1. Introduction to SQL

LAB EXERCISES: Create a database called library_db and a table books with columns: b

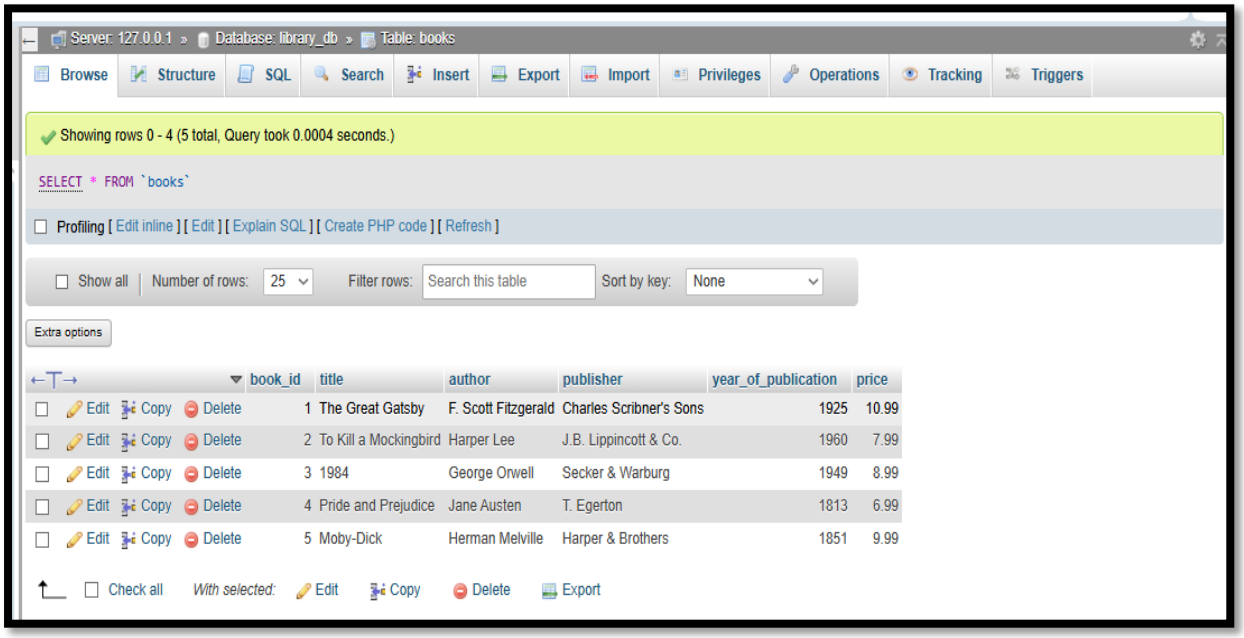
 Lab 3: book_id, title, author, publisher, year_of_publication, and price. Insert five records into the table.

- CREATE DATABASE library_db;

- Create the books table
  - CREATE TABLE books (
  - book_id INT AUTO_INCREMENT PRIMARY KEY,
  - title VARCHAR (255),
  - author VARCHAR(255),
  - publisher VARCHAR (255),
  - year_of_publication INT,
  - price DECIMAL (10, 2)

  );

- INSERT INTO books (title, author, publisher, year_of_publication, price)
  VALUES
  ('The Great Gatsby', 'F. Scott Fitzgerald', 'Charles Scribner\'s Sons', 1925, 10.99),
  ('To Kill a Mockingbird', 'Harper Lee', 'J.B. Lippincott & Co.', 1960, 7.99),
  ('1984', 'George Orwell', 'Secker & Warburg', 1949, 8.99),
  ('Pride and Prejudice', 'Jane Austen', 'T. Egerton', 1813, 6.99),
  ('Moby-Dick', 'Herman Melville', 'Harper & Brothers', 1851, 9.99);

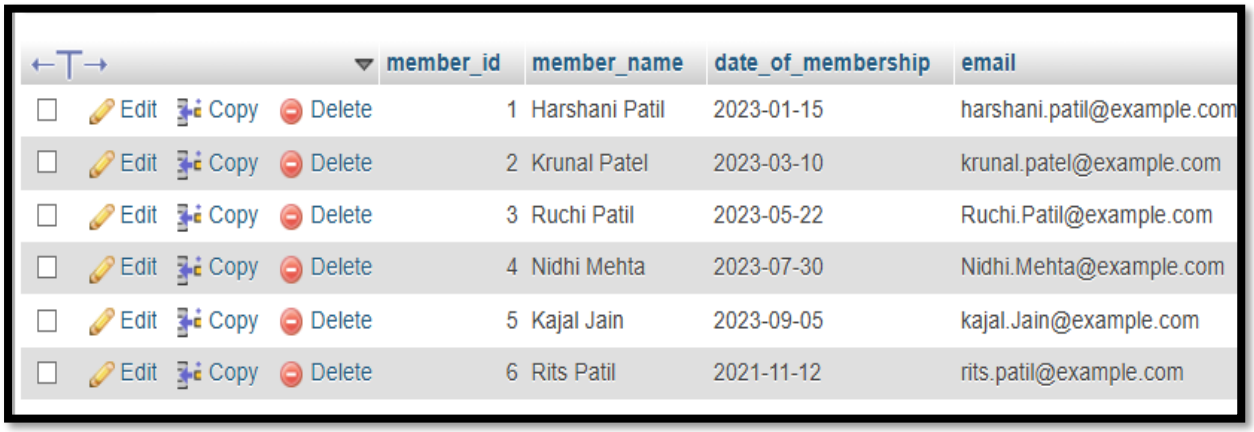


 Lab 4: Create a table member in library_db with columns: member, member name,

date_of_membership, and email. Insert five records into this table.

- CREATE TABLE members (
  member_id INT AUTO_INCREMENT PRIMARY KEY,
  member_name VARCHAR(100),
  date_of_membership DATE,
  email VARCHAR(100)
);

- INSERT INTO members (member_name, date_of_membership, email)
  VALUES
  ('Harshani Patil', '2023-01-15', 'harshani.patil@example.com'),
  ('Krunal Patel', '2023-03-10', 'krunal.patel@example.com'),
  ('Ruchi Patil', '2023-05-22', 'Ruchi.Patil@example.com'),
  ('Nidhi Mehta', '2023-07-30', 'Nidhi.Mehta@example.com'),
  ('Kajal Jain', '2023-09-05', 'kajal.Jain@example.com');
  ('Rits Patil', '2021-11-12', 'rits.patil@example.com');



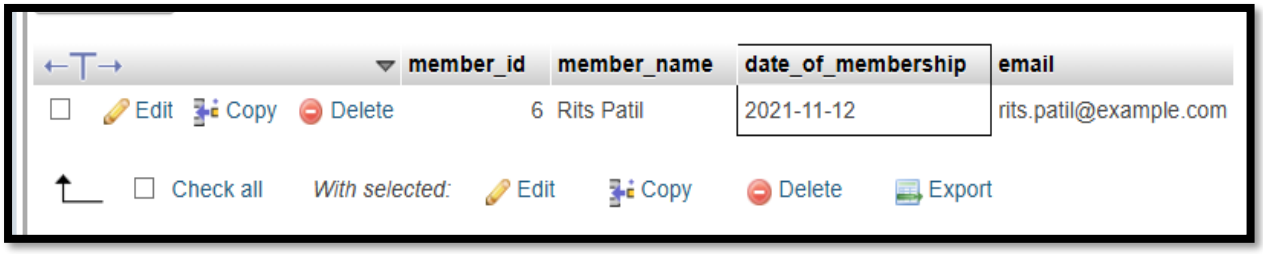
	member_id	member_name	date_of_membership	email
Edit Copy Delete	1	Harshani Patil	2023-01-15	harshani.patil@example.com
Edit Copy Delete	2	Krunal Patel	2023-03-10	krunal.patel@example.com
Edit Copy Delete	3	Ruchi Patil	2023-05-22	Ruchi.Patil@example.com
Edit Copy Delete	4	Nidhi Mehta	2023-07-30	Nidhi.Mehta@example.com
Edit Copy Delete	5	Kajal Jain	2023-09-05	kajal.Jain@example.com
Edit Copy Delete	6	Rits Patil	2021-11-12	rits.patil@example.com

2. SQL Syntax

LAB EXERCISES:

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

- SELECT * FROM members WHERE date_of_membership < '2022-01-01' ORDER BY date_of_membership ASC;

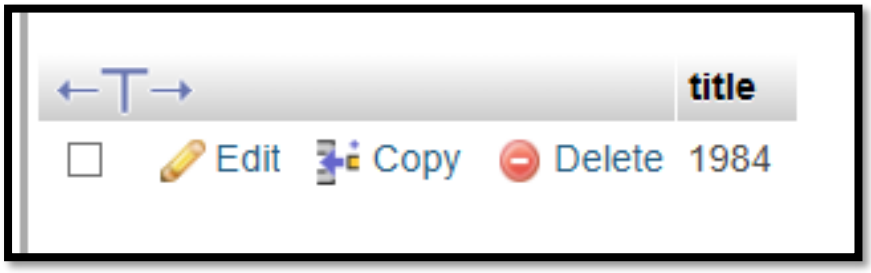


	member_id	member_name	date_of_membership	email
Edit Copy Delete	6	Rits Patil	2021-11-12	rits.patil@example.com

Check all With selected: Edit Copy Delete Export

 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.

