Development Individual Project:

Executive Summary

Retail Store – Database Design

Name: Naveed Jamal

Module: Deciphering Big Data

Date: January 19th, 2025

Table of Contents

[**Overview** 3](#_Toc188151069)

[**Summary of Work Carried Out** 3](#_Toc188151070)

[**Review of Database Modeling Concepts** 4](#_Toc188151071)

[**Outcomes of DBMS Analysis: SQL vs. No-SQL** 4](#_Toc188151072)

[**Compliance with Legal Standards** 5](#_Toc188151073)

[**Conclusions and Recommendations** 5](#_Toc188151074)

[**Recommendations** 5](#_Toc188151075)

[**References** 6](#_Toc188151076)

[Appendix 7](#_Toc188151077)

**List of Figures**

[Figure 1 Entity Relationship Diagram for Retail Store 7](#_Toc183435034)

[Figure 2 Explaining Data Types and Reason 8](#_Toc183435035)

[Figure 3 Code to Create Tables in SQL 10](#_Toc183435036)

**Overview**

For our project in Deciphering Big Data, our team chose a retail store as the client organization. This company seeks to migrate its local data to the cloud and implement a loyalty program to better understand customer behavior and improve customer retention. Retail data is inherently complex, encompassing customer demographics, store locations, sales records, supplier information, and more. Dhanushkodi et al. (2024) highlight that analyzing customer behavior plays a vital role in shaping effective business strategies and driving profitability in the ever-changing retail industry.

To address these needs, we propose designing a cloud-based database using Snowflake to handle large-scale data storage and analytics, while MySQL Workbench serves as the relational database management system for efficient backend processing. Snowflake is chosen for its flexibility, scalability, and ability to facilitate efficient data storage, processing, and analysis (Altexsoft, 2024). A key feature of this design is the loyalty program, which incentivizes customers by offering vouchers and discounts tailored to their spending patterns. Lacey and Sneath (2006) note that such programs are effective in encouraging customers to share their personal information, driven by the promise of exclusive benefits.

**Summary of Work Carried Out**

The development of our retail database involved several critical steps to ensure the design is both functional and aligned with the client’s business objectives. Initially, we conducted a detailed requirements analysis to identify the primary data sources, business processes, and analytical needs. The complexity of retail data, with its diverse elements such as customer demographics, sales data, and inventory records, necessitated a robust and scalable solution.

Our proposed solution integrates two key technologies: MySQL Workbench as the relational database management system (RDBMS) and Snowflake as the cloud-based data warehouse. This hybrid approach combines the structured, transactional capabilities of MySQL with the analytical power and scalability of Snowflake. The ETL (Extract, Transform, Load) pipeline serves as the bridge between these systems, ensuring seamless data transfer and transformation. This architecture allows the retailer to maintain operational efficiency while leveraging advanced analytics for strategic decision-making.

The loyalty program, a centerpiece of the design, was meticulously developed to enhance customer retention. Customers enroll by providing personal details such as name, email, and date of birth, either in-store or online. This information is stored in the MySQL database and then transferred to Snowflake for analysis. Rewards, tailored to individual spending patterns and demographics, are automatically generated and communicated to customers through email. To ensure data quality, duplicate accounts are flagged and removed, and anomalies like extreme outlier spending are excluded from analysis. Security measures, including encryption and role-based access control, are implemented to safeguard sensitive customer information.

To support these processes, we designed a structured and normalized relational database model, adhering to third normal form (3NF). This ensures data consistency and minimizes redundancy. Key tables include Customer, Order, Store, OrderItems, Employee, Product, and Supplier. Each table is interconnected through carefully designed relationships, enabling comprehensive data analysis and reporting.

**Review of Database Modeling Concepts**

Database modeling forms the foundation of any data management system, and the choice of data model significantly impacts the system's functionality and scalability. For this project, we adopted a relational data model, implemented using MySQL Workbench, due to its proven ability to handle structured, transactional data efficiently. The relational model’s strengths lie in its simplicity, support for ACID (Atomicity, Consistency, Isolation, Durability) properties, and widespread adoption across industries.

