SQL Database Simulation for SME Company

Technical Stakeholder Report

Written by Tolulope Olarewaju

7th July 2025

Table of Contents

Executive Summary

Purpose: This project simulates a robust backend data system for a fictional small-to-medium enterprise (SME) using MySQL. It demonstrates the creation, manipulation, analysis, and automation of business data across customer, order, product, and employee domains.

Key Findings: Successfully demonstrated end-to-end SQL capabilities, including schema design, CRUD operations, joins, temporary tables, stored procedures, and analytical queries with window functions and CASE logic.

Recommendations: Extend this system to integrate with BI dashboards (e.g., Tableau), normalize product categories for better scalability, and apply additional stored procedures for real-time analytics and automation.

Introduction

Background: SMEs often lack advanced tools to manage, clean, and analyze operational data. This project simulates a full database pipeline from schema creation to advanced SQL analysis to support decision-making.

Objectives:

- Build a relational database for a fictional SME.
- Populate the database with mock transactional data.
- Perform data cleaning, transformation, and analysis.
- Demonstrate practical use of SQL joins, aggregations, window functions, and procedures.

Scope: The project is limited to backend database operations without frontend interface or live integration. Data is static and used for simulation only.

Data Sources

Data Description: Synthetic data was manually inserted into SQL tables. The schema includes:

- customers: Personal and contact information.

- orders: Order amount, date, and linkage to customers.
- products: Inventory with pricing and quantity.
- employees: HR data.
- employees_error: Messy data used for transformation demonstration.

Data Collection: All data was created and inserted using INSERT statements. Preprocessing was applied using SQL commands like TRIM(), REPLACE(), and LOWER().

Methodology

Analysis Techniques:

- Aggregations: GROUP BY, HAVING
- Filtering: WHERE, LIKE, IN, IS NULL
- Joins: INNER, LEFT, RIGHT, UNION, UNION ALL
- CASE logic and window functions for segmentation and scoring

Tools and Technologies:

- **MySQL:** Database management
- **SQL:** Querying, transformation, and analysis language

Modeling Approach: While no ML modeling was applied, this project used relational modeling principles with foreign key constraints, normalization, and entity relationships.

Findings

Descriptive Statistics:

- Customer order value distribution and averages
- Product inventory quantity by category
- Employee title frequencies
- Ranking orders by value and time
- Discounts calculated by tier using CASE

- Regex queries to filter products with advanced conditions

Detailed Analysis:

- 3 revenue segments: Big, Medium, Normal Customers
- Discounts are simulated dynamically by revenue threshold
- Joins across 3+ tables for relationship tracing (customer-product-order)
- Use of CTEs and window functions for trend and frequency tracking

---

Discussion

Interpretation of Results:

- The SMECompany database reflects real-world complexity.
- SQL tools like joins and CASE logic are powerful for data segmentation.
- Data cleaning is critical in preparing for accurate business analysis.

Limitations:

- All data is synthetic and lacks true behavioral variety.
- Some joins are simulated due to a fictional foreign key design.
- No frontend or BI layer integration at this stage.

Comparative Analysis:

- Compared to flat file or spreadsheet analysis, SQL enables much deeper querying, data cleaning, and auditability.

Recommendations

Actionable Steps:

- Add time-series tables for real order tracking and forecasting.
- Create views for repeated queries and performance optimization.
- Extend the system to include payment methods, supplier data, and logistics.

Implementation Considerations:

- May require indexing to improve performance as table size increases.
- Use stored procedures and scheduled events for automation and monitoring.

Conclusion

Summary of Key Points:

- Built a relational SQL database from scratch with practical table relationships
- Implemented advanced SQL operations including cleaning, CASE logic, joins, CTEs, and procedures
- Demonstrated analytical thinking in querying and interpretation

Future Work:

- Integrate this SQL backend with a BI dashboard using Tableau or Power BI
- Simulate a REST API connection for CRUD operations through a web application
- Apply this structure to real business data for consulting or freelancing work

---

Sample SQL Code Snippets

a) Categorizing Customers by Spending Tier

```
SELECT Order_id, Total_amount,
  CASE
    WHEN total_amount >= 6000 THEN 'Big Customer'
    WHEN total_amount BETWEEN 2000 AND 5999 THEN 'Medium Customer'
    ELSE 'Normal Customer'
  END AS customer_segment
FROM orders;
```

b) Joining Customers and Orders for Discount Calculation

```
SELECT sn, email, Order_date, Total_amount,
  CASE
    WHEN total_amount >= 8000 THEN total_amount * 0.90
    WHEN total_amount BETWEEN 4000 AND 7999 THEN total_amount * 0.92
    ELSE total_amount * 0.95
  END AS Discounted_Amount
FROM customers
JOIN orders ON customers.SN = orders.Customer_id;
```

c) Using Window Function to Count Repeating Order Values

```
SELECT Customer_id, Name, Order_date, Total_amount,
  COUNT(total_amount) OVER (PARTITION BY total_amount) AS Amount_Occurrence
FROM customers
JOIN orders ON customers.SN = orders.Customer_id;
```

d) Common Table Expression (CTE) for Reusability

```
WITH cte_customers AS (
  SELECT Customer_id, Name, Order_date, Total_amount,
    COUNT(total_amount) OVER (PARTITION BY total_amount) AS Amount
  FROM customers
  JOIN orders ON customers.SN = orders.Customer_id
)
SELECT Name, Total_amount FROM cte_customers;
```

e) Data Cleaning Using TRIM and REPLACE

```
SELECT employee_id, TRIM(employee_id) AS Cleaned_ID,
  REPLACE(Last_name, 'Fired- Man', 'Man') AS Cleaned_Lastname
FROM employees_error;
```