- SELECT title FROM books WHERE author = 'George Orwell’ ORDER BY year_of_publication DESC;



3. SQL Constraints

LAB EXERCISES:

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

- ALTER TABLE books ADD CONSTRAINT chk_price_positive CHECK (price > 0);

Show Constraints in MySQL

- SELECT CONSTRAINT_NAME, TABLE_NAME, CONSTRAINT_TYPE FROM information_schema.TABLE_CONSTRAINTS WHERE TABLE_SCHEMA = 'library_db' AND TABLE_NAME = 'books';

CONSTRAINT_NAME	TABLE_NAME	CONSTRAINT_TYPE
PRIMARY	books	PRIMARY KEY
chk_price_positive	books	CHECK

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

- ALTER TABLE members ADD CONSTRAINT unique_email UNIQUE (email);
- SHOW INDEX FROM members;

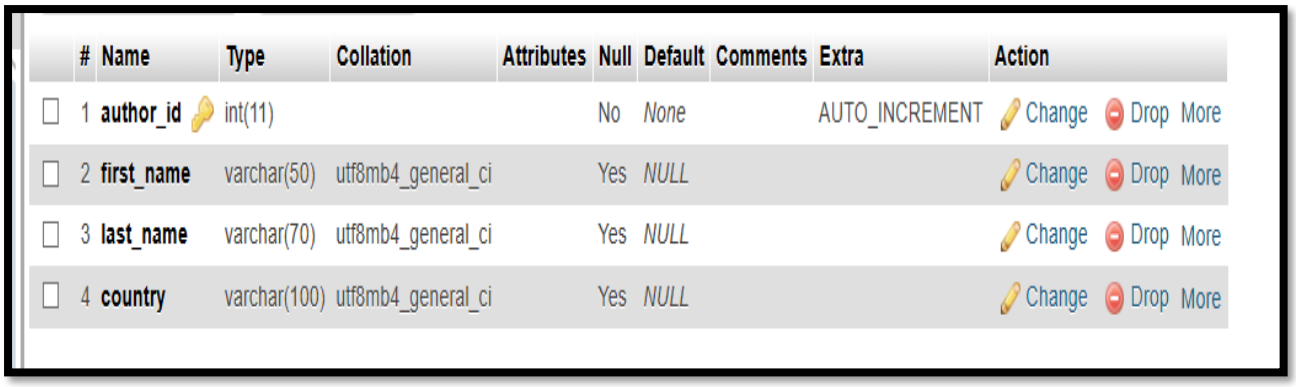
Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comm
members	0	PRIMARY	1	member_id	A	6	NULL	NULL		BTREE	
members	0	unique_email	1	email	A	6	NULL	NULL	YES	BTREE	

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

LAB EXERCISES:

 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.

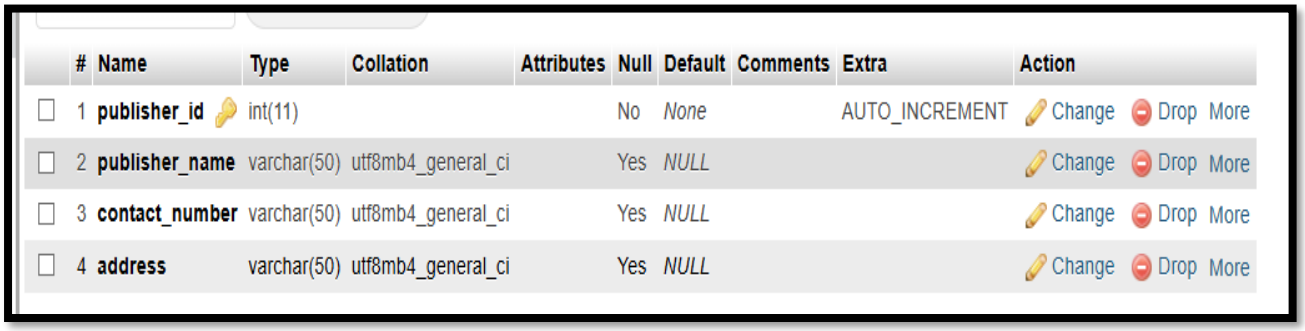
- CREATE TABLE authors (author_id INT AUTO_INCREMENT PRIMARY KEY, first_name VARCHAR(50), last_name VARCHAR(50),country VARCHAR(70));



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	author_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	first_name	varchar(50)	utf8mb4_general_ci		Yes	NULL			Change Drop More
3	last_name	varchar(70)	utf8mb4_general_ci		Yes	NULL			Change Drop More
4	country	varchar(100)	utf8mb4_general_ci		Yes	NULL			Change Drop More

 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.

- CREATE table publishers (publisher_id int AUTO_INCREMENT PRIMARY KEY, publisher_name varchar (50), contact_number varchar (50), address varchar (50));



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	publisher_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	publisher_name	varchar(50)	utf8mb4_general_ci		Yes	NULL			Change Drop More
3	contact_number	varchar(50)	utf8mb4_general_ci		Yes	NULL			Change Drop More
4	address	varchar(50)	utf8mb4_general_ci		Yes	NULL			Change Drop More

5. ALTER Command

LAB EXERCISES:

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

- ALTER TABLE books ADD COLUMN genre VARCHAR(100);
- UPDATE books SET genre = 'Fiction' WHERE title = 'To Kill a Mockingbird';
- UPDATE books SET genre = 'Science Fiction' WHERE title = 'Dune';
- UPDATE books SET genre = 'Fantasy' WHERE title = 'Harry Potter and the Sorcerer''s Stone';
- UPDATE books SET genre = 'Historical' WHERE title = 'The Book Thief';

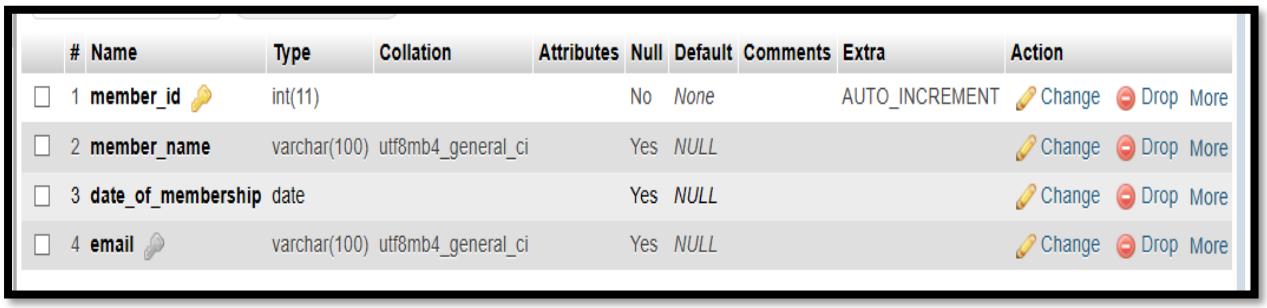
-- Add more UPDATE statements based on the actual titles in your table

- UPDATE books SET genre = 'Unknown';

	book_id	title	author	publisher	year_of_publication	price	genre
Edit Copy Delete	1	The Great Gatsby	F. Scott Fitzgerald	Charles Scribner's Sons	1925	10.99	Unknown
Edit Copy Delete	2	To Kill a Mockingbird	Harper Lee	J.B. Lippincott & Co.	1960	7.99	Unknown
Edit Copy Delete	3	1984	George Orwell	Secker & Warburg	1949	8.99	Unknown
Edit Copy Delete	4	Pride and Prejudice	Jane Austen	T. Egerton	1813	6.99	Unknown
Edit Copy Delete	5	Moby-Dick	Herman Melville	Harper & Brothers	1851	9.99	Unknown

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

- ALTER TABLE members MODIFY COLUMN email VARCHAR (100);



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	member_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	member_name	varchar(100)	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

6. DROP Command

LAB EXERCISES:

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

- DESCRIBE publishers;

Field	Type	Null	Key	Default	Extra
publisher_id	int(11)	NO	PRI	*NULL*	auto_increment
publisher_name	varchar(50)	YES		*NULL*	
contact_number	varchar(50)	YES		*NULL*	
address	varchar(50)	YES		*NULL*	

- DROP TABLE publishers;

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

- SELECT * FROM `members_backup`

member_id	member_name	date_of_membership	email
1	Harshani Patil	2023-01-15	harshani.patil@example.com
2	Krunal Patel	2023-03-10	krunal.patel@example.com
3	Ruchi Patil	2023-05-22	Ruchi.Patil@example.com
4	Nidhi Mehta	2023-07-30	Nidhi.Mehta@example.com
5	Kajal Jain	2023-09-05	kajal.Jain@example.com
6	Rits Patil	2021-11-12	rits.patil@example.com

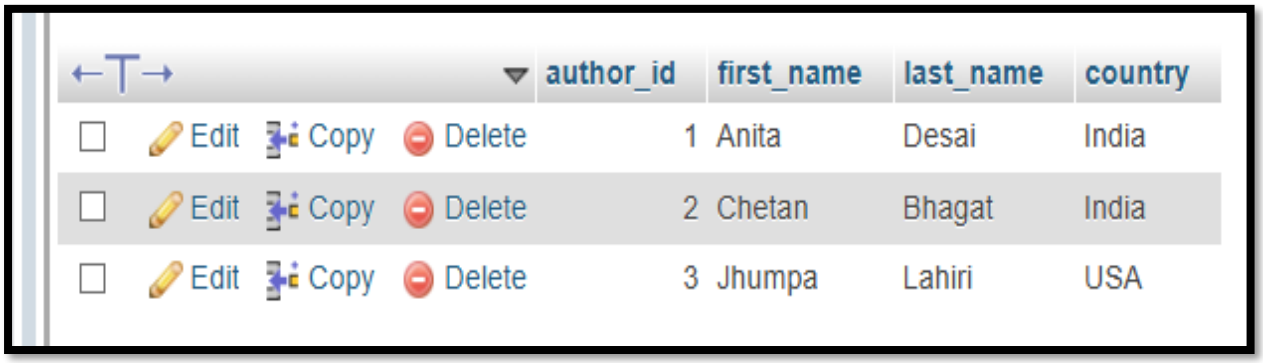
- DROP TABLE members;

7. Data Manipulation Language (DML)

LAB EXERCISES:

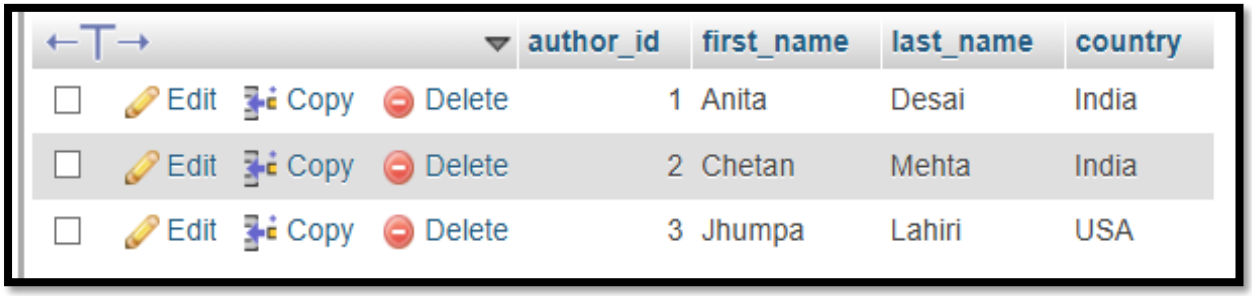
 Lab 4: Insert three new authors into the authors table, then update the last name of one of the authors.