One of the primary advantages of the relational model is its ability to ensure data consistency through normalization. By organizing data into related tables and enforcing relationships through primary and foreign keys, the design eliminates redundancy and ensures data integrity. For example, in our design, the Customer table connects to the Order table via CustomerID, creating a one-to-many relationship. This structure allows us to track customer purchases while avoiding data duplication.

However, the relational model is not without its limitations. Its rigid schema can make it challenging to accommodate unstructured or semi-structured data, such as customer feedback or social media interactions. Additionally, the reliance on JOIN operations to retrieve data from multiple tables can lead to performance issues as the dataset grows. To address these challenges, we incorporated Snowflake as a complementary solution. Snowflake’s support for semi-structured data formats like JSON and its dynamic scalability make it an ideal choice for handling large-scale analytical workloads.

**Outcomes of DBMS Analysis: SQL vs. No-SQL**

The choice between SQL and No-SQL databases is a critical decision in database design. SQL databases, like MySQL, are well-suited for structured, transactional data, offering robust querying capabilities and adherence to ACID properties. In contrast, No-SQL databases, such as MongoDB, excel in handling unstructured or semi-structured data and provide flexibility in schema design.

For our project, we selected MySQL as the primary DBMS due to its compatibility with the relational data model and its ability to handle transactional data efficiently. MySQL’s support for complex queries and relationships made it the ideal choice for managing structured data like customer profiles, sales records, and inventory.

Snowflake, a cloud-based data warehouse, was chosen to complement MySQL. Unlike traditional SQL databases, Snowflake provides elasticity and scalability, enabling the system to handle large-scale analytical workloads without compromising performance. Its ability to integrate structured and semi-structured data makes it a valuable addition to the architecture. By combining MySQL and Snowflake, we created a hybrid system that leverages the strengths of both SQL and No-SQL approaches, ensuring operational efficiency and analytical depth.

**Compliance with Legal Standards**

Meeting legal and compliance requirements is essential in database design, particularly for businesses handling sensitive customer data. The General Data Protection Regulation (GDPR) sets stringent guidelines for data protection, requiring organizations to ensure data privacy and security.

Our design incorporates several measures to comply with GDPR. Customer data is encrypted both in transit and at rest, safeguarding it from unauthorized access. Role-based access control (RBAC) ensures that employees can only access data relevant to their roles. For example, store employees can view customer information necessary for transactions but cannot access sensitive details like loyalty points or purchase history. Customers access their data through password-protected portals, ensuring privacy.

To address GDPR’s "right to be forgotten" provision, we implemented mechanisms for deleting customer data upon request. Additionally, accounts with prolonged inactivity are flagged for deletion. Regular audits ensure compliance with GDPR and other relevant regulations.

**Conclusions and Recommendations**

In conclusion, our project successfully addresses the retailer’s needs for scalable data management and customer retention. The combination of MySQL and Snowflake provides a robust foundation for handling both transactional and analytical workloads. The loyalty program, supported by a structured relational database, enhances customer engagement and drives retention by offering personalized rewards.

**Recommendations**

1. **Expand Data Sources**: Incorporate additional data sources, such as social media interactions and customer feedback, to gain deeper insights into customer behavior.
2. **Enhance Analytics**: Leverage Snowflake’s advanced analytical capabilities to implement predictive modeling and customer segmentation.
3. **Strengthen Security**: Invest in advanced threat detection and monitoring systems to further enhance data security.
4. **Optimize ETL Processes**: Streamline the ETL pipeline to improve data transfer efficiency between MySQL and Snowflake.
5. **Monitor Performance**: Regularly evaluate system performance and scalability to ensure it meets evolving business needs.

This design not only meets current business requirements but also provides a scalable framework for future growth. By integrating advanced technologies and adhering to best practices in database design, the retailer is well-positioned to achieve its objectives and maintain a competitive edge in the market.

**References**

Altexsoft. (2024) The Good and the Bad of Snowflake Data Warehouse. Available from: <https://www.altexsoft.com/blog/snowflake-data-warehouse-pros-cons/> [Accessed 24 November 2024].

Dhanushkodi, K., Bala, A., Kodipyaka, N. & Shreyas, V. (2024) Customer Behaviour Analysis and Predictive Modelling in Supermarket Retail: A Comprehensive Data Mining Approach. IEEE Access. DOI: 10.1109/ACCESS.2024.3407151

European Commission (2016) GDPR Regulation of the European Parliament and Council. Available at: <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679&qid=1713388284638> [Accessed 24 November 2024].

