Building Pet's n' Paws Database Schema Using Structured Query Language

Robert Brown

Northcentral University

TIM-5020 Database Design and Business Intelligence

Dr. Martin Crossland

March 12, 2023

Building Pet's n' Paws Database Schema Using Structured Query Language

Introduction

This assignment aims to develop a relational database system for a local pet supply company that recently expanded its retail business. The database system will handle customer orders, the product catalog, suppliers, and shipping. Customers can order in-store, online, and via a custom mobile app. The database system will support the organization's goals of improving order-entry business processes, better understanding customers' needs, making effective marketing decisions, and enhancing decision-making using business intelligence analysis and reporting.

To complete this assignment, the entity-relationship diagram (ERD) from Week 3 will be referenced to build the database schema using SQL CREATE statements. The schema will be designed in the third standard form (3NF) (Kroenke et al., 2022). A SQL script called CREATE.sql will be drafted to include the CREATE statements needed to build the schema in the correct order of dependency based on the relationships between tables (Bagui & Earp, 2011; Kroenke et al., 2022). Additionally, a SQL script called INSERT. SQL will be written to add sample data to the database to test the design of the database.

Throughout the process, attention will be paid to ensure that all constraints are met and that no INSERT, UPDATE, or DELETE anomalies are present in the database. The completed assignment will demonstrate the creation of a 3NF database schema for the Pet’s n’ Paw’s supply company that meets the business rules outlined in the business case.

Designing the Database Schema

Designing the database schema for the pet supply company involved creating an entity-relationship diagram (ERD) based on the business case and the guidelines provided by scholarly resources (Bagui & Earp, 2011; Kroenke et al., 2022). The ERD served as a visual representation of the relationships between the entities in the database and ensured that the schema was designed logically and efficiently (Bagui & Earp, 2011).

The first step in designing the schema was identifying the database entities based on the business requirements. The entities identified included customers, orders, products, categories, suppliers, and shipping methods. Each entity was defined with the necessary attributes, such as customer contact information, product details, and shipping addresses.

Next, relationships were established between the entities, including one-to-one, one-to-many, and many-to-many relationships (Bagui & Earp, 2011). Unique primary keys were identified for each entity, and foreign keys were assigned to establish the relationships between the entities (Bagui & Earp, 2011; Kroenke et al., 2022).

While creating the tables, a SQL notebook was used to write and store the SQL code for creating the tables in SQL Server Management Studio Express. The notebook allowed revisions to the SQL code as modifications were made to the ERD in preparation for creating the database.

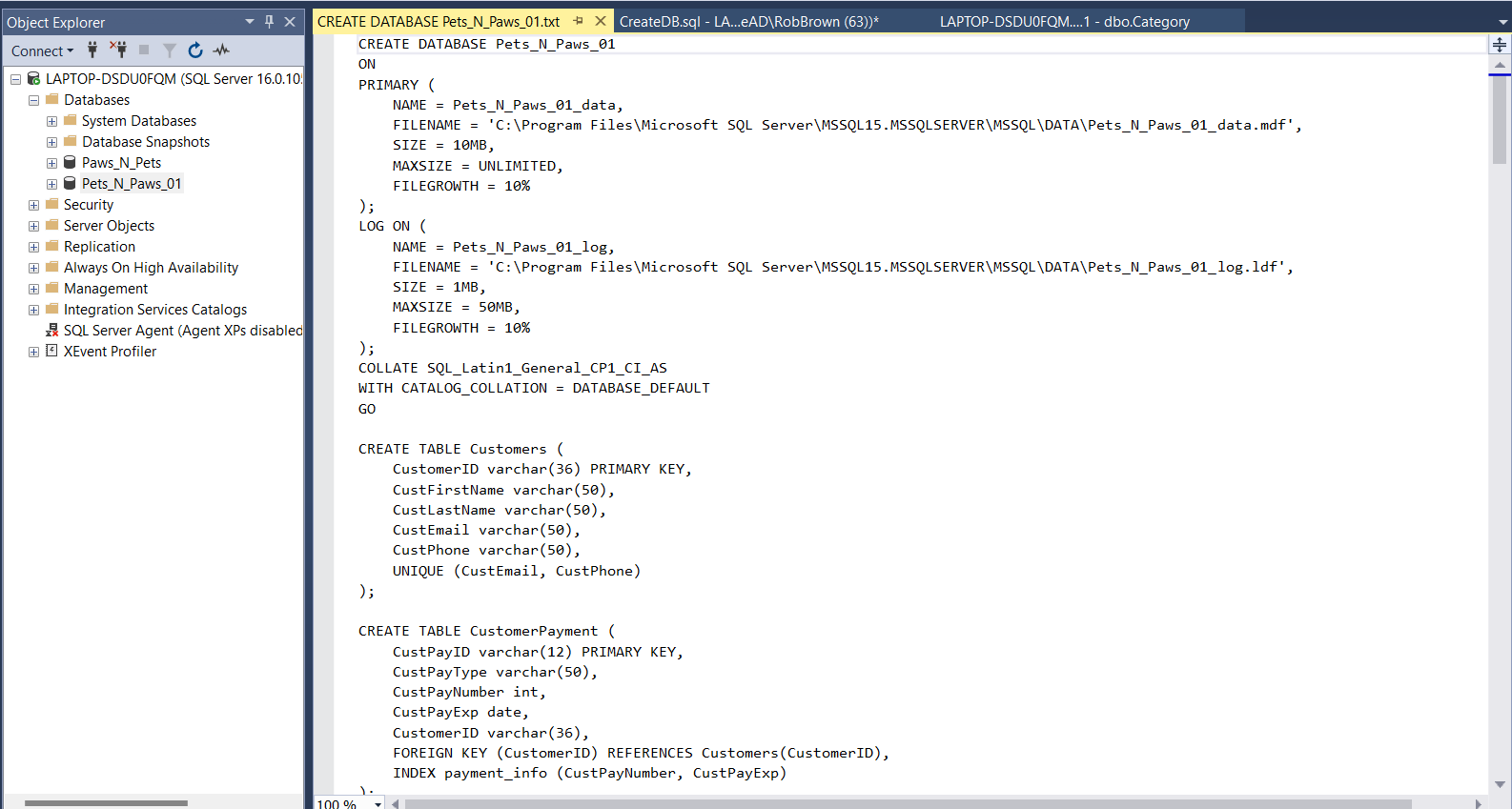
It was not until later in the process that it was discovered that a table was needed to prevent redundancy in the database. A revision was made to the ERD to reduce unnecessary keys and columns in a few tables. A new table called 'OrderStatus' was created to reduce redundancy and create a more streamlined relationship in the database.

