Student's Full Name: Sundar Ram Subramanian

Course Number and Title: INFO531 – Data Warehousing and Analytics in Cloud

Term name and year: Fall 2025

Submission Week: Week 16 Project Report

Instructor's Name: Dr. Nayem Rahman, PhD

Date of Submission: 10-Dec-2025

Table of Contents

Table of Figures

Project Report

Executive Summary

This project presents the design and development of a comprehensive Data Management and Analytics System powered by MySQL and Python Streamlit and built on the Northwind dataset, that has been modified (refer Appendix section for links to the datasets) to suit academic nature of the project. The primary objective of the project was to transform a traditional transactional database into a fully interactive platform capable of performing seamless data management operations (CRUD) and providing actionable business insights (Analytics). The system integrates a well-structured backend, a robust relational database, and an intuitive front-end interface to support a complete analytical workflow, including data ingestion, validation, CRUD operations, and visualization.

Throughout the development cycle, the system evolved through structured phases guided by milestone updates. The final product includes a refined database schema with enforced relational integrity, an automated validation and update engine, and a multi-layer Streamlit UI that facilitates interaction with the data in a seamless and user-friendly manner. The analytics dashboard provides meaningful visualizations of sales performance, customer trends**,** and product movement, enabling decision-makers to gain a deeper understanding of business operations.

Introduction

For this project, I selected option 1, that proposes the development of an interactive, web-based user interface. The purpose of this project is to develop a modular and scalable prototype of data management platform that allows users to interact with a real-world relational dataset in a controlled, validated, and visually interpretable environment. The project uses the Northwind dataset, a classic dataset representing an order management workflow comprising customers, suppliers, products, categories, orders, and employees that has been modified to suit the syntax and requirements of MySQL server.

The system supports complete CRUD operations, and a fully functional analytics dashboard. The project draws inspiration from enterprise-level data warehousing workflows in which operational and analytical components are interconnected. The system was built using Python Streamlit to create a unified web interface, MySQL to manage the relational database, and custom defined helper modules to handle data validation and transformation. Then entire code repository is included in the appendix section of the document.

System Architecture

The system is built on a three-layer architecture that separates data storage, backend processing, and user interaction. This separation ensures modularity, scalability, and ease of maintenance. The layers are as below

- **Database laye**r - integrating MySQL for data storage
- **Backend layer** - Python for backend logic
- **UI Layer** - Streamlit for the user interface

Database layer

The database layer forms the foundation of the system, consisting of a fully normalized MySQL relational structure. This layer stores all master and transactional entities such as Customers, Categories, Products, Employees, Orders, and OrderDetails. Referential integrity is enforced using primary keys, foreign keys, and CHECK constraints that ensure valid data entry and prevent anomalies. The database was created through a structured SQL script (refer to database creation code in appendix section) that defines table structures, field types, relational dependencies, and indexes to optimize performance. This layer handles all persistent data operations and forms the backbone for both CRUD functionalities and analytical operations.

Backend layer

The backend layer consists of Python helper modules (refer appendix section for code) that manage communication between the UI and the database. It includes functions for establishing *database connections using mysql-connector-python library, retrieving reference*

values, performing data validation, executing SQL queries, and orchestrating CRUD operations. The helper functions were designed to be reusable and modular, enabling clean integration with the UI. This layer also contains the logic for bulk data updates, validation rules that compare uploaded CSV values with database records, and analytics queries that compute sales metrics, product performance, and customer trends. The backend ensures that user interactions translate into safe and accurate operations in the database.

Frontend layer

The front-end layer is built using Streamlit and provides an accessible and intuitive interface for users to interact with the system. The UI is divided into three main pages: the Home Page, the Data Management CRUD Page, and the Analytics Dashboard. The interface module organizes navigation, forms, file uploaders, tables, editors, and instruction popovers to guide the user across the system. Users can perform operations such as creating orders, editing product details, uploading datasets for mass updates, and visualizing business trends through charts and metrics (matplotlib, plotly & altair library) . The UI layer incorporates appropriate error handling, data previews, and confirmation messages, ensuring smooth interactivity and usability.

Dataset Preparation

Dataset preparation was a critical component of the project, forming the basis of reliable data operations and accurate analytics.

Dataset Source:

The dataset used in this project is the modified Northwind dataset, representing a multinational food distribution business. It captures key business entities such as customers, employees, suppliers, shippers, products, categories, orders & order details. The dataset is typically used for demonstrating relational modelling, querying, and analytical capabilities. The original dataset was first reviewed during Update 1, where the project scope and requirements were defined based on the nature and structure of the modified tables. The modifications in

table include adding CreatedDate & UpdatedDate Columns and removing columns that stored Photo blob data. Later, during Update 2, significant preparation work was conducted to clean and optimize the dataset for modern SQL usage and UI integration. The link to modified Dataset creation code is included in the appendix section of the report.

Dataset Explanation

The Northwind dataset is rich in structure and comprises both master data and transactional data. Master entities include Customers, Suppliers, Employees, Categories, and Shippers, while Products and Orders represent core business processes. OrderDetails serves as a bridge table linking Orders and Products, enabling analysis at a granular item-level. During preparation, unused fields such as the Picture column in Categories were dropped to simplify the schema. Various date fields were converted to consistent formats, and textual inconsistencies were addressed. The cleaned dataset was then validated to ensure that it conformed to relational constraints and that no orphan or inconsistent records existed.

The dataset was curated to support both CRUD and analytical operations. It now enables efficient retrieval, update, and summarization of business activities, aligning with the reporting and visualization goals of the project.

Database Design

The database design reflects careful normalization and relationship modelling across all entities. The creation script constructs tables with explicit primary and foreign key relationships to ensure data integrity. The structure allows seamless joins across Customers, Orders, Products, and Employees. Field-level constraints ensure logical correctness; for example, UnitPrice and Quantity fields follow CHECK rules to enforce non-negative values, while ReorderLevel supports inventory modelling.

Indexes were added to frequently accessed columns such as ProductName, CategoryName, CustomerID, and OrderDate to improve performance. This database design enables consistent CRUD operations and forms the analytical backbone for visualizations. The

schema is optimized for referential integrity and supports error-free insertions and updates through the UI.

The Underlying Model of the Data

The Entity-Relationship Diagram represents the structure, dependencies, and cardinalities between entities.

Logical Database Model:

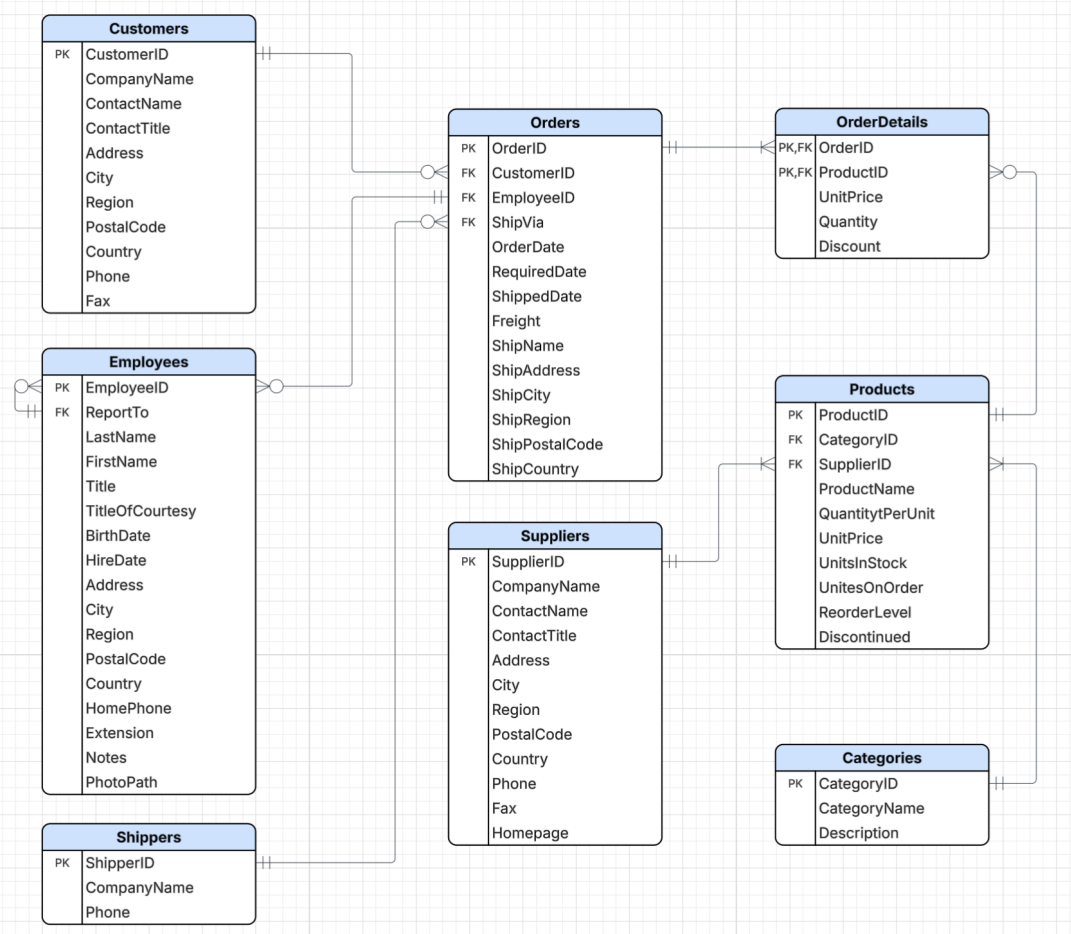

Figure 1 Logical Data Model

Categories and Suppliers maintain one-to-many relationships with Products, Customers maintain a one-to-many relationship with Orders, and Orders maintain a one-to-many relationship with OrderDetails. OrderDetails acts as a junction table that establishes a many-to-many relationship between Orders and Products, capturing the line-item details within each transaction. Shippers and Employees also have relational dependencies with Orders, reflecting real-world business processes.

Physical Database Model:

