|  |
| --- |
| **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.**  **ANS :-**  CREATE DATABASE library\_db;  CREATE TABLE books (  book\_id int ,  title text ,  AUTHOR text ,  publisher varchar(60),  YEAR\_of\_publication int,  price INT  );  INSERT INTO books VALUES(101, 'To Kill a Mockingbird', 'Harper Lee', 'J.B. Lippincott & Co.', 1960, 1800),  (102, '1984', 'George Orwell', 'Secker & Warburg', 1949, 1599),  (103, 'The Great Gatsby', 'F. Scott Fitzgerald', 'Charles Scribner\'s Sons', 1925, 1099),  (104, 'Pride and Prejudice', 'Jane Austen', 'T.Egerton', 1813, 1299),  (105, 'Moby-Dick', 'Herman Melville', 'Harper & Brothers', 1851, 1799);    **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.**  **ANS :-**  CREATE TABLE members (  member\_id int ,  member\_name varchar(30),  date\_of\_membership date,  email text  );  INSERT INTO members VALUES  (1, 'John Doe', '2022-05-10', 'john.doe@example.com'),  (2, 'Jane Smith', '2023-01-15', 'jane.smith@example.com'),  (3, 'Alice Johnson', '2021-11-25', 'alice.johnson@example.com'),  (4, 'Bob Brown', '2023-03-30', 'bob.brown@example.com'),  (5, 'Charlie Davis', '2022-07-20', 'charlie.davis@example.com'); |
| **Lab 3: Retrieve all members who joined the library before 2022. Use appropriate SQL syntax with WHERE and ORDER BY.**  **ANS :-**  [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) member\_name , date\_of\_membership FROM members WHERE date\_of\_membership <='2022-01-01' ORDER BY date\_of\_membership;    SELECT \* FROM members ORDER BY member\_name;    **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.**  **ANS :-**  SELECT AUTHOR, title FROM books; |
| **Lab 3: Add a CHECK constraint to ensure that the price of books in the books table is greater than 0.**  **ANS :-**  CREATE TABLE books(  book\_id int PRIMARY KEY ,  title text ,  AUTHOR text ,  publisher varchar(50),  YEAR\_of\_publication int ,  price int CHECK(price>0));    **Lab 4: Modify the members table to add a UNIQUE constraint on the email column, ensuring that each member has a unique email address.**  **ANS :-**  ALTER TABLE members MODIFY COLUMN email text UNIQUE; |
| **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.**  **ANS :-**  CREATE TABLE AUTHOR(  AUTHOR\_id int ,  FIRST\_name text ,  LAST\_name varchar(50),  country text,  PRIMARY KEY(AUTHOR\_id));    **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.**  **ANS :-**  CREATE TABLE publishers(  publisher\_id int PRIMARY KEY ,  publisher\_name varchar(50) ,  concat\_number int UNIQUE ,  address text  );    **Lab 3: Add a new column genre to the books table. Update the genre for all existing records.**  **ANS :-**  ALTER TABLE books ADD COLUMN genre text;  UPDATE books SET genre='Modern Indian Social Drama' WHERE book\_id=101;  UPDATE books SET genre='Political Fiction, Dystopian' WHERE book\_id=102;  UPDATE books SET genre='Modernist Indian Fiction' WHERE book\_id=103;  UPDATE books SET genre='Indian Romantic Literature' WHERE book\_id=104;  UPDATE books SET genre='Adventure, Mythological' WHERE book\_id=105;    **Lab 4: Modify the members table to increase the length of the email column to 100 characters.**  **ANS :-**  ALTER TABLE members MODIFY COLUMN email varchar(100) ; |
| **Lab 3: Drop the publishers table from the database after verifying its structure.**  **ANS :-**  DROP TABLE publishers;  Lab 4: Create a backup of the members table and then drop the original members table.  ANS :-  CREATE TABLE members\_backup SELECT \* FROM members;  DROP TABLE members; |
| **Lab 4: Insert three new authors into the authors table, then update the last name of one of the authors.**  **ANS :-**  INSERT INTO author VALUES  (11,'narsinh','mehta','talaja'),  (12,'premannad','bhatt','vadodra'),  (13,'javerchand','meghani','botad');    UPDATE author SET LAST\_name='jayant meghani' WHERE AUTHOR\_id=13;    **Lab 5: Delete a book from the books table where the price is higher than $100.**  **ANS :-**  DELETE FROM books WHERE price>8500; |

|  |
| --- |