Each table was reviewed to ensure the schema was in third normal form (3NF), ensuring that each non-key attribute depended on the primary key (Bagui & Earp, 2011). Tables were also reviewed to ensure that there were no transitive dependencies.

Overall, the design process involved iteratively refining the ERD and SQL code until a logical and efficient schema was created that met the business requirements. The annotated screenshots of the schema creation steps in SQL Server Management Studio Express are included in Figure 1. The creation of the database had its troubles.

Figure

SQL Code to Create Pet's n' Paw's Database

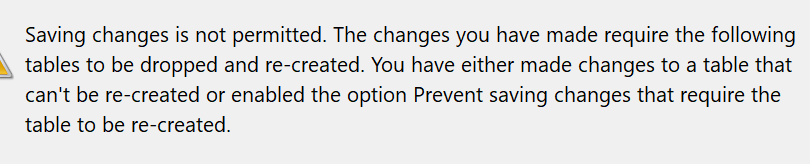


Note. Code was created in a text document and then transferred to Microsoft SQL Server Management Studio for proper syntax and format.

At some point, the master database file in an original installation was overwritten. During the sample data installation, the table attributes became incompatible, which returned continuous memory errors preventing further manipulation of the database in the format it needed to be configured, see Figure 2. Multiple attempts and methods to correct the database failed because the master database configuration files in SQL Server Management Studio (SSMS) do not allow the master file to be deleted. At one point, after deleting the program, during a second installation of the program, a new master database file was created.

Figure 2

Memory and Configuration Error Message Example



Note. This is just one example of the stream of errors that plagued the configuration installation and manipulation of the database.

An attempt to overwrite the original install was successful; however, when attempting to boot SMSS, the program will not connect to any database programs. Five attempts to uninstall and reinstall the program (using unique and default settings) yielded no success. The program was then installed on an older computer after configuring it to meet the base system requirements for the program.

After downloading the program and configuring a unique environment to avoid potential collisions, the efforts successfully installed a working system. One issue that was causing problems in the original install was that when the user attempted to create a new database, the master database attributes were overwriting the unique database attributes somehow. Once the program was completely installed and configured, the database was successfully created and configured for sample data sets.

However, some columns and keys needed to be correctly configured as the database was being populated. The ‘Orders’ tables were not adequately related to several other tables or had incorrect data types. This was solved by carefully examining the data types and relationships between the tables to ensure the foreign keys were assigned to the correct primary keys, and the data types and constraints were consistent across the relevant columns.

When the table 'OrderStatus' was created to reduce redundancy and create a more streamlined relationship in the database, it appeared to have caused an issue with the ‘Orders’ table. This addition called for a revision to the ERD to visualize the problem to reduce unnecessary keys and columns in the “troublesome” tables.

Although the relativistic relations were correctly created, the database will still not populate data into the ‘Orders’ table from the ‘Shipments’ table—specifically, the association to ‘Shipments’, ‘Shippers’, ‘ShipperCost’, and ‘OrderItemID.’