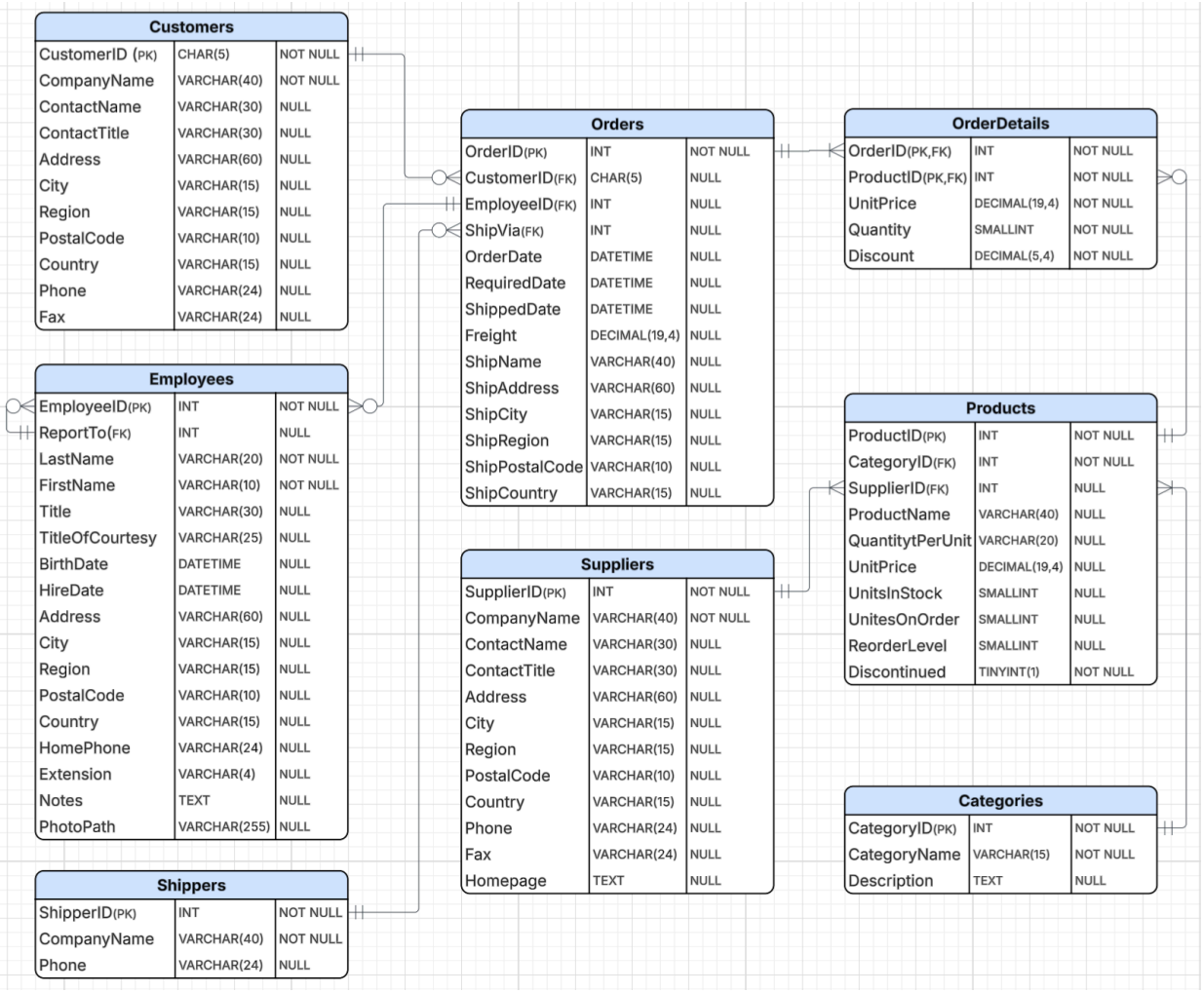

Figure 2 Physical Data Model

The ERD highlights the relational interactions essential for CRUD operations and analytical computations. For example, product sales trends rely on the linkage between Orders, OrderDetails, and Products, while customer segmentation is derived from the Customers and Orders tables. The ERD thus supports the design principles of normalization, data integrity, and analytical utility.

Methodology

The project followed an iterative, milestone-driven methodology that ensured systematic progression from concept to implementation. Following are the different phases of the project.

- Phase 1 – Problem Understanding and Initial Setup
- Phase 2 – Data Preparation & System Architecture
- Phase 3 – Final Build-Out

Phase 1 – Problem Understanding and Initial Setup:

The Phase 1 of the project involved Database collection, reviewing the Northwind dataset for clearly defining & ensuring the problem statement and its scope. The primary objective was to develop a unified system capable of data management and analytics built over a database. Initial analysis identified the necessary entities, relationships, and functionalities required for CRUD operations and the analytics dashboard. The project setup included configuring the MySQL environment, establishing initial table designs, and outlining the system architecture and navigation flow.

Phase 2 – Data Preparation & System Architecture:

Phase 2 focused on preparing the dataset for integration into the system. Data Cleaning, Data transformation, ensuring key constraints key constraints were appropriately applied, and relational model validation in the backend were the main objectives. The system architecture was also formalized in this phase, dividing the solution into database, backend, and UI layers. The CRUD and Data Management framework was structured to support both individual record entry and batch updates. During this phase, the analytics requirements were identified, and the dashboard metrics were drafted. Popovers, UI structures, and helper module designs were developed to improve code reusability and maintainability.

Phase 3 – Final Build-Out:

The final phase involved implementing the remaining features and integrating all components into a cohesive application. CRUD operations were completed with robust error-handling, data validation, and update logic that tracked field-level changes. Several test files were designed and developed to ensure robustness across a variety of use cases that involved error handling and data entry. The analytics dashboard was developed with visual components that summarized sales, customer activity, and product performance. Final refinements included

helper-function consolidation, and preparation of technical documentation and reporting sections.

CRUD Operations & Data Validation

Create

Create operations involve form-based input fields with embedded validation that ensures data consistency and adherence to relational constraints. For instance, creating an OrderDetails record automatically retrieves the UnitPrice from the Products table, ensuring accuracy in line-item calculations.

Read

Read operations allow users to view datasets and preview uploaded CSV files before updates take place.

Update

Update operations represent the most advanced component of the CRUD system. The bulk update functionality compares uploaded CSV records with the existing database data. The system evaluates column mismatches, NOT NULL violations, and primary key inconsistencies. It then identifies field-level differences and updates only the modified values, preserving data accuracy while enhancing efficiency. This selective update mechanism prevents unnecessary database writes and improves system performance.

Delete

Delete operations were intentionally excluded to maintain referential integrity across transactional data, aligning with safe data-management practices common in enterprise systems. However, Delete is made available for mis-placed orders or order changes.

Data Validation for Create & Update

Data validation forms one of the most critical components of this project and is embedded throughout the system to ensure that every user interaction results in clean, accurate, and rule-compliant data being written to the database. The system performs validation at multiple levels

including UI-level validation, schema-level validation, database-level constraint checks, and change-detection logic in bulk-update workflows. The combination of these layers ensures that the integrity of the Northwind database is maintained at all times, regardless of how many records are inserted or updated.

The validation process begins at the user interface, where many of the input fields enforce type safety and constraint limits before any data reaches the backend. For example, when creating an OrderDetails record, the user is prompted to enter a ProductID. As soon as a ProductID is provided, the system automatically connects to the Products table and retrieves the corresponding UnitPrice, which is then displayed back to the user. This prevents users from manually entering a price that may not align with the company's product catalog. In addition, the user is not allowed to proceed unless the ProductID is valid. This ensures referential integrity long before the INSERT statement is executed. Similarly, fields such as Quantity and Discount enforce minimum and maximum values preventing invalid entries, such as negative quantities or discounts exceeding 100%.

Validation becomes even more structured in the bulk-update module, which is the most advanced part of your system's data governance logic. When a user uploads a CSV file for updating a table such as Categories, Employees, Customers, or Suppliers, the file undergoes rigorous verification before any updates are performed. The first stage checks whether the CSV includes all required columns. Using the table's metadata extracted from MySQL (retrieved using table_column_details), the system ensures that all expected fields are present. If any column is missing even if it's just one the system halts further processing and prompts the user to correct the input.

After column validation, the system checks for violations of NOT NULL constraints. Each field that is marked "NO" for nullability in the database schema is scanned across the entire CSV. If the uploaded file contains null values, NaN entries, or even empty strings in these mandatory

fields, the system raises an error. This prevents incomplete or malformed data from entering the database.

A third layer of validation ensures that every primary key value in the uploaded CSV exists in the corresponding base table of the database. This prevents accidental creation of orphan records or attempts to update non-existent rows. For example, if the user tries to update a CategoryID that does not exist in the Categories table, the system flags the issue and prevents execution. This check is performed by comparing the list of primary keys from the CSV with the existing primary keys extracted via a live SQL query.

Finally, the most sophisticated validation step occurs during the record update workflow, where the system performs row-by-row, column-by-column change detection. Once the CSV is verified to be structurally sound, the system extracts each row's primary key and fetches the corresponding row from the database. Every non-PK field is then compared between the CSV and database values. Differences are identified using a combination of string comparisons and NaN equivalence checks. Only the fields whose values actually differ are selected for update. This selective update mechanism is an important part of the system's validation strategy because it prevents unnecessary database writes, reduces performance overhead, and safeguards against overwriting unchanged data. Moreover, it ensures that the database always reflects the user's intended changes and nothing more.

Beyond the bulk workflow, validation is also supported by MySQL's own structural safeguards, such as CHECK constraints for fields like UnitPrice, Quantity, and Discount. These constraints prevent any out-of-range or logically incorrect values from being stored, even if they somehow bypass UI or backend checks. Similarly, foreign key constraints ensure that relationships between tables remain valid for example, that Orders cannot reference non-existent Customers, and OrderDetails cannot reference invalid ProductIDs or OrderIDs.

Together, these multiple layers UI validations, backend schema validations, PK existence checks, NOT NULL enforcement, type and constraint verification, change detection, and

database-level constraints form a comprehensive validation framework. This ensures that all data entering or being modified within the system adheres strictly to structural, relational, and logical rules. Such a layered approach not only maintains high data quality but also mirrors real-world enterprise-grade data governance standards, making the system robust, reliable, and production-ready.

Through-out the Create and Update operations, the CreatedDate and UpdatedDate columns are also automatically updated as applicable, ensuring tracking of changes.

Analytics Dashboard for Visualization & Insights

The analytics dashboard provides comprehensive business insights through a combination of charts, KPIs, and aggregated metrics. The dashboard offers an overview of total orders, total sales, number of customers, and inventory metrics. Users can view monthly revenue trends, product-wise performance charts, customer segmentation summaries, and top-selling product lists.

Each visualization was crafted to deliver intuitive insights. For example, the sales trend chart helps users understand seasonal fluctuations, while the category-based revenue distribution offers clarity on product line performance. These analytical insights complement the CRUD functionalities by helping users review operational efficiency and make informed decisions based on data patterns.

Conclusion

This project successfully delivered a comprehensive system that integrates data management and analytics within a unified platform. By leveraging MySQL for data storage, Python for backend logic, and Streamlit for the user interface, the system demonstrates a practical, real-world approach to managing business data. The solution supports accurate CRUD operations, reliable data validation, and insightful analytics, making it suitable for operational teams, analysts, and decision-makers.

The project offers a strong foundation for future enhancements such as user authentication, cloud deployment, machine learning-based forecasting, and extended reporting capabilities. Overall, the application showcases a successful blend of database design, and analytical visualization aligned with modern data warehousing practices.

Appendix

Hyperlinks to codes & dataset

- Original Northwind Dataset
- Modified Dataset used in this project – Refer to dataset_sample.xlsx in GitHub. You need to download the file to view the data.
- Database creation code
- Helpers module
- Code Repository and Technical documentation