Heck, W. (2009) Using MySQL in Your Organisation. Available from: <https://library.iated.org/view/HECK2009USI> [Accessed 24 November 2024].

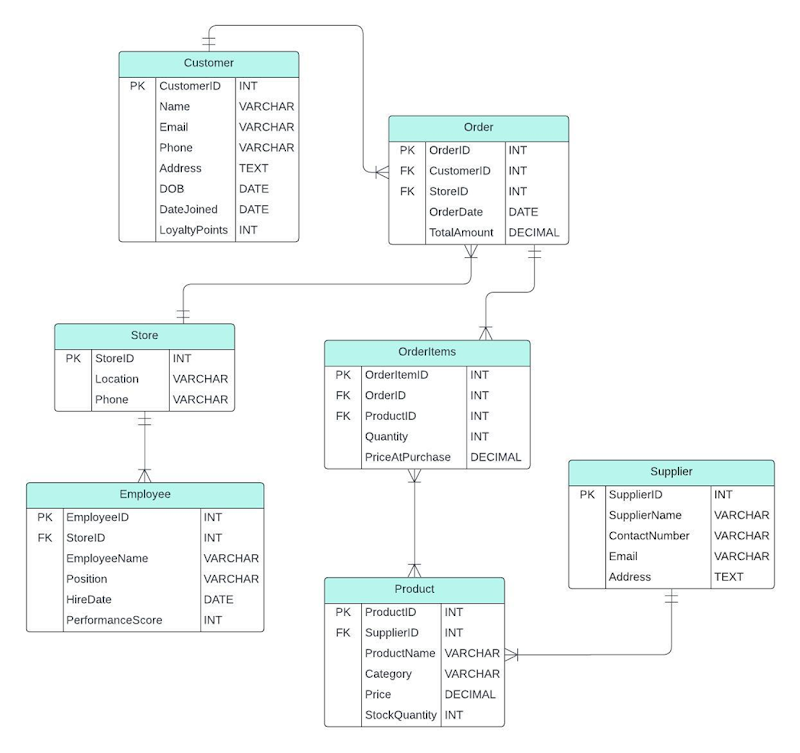
Lacey, R. & Sneath, J.Z. (2006) Customer Loyalty Programs: Are they Fair to Consumers? The Journal of Consumer Marketing 23(7): 458-464. DOI: 10.1108/07363760610713000

Sargo, M. (N.D.) Top 5 Benefits to Centralized Data. Available from: <https://www.dataideology.com/top-5-benefits-to-centralized-data/> [Accessed 24 November 2024].

Slingerland, C. (2024) Horizontal vs Vertical Scaling: Which Should You Choose? Available from: <https://www.cloudzero.com/blog/horizontal-vs-vertical-scaling/#:~:text=While%20horizontal%20scaling%20refers%20to,%2C%20storage%2C%20or%20network%20speed> [Accessed 25 November 2024].

