Setting Up a Free Online MySQL Database

First, let's set up our MySQL database that we'll connect to Appsmith.

1. **Register for an Account**:
    - Go to https://www.freemysqlhosting.net/
    - Scroll down to the **"Sign Up"** section.
    - Fill in the requested information:
        - **Username**
        - **Password**
        - **Database Name** (Optional, if not provided, it will be the same as the username)
    - Accept the terms and click **"Create Database"**.
    - You'll receive a confirmation email. Click on the link to confirm your registration.
2. **Note Database Credentials**:
    - **Hostname**: ….
    - **Port**: `3306`
    - **Database Name**: As provided during sign-up.
    - **Username**: As provided during sign-up.
    - **Password**: As provided during sign-up.
3. **Access phpMyAdmin**:
    - Go to phpMyAdmin.
    - Log in using your database credentials.
    - We'll use this interface to run SQL scripts and manage our database.

Designing the Database Schema

Next, we'll design the schema for our library management system.

Entities in Our System

1. **Books**
2. **Authors**

3. **Categories**
4. **Users** (optional, for extended functionalities)
5. **BorrowRecords** (optional, for tracking book loans)

For this tutorial, we'll focus on the core functionalities: managing books, authors, and categories.

SQL Script to Create Tables

```
sql
Copy
-- Create Authors Table
CREATE TABLE Authors (
    AuthorID INT AUTO_INCREMENT PRIMARY KEY,
    AuthorName VARCHAR(255) NOT NULL
);

-- Create Categories Table
CREATE TABLE Categories (
    CategoryID INT AUTO_INCREMENT PRIMARY KEY,
    CategoryName VARCHAR(255) NOT NULL
);

-- Create Books Table
CREATE TABLE Books (
    BookID INT AUTO_INCREMENT PRIMARY KEY,
    Title VARCHAR(255) NOT NULL,
    AuthorID INT NOT NULL,
    CategoryID INT NOT NULL,
    ISBN VARCHAR(20),
    Publisher VARCHAR(255),
    YearPublished YEAR,
    FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID),
    FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)
);
```

Copy and run this script in phpMyAdmin to create the necessary tables.

Sample Data Insertion (Optional)

To test our application, we can insert some sample data.

```
sql
Copy
-- Insert Sample Authors
INSERT INTO Authors (AuthorName) VALUES ('J.K. Rowling'), ('George R.R. Martin'),
('J.R.R. Tolkien');

-- Insert Sample Categories
INSERT INTO Categories (CategoryName) VALUES ('Fantasy'), ('Science Fiction'),
('Mystery');

-- Insert Sample Books
```

```sql
INSERT INTO Books (Title, AuthorID, CategoryID, ISBN, Publisher, YearPublished)
VALUES
('Harry Potter and the Sorcerer''s Stone', 1, 1, '978-0439708180', 'Scholastic',
1998),
('A Game of Thrones', 2, 1, '978-0553103540', 'Bantam', 1996),
('The Hobbit', 3, 1, '978-0618968633', 'Houghton Mifflin Harcourt', 1937);
```

Setting Up Appsmith

Now, let's set up our Appsmith application.

1. **Sign In to Appsmith**:
   - Visit Appsmith and sign in to your account.
2. **Create a New Application**:
   - Click on **"Create New"** on your dashboard.
   - Name your app **"Library Management System"**.

Connecting Appsmith to the MySQL Database

We'll connect Appsmith to our MySQL database.

1. **Add a New Datasource**:
   - In Appsmith, navigate to **Datasources** > **Create New** > **MySQL**.
2. **Configure the Connection**:
   - **Host Address**: `db4free.net`
   - **Port**: `3306`
   - **Database Name**: Your database name.
   - **Username**: Your username.
   - **Password**: Your password.
3. **Test and Save**:
   - Click **"Test"** to ensure the connection works.
   - Click **"Save"** to store the datasource.
   - Name the datasource `LibraryDB`.

Building the User Interface

Let's design the UI of our app using Appsmith's widgets.

1. Displaying Books

1. **Add a Table Widget**:
    - Drag and drop a **Table** onto the canvas.
    - Rename it to `BooksTable`.
2. **Create a Query to Fetch Books**:
    - Under **LibraryDB**, click on **"New Query"**.
    - Name it `getBooks`.
    - SQL Command:

      sql
      Copy
      ```
      SELECT
          Books.BookID,
          Books.Title,
          Authors.AuthorName,
          Categories.CategoryName,
          Books.ISBN,
          Books.Publisher,
          Books.YearPublished
      FROM Books
      JOIN Authors ON Books.AuthorID = Authors.AuthorID
      JOIN Categories ON Books.CategoryID = Categories.CategoryID;
      ```
    - Click **"Run"** to test the query.
3. **Bind Data to Table Widget**:
    - Set `BooksTable`'s **Table Data** to `{{getBooks.data}}`.
4. **Customize Table Columns**:
    - Adjust column names and visibility as needed.