Screenshots of the application

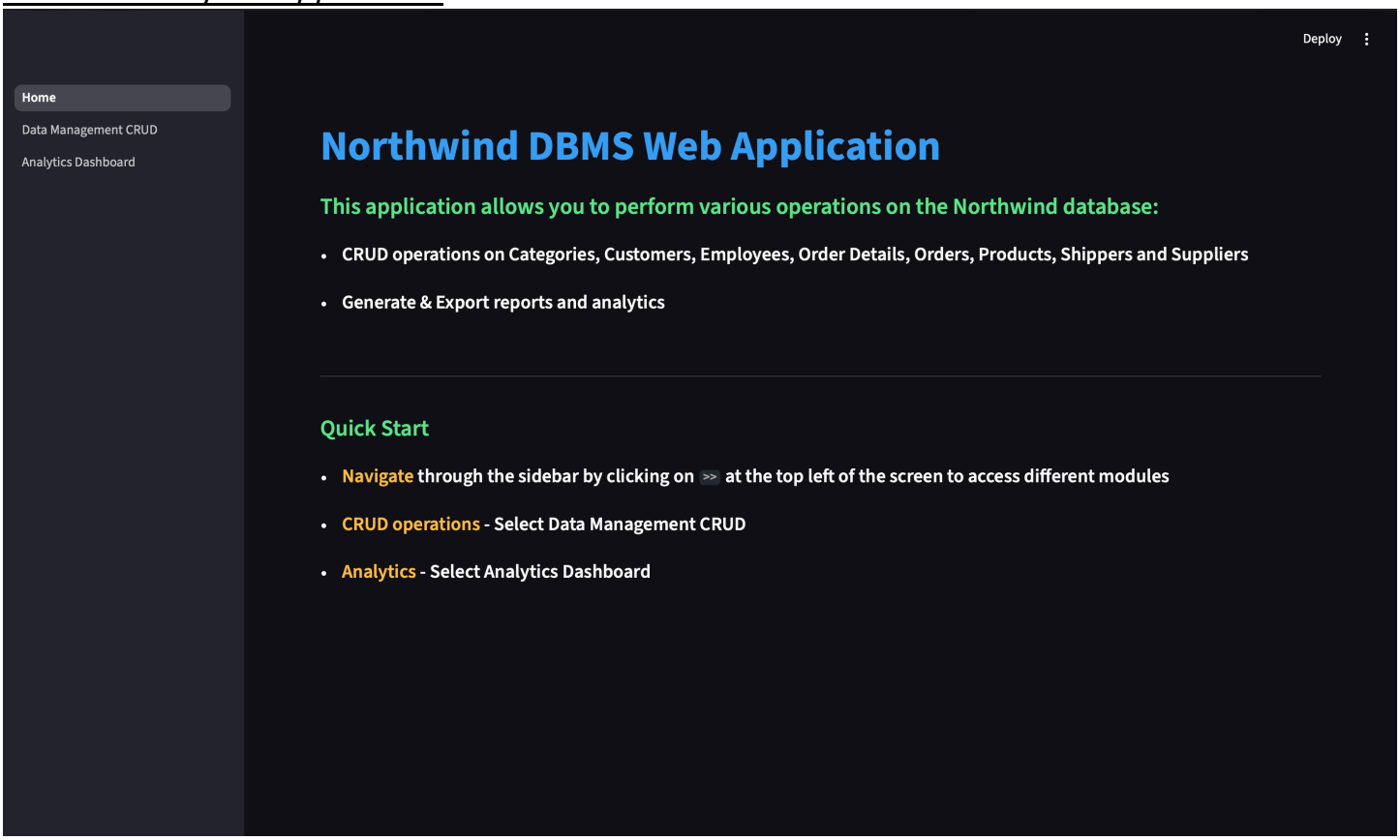

Figure 3 Home Page of the application

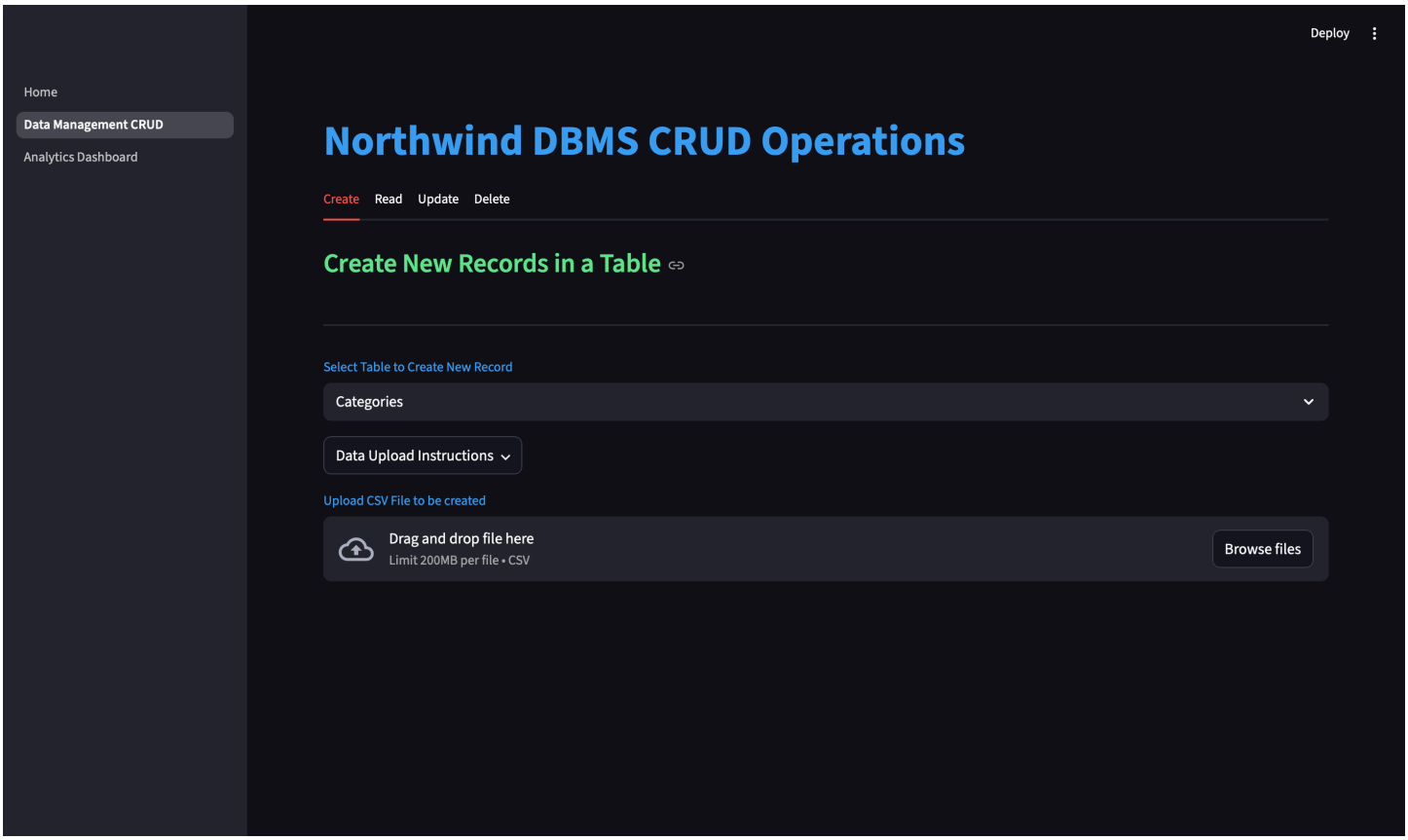

Figure 4 Data Management Page with CRUD Operations tabs

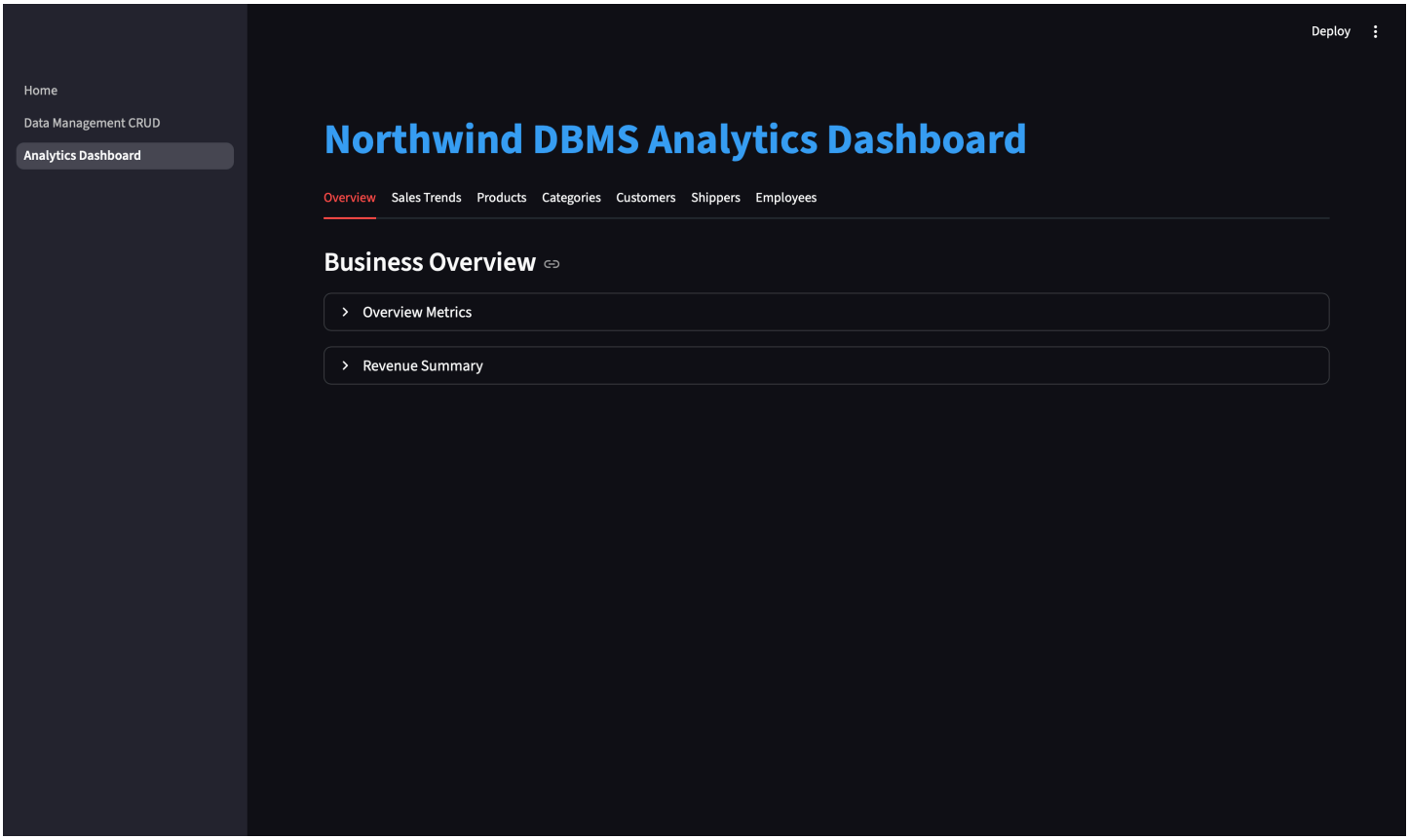

Figure 5 Analytics Dashboard Page

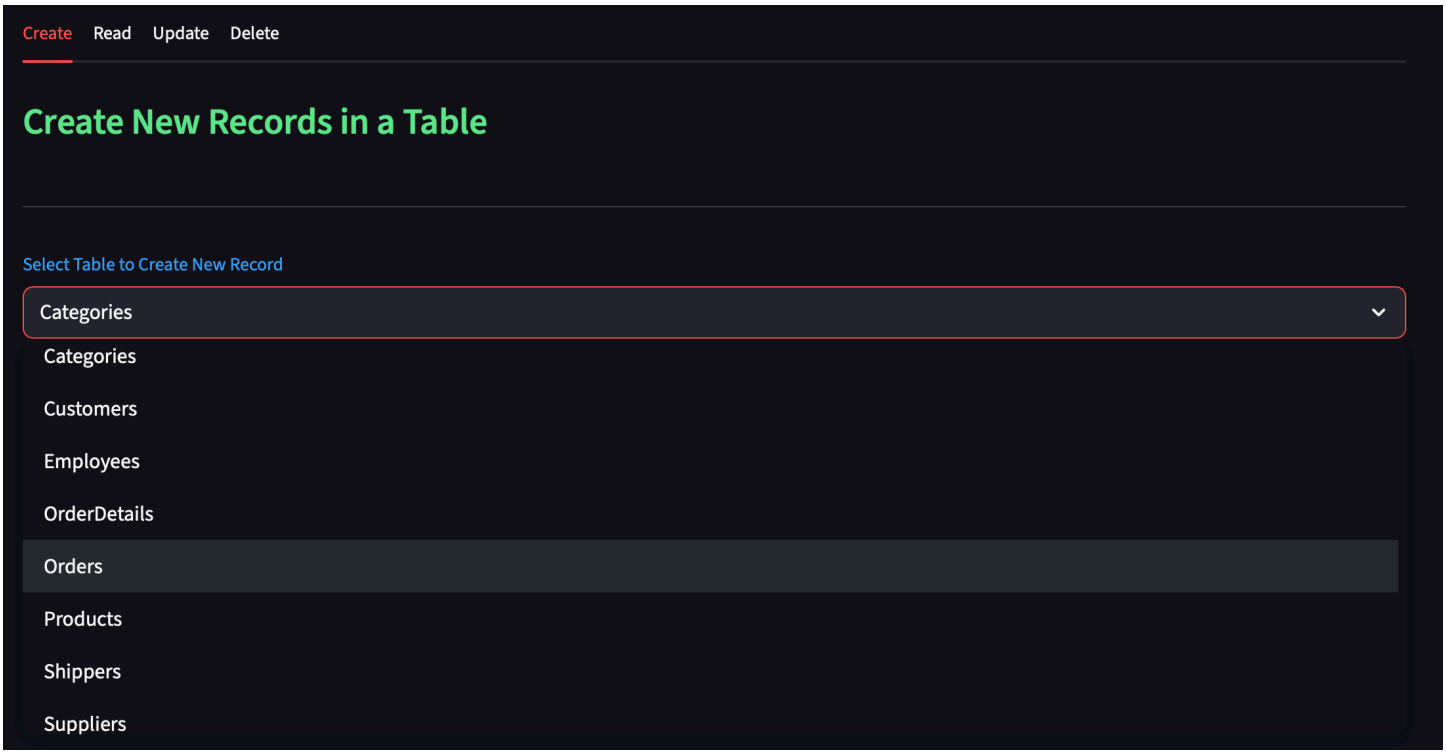

Figure 6 List of Tables in Create Tab

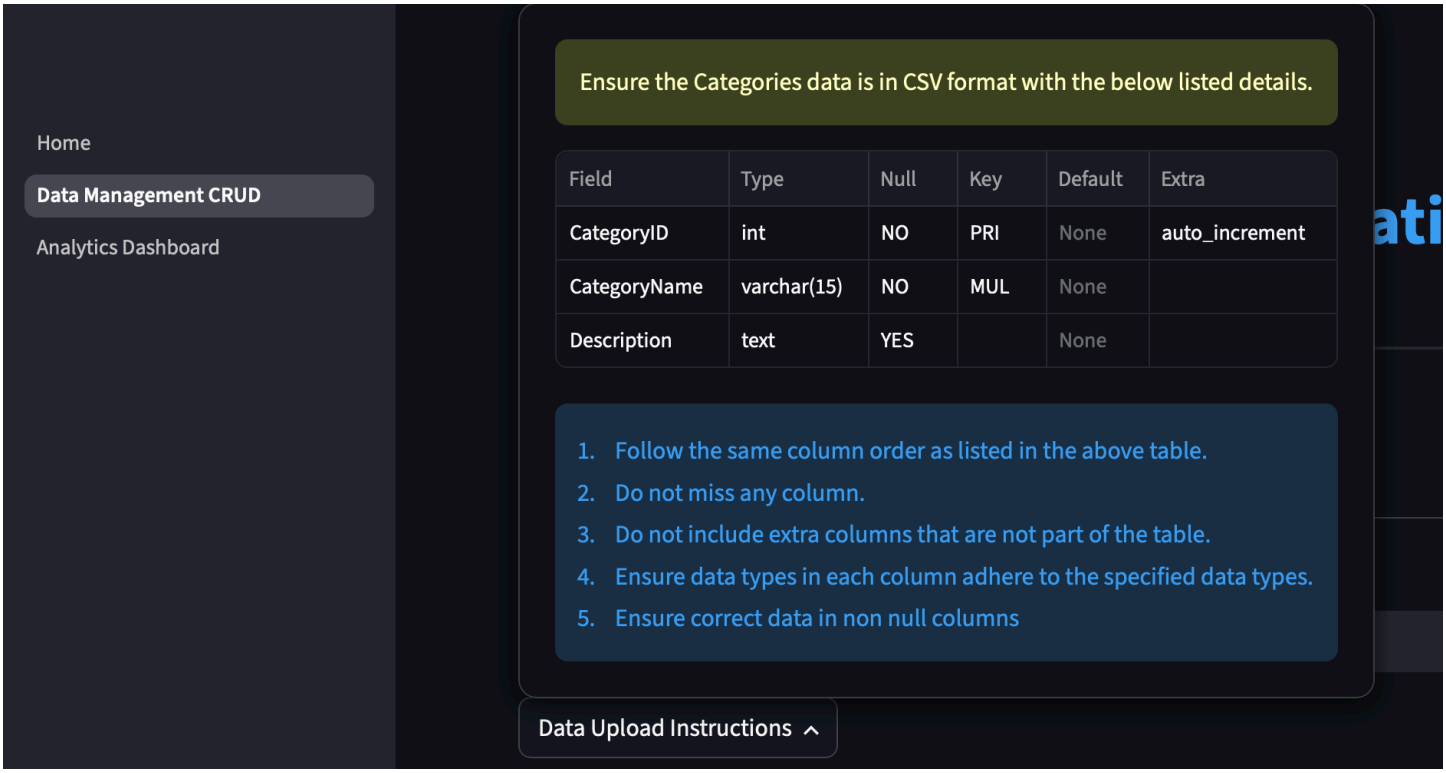

Figure 7 Upload instructions for Bulk upload tables (Example shown for Categories)