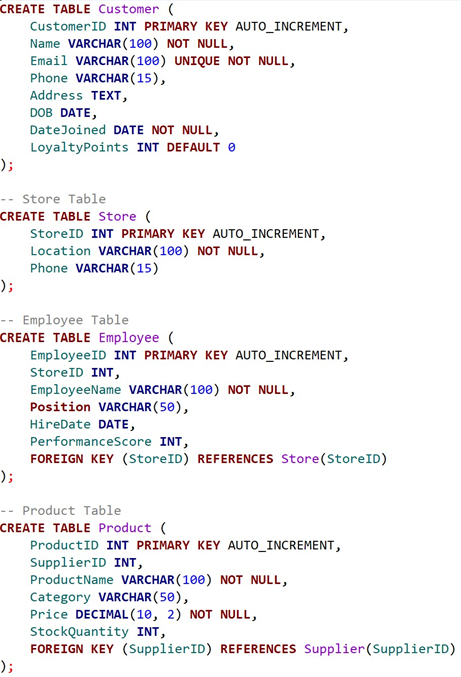
Appendix

Figure 1 Entity Relationship Diagram for Retail Store



|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Attribute** | **Data Type** | **Reason** |
| Customer | CustomerID | INT | A unique integer identifier for each customer, efficient for indexing and queries. |
| Name | VARCHAR | Flexible text format to store customer names of varying lengths (e.g., "John Doe"). |
| Email | VARCHAR | Emails are alphanumeric and variable-length; VARCHAR ensures storage efficiency. |
| Phone | VARCHAR | Phone numbers often include formatting characters (e.g., +1, -), making VARCHAR suitable. |
| Address | TEXT | Addresses can be lengthy and vary significantly, so TEXT provides sufficient flexibility. |
| DOB | DATE | A specific date format is needed to store the customer's birth date. |
| DateJoined | DATE | Captures the date the customer joined the loyalty program for tracking membership history. |
| LoyaltyPoints | INT | An integer value to store points earned by customers in the loyalty program. |
| Order | OrderID | INT | Unique identifier for each order, optimised for indexing and relational links. |
| CustomerID | INT | Foreign key linking to the Customer table. Matches the data type of CustomerID. |
| StoreID | INT | Foreign key linking to the Store table. Matches the data type of StoreID. |
| OrderDate | DATE | Tracks the specific date the order was placed. |
| TotalAmount | DECIMAL | Stores the total cost of the order, including decimals for accuracy in monetary values. |
| Store | StoreID | INT | Unique identifier for each store, suitable for indexing. |
| Location | VARCHAR | Text format to store store locations (e.g., city names or addresses). |
| Phone | VARCHAR | Allows storage of phone numbers with varying formats. |
| OrderItems | OrderItemID | INT | Unique identifier for each order item, optimised for indexing. |
| OrderID | INT | Foreign key linking to the Order table. Matches the data type of OrderID. |
| ProductID | INT | Foreign key linking to the Product table. Matches the data type of ProductID. |
| Quantity | INT | Integer to store the number of items purchased. |
| PriceAtPurchase | DECIMAL | Captures the price of the product at the time of purchase, with decimal precision. |
| Employee | EmployeeID | INT | Unique identifier for each employee, efficient for indexing and queries. |
| StoreID | INT | Foreign key linking to the Store table. Matches the data type of StoreID. |
| EmployeeName | VARCHAR | Stores employee names of varying lengths. |
| Position | VARCHAR | Tracks employee job titles (e.g., "Manager"). |
| HireDate | DATE | Tracks when the employee was hired. |
| PerformanceScore | INT | Integer used for performance tracking or evaluations. |
| Product | ProductID | INT | Unique identifier for each product, optimised for indexing. |
| SupplierID | INT | Foreign key linking to the Supplier table. Matches the data type of SupplierID. |
| ProductName | VARCHAR | Stores product names, allowing flexibility for different lengths. |
| Category | VARCHAR | Tracks product categories (e.g., "Electronics"). |
| Price | DECIMAL | Stores product prices with decimal precision for accuracy. |
| StockQuantity | INT | Tracks the quantity of the product in stock. |
| Supplier | SupplierID | INT | Unique identifier for each supplier. |
| SupplierName | VARCHAR | Stores supplier names with varying lengths. |
| ContactNumber | VARCHAR | Allows flexibility for phone numbers with varying formats. |
| Email | VARCHAR | Suitable for storing email addresses. |
| Address | TEXT | Provides flexibility for lengthy addresses. |

Figure 2 Explaining Data Types and Reason



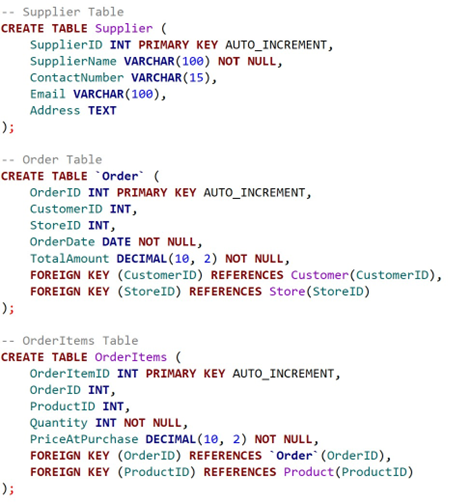


Figure 3 Code to Create Tables in SQL