Milestone Four: Databases

Robert Heavner

SNHU

CS-499

Professor Bryant

03/31/2025

Artifact Overview

A fully operational PostgreSQL database intended to handle product inventories, customer information, sales, and refunds serves as the artifact for this milestone. This solution was developed to use a normalized, query-optimized relational database schema in place of static or spreadsheet-based data tracking. To facilitate effective sales and return tracking and reporting, the solution consists of tables, foreign key connections, views, materialized views, indexes, and stored procedures.

This artifact started off as a fundamental schema for another class, but it was much improved to conform to professional database standards, such as improved indexing, validation requirements, and business logic contained in triggers and stored procedures.

Justification for Inclusion in ePortfolio

This artifact was chosen because it showcases:

- Advanced relational schema design with referential integrity
- Complex query handling via stored functions
- Real-world reporting solutions with views and materialized views
- Performance optimization via indexing
- Trigger usage for data auditing/logging

Key areas of strength demonstrated:

- Ability to design a normalized schema
- Creating and testing stored functions to calculate return rates and state-level reporting
- Implementing data integrity constraints (i.e. CHECK, UNIQUE, ON DELETE CASCADE)
- SQL views for simplified business reporting
- Using materialized views for snapshot reporting
- Trigger creation to automatically log events, showcasing understanding of audit requirements in enterprise systems.

This artifact demonstrates the software engineering and database design skills required in modern backend development, particularly in domains like sales tracking, business intelligence, and transactional systems.

Enhancements & Improvements

Enhancements made in this milestone include:

1. Referential Integrity

- Added FOREIGN KEY constraints to ensure product and customer data integrity in sales and returns tables.
- Used ON DELETE CASCADE to ensure cascading cleanup of child records.

2. Indexing for Performance

- Added indexes to frequently joined columns (product ID, customer ID, SKU, email).
- Ensured that lookups and joins in reporting queries are optimized.

3. Reporting Features

- Created get_return_rate() to calculate return percentages per SKU.
- Created get_returns_by_state() to provide insight into geographic trends.
- Designed a sales_summary view and return_summary view to present quick overviews.
- Built a mv_return_summary materialized view for performance-tuned snapshots.

4. Trigger & Logging

- Created a return_log table to track inserts into the returns table.
- Used a trigger (trg_log_return_insert) to automatically log any return record insert.

5. Validation & Constraints

- Ensured all quantity and refund values are positive with CHECK constraints.
- Added UNIQUE constraints to prevent duplicate sales records (same customer/product/date).

Alignment with Course Outcomes

This milestone showcases several core Computer Science program outcomes. The database schema, created with professional best practices, directly addresses the ability to design, develop, and deliver high quality technical solutions. The utilization of PostgreSQL, alongside stored procedures, triggers, and materialized views, demonstrates the practical application of various techniques. The evaluation of computing solutions is evident in the implementation of normalization, referential integrity, and the strategic consideration of trade-offs inherent in different view types. This shows my understanding of databases and how a great database can be used in backend development. The project also reinforces software engineering and design principles through schema validation, logging triggers, and the creation of reusable views and reports. This enhancement aligns seamlessly with the planned outcome coverage, effectively reinforcing the goals established in previous modules without necessitating any adjustments.

Reflection on Enhancement Process

This development process showcases significant learning experiences, involving the transformation of a schemas into a production ready database. A key takeaway was the critical role of type casting in PostgreSQL functions. The importance of structuring grouping

logic and JOINs became clear when designing practical, real-world reports. The implementation of ON DELETE CASCADE highlighted its ability to streamline long-term database maintenance. The use of null's proved invaluable for safely handling division within functions, preventing division-by-zero errors. The automation of insert action capture through triggers and logs was also a valuable lesson. Issues I had were issues with mismatches between BIGINT and INT in stored procedures, the prevention of division-by-zero errors in return rate calculations, and the decision between views and materialized views for performance optimization. Using pgAdmin was very helpful long the way for testing purposes to ensure everything works as it should. This script can be ran over and over again, which allows this script to be a great starting script for anyone looking to use this system as a simple database.

Summary

This artifact reflects my growth as a computer science student specializing in backend systems and data management. The final database solution:

- Uses PostgreSQL effectively for structured, scalable storage
- Automates reporting through functions and views
- Handles real-world logic like returns and logging
- Demonstrates clean schema design, constraint usage, and performance tuning

This project would be an excellent foundation for building a full-stack application or plugging into a business intelligence dashboard. It clearly showcases my skills in relational database design, query optimization, data integrity, and reporting logic core areas in modern software development.