- INSERT INTO authors (first_name, last_name, country) VALUES ('Anita', 'Desai', 'India'), ('Chetan', 'Bhagat', 'India'), ('Jhumpa', 'Lahiri', 'USA');



	author_id	first_name	last_name	country
Edit Copy Delete	1	Anita	Desai	India
Edit Copy Delete	2	Chetan	Bhagat	India
Edit Copy Delete	3	Jhumpa	Lahiri	USA

- UPDATE authors SET last_name = 'Mehta' WHERE first_name = 'Chetan' AND last_name = 'Bhagat';



	author_id	first_name	last_name	country
Edit Copy Delete	1	Anita	Desai	India
Edit Copy Delete	2	Chetan	Mehta	India
Edit Copy Delete	3	Jhumpa	Lahiri	USA

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

- DELETE FROM books WHERE price > 100;

	book_id	title	author	publisher	year_of_publication	price	genre
Edit Copy Delete	1	The Great Gatsby	F. Scott Fitzgerald	Charles Scribner's Sons	1925	10.99	Unknown
Edit Copy Delete	2	To Kill a Mockingbird	Harper Lee	J.B. Lippincott & Co.	1960	7.99	Unknown
Edit Copy Delete	3	1984	George Orwell	Secker & Warburg	1949	8.99	Unknown
Edit Copy Delete	4	Pride and Prejudice	Jane Austen	T. Egerton	1813	6.99	Unknown
Edit Copy Delete	5	Moby-Dick	Herman Melville	Harper & Brothers	1851	9.99	Unknown

8. UPDATE Command

LAB EXERCISES:

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

- UPDATE books SET year_of_publication = 2022 WHERE book_id = 5;

	book_id	title	author	publisher	year_of_publication	price	genre
Edit Copy Delete	1	The Great Gatsby	F. Scott Fitzgerald	Charles Scribner's Sons	1925	10.99	Unknown
Edit Copy Delete	2	To Kill a Mockingbird	Harper Lee	J.B. Lippincott & Co.	1960	7.99	Unknown
Edit Copy Delete	3	1984	George Orwell	Secker & Warburg	1949	8.99	Unknown
Edit Copy Delete	4	Pride and Prejudice	Jane Austen	T. Egerton	1813	6.99	Unknown
Edit Copy Delete	5	Moby-Dick	Herman Melville	Harper & Brothers	2022	9.99	Unknown

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

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

	book_id	title	author	publisher	year_of_publication	price	genre
Edit Copy Delete	1	The Great Gatsby	F. Scott Fitzgerald	Charles Scribner's Sons	1925	12.09	Unknown
Edit Copy Delete	2	To Kill a Mockingbird	Harper Lee	J.B. Lippincott & Co.	1960	8.79	Unknown
Edit Copy Delete	3	1984	George Orwell	Secker & Warburg	1949	9.89	Unknown
Edit Copy Delete	4	Pride and Prejudice	Jane Austen	T. Egerton	1813	7.69	Unknown
Edit Copy Delete	5	Moby-Dick	Herman Melville	Harper & Brothers	2022	9.99	Unknown

9. DELETE Command

LAB EXERCISES:

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

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

- SELECT * FROM members WHERE date_of_membership < '2020-01-01';

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

- DELETE FROM books WHERE author IS NULL;

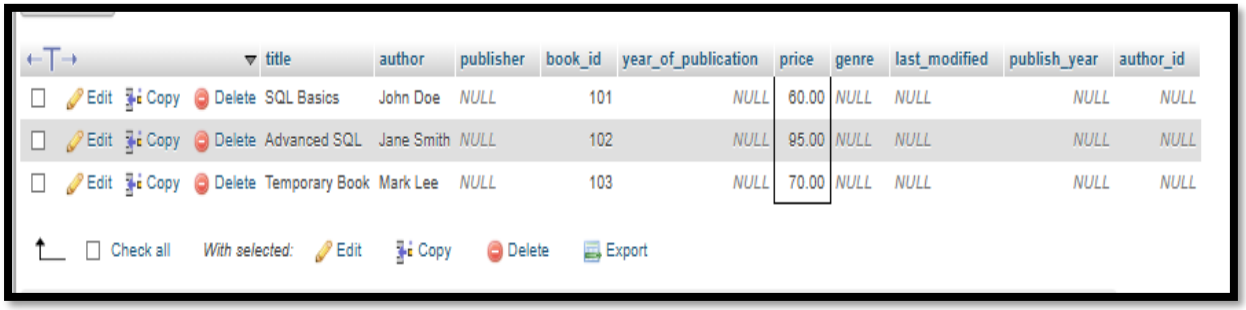
- SELECT * FROM books WHERE author IS NULL;

10. Data Query Language (DQL)

LAB EXERCISES:

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

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



	title	author	publisher	book_id	year_of_publication	price	genre	last_modified	publish_year	author_id
Edit Copy Delete	SQL Basics	John Doe	NULL	101	NULL	60.00	NULL	NULL	NULL	NULL
Edit Copy Delete	Advanced SQL	Jane Smith	NULL	102	NULL	95.00	NULL	NULL	NULL	NULL
Edit Copy Delete	Temporary Book	Mark Lee	NULL	103	NULL	70.00	NULL	NULL	NULL	NULL

Check all With selected: Edit Copy Delete Export

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

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

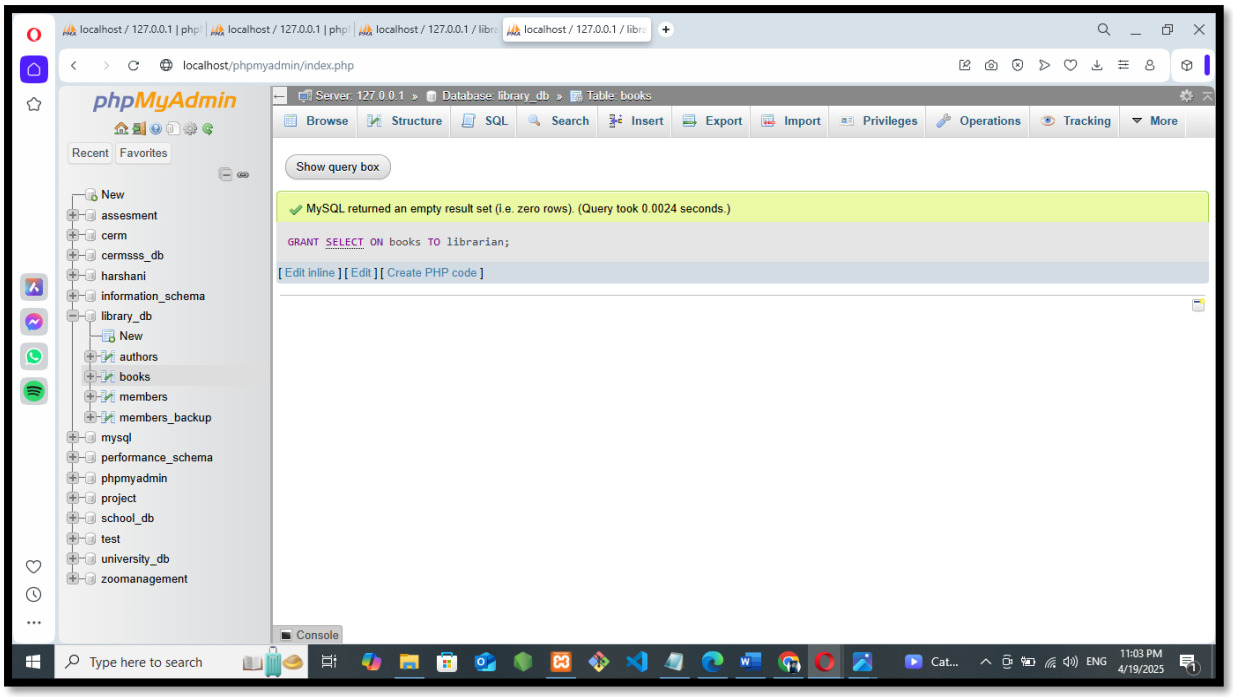
	book_id	title	author ▲ 1	publisher	year_of_publication	price	genre
Edit Copy Delete	1	The Great Gatsby	F. Scott Fitzgerald	Charles Scribner's Sons	1925	12.09	Unknown
Edit Copy Delete	3	1984	George Orwell	Secker & Warburg	1949	9.89	Unknown
Edit Copy Delete	2	To Kill a Mockingbird	Harper Lee	J.B. Lippincott & Co.	1960	8.79	Unknown

11. Data Control Language (DCL)

LAB EXERCISES:

