<u>NodeJS SQL Practice session</u>

Imagine you are developing a web application that manages a library's book inventory. The application has a database with two tables: **books** and **authors**. The **books** table has columns like **id**, **title**, **author_id**, **published_year**, and **quantity_available**. The **authors** table has columns such as **author_id**, **author_name**, and **author_bio**.

Your task is to create a Node.js server that interacts with the database using SQL queries to perform the following operations:

1. Retrieve the list of all books in the inventory along with their authors' names.
2. Add a new book to the inventory, ensuring that the author already exists in the **authors** table. If not, add the author first.
3. Update the quantity_available for a specific book.
4. Delete a book from the inventory based on its ID.

How would you structure the SQL queries in your Node.js application to achieve these operations efficiently?

SOLUTION

1. **Retrieve the list of all books in the inventory along with their authors' names:**

```
SELECT books.id, books.title, authors.author_name, books.published_year, books.quantity_available

FROM books

JOIN authors ON books.author_id = authors.author_id;
```

2. **Add a new book to the inventory, ensuring that the author already exists in the authors table. If not, add the author first:**

First, check if the author exists, and if not, add the author:

```
INSERT INTO authors (author_name, author_bio)

VALUES ('New Author Name', 'Author Bio')

ON DUPLICATE KEY UPDATE author_id = LAST_INSERT_ID(author_id);
```

Then, add the new book:

```
INSERT INTO books (title, author_id, published_year, quantity_available)

VALUES ('New Book Title', LAST_INSERT_ID(), 2023, 10);
```

3. **Update the quantity_available for a specific book:**

```
UPDATE books

SET quantity_available = 15

WHERE id = 1; -- Replace 1 with the specific book ID you want to update
```

4. **Delete a book from the inventory based on its ID:**

```
DELETE FROM books

WHERE id = 1; -- Replace 1 with the specific book ID you want to delete
```

<u>NodeJS SQL Practice session</u>

You are developing a blog application where users can create, view, and interact with blog posts. The application has a database with three tables: users, posts, and comments. The users table has columns like user_id, username, and email. The posts table has columns such as post_id, title, content, user_id, and created_at. The comments table has columns like comment_id, post_id, user_id, and comment_text.

Your task is to create a Node.js server that interacts with the database using SQL queries to perform the following operations:

1. Retrieve the latest 5 blog posts, including the username of the user who created each post.
2. Insert a new blog post into the database.
3. Retrieve all comments for a specific blog post, including the username of the user who posted each comment.
4. Delete a comment from the database based on its ID.

How would you structure the SQL queries in your Node.js application to achieve these operations efficiently?

SOLUTION

1. **Retrieve the latest 5 blog posts, including the username of the user who created each post:**

```
SELECT posts.post_id, posts.title, posts.content, users.username AS author, posts.created_at

FROM posts

JOIN users ON posts.user_id = users.user_id

ORDER BY posts.created_at DESC

LIMIT 5;
```

2. **Insert a new blog post into the database:**

```
INSERT INTO posts (title, content, user_id, created_at)

VALUES ('New Post Title', 'New Post Content', 1, NOW());

-- Replace 1 with the user_id of the user creating the post
```

3. **Retrieve all comments for a specific blog post, including the username of the user who posted each comment:**

```
SELECT comments.comment_id, comments.comment_text, users.username AS commenter

FROM comments

JOIN users ON comments.user_id = users.user_id

WHERE comments.post_id = 1;

-- Replace 1 with the post_id of the specific blog post
```

4. **Delete a comment from the database based on its ID:**

```
DELETE FROM comments

WHERE comment_id = 1;

-- Replace 1 with the comment_id of the specific comment to delete
```