**Online Food Ordering Database – Final Report**

Darrell Douglas

CPT 310 Database Systems & Management

Professor: Lisa Sims

February 9, 2025

This final report brings together all the elements of the Online Food Ordering Database project, highlighting its implementation, challenges faced, and key learnings gained throughout the journey. From creating normalized tables to adding triggers for automated auditing, this project has been a hands-on deep dive into database design and management. Let’s break it all down.

## **Implementation**

### Tables and Relationships

The database comprises six main tables:

1. **tbl\_customer**: Captures customer details such as first\_name, last\_name, email, phone, and created\_at. The primary key is customer\_id.
2. **tbl\_order**: Tracks customer orders, linked to tbl\_customer through a foreign key (customer\_id).
3. **tbl\_order\_item**: Logs items within orders, connecting tbl\_order and tbl\_product with foreign keys (order\_id and product\_id).
4. **tbl\_product**: Stores product details, including product\_name, price, stock\_quantity, and created\_at.
5. **tbl\_employee**: Tracks employee details with fields like first\_name, last\_name, position, and created\_at.
6. **tbl\_employee\_audit**: Logs actions like adding or deleting employees, with fields for id and audit\_data.

Relationships:

* **One-to-Many**: Between tbl\_customer and tbl\_order, tbl\_order and tbl\_order\_item, and tbl\_product and tbl\_order\_item.
* **Audit Trails**: Triggers connect tbl\_employee to tbl\_employee\_audit.

### SQL Scripts

1. **CreateDB.sql**:
   * Contains all CREATE TABLE statements with primary and foreign key constraints.
   * Defines triggers for automating logs in tbl\_employee\_audit.
2. **PopulateData.sql**:
   * Inserts realistic sample data into each table, including customer details, product information, and employee data.
   * Populates tbl\_employee\_audit via triggers during test operations.

### EER Diagram

The EER diagram visually demonstrates the relationships among all tables and includes triggers documented as part of tbl\_employee. This diagram serves as the database blueprint, ensuring clarity and consistency.

## **Challenges and Solutions**

### Challenge 1: Ensuring Data Integrity

**Issue**: Balancing normalization and performance without compromising data integrity. **Solution**: Proper use of primary and foreign keys alongside data constraints (e.g., NOT NULL, UNIQUE) prevented invalid entries.

### Challenge 2: Implementing Triggers

**Issue**: Developing effective triggers for automated auditing. **Solution**: Created AFTER INSERT and AFTER DELETE triggers for tbl\_employee, leveraging the CONCAT function to log detailed actions in tbl\_employee\_audit.

### Challenge 3: Building an EER Diagram in MySQL Workbench

**Issue**: Displaying triggers and maintaining connections between all tables. **Solution**: Used MySQL Workbench’s Diagram View to represent relationships and included trigger annotations for clarity.

## **Key Learnings**

### SQL Mastery

This project solidified my understanding of:

* **Data Definition Language (DDL)** for creating and altering database structures.
* **Data Manipulation Language (DML)** for inserting, updating, and deleting records.
* **Data Control Language (DCL)** for defining permissions, though not implemented here.

### Automation with Triggers

Triggers save time and ensure consistency. By automating audit trails, they reduce manual intervention while enforcing business rules.

### Practical Design

Normalization to 3NF and proper indexing enhance both database scalability and performance. Visualizing the design in an EER diagram provides a clear picture of table relationships and constraints.

**CreatedDB:**

-- Drop existing tables if they exist to avoid conflicts

DROP TABLE IF EXISTS tbl\_employee\_audit, tbl\_employee, tbl\_order\_item, tbl\_order, tbl\_product, tbl\_customer;

-- Create tbl\_customer

CREATE TABLE tbl\_customer (

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

email VARCHAR(100) NOT NULL UNIQUE,

phone VARCHAR(15) NOT NULL,

created\_at DATE NOT NULL

);

-- Create tbl\_product

CREATE TABLE tbl\_product (

product\_id INT AUTO\_INCREMENT PRIMARY KEY,

product\_name VARCHAR(100) NOT NULL,

price DECIMAL(10, 2) NOT NULL,

stock\_quantity INT NOT NULL,

created\_at DATE NOT NULL

);

-- Create tbl\_order

CREATE TABLE tbl\_order (

order\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_id INT NOT NULL,

order\_date DATE NOT NULL,

total\_amount DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (customer\_id) REFERENCES tbl\_customer(customer\_id)

);

-- Create tbl\_order\_item

CREATE TABLE tbl\_order\_item (

order\_item\_id INT AUTO\_INCREMENT PRIMARY KEY,

order\_id INT NOT NULL,

product\_id INT NOT NULL,

quantity INT NOT NULL,

price DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (order\_id) REFERENCES tbl\_order(order\_id),

FOREIGN KEY (product\_id) REFERENCES tbl\_product(product\_id)

);

-- Create tbl\_employee

CREATE TABLE tbl\_employee (

employee\_id INT AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

position VARCHAR(50) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Create tbl\_employee\_audit

CREATE TABLE tbl\_employee\_audit (

id INT AUTO\_INCREMENT PRIMARY KEY,

audit\_data VARCHAR(255) NOT NULL

);