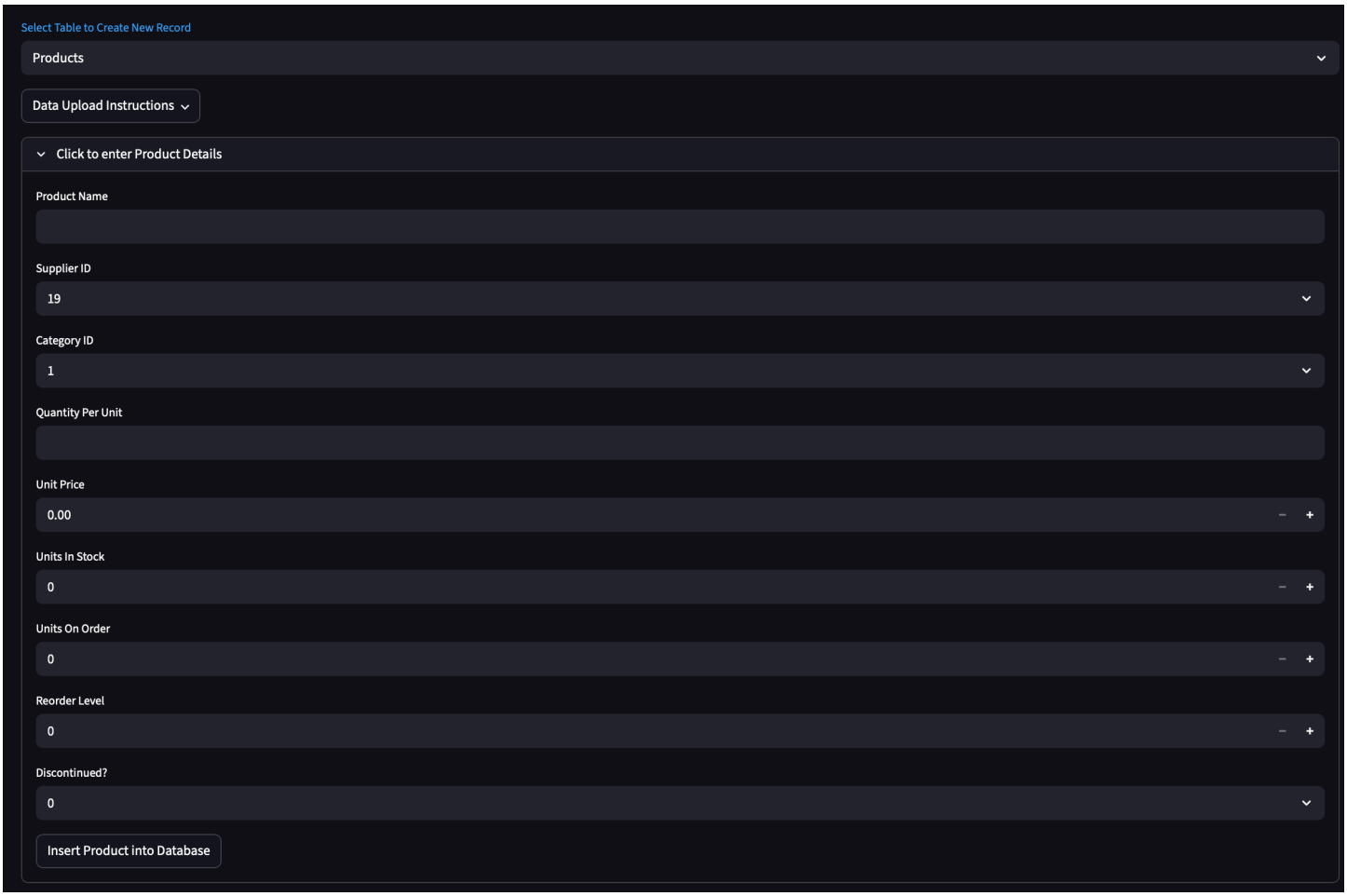

Figure 8 Data Creation by individual Records

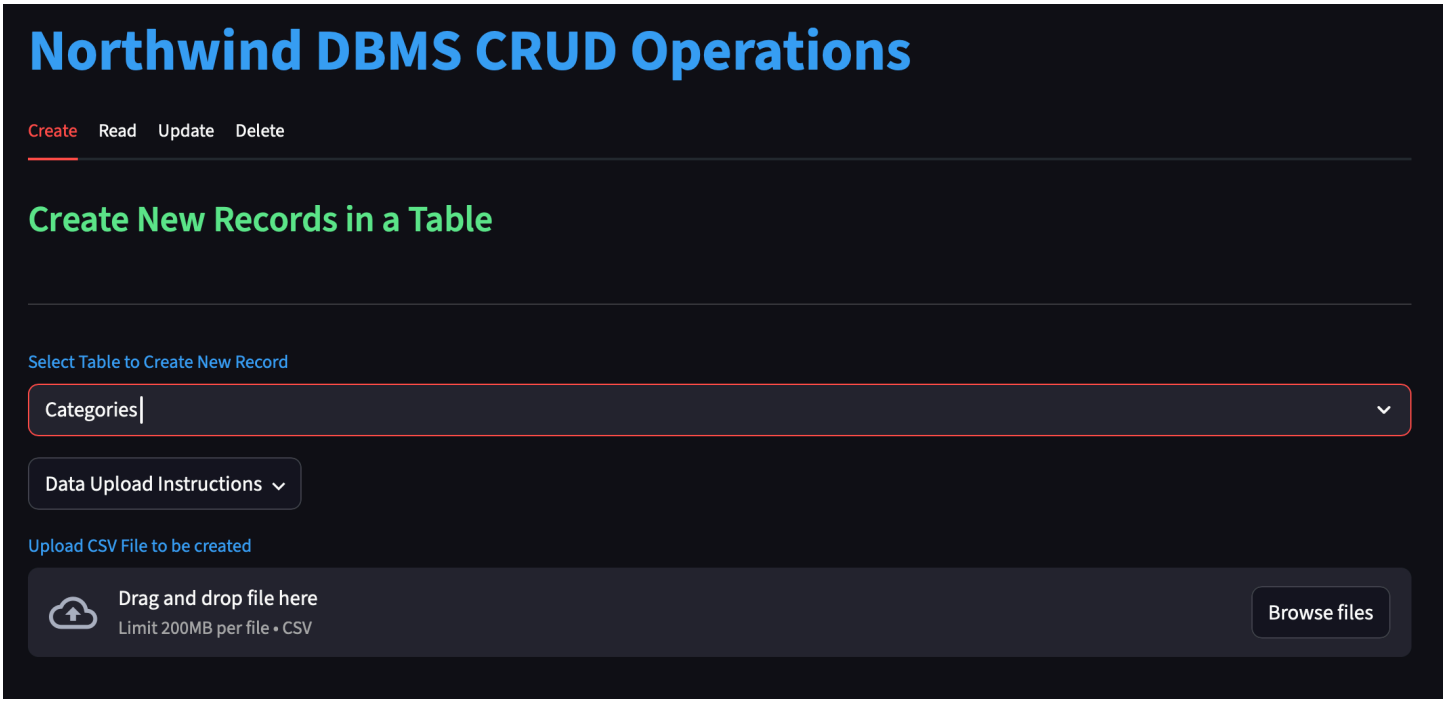

Figure 9 Data Creation by Bulk upload of csv

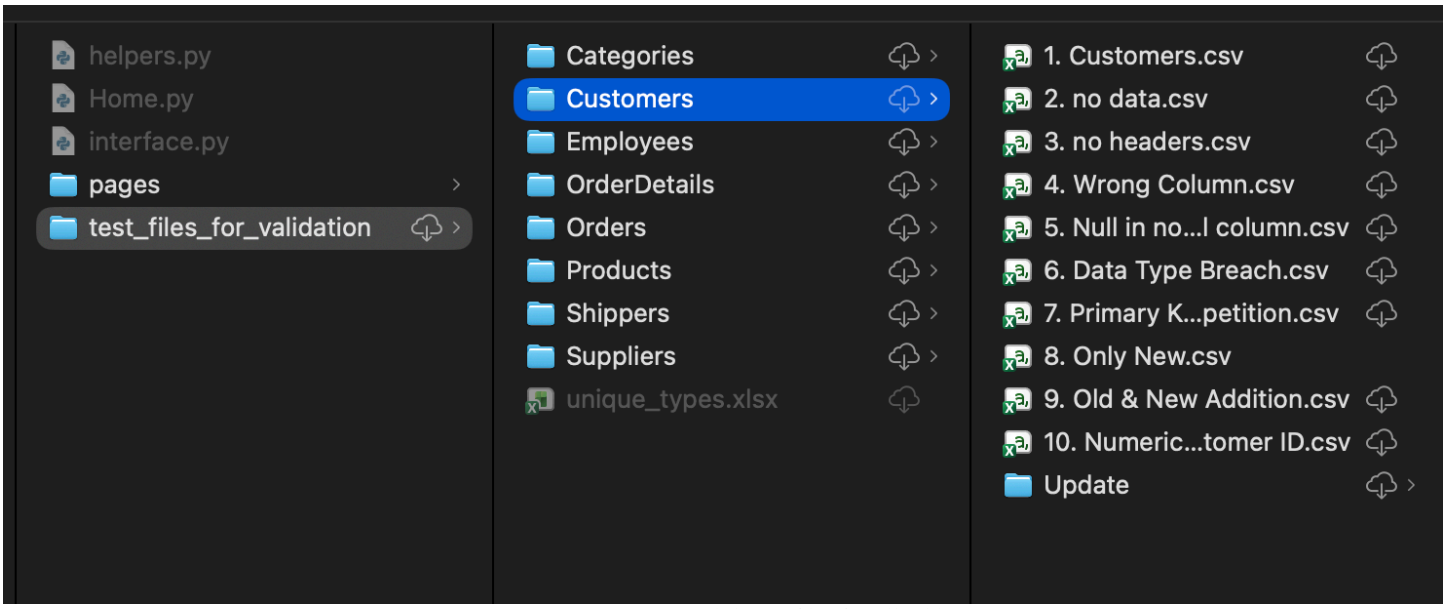

Figure 10 Test Case validation files for all tables

Figure 11 Validation of uploaded file & Error handling instructions

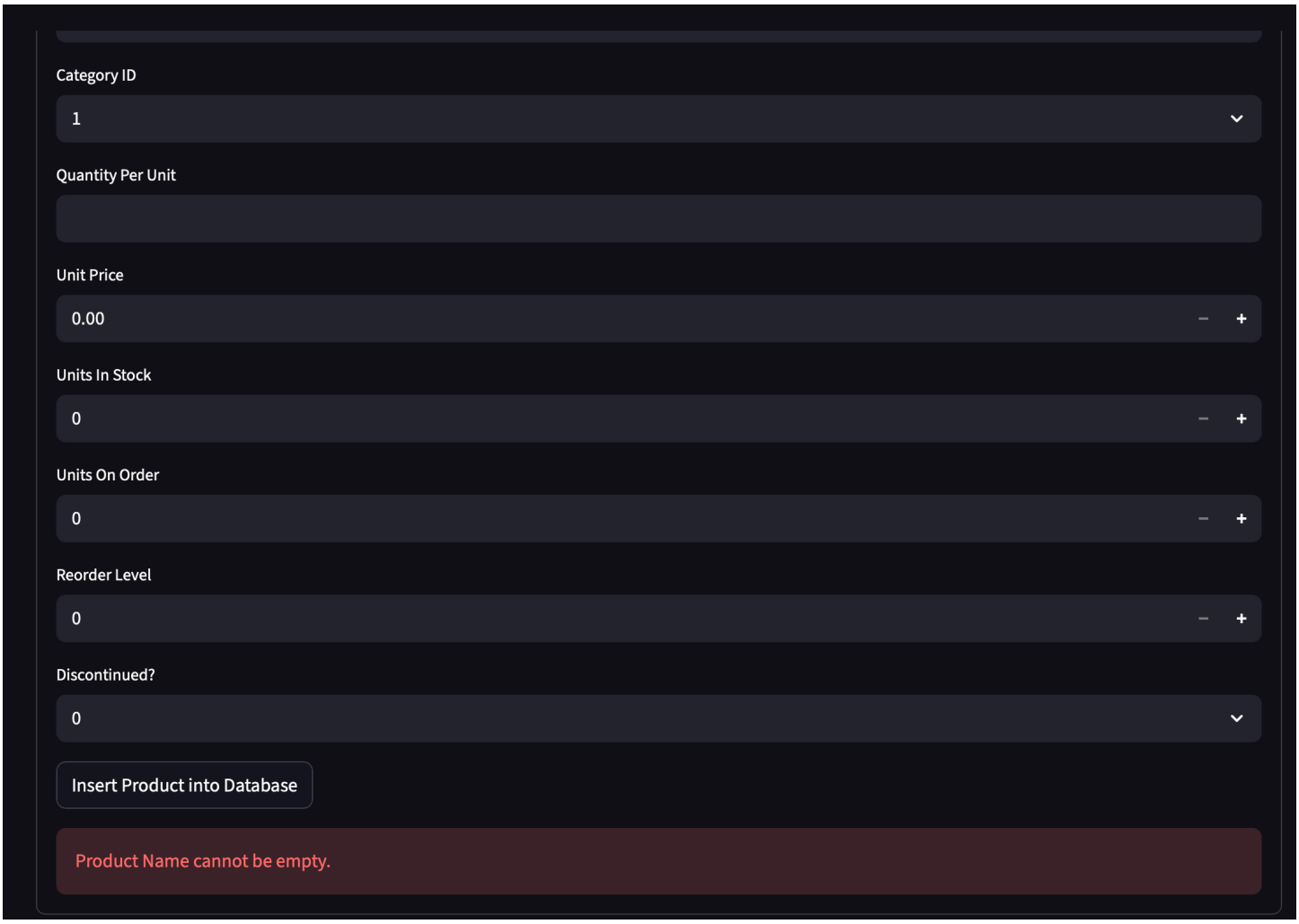

