanent)
```

```
COMMIT;
```

-- 3 Start a new transaction

```
START TRANSACTION;
```

-- Make some changes

```
UPDATE books SET price = price + 50 WHERE book_id = 7;
```

```
UPDATE books SET price = price + 100 WHERE book_id = 8;
```

-- 4 Set a savepoint

```
SAVEPOINT before_rollback;
```

-- Make additional changes

```
UPDATE books SET price = price * 2 WHERE book_id = 7;
```

-- 5 Rollback to savepoint (undo last update only)

```
ROLLBACK TO SAVEPOINT before_rollback;
```

-- 6 Check table state

```
SELECT * FROM books;
```

-- Commit final changes

```
COMMIT;
```

book_id	title	author	publisher	year_of_publication	price	genre	author_id	last_modified
2	Lord of Rings	J.R.R. Tolkien	Allen & Unwin	1954	660.00	Fantasy	2	2025-11-20 17:19:55
3	The Alchemist	Paulo Coelho	HarperCollins	1990	330.00	Fiction	3	2025-11-20 17:19:55
4	Pride & Prejudice	Jane Austen	T. Egerton	2020	250.00	Romance	1	2025-11-20 17:19:55
5	Inferno	Dan Brown	Doubleday	2013	495.00	Thriller	2	2025-11-20 17:19:55
6	The Great Gatsby	F. Scott Fitzgerald	Scribner	1925	450.00	NULL	NULL	2025-11-20 17:43:44
7	Pride and Prejudice	Jane Austen	T. Egerton	0000	400.00	NULL	NULL	2025-11-20 18:37:45
8	1984	George Orwell	Secker & Warburg	1949	500.00	NULL	NULL	2025-11-20 18:37:45
201	Book A	Author A	Publisher A	2018	250.00	NULL	NULL	2025-11-20 17:19:55
202	Book B	Author B	Publisher B	2020	300.00	NULL	NULL	2025-11-20 17:19:55