2. Adding Search Functionality

1. **Add a Text Input Widget**:
    - Drag and drop a **Text Input** onto the canvas above the table.
    - Rename it to `SearchInput`.
    - Placeholder text: `"Search books..."`
2. **Modify the `getBooks` Query**:

- Update the SQL Command to include a WHERE clause:
  sql
  Copy
  ```
  SELECT
      Books.BookID,
      Books.Title,
      Authors.AuthorName,
      Categories.CategoryName,
      Books.ISBN,
      Books.Publisher,
      Books.YearPublished
  FROM Books
  JOIN Authors ON Books.AuthorID = Authors.AuthorID
  JOIN Categories ON Books.CategoryID = Categories.CategoryID
  WHERE Books.Title LIKE '%{{SearchInput.text}}%'
      OR Authors.AuthorName LIKE '%{{SearchInput.text}}%'
      OR Categories.CategoryName LIKE '%{{SearchInput.text}}%';
  ```
- Ensure **Prepared Statement** is enabled (to prevent SQL injection).

3. **Set Up Search Input Actions**:
   - In `SearchInput`, under **Events**, set **onTextChanged** to execute `getBooks`.

3. Adding New Books

1. **Add a Button Widget**:
   - Label it **"Add New Book"**.
   - Configure it to open a modal when clicked.
2. **Create a Modal Form**:
   - **Form Inputs**:
     - **Title** (Text Input)
     - **Author** (Dropdown populated from `Authors` table)
     - **Category** (Dropdown populated from `Categories` table)
     - **ISBN** (Text Input)
     - **Publisher** (Text Input)
     - **Year Published** (Text Input)
3. **Create Queries to Fetch Authors and Categories**:
   - **Get Authors Query** (`getAuthors`):
     sql
     Copy
     ```
     SELECT AuthorID, AuthorName FROM Authors;
     ```
   - **Get Categories Query** (`getCategories`):
     sql
     Copy

```
SELECT CategoryID, CategoryName FROM Categories;
```

- Use these queries to populate the dropdowns in the form.

4. **Create an 'Insert Book' Query**:
   - **Name**: `insertBook`
   - SQL Command:
     ```
     sql
     Copy
     INSERT INTO Books (Title, AuthorID, CategoryID, ISBN, Publisher,
     YearPublished)
     VALUES (
       '{{TitleInput.text}}',
       '{{AuthorDropdown.selectedOptionValue}}',
       '{{CategoryDropdown.selectedOptionValue}}',
       '{{ISBNInput.text}}',
       '{{PublisherInput.text}}',
       '{{YearPublishedInput.text}}'
     );
     ```
   - Ensure **Prepared Statement** is enabled.
5. **Configure the Form Submission**:
   - On form submission, execute `insertBook`, then `getBooks`, and close the modal.

4. Editing and Deleting Books

1. **Enable Row Selection on BooksTable**:
   - Ensure that rows can be selected to edit or delete.
2. **Add "Edit" and "Delete" Actions in Table**:
   - In `BooksTable`'s **Column Settings**, add two new columns:
     - **Edit**: An **Edit** button that opens an edit modal.
     - **Delete**: A **Delete** button that triggers deletion.
3. **Create an 'Update Book' Query**:
   - **Name**: `updateBook`
   - SQL Command:
     ```
     sql
     Copy
     UPDATE Books
     SET
       Title = '{{EditTitleInput.text}}',
       AuthorID = '{{EditAuthorDropdown.selectedOptionValue}}',
       CategoryID = '{{EditCategoryDropdown.selectedOptionValue}}',
       ISBN = '{{EditISBNInput.text}}',
     ```

```
  Publisher = '{{EditPublisherInput.text}}',
  YearPublished = '{{EditYearPublishedInput.text}}'
WHERE BookID = {{BooksTable.selectedRow.BookID}};
```

4. **Configure the Edit Modal**:
   - Pre-populate form fields with `{{BooksTable.selectedRow.ColumnName}}`.
   - On submission, execute `updateBook`, then `getBooks`, and close the modal.

5. **Create a 'Delete Book' Query**:
   - **Name**: `deleteBook`
   - SQL Command:
     sql
     Copy
     ```
     DELETE FROM Books WHERE BookID = {{BooksTable.selectedRow.BookID}};
     ```
   - On delete button click, confirm deletion, execute `deleteBook`, then `getBooks`.