Figure 12 Validation & Error Handling instructions for individual record creations

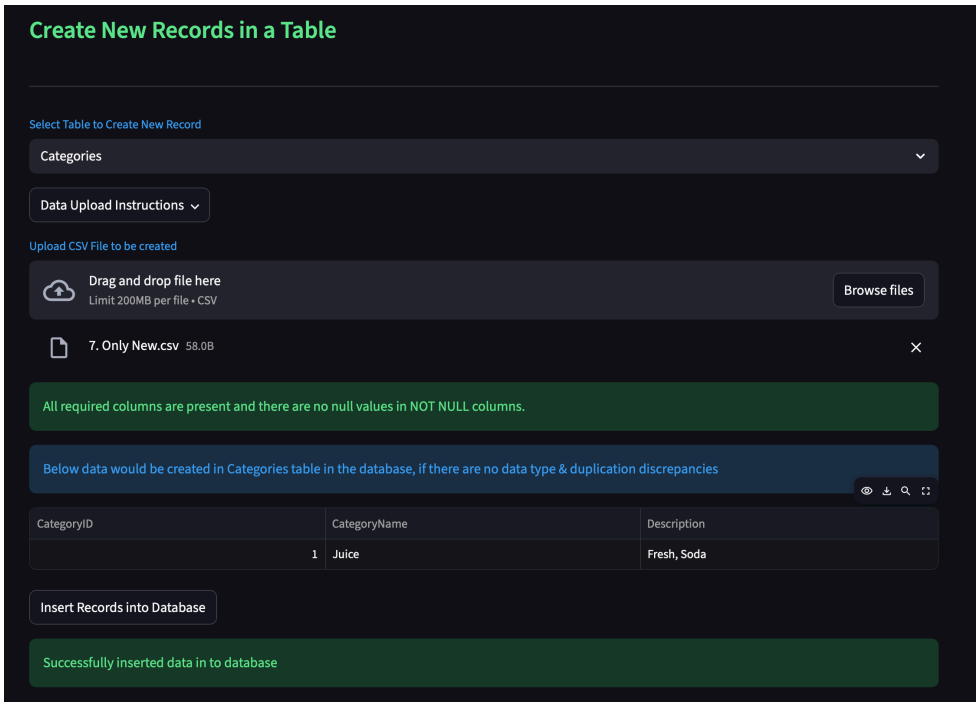

Figure 13 Info and success messages for successful creation of data using bulk upload

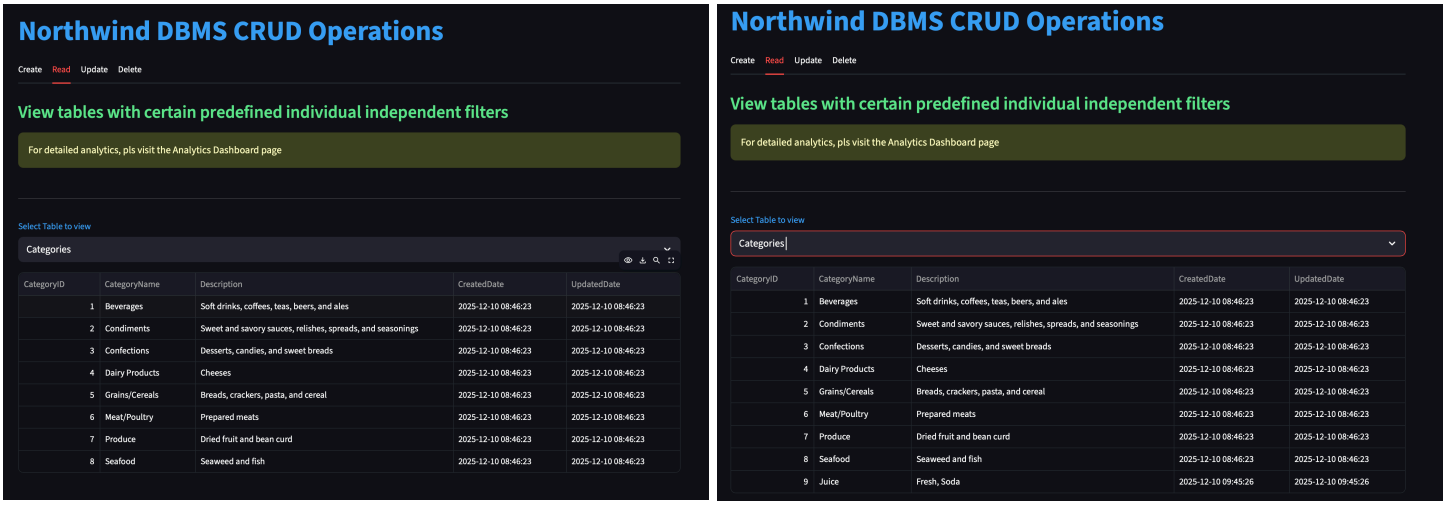

Figure 14 Read tab of CRUD: Left panel – Data before creation of new record. Right Panel – Data after creation of new category. Creation Time stamp variation can be seen