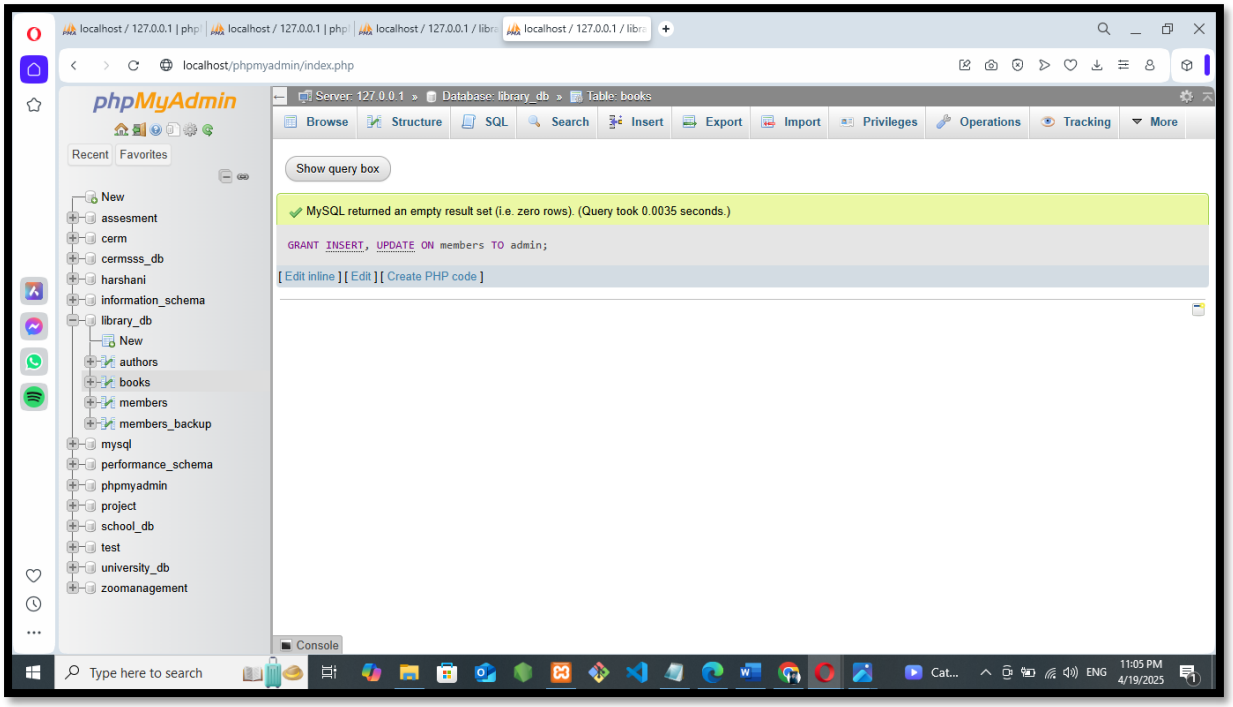
 Lab 3: Grant SELECT permission to a user named librarian on the books table.

- GRANT SELECT ON books TO librarian;



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

- GRANT INSERT, UPDATE ON members TO admin;

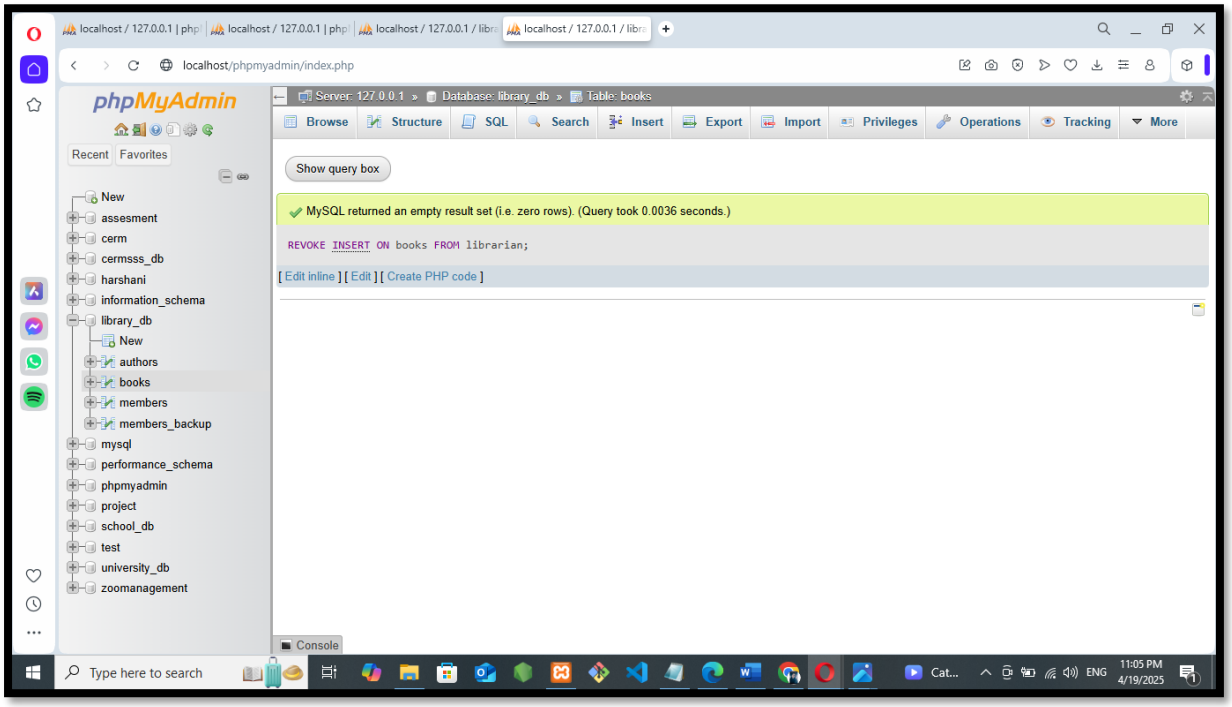


12. REVOKE Command

LAB EXERCISES:

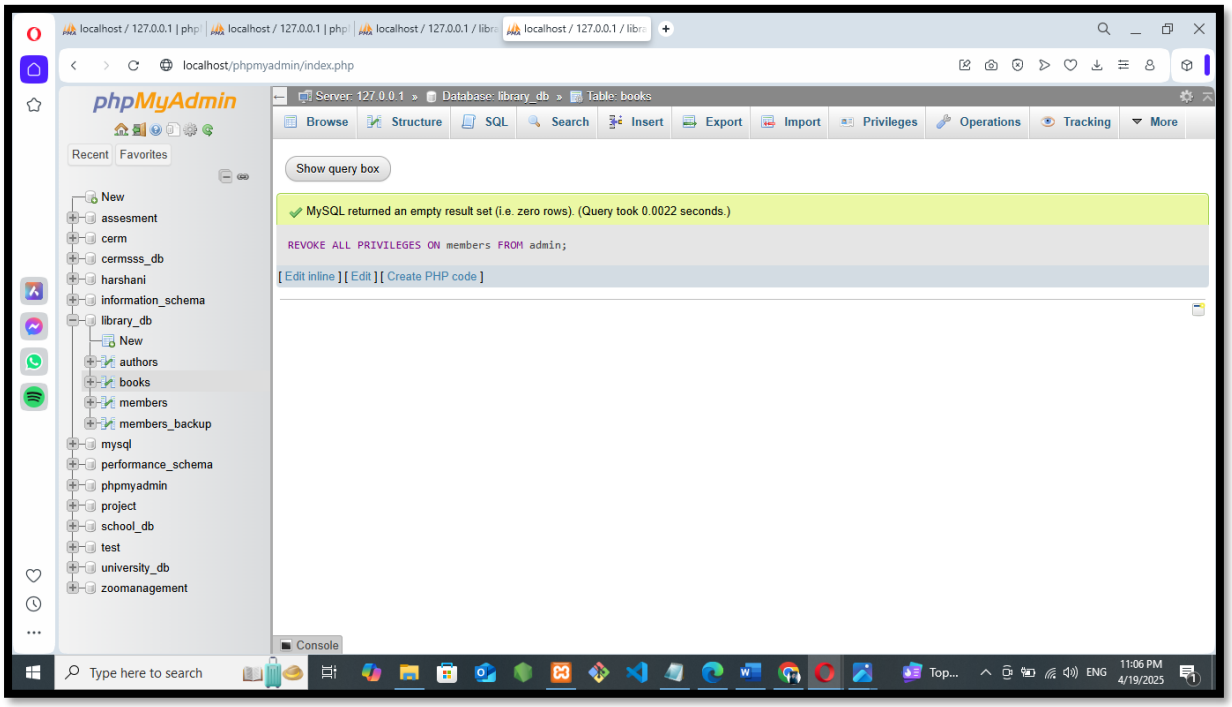
 Lab 3: Revoke the INSERT privilege from the user librarian on the books table.

- REVOKE INSERT ON books FROM librarian;



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

- REVOKE ALL PRIVILEGES ON members FROM admin;



13. Transaction Control Language (TCL)

LAB EXERCISES:

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

Start a transaction (optional in some DBs)

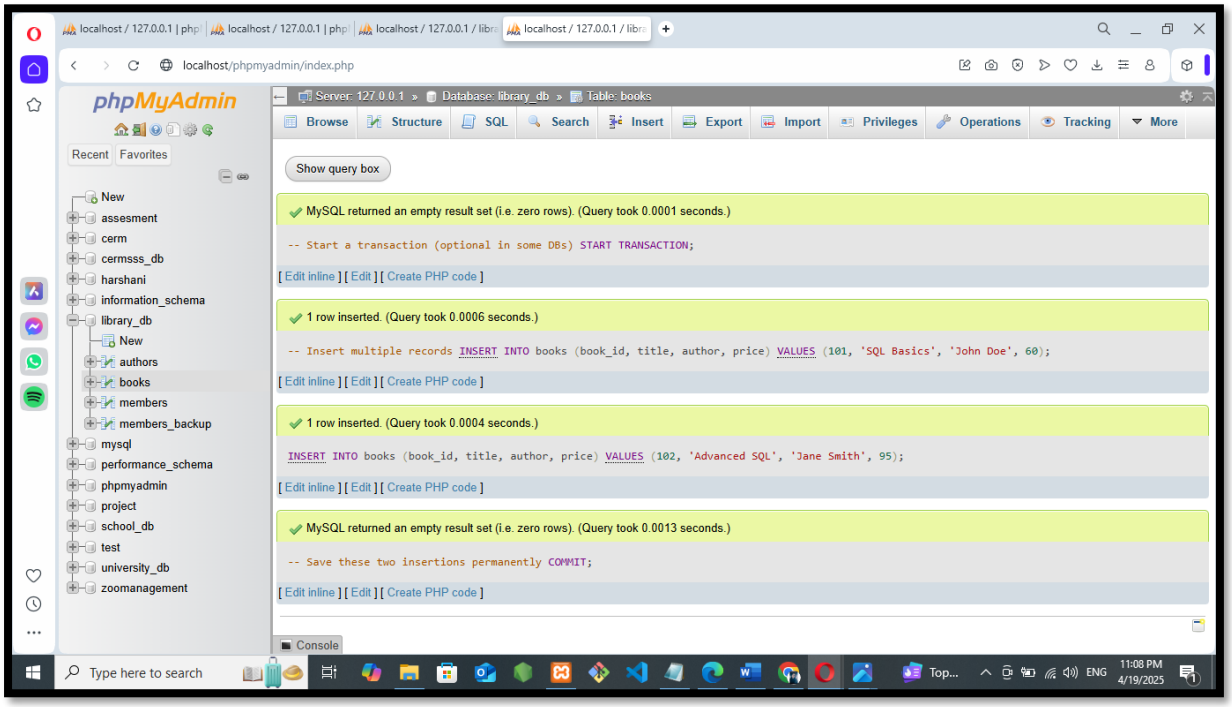
- START TRANSACTION;