| **Lab 3: Update the year\_of\_publication of a book with a specific book\_id.**  **ANS :-**  UPDATE books SET YEAR\_of\_publication=1960 WHERE book\_id=101;  UPDATE books SET YEAR\_of\_publication=1956 WHERE book\_id=102;  UPDATE books SET YEAR\_of\_publication=1872 WHERE book\_id=103;  UPDATE books SET YEAR\_of\_publication=2001 WHERE book\_id=105;    **Lab 4: Increase the price of all books published before 2015 by 10%.**  **ANS :-**  UPDATE books SET price=price\*0.10+price WHERE YEAR\_of\_publication<2015;  UPDATE books SET price=price\*0.10+price WHERE YEAR\_of\_publication<2015;  UPDATE books SET price=price\*0.10+price WHERE YEAR\_of\_publication<2015;  UPDATE books SET price=price\*0.10+price WHERE YEAR\_of\_publication<2015; |
| **Lab 3: Remove all members who joined before 2020 from the members table.**  **ANS :-**  DELETE FROM members\_backup WHERE date\_of\_membership<2020-01-01;    **Lab 4: Delete all books that have a NULL value in the author column.**  **ANS :-**  DELETE FROM books WHERE AUTHOR is null; |
| **Lab 4: Write a query to retrieve all books with price between $50 and $100.**  **ANS :-**  SELECT \* FROM books WHERE price BETWEEN 4200 AND 8500;    **Lab 5: Retrieve the list of books sorted by author in ascending order and limit the results to the top 3 entries.**  **ANS :-**  SELECT \* FROM books ORDER BY AUTHOR LIMIT 3 ; |
| **Lab 3: Perform an INNER JOIN between books and authors tables to display the title of books and their respective authors' names.**  **ANS :-**  SELECT book.b\_id , book.b\_title , author1.A\_name FROM book INNER JOIN author1 ON book.b\_id = author1.b\_id;    **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.**  **ANS :-**  SELECT book.b\_id , book.b\_title , author1.A\_name FROM book FULL JOIN author1 ON book.b\_id = author1.b\_id; |
| **Lab 3: Group books by genre and display the total number of books in each genre.**  **ANS :-**  SELECT genre , COUNT(title) FROM books GROUP BY genre;    **Lab 4: Group members by the year they joined and find the number of members who joined each year.**  **ANS :-**  SELECT year , COUNT(member\_name) FROM members\_backup GROUP BY year; |
| **Lab 3: Write a stored procedure to retrieve all books by a particular author.**  **ANS :-**  DELIMITER $$  CREATE PROCEDURE pro\_1 (b\_id int , book\_name varchar(60) , author\_name varchar(60) , price int )  BEGIN  INSERT INTO books VALUES(b\_id , book\_name , author\_name , price);  END;  CALL pro\_1(1, 'Godan', 'Premchand', 3000);  CALL pro\_1(2, 'Rangbhoomi', 'Premchand', 3500);  CALL pro\_1(3, 'Madhushala', 'Harivansh Rai Bachchan', 2500);    **Lab 4: Write a stored procedure that takes book\_id as an argument and returns the price of the book.**  **ANS :-**  CREATE PROCEDURE GetBookPrice(IN book\_id INT, OUT book\_price DECIMAL(10, 2))  BEGIN  SELECT price INTO book\_price  FROM books  WHERE b\_id = book\_id;  END;  CALL GetBookPrice(1, @price);  SELECT @price; |
| **Lab 3: Create a view to show only the title, author, and price of books from the books table.**  **ANS :-**  CREATE VIEW v\_1 AS SELECT title , AUTHOR , price FROM books;  SELECT \* FROM v\_1;    **Lab 4: Create a view to display members who joined before 2020.**  **ANS :-**  CREATE VIEW v\_3 AS SELECT member\_name FROM members\_backup WHERE year<2020;  SELECT \* FROM v\_3; |
| **Lab 3: Create a trigger to automatically update the last\_modified timestamp of the books table whenever a record is updated.**  **ANS :-**  DELIMITER $$  CREATE TRIGGER t\_22 AFTER UPDATE ON book\_1 FOR EACH ROW  BEGIN  INSERT INTO author(id , b\_title , pub\_years , records)VALUES(new.book\_id ,new.book\_name , new.pub\_year , 'Record is Successfully update.');  END  UPDATE book\_1 SET pub\_year=1944 WHERE book\_id=101;    **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.**  **ANS :-**  DELIMITER $$  CREATE TRIGGER t\_22 AFTER INSERT ON book\_1 FOR EACH ROW  BEGIN  INSERT INTO author(id , b\_title , pub\_years , records)VALUES(new.book\_id ,new.book\_name , new.pub\_year , 'Record is Successfully inserted.');  END    DELIMITER $$  CREATE TRIGGER t\_24 AFTER DELETE ON book\_1 FOR EACH ROW  BEGIN  INSERT INTO author(id , b\_title , pub\_years , records)VALUES(old.book\_id ,old.book\_name , old.pub\_year , 'Record is Successfully deleted.');  END |