Figure 15 Update tab of CRUD

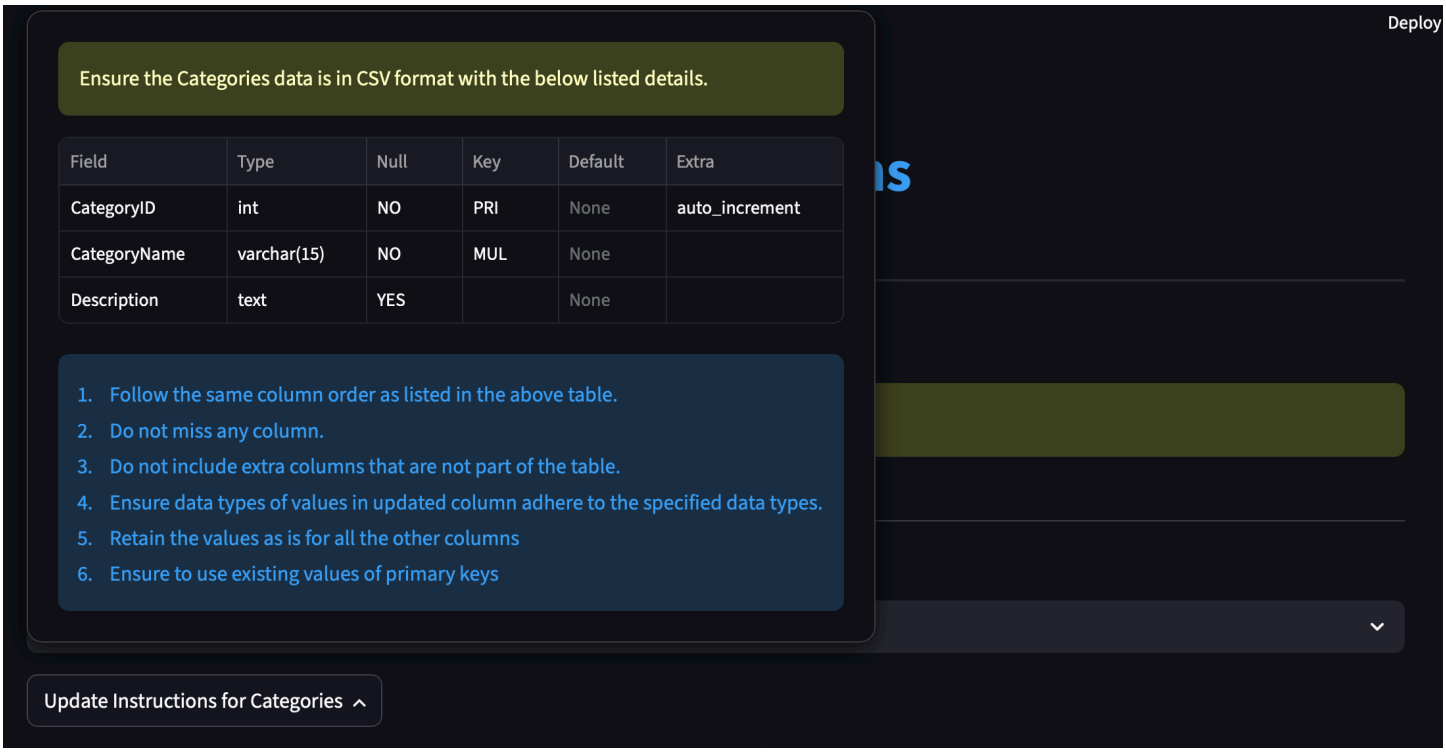

Figure 16 Data Upload instructions for update tab's selected table

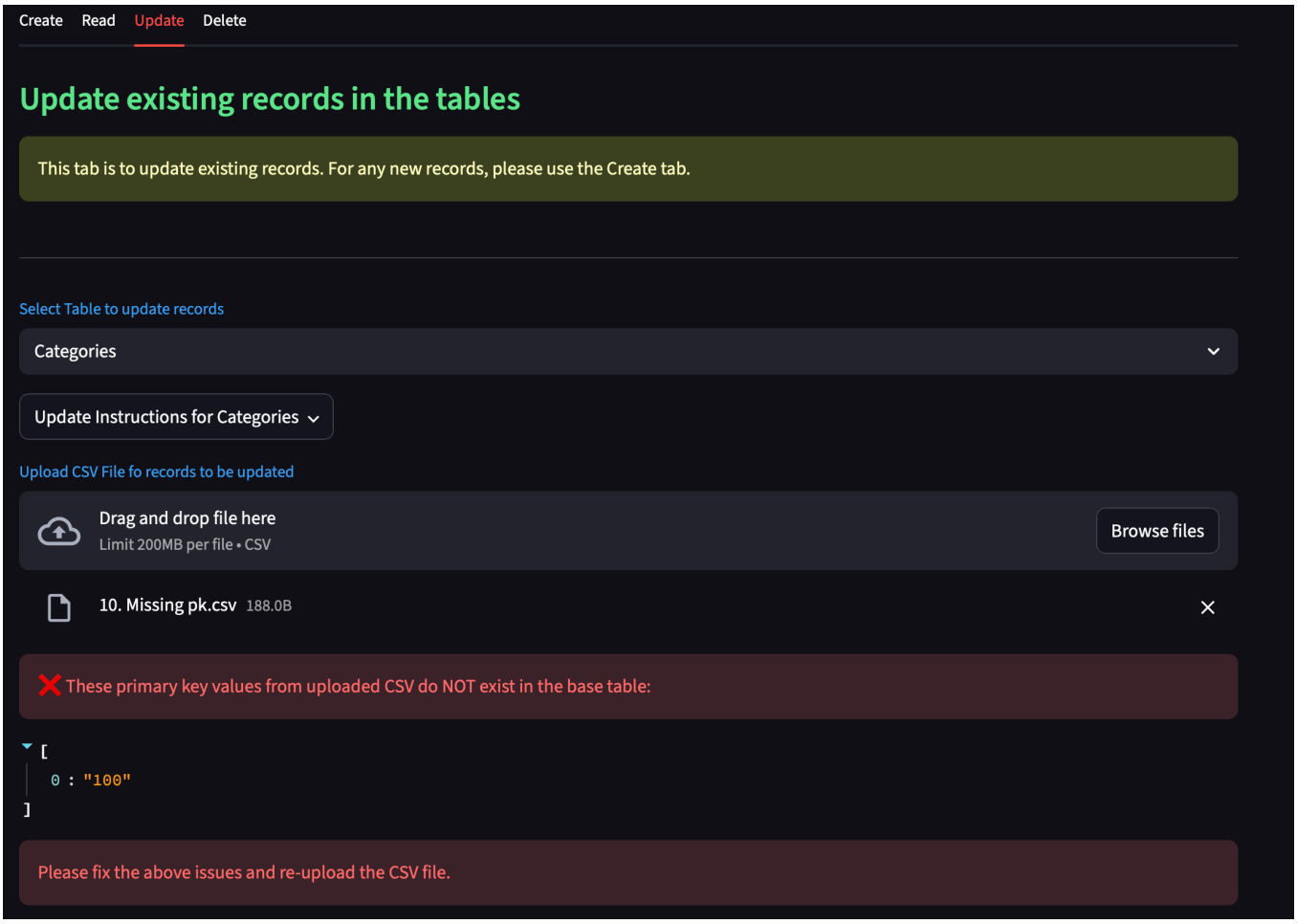

Figure 17 Data Validation & Error Handling in Update Tab

Figure 18 Success messages of updated Records

CategoryID	CategoryName	Description	CreatedDate	UpdatedDate
1	Beverages	Soft drinks, coffees, teas, beers, and ales	2025-12-10 08:46:23	2025-12-10 08:46:23
2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings	2025-12-10 08:46:23	2025-12-10 08:46:23
3	Confections	Desserts, candies, and sweet breads	2025-12-10 08:46:23	2025-12-10 08:46:23
4	Dairy Products	Cheeses, Milk, Paneer, Changed column	2025-12-10 08:46:23	2025-12-10 09:50:57
5	Grains/Cereals	Breads, crackers, pasta, and cereal	2025-12-10 08:46:23	2025-12-10 08:46:23
6	Meat/Poultry	Prepared meats	2025-12-10 08:46:23	2025-12-10 08:46:23
7	Produce	Dried fruit and bean curd	2025-12-10 08:46:23	2025-12-10 08:46:23
8	Seafood	Seaweed and fish (updated)	2025-12-10 08:46:23	2025-12-10 09:50:57
9	Juice	Fresh, Soda	2025-12-10 09:45:26	2025-12-10 09:45:26

Figure 19 Changes in UpdatedDate for CategoryID 4 after successful update operations