Insert multiple records

- INSERT INTO books (book_id, title, author, price) VALUES (101, 'SQL Basics', 'John Doe', 60);
- INSERT INTO books (book_id, title, author, price) VALUES (102, 'Advanced SQL', 'Jane Smith', 95);

Save these two insertions permanently

- COMMIT;

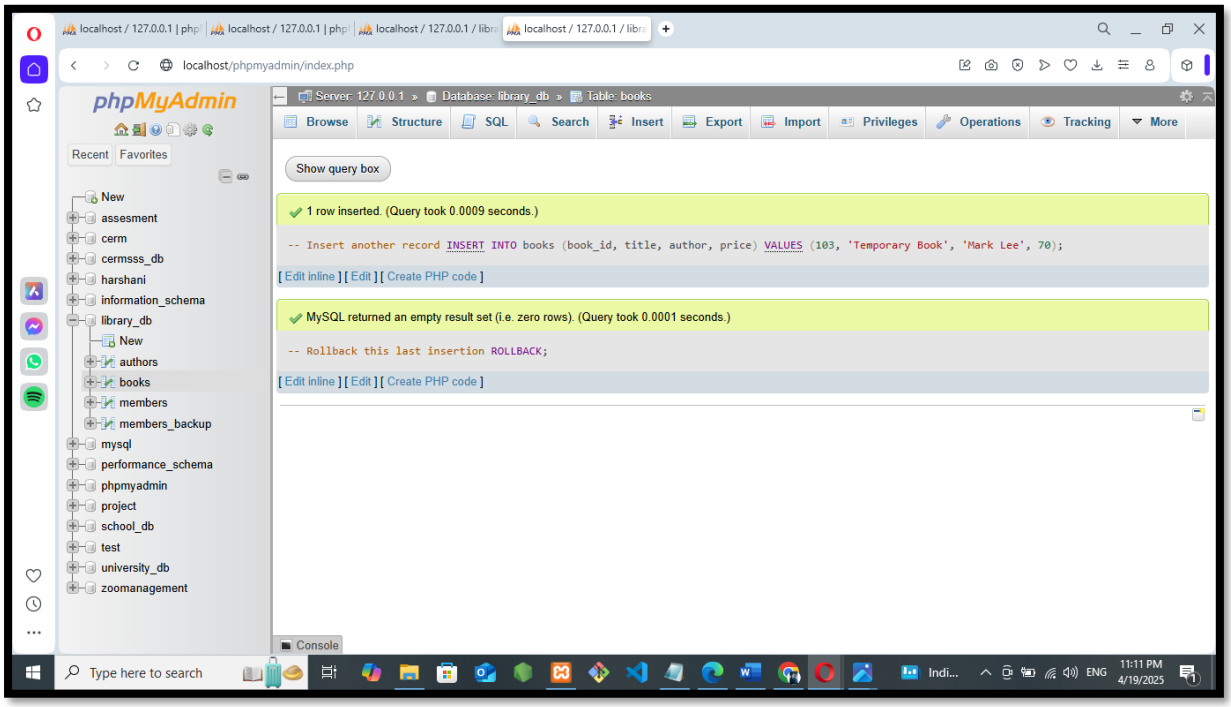


Insert another record

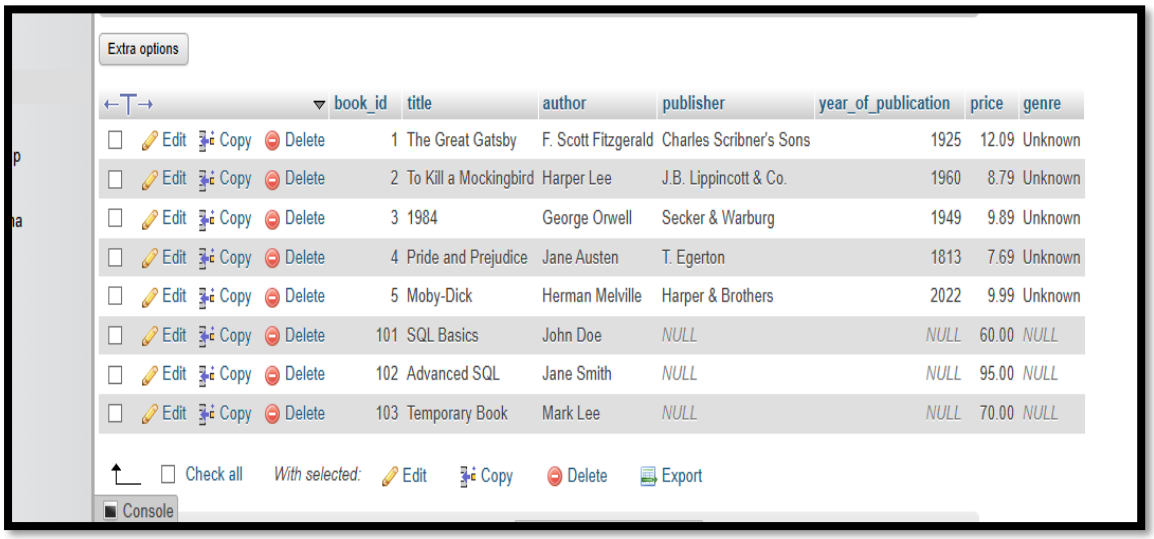
- INSERT INTO books (book_id, title, author, price) VALUES (103, 'Temporary Book', 'MarkLee',70);

Rollback this last insertion

- ROLLBACK;



- SELECT * FROM books;



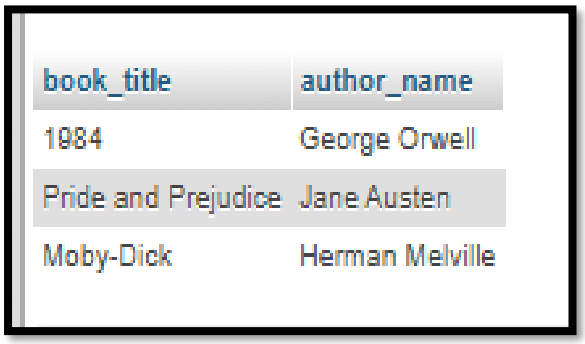
Lab 4: Set a SAVEPOINT before making updates to the members table, perform some updates, and then roll back to the SAVEPOINT.

14. SQL Joins

LAB EXERCISES:

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

- SELECT books.title AS book_title, CONCAT(authors.first_name, ' ', authors.last_name) AS author_name FROM books
  INNER JOIN
  authors ON books.author_id = authors.author_id;



book_title	author_name
1984	George Orwell
Pride and Prejudice	Jane Austen
Moby-Dick	Herman Melville

 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.

- SELECT books.title AS book_title,CONCAT(authors.first_name, ' ', authors.last_name) AS author_name,books.author_id AS book_author_id, authors.author_id AS author_table_id
  FROM books
  LEFT JOIN authors ON books. author_id = authors.author_id
  UNION
  SELECT books.title AS book_title,
  CONCAT(authors.first_name, ' ', authors.last_name) AS author_name,
  books.author_id AS book_author_id,
  authors.author_id AS author_table_id
  FROM
  authors
  LEFT JOIN books ON books.author_id = authors.author_id;

book_title	author_name	book_author_id	author_table_id
Updated Title	NULL	NULL	NULL
1984	George Orwell	4	4
Pride and Prejudice	Jane Austen	5	5
Moby-Dick	Herman Melville	6	6
SQL Basics	NULL	NULL	NULL
Advanced SQL	NULL	NULL	NULL
Temporary Book	NULL	NULL	NULL
PL/SQL Programming	NULL	NULL	NULL
Wings of Fire	NULL	NULL	NULL
The Alchemist	NULL	NULL	NULL
The Secret	NULL	NULL	NULL
NULL	Anita Desai	NULL	1
NULL	Chetan Mehta	NULL	2
NULL	Jhumpa Lahiri	NULL	3
NULL	Jhumpa Lahiri	NULL	10

15. SQL Group By

LAB EXERCISES:

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

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

genre	total_books
NULL	3
Unknown	5

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

- SELECT YEAR (date_of_membership) AS join_year,COUNT(*) AS member_count
  FROM members
  GROUP BY YEAR(date_of_membership)
  ORDER BY join_year LIMIT 0, 25;

join_year ▲ 1	member_count
2019	1
2021	1
2023	5
2025	1

16. SQL Stored Procedure

LAB EXERCISES:

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

- DELIMITER $$

  CREATE PROCEDURE GetBooksByAuthor(IN author_name VARCHAR(255))
  BEGIN
  SELECT * FROM books WHERE author = author_name;
  END $$

  DELIMITER ;

- CALL GetBooksByAuthor('John Doe');

Extra options

book_id	title	author	publisher	year_of_publication	price	genre
101	SQL Basics	John Doe	*NULL*	*NULL*	60.00	*NULL*

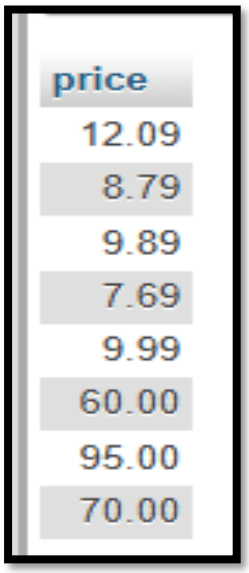
 Lab 4: Write a stored procedure that takes book_id as an argument and returns the price of the book.

