Task 2: Analyzing Database Performance for Food-order Project and Applying Enhancement to It

Introduction

The `food-order` project is a web-based application designed to manage food orders for a restaurant. The system includes functionalities for administrators to manage categories, food items, and customer orders. The database is structured to handle various types of data related to these functionalities, ensuring efficient storage and retrieval of information. Also, it has the home page for users orders.

Database Analysis Report

Overview of the Database

The `food-order` database consists of the following tables:

1. `tbl_admin` - Stores admin user information.
2. `tbl_category` - Contains food category data.
3. `tbl_food` - Holds information about food items.
4. `tbl_order` - Records customer orders.

Analyzing Slow Queries

After reviewing the database structure and potential queries, here are some areas where performance issues might arise:

1. **Full Table Scans in Large Tables**

- Queries that do not utilize indexes effectively can result in full table scans, significantly slowing down performance as the number of records grows.

2. **Joins on Non-Indexed Columns**
    - Performing joins on columns that are not indexed can lead to slow query performance.

Suggestions and Enhancements

1. **Indexing**
    - Ensure appropriate indexing on columns frequently used in `WHERE`, `JOIN`, and `ORDER BY` clauses.
2. **Query Optimization**
    - Optimize queries to minimize the data scanned and retrieved.
3. **Caching**
    - Implement caching strategies for frequently accessed data to reduce database load.

Enhanced Code

1. **Indexing**

Adding indexes to frequently queried columns can significantly improve query performance.

```
-- Adding an index to the `username` column in `tbl_admin`
CREATE INDEX idx_username ON tbl_admin (username);

-- Adding indexes to the `category_id` and `price` columns in `tbl_food`
CREATE INDEX idx_category_id ON tbl_food (category_id);
```

```
CREATE INDEX idx_price ON tbl_food (price);

-- Adding indexes to the `customer_email` and `order_date` columns in `tbl_ord
er`
CREATE INDEX idx_customer_email ON tbl_order (customer_email);
CREATE INDEX idx_order_date ON tbl_order (order_date);
```

2. **Optimized Queries**

   Below are some examples of optimized queries for common operations:

- **Fetching Orders by Customer Email**

  Before:

```
SELECT * FROM tbl_order WHERE customer_email = 'example@example.com';
```

  Enhanced:

```
SELECT * FROM tbl_order WHERE customer_email = 'example@example.com' AND order
_date > '2024-01-01';
```

- **Retrieving Food Items by Category**

  Before:

```
SELECT * FROM tbl_food WHERE category_id = 1;
```

  Enhanced:

```
SELECT id, title, price FROM tbl_food WHERE category_id = 1;
```

3. **Query Execution Plan**

   Using the `EXPLAIN` statement can help analyze and optimize queries further.

```
EXPLAIN SELECT * FROM tbl_order WHERE customer_email = 'example@example.com';
```

This will provide insights into how the query is executed, allowing further optimization.

How to open the project:

1. Open xampp and start apache and MySQL
2. Open MySQL admin dashboard
3. Create new databases named food_order and food_order_enhanced
4. Copy the project folder to xampp/htdocs
5. Open the project using VSC
6. Go to the browser and open the projects file: localhost/food-order
7. Brows the home page and ordering page
8. Go to the admin page: localhost/food-order/admin
9. Enter admin as username and password
10. Try to use the CRUD operations in all pages

Conclusion

By implementing these enhancements, the `food-order` database can handle larger volumes of data more efficiently, resulting in faster query performance and an overall improved user experience. Regular monitoring and optimization of queries are essential to maintain optimal performance as the database grows.