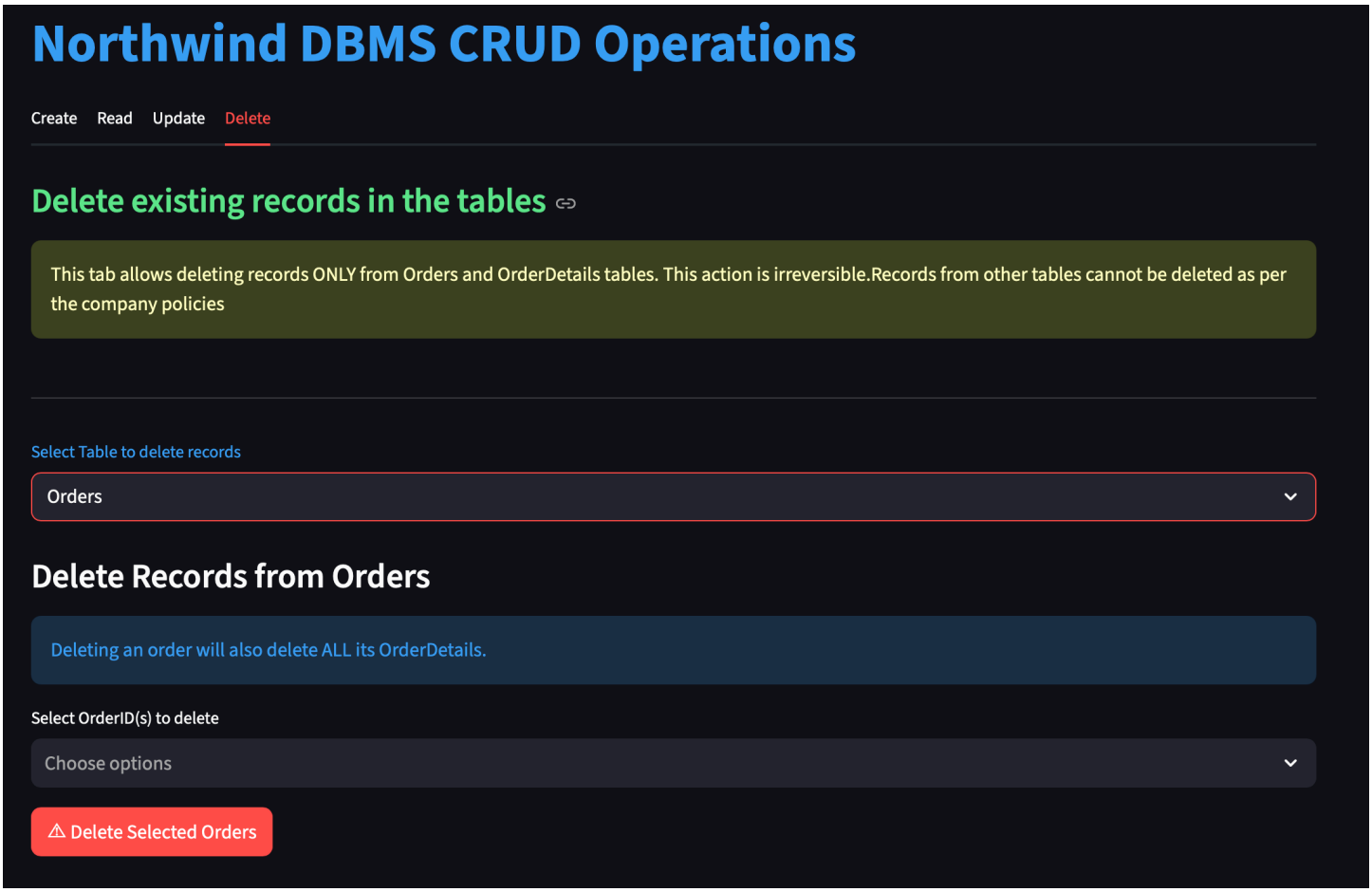

Figure 20 Delete Tab of CRUD Operations

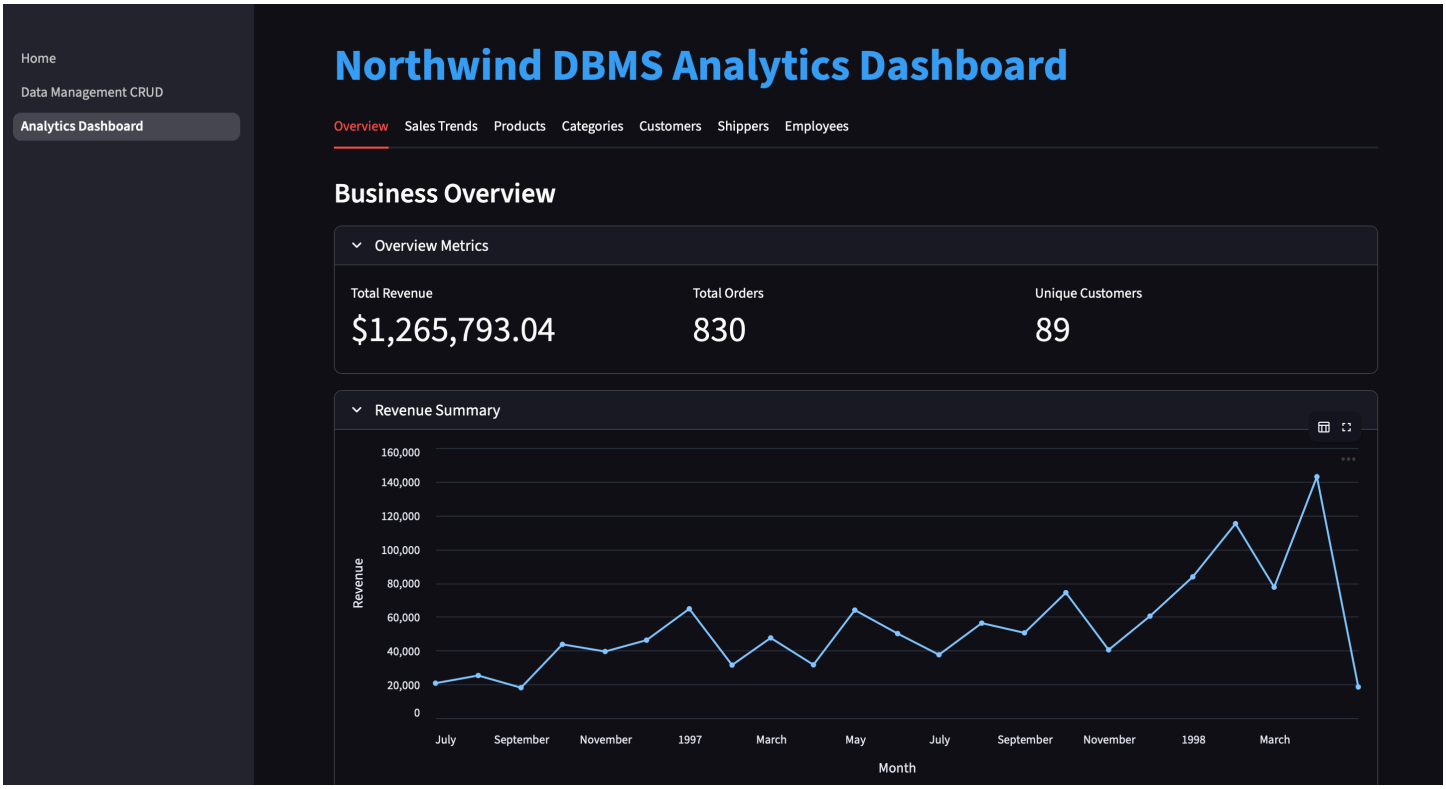

Figure 21 Analytics Dashboard Overview

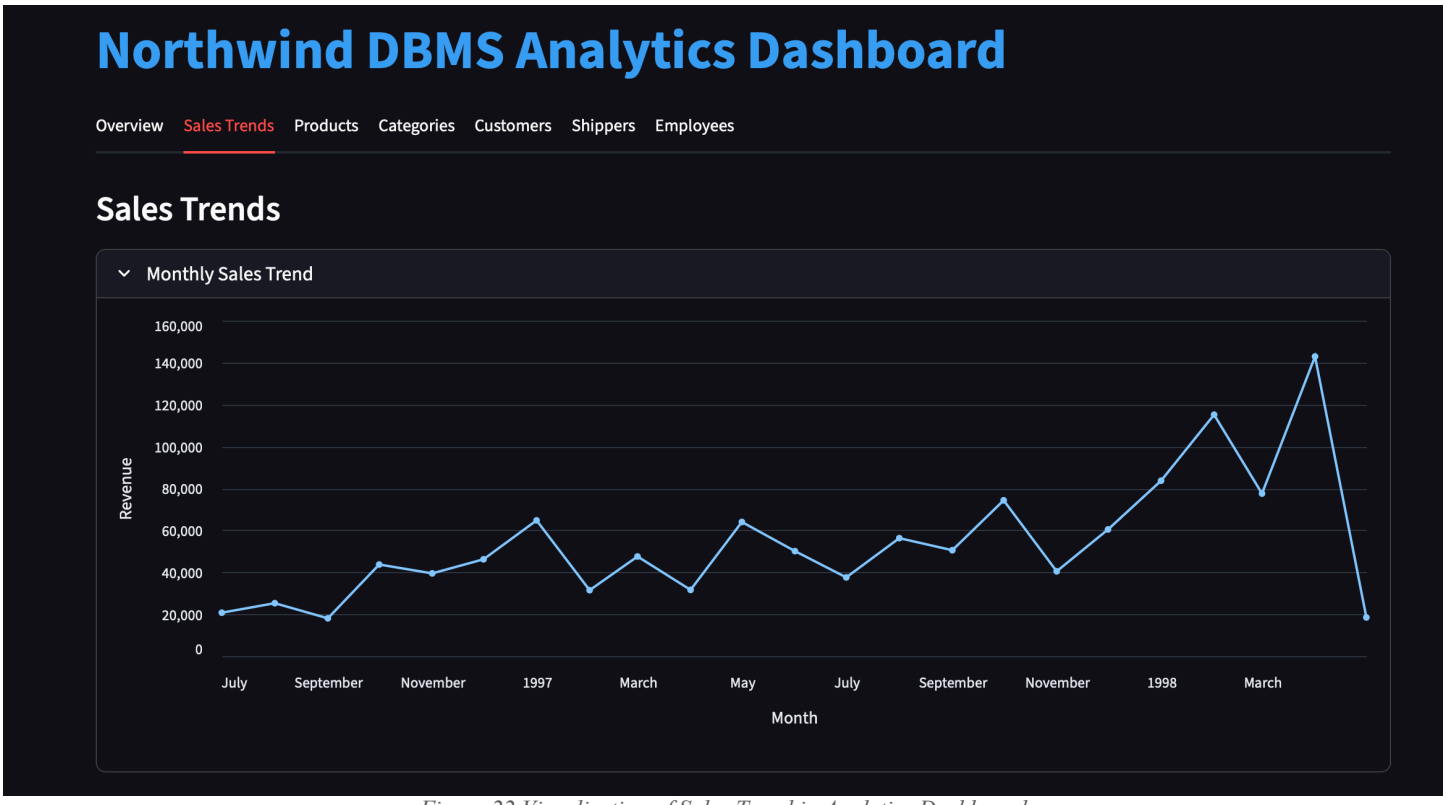

Figure 22 Visualization of Sales Trend in Analytics Dashboard

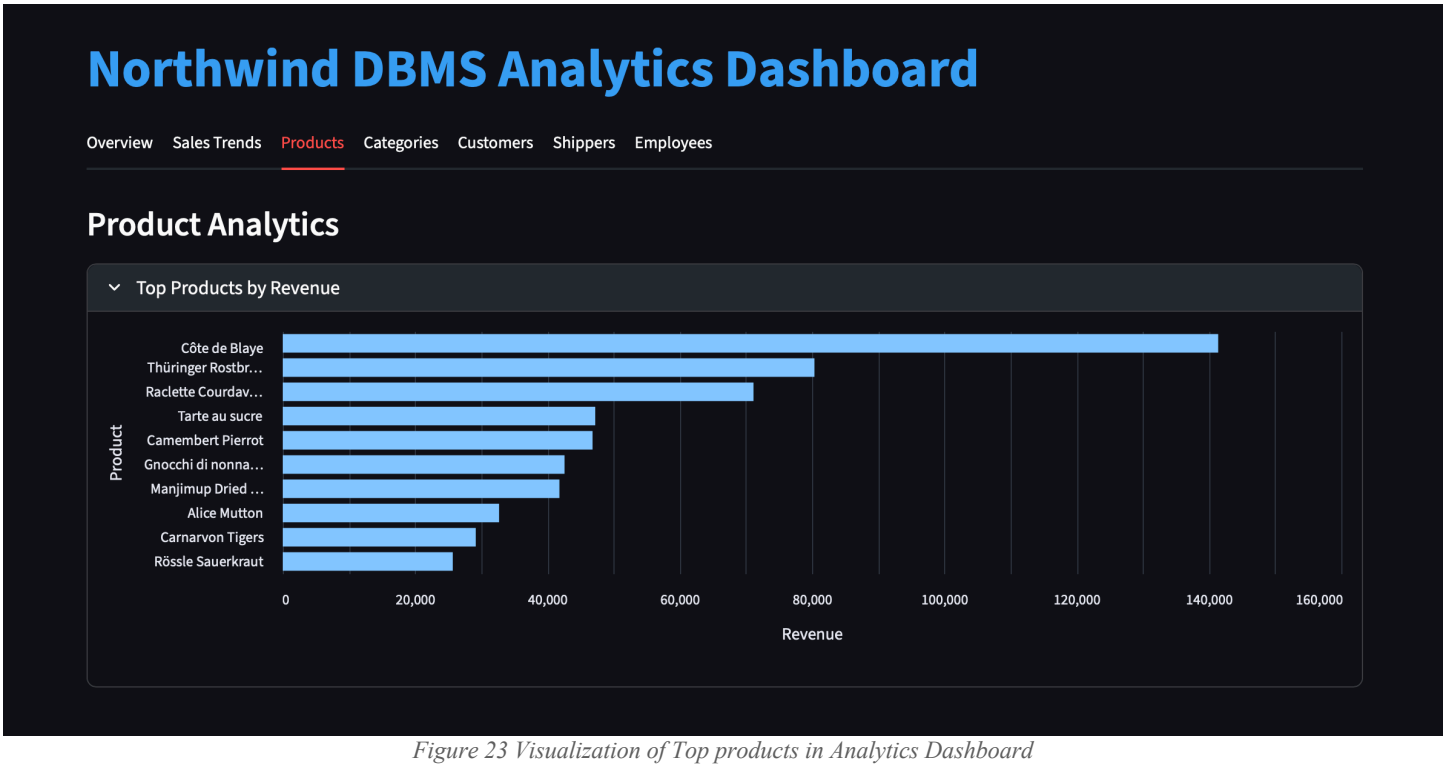

Figure 23 Visualization of Top products in Analytics Dashboard

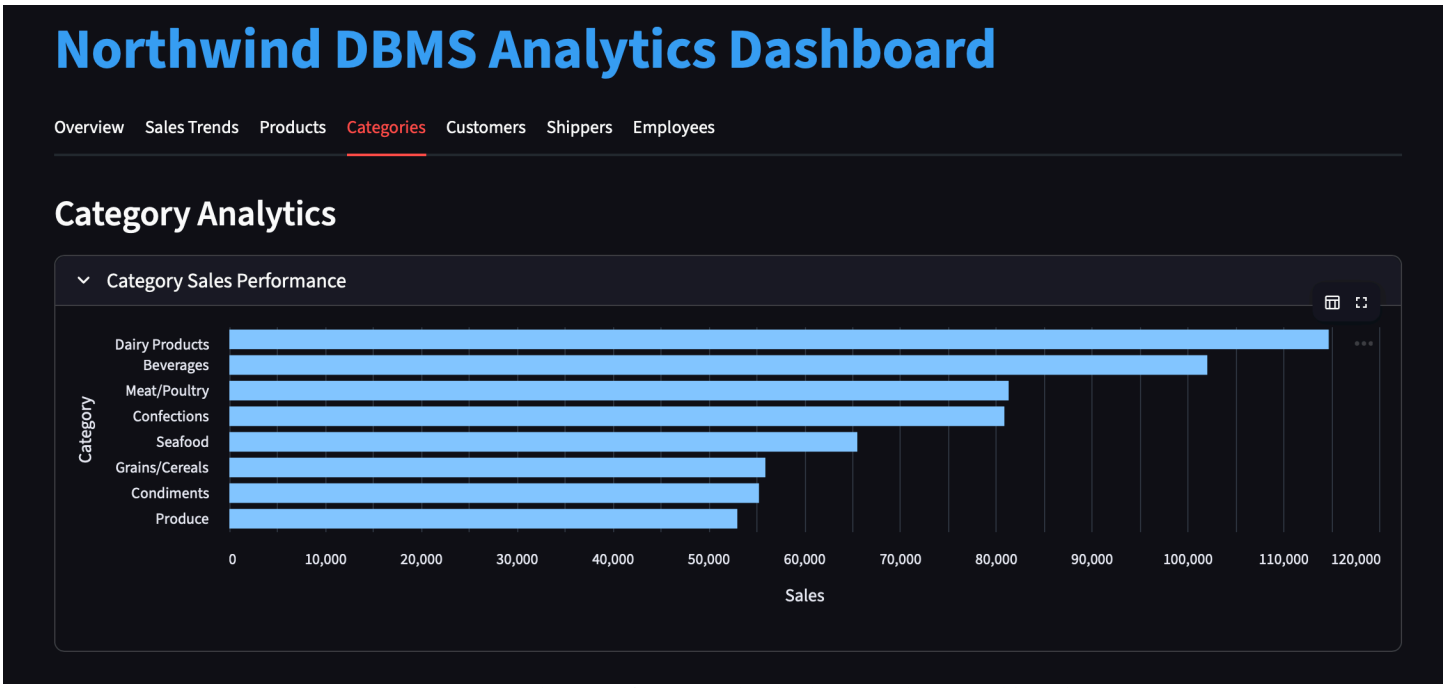