-- Create Insert Trigger for tbl\_employee

DELIMITER //

CREATE TRIGGER trg\_employee\_insert

AFTER INSERT ON tbl\_employee

FOR EACH ROW

BEGIN

INSERT INTO tbl\_employee\_audit (audit\_data)

VALUES (CONCAT('New employee with ID = ', NEW.employee\_id, ' was added on ', NOW()));

END;

//

DELIMITER ;

-- Create Delete Trigger for tbl\_employee

DELIMITER //

CREATE TRIGGER trg\_employee\_delete

AFTER DELETE ON tbl\_employee

FOR EACH ROW

BEGIN

INSERT INTO tbl\_employee\_audit (audit\_data)

VALUES (CONCAT('Employee with ID = ', OLD.employee\_id, ' was deleted on ', NOW()));

END;

//

DELIMITER ;

**PopulateData:**

-- Insert data into tbl\_customer

INSERT INTO tbl\_customer (first\_name, last\_name, email, phone, created\_at)

VALUES

('John', 'Doe', 'johndoe@example.com', '1234567890', '2025-02-01'),

('Jane', 'Smith', 'janesmith@example.com', '0987654321', '2025-02-01');

-- Insert data into tbl\_product

INSERT INTO tbl\_product (product\_name, price, stock\_quantity, created\_at)

VALUES

('Protein Shake', 10.99, 50, '2025-02-01'),

('Yoga Mat', 20.00, 30, '2025-02-01'),

('Dumbbells Set', 50.00, 20, '2025-02-01'),

('Resistance Bands', 15.00, 40, '2025-02-01');

-- Insert data into tbl\_order

INSERT INTO tbl\_order (customer\_id, order\_date, total\_amount)

VALUES

(1, '2025-02-02', 35.99),

(2, '2025-02-02', 75.00);

-- Insert data into tbl\_order\_item

INSERT INTO tbl\_order\_item (order\_id, product\_id, quantity, price)

VALUES

(1, 1, 2, 21.98),

(1, 2, 1, 20.00),

(2, 3, 1, 50.00),

(2, 4, 2, 30.00);

-- Insert data into tbl\_employee

INSERT INTO tbl\_employee (first\_name, last\_name, position)

VALUES

('Mike', 'Thompson', 'Manager'),

('Lisa', 'Brown', 'Cashier');

-- Delete an employee to trigger the audit log

DELETE FROM tbl\_employee WHERE employee\_id = 1;

-- Verify the employee audit log

SELECT \* FROM tbl\_employee\_audit;

## **Conclusion**

The Online Food Ordering Database project combined technical depth with real-world applicability. By designing and implementing a structured, normalized database, I’ve honed skills in SQL scripting, data modeling, and relational database management. The hands-on experience of troubleshooting and optimizing processes has been invaluable, a solid foundation for future endeavors.

The journey wasn’t without its hiccups, but every challenge was a step toward mastery. This project stands as a testament to the power of structured data and the elegance of relational design.

**User Guide: Online Food Ordering Database**

**Introduction**

This guide provides step-by-step instructions on how to set up and populate the Online Food Ordering Database using MySQL and PHPMyAdmin. The guide will cover executing the provided SQL scripts to create database tables, relationships, constraints, and triggers, as well as inserting sample data into the tables.

**Prerequisites**

Before running the scripts, ensure that you have the following:

* A MySQL server installed (e.g., via XAMPP or MySQL Workbench).
* PHPMyAdmin or MySQL CLI access.
* The provided SQL scripts (CreateDB.sql and PopulateData.sql).

**Step 1: Creating the Database and Tables**

1. Open **PHPMyAdmin** or **MySQL Workbench**.
2. Create a new database:

CREATE DATABASE online\_food\_ordering;

1. Select the database:

USE online\_food\_ordering;

1. Open the **CreateDB.sql** script in your MySQL query editor.
2. Execute the script to generate all tables, relationships, constraints, and triggers.
3. Verify that tables have been created by running:

SHOW TABLES;

**Step 2: Populating Data**

1. Open the **PopulateData.sql** script in your MySQL query editor.
2. Execute the script to insert sample data.
3. Verify the inserted data using:
4. SELECT \* FROM tbl\_customer;
5. SELECT \* FROM tbl\_product;
6. SELECT \* FROM tbl\_order;
7. SELECT \* FROM tbl\_order\_item;
8. SELECT \* FROM tbl\_employee;

SELECT \* FROM tbl\_employee\_audit;

**Step 3: Testing Triggers**

To test the tbl\_employee\_audit triggers:

1. Insert a new employee:
2. INSERT INTO tbl\_employee (first\_name, last\_name, position)

VALUES ('John', 'Doe', 'Supervisor');

1. Check the audit log:

SELECT \* FROM tbl\_employee\_audit;

1. Delete an employee:

DELETE FROM tbl\_employee WHERE employee\_id = 1;

1. Verify the deletion has been logged:

SELECT \* FROM tbl\_employee\_audit;

**Screencast Video**

For a step-by-step demonstration of running these scripts, view the screencast video at the following link: <https://go.screenpal.com/watch/cTn6etnhlUN>

**Conclusion**

This guide provides a comprehensive overview of setting up and managing the Online Food Ordering Database. By following these steps, users can create, manage, and audit the database efficiently.

**References**

Coronel, C., & Morris, S. (2019). *Database systems: Design, implementation, and management* (13th ed.). Boston, MA: Cengage.