- DELIMITER $$

  CREATE PROCEDURE GetBookPriceById(IN book_id INT)
  BEGIN
  SELECT price FROM books WHERE book_id = book_id;
  END $$

  DELIMITER ;

- CALL GetBookPriceById(101);



price
12.09
8.79
9.89
7.69
9.99
60.00
95.00
70.00

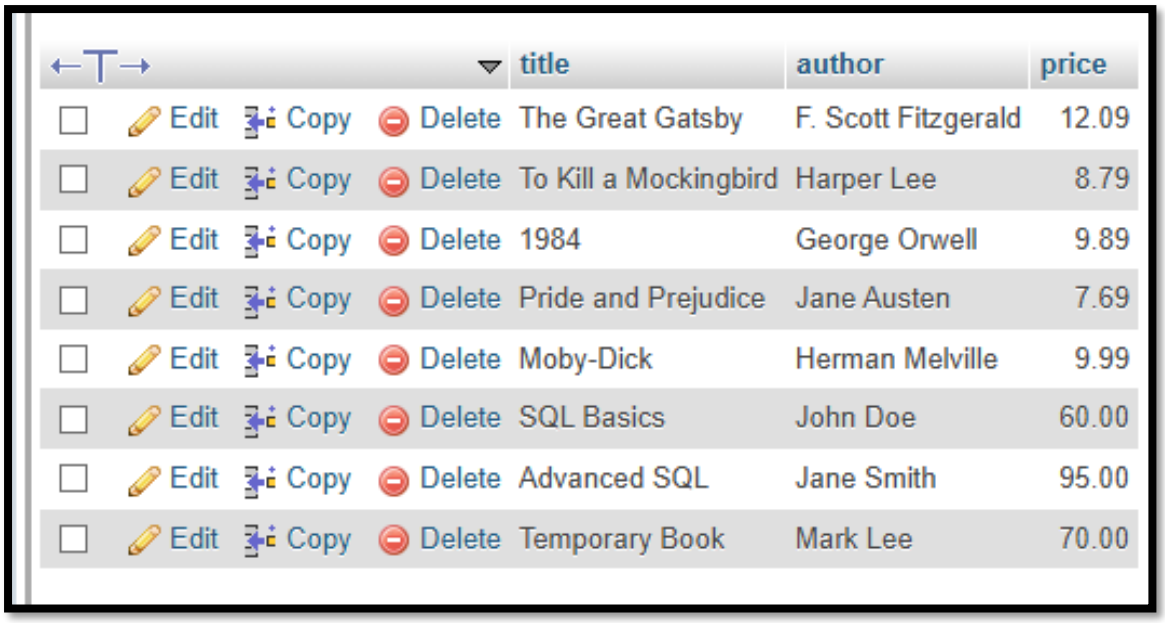
17. SQL View

LAB EXERCISES:

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

- CREATE VIEW book_details AS SELECT title, author, price FROM books;

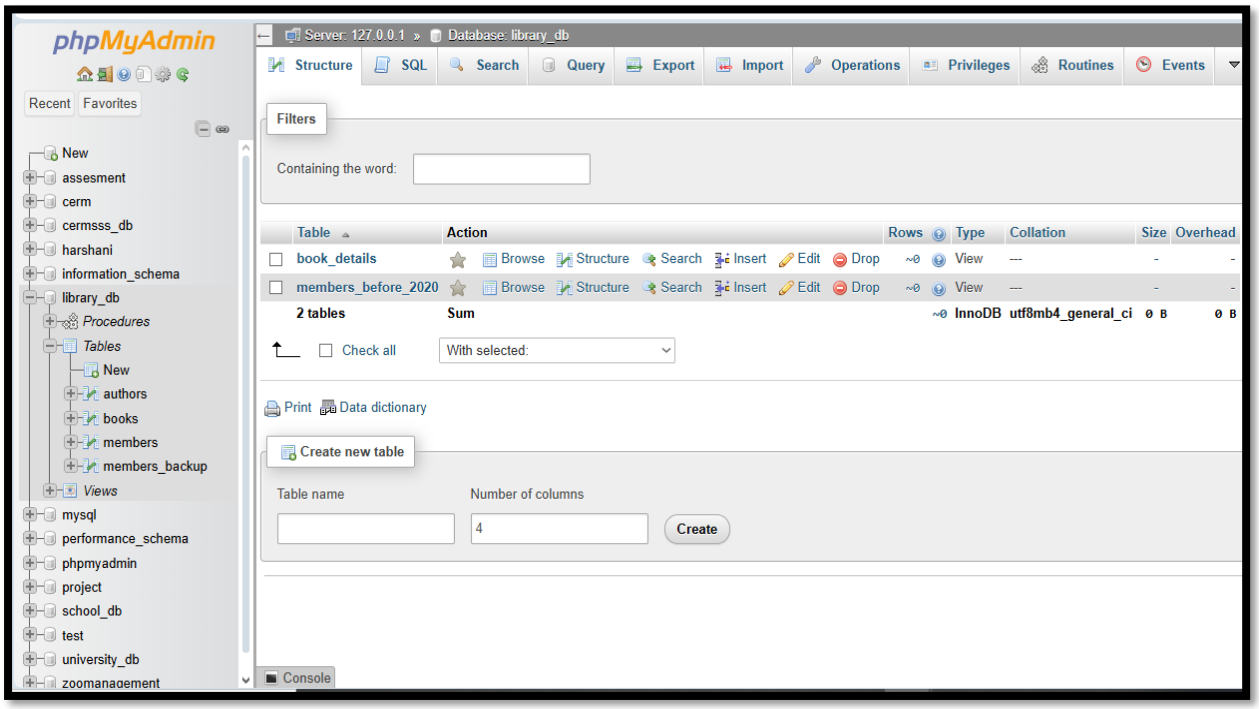
- SELECT * FROM book_details;



	title	author	price
Edit Copy Delete	The Great Gatsby	F. Scott Fitzgerald	12.09
Edit Copy Delete	To Kill a Mockingbird	Harper Lee	8.79
Edit Copy Delete	1984	George Orwell	9.89
Edit Copy Delete	Pride and Prejudice	Jane Austen	7.69
Edit Copy Delete	Moby-Dick	Herman Melville	9.99
Edit Copy Delete	SQL Basics	John Doe	60.00
Edit Copy Delete	Advanced SQL	Jane Smith	95.00
Edit Copy Delete	Temporary Book	Mark Lee	70.00

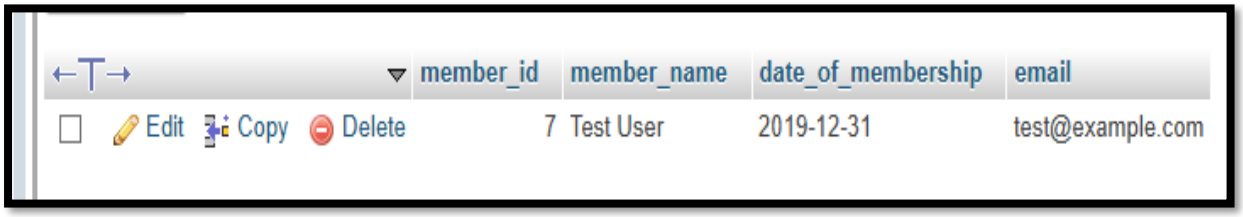
 Lab 4: Create a view to display members who joined before 2020.

- CREATE VIEW members_before_2020 AS
  SELECT *FROM members WHERE date_of_membership < '2020-01-01';



- INSERT INTO members (member_id, member_name, date_of_membership, email)VALUES (7, 'Test User', '2019-12-31', 'test@example.com');

- SELECT * FROM members_before_2020;



	member_id	member_name	date_of_membership	email
Edit Copy Delete	7	Test User	2019-12-31	test@example.com

18. SQL Trigger

LAB EXERCISES:

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

- ALTER TABLE books ADD last_modified DATETIME;

Create the trigger:

- DELIMITER $$

```
CREATE TRIGGER update_books_last_modified
BEFORE UPDATE ON books
FOR EACH ROW
BEGIN
SET NEW.last_modified = NOW();
END$$

DELIMITER ;
```

Test the Trigger:

- UPDATE books SET title = 'Updated Title' WHERE book_id = 1;

Then check:

- SELECT * FROM books WHERE book_id = 1;

	book_id	title	author	publisher	year_of_publication	price	genre	last_modified
Edit Copy Delete	1	Updated Title	F. Scott Fitzgerald	Charles Scribner's Sons	1925	12.09	Unknown	2025-04-20 13:49:49

 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.