Figure 24 Visualization of Category level sales in Analytics Dashboard

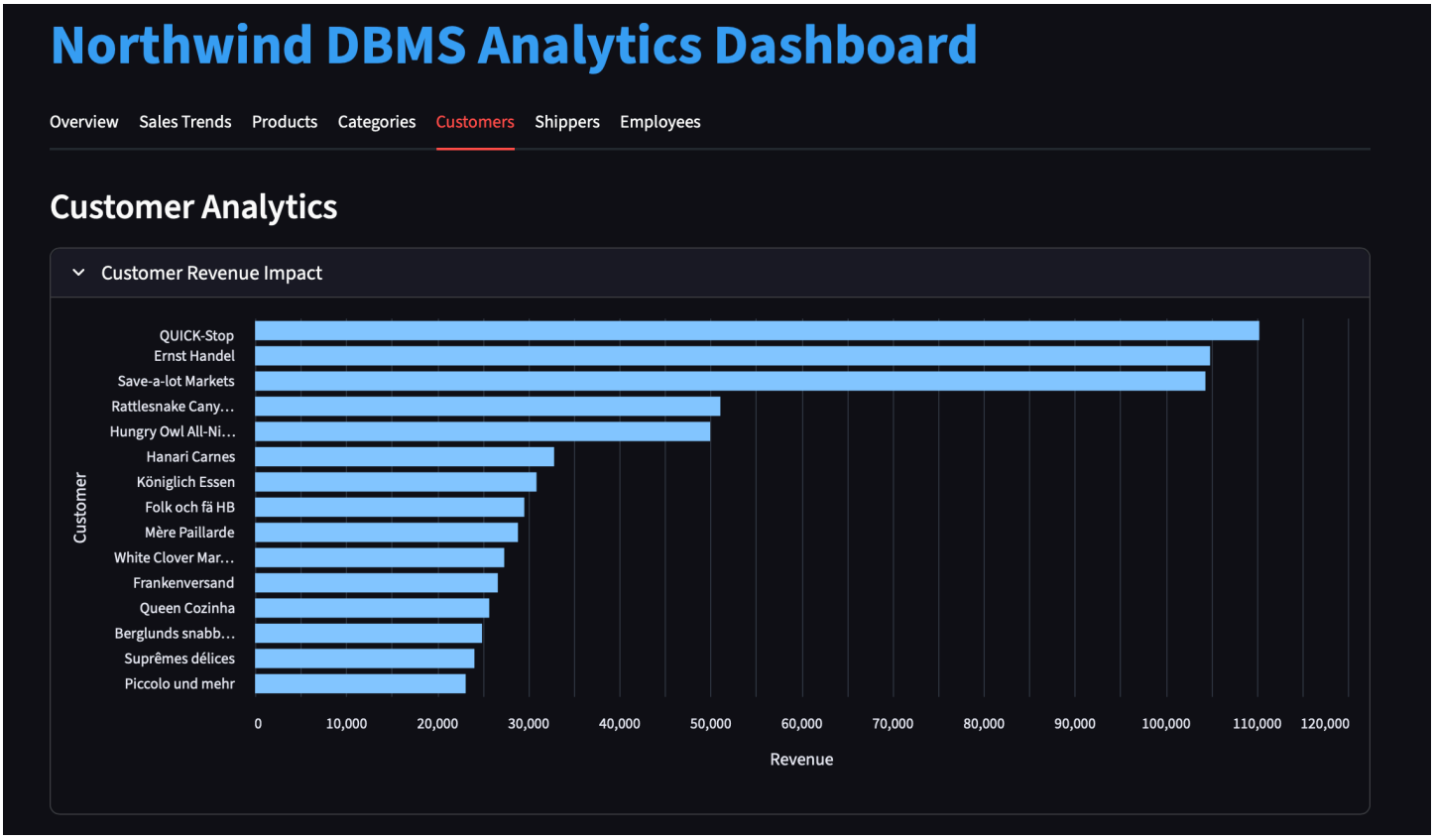

Figure 25 Visualization of Revenue per customer in Analytics Dashboard

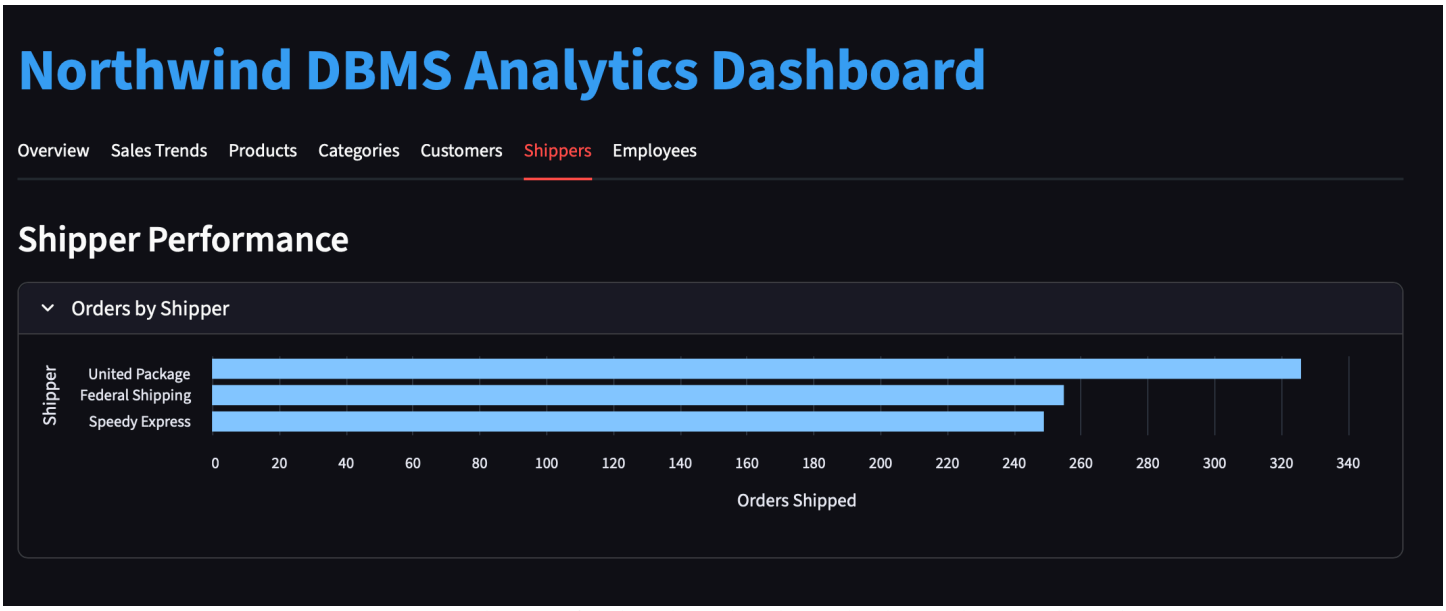

Figure 26 Visualization of Orders shipped at Shipper level in Analytics Dashboard

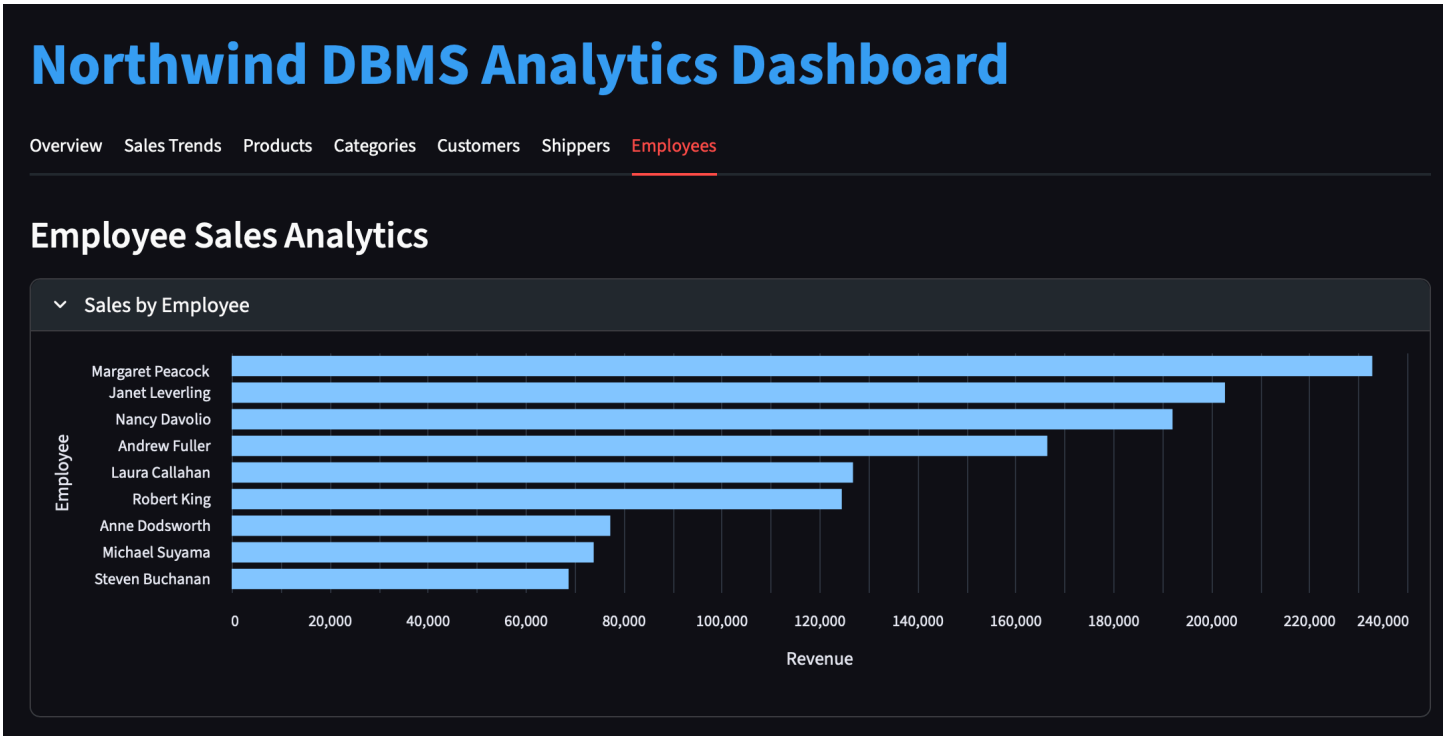

Figure 27 Visualization of Revenue by Employee in Analytics Dashboard