Create the log_changes table:

- CREATE TABLE log_changes (log_id INT AUTO_INCREMENT PRIMARY KEY,book_id INT,title VARCHAR(255), author VARCHAR(255),deleted_at DATETIME);

Create the AFTER DELETE trigger:

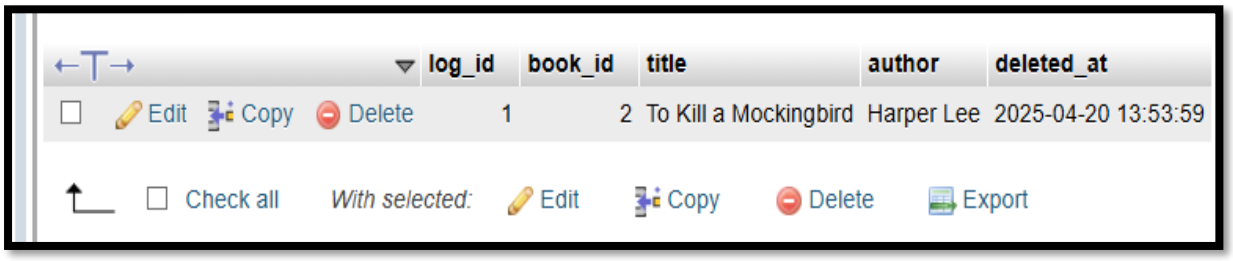
- DELIMITER $$

```
CREATE TRIGGER after_book_delete
AFTER DELETE ON books
FOR EACH ROW
BEGIN
INSERT INTO log_changes (book_id, title, author, deleted_at)
VALUES (OLD.book_id, OLD.title, OLD.author, NOW ());
END$$

DELIMITER ;
```

Delete a book:

- DELETE FROM books WHERE book_id = 2;

- SELECT * FROM log_changes;



19. Introduction to PL/SQL

LAB EXERCISES:

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

-- Change delimiter to $$ to allow semicolons inside the procedure body

- DELIMITER $$

Create the procedure

```
CREATE PROCEDURE insert_book(
IN book_id INT,
```

```
    IN title VARCHAR(255),
    IN author VARCHAR(255),
    IN publish_year INT
    )
    BEGIN
```

Insert a new book into the books table

```
    INSERT INTO books (book_id, title, author, publish_year) VALUES (book_id, title,
    author, publish_year);
```

Display a confirmation message

```
    SELECT CONCAT('Book "', title, '" by ', author, ' has been successfully added.') AS
    confirmation_message;
    END $$
```

Reset the delimiter back to semicolon

```
    DELIMITER ;
```

- DESCRIBE books;

Field	Type	Null	Key	Default	Extra
book_id	int(11)	NO	PRI	*NULL*	auto_increment
title	varchar(255)	YES		*NULL*	
author	varchar(255)	YES		*NULL*	
publisher	varchar(255)	YES		*NULL*	
year_of_publication	int(11)	YES		*NULL*	
price	decimal(10,2)	YES		*NULL*	
genre	varchar(100)	YES		*NULL*	
last_modified	datetime	YES		*NULL*	

- ALTER TABLE books ADD COLUMN publish_year INT;

- CALL insert_book(201, 'PL/SQL Programming', 'Harshani Patil', 2025);

confirmation_message
Book "PL/SQL Programming" by Harshani Patil has be...

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

- DELIMITER $$

  CREATE PROCEDURE get_total_books()
  BEGIN
  SELECT COUNT(*) AS total_books FROM books;
  END $$

  DELIMITER ;

- CALL get_total_books();

20. PL/SQL Syntax

LAB EXERCISES:

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

- DELIMITER $$
  CREATE PROCEDURE display_book_details()
  BEGIN

Declare variables

  DECLARE book_id INT DEFAULT 101;
  DECLARE price DECIMAL(8,2) DEFAULT 499.99;

Display results

  SELECT CONCAT ('Book ID: ', book_id) AS book_id_message, CONCAT ('Price: ₹', price) AS price_message;
  END $$
  DELIMITER;

- CALL display_book_details ();

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

- DELIMITER $$

```
CREATE PROCEDURE book_price_arithmetic ()
BEGIN
Declare constants using variables (MySQL doesn't support true constants in procedures)
DECLARE book_price1 DECIMAL (8,2) DEFAULT 350.75;
DECLARE book_price2 DECIMAL (8,2) DEFAULT 499.25;

Declare result variables
DECLARE total_price DECIMAL (8,2);
DECLARE price_difference DECIMAL (8,2);
DECLARE average price DECIMAL (8,2);

Perform arithmetic operations
SET total_price = book_price1 + book_price2;
SET price_difference = ABS (book_price1 - book_price2);
SET average_price = (book_price1 + book_price2) / 2;

Display results
SELECT
book_price1 AS 'Book Price 1',
book_price2 AS 'Book Price 2',
total_price AS 'Total Price',
price_difference AS 'Price Difference',
average_price AS 'Average Price';
END $$

DELIMITER;
```

- CALL book_price_arithmetic ();

Book Price 1	Book Price 2	Total Price	Price Difference	Average Price
350.75	499.25	850.00	148.50	425.00

21. PL/SQL Control Structures

LAB EXERCISES:

 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.

- DELIMITER $$

  CREATE PROCEDURE check_book_price()
  BEGIN
  DECLARE book_price DECIMAL(8,2) DEFAULT 120.50; -- You can change this value

  IF book_price > 100 THEN
  SELECT 'The book price is above $100.' AS message;
  ELSE
  SELECT 'The book price is $100 or below.' AS message;
  END IF;
  END $$

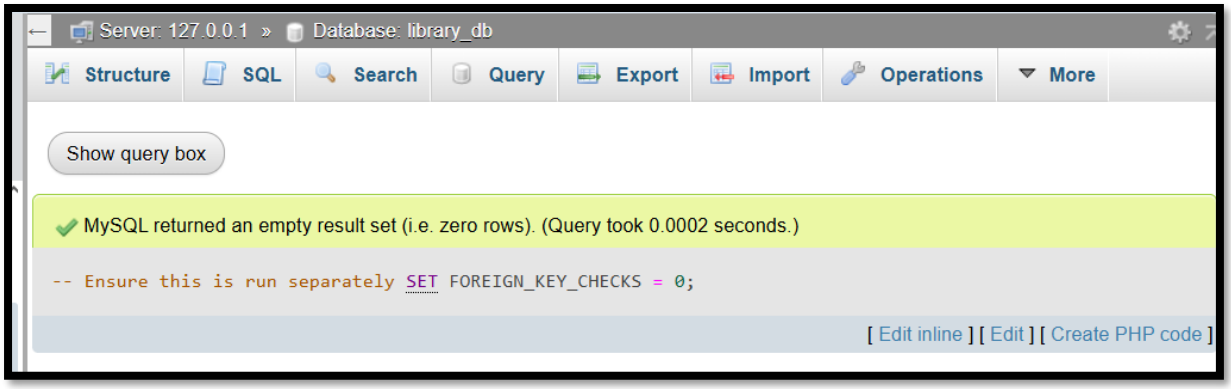
  DELIMITER;

- CALL check_book_price();

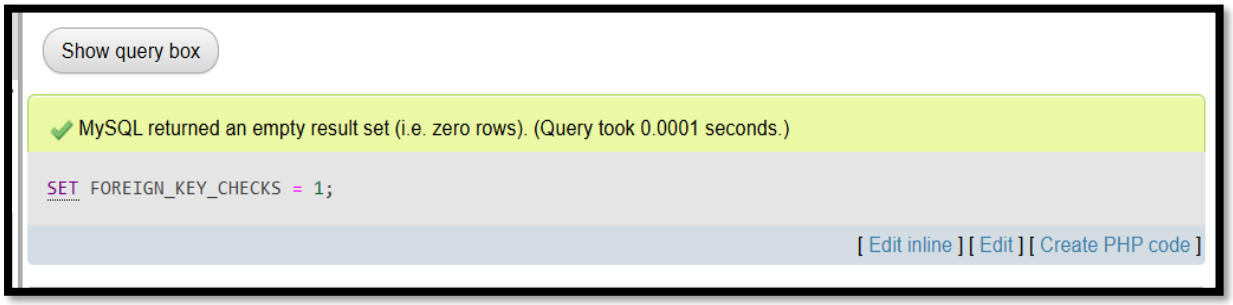
 Lab 4: Use a FOR LOOP in PL/SQL to display the details of all books one by one.

-- Ensure this is run separately

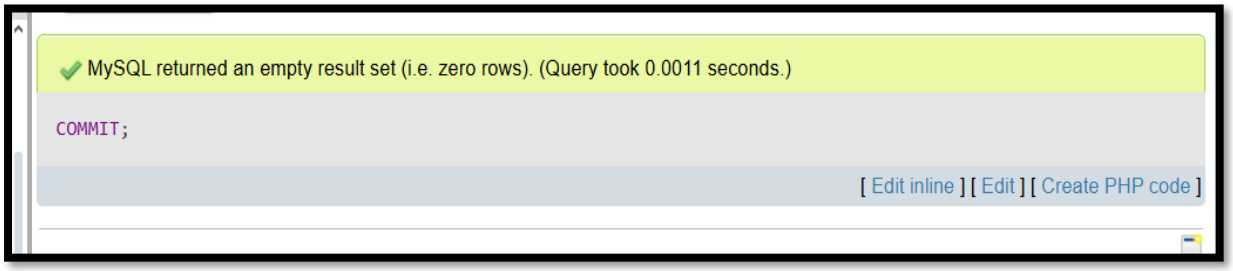
- SET FOREIGN_KEY_CHECKS = 0;



- SET FOREIGN_KEY_CHECKS = 1;



- COMMIT;



22. SQL Cursors

LAB EXERCISES:

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

- DELIMITER $$
  CREATE PROCEDURE FetchAllMembers()
  BEGIN

Declare variables to hold column data

  DECLARE done INT DEFAULT FALSE;
  DECLARE v_id INT;
  DECLARE v_name VARCHAR(100);
  DECLARE v_join_date DATE;
  DECLARE v_email VARCHAR(100);

Declare the cursor

  DECLARE member_cursor CURSOR FOR
  SELECT member_id, member_name, date_of_membership, email FROM members;

Declare handler for when there are no more rows

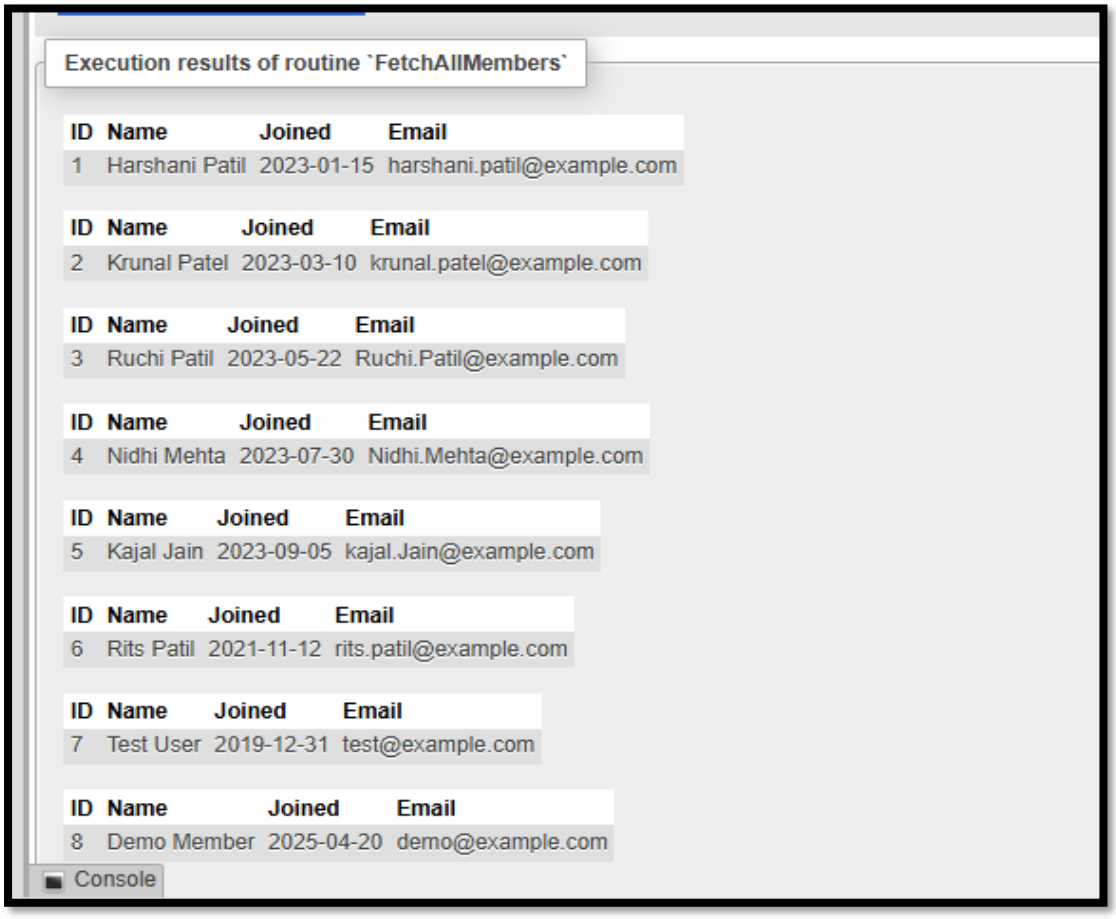
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

Open the cursor

  OPEN member_cursor;
  read_loop: LOOP
  FETCH member_cursor INTO v_id, v_name, v_join_date, v_email;
  IF done THEN
  LEAVE read_loop;

```
        END IF;
 Since MySQL does not support DBMS_OUTPUT, we'll use SELECT to display
        SELECT v_id AS 'ID', v_name AS 'Name', v_join_date AS 'Joined', v_email AS 'Email';
        END LOOP;
  Close the cursor
        CLOSE member_cursor;
        END$$

        DELIMITER ;
```

- `CALL `FetchAllMembers`();`



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

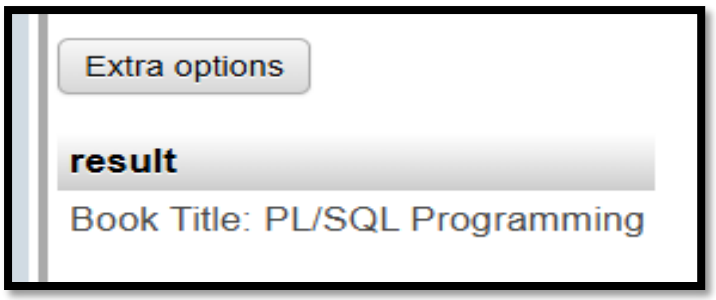
- `DROP PROCEDURE IF EXISTS get_books_by_author;`

Create the procedure

- `DELIMITER $$`

```
CREATE PROCEDURE get_books_by_author(IN author_name VARCHAR(100))
BEGIN
DECLARE done INT DEFAULT 0;
DECLARE book_title VARCHAR(255);

Declare cursor
DECLARE book_cursor CURSOR FOR
SELECT title FROM books WHERE author = author_name;

Exit handler
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

Open cursor
OPEN book_cursor;

read_loop: LOOP
 FETCH book_cursor INTO book_title;
 IF done THEN
LEAVE read_loop;
END IF;

Display the book title
SELECT CONCAT('Book Title: ', book_title) AS result;
END LOOP;

CLOSE book_cursor;
END $$

DELIMITER ;
```

- CALL get_books_by_author('Harshani Patil');



- DESCRIBE books;

Field	Type	Null	Key	Default	Extra
book_id	int(11)	NO	PRI	*NULL*	auto_increment
title	varchar(255)	YES		*NULL*	
author	varchar(255)	YES		*NULL*	
publisher	varchar(255)	YES		*NULL*	
year_of_publication	int(11)	YES		*NULL*	
price	decimal(10,2)	YES		*NULL*	
genre	varchar(100)	YES		*NULL*	
last_modified	datetime	YES		*NULL*	
publish_year	int(11)	YES		*NULL*	

23. Rollback and Commit Savepoint

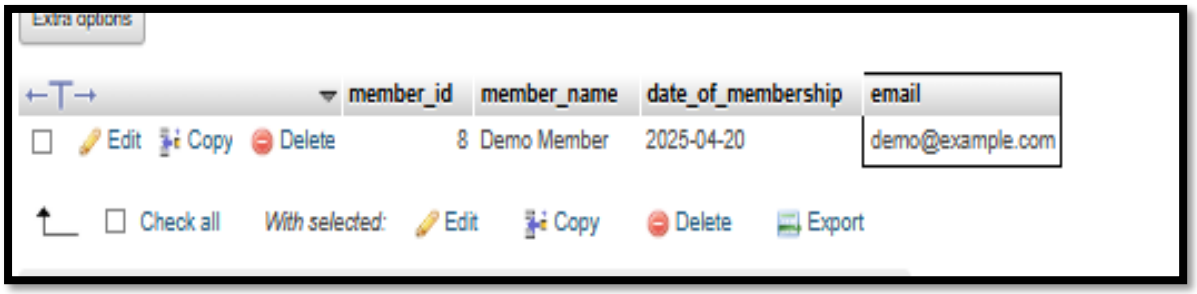
LAB EXERCISES:

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

```
-- Start the transaction
START TRANSACTION;

-- Insert a new member
INSERT INTO members (member_name, date_of_membership, email)
VALUES ('Demo Member', '2025-04-20', 'demo@example.com');

-- Set a savepoint
SAVEPOINT after_insert;

-- Update the newly inserted member's name (this will be undone)
UPDATE members
SET member_name = 'Updated Demo'
WHERE email = 'demo@example.com';

-- Rollback to the savepoint (undo the update)
ROLLBACK TO SAVEPOINT after_insert;

-- Commit the transaction (only the insert remains)
```

COMMIT;

- SELECT * FROM members WHERE email = 'demo@example.com';

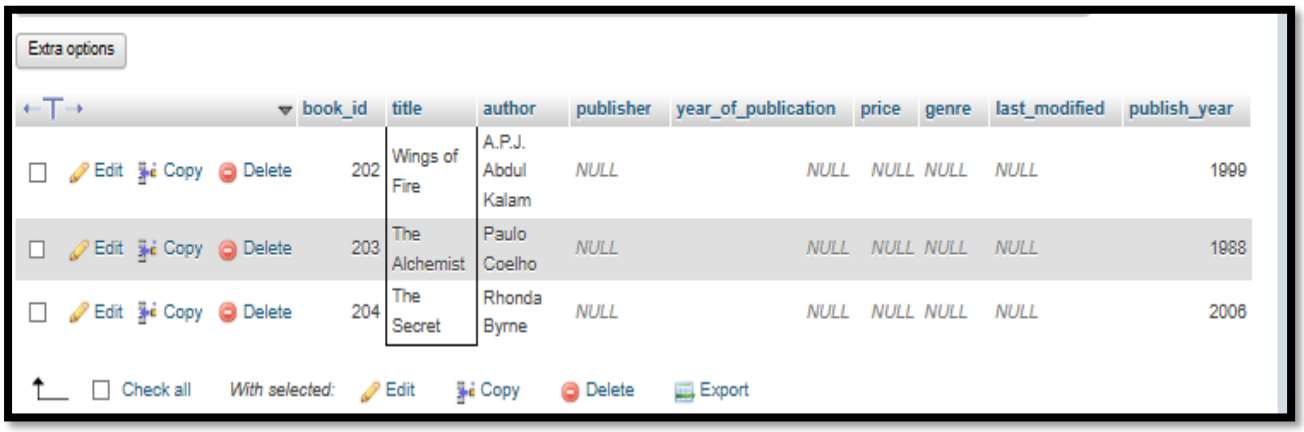


Lab 4: Use COMMIT after successfully inserting multiple books into the books table, then use

ROLLBACK to undo a set of changes made after a savepoint.

```
-- Step 1: Start a transaction
START TRANSACTION;

-- Step 2: Insert multiple new books
INSERT INTO books (title, author, publish_year)
VALUES
('Wings of Fire', 'A.P.J. Abdul Kalam', 1999),
('The Alchemist', 'Paulo Coelho', 1988),
('The Secret', 'Rhonda Byrne', 2006);

-- Step 3: Commit the successful inserts
COMMIT;

-- Step 4: Start a new transaction to make changes
START TRANSACTION;

-- Step 5: Set a savepoint
SAVEPOINT before_update;

-- Step 6: Perform updates (which we will undo)
UPDATE books SET author = 'Unknown Author' WHERE title = 'The Secret';
UPDATE books SET publish_year = 2025 WHERE title = 'The Alchemist';

-- Step 7: Rollback the updates (back to savepoint)
ROLLBACK TO SAVEPOINT before_update;

-- Step 8: Final commit to save only what came before the savepoint
COMMIT;
```

- SELECT * FROM books WHERE title IN ('Wings of Fire', 'The Alchemist', 'The Secret');



Extra options

	book_id	title	author	publisher	year_of_publication	price	genre	last_modified	publish_year
Edit Copy Delete	202	Wings of Fire	A.P.J. Abdul Kalam	NULL	NULL	NULL	NULL	NULL	1999
Edit Copy Delete	203	The Alchemist	Paulo Coelho	NULL	NULL	NULL	NULL	NULL	1988
Edit Copy Delete	204	The Secret	Rhonda Byrne	NULL	NULL	NULL	NULL	NULL	2006

Check all With selected: Edit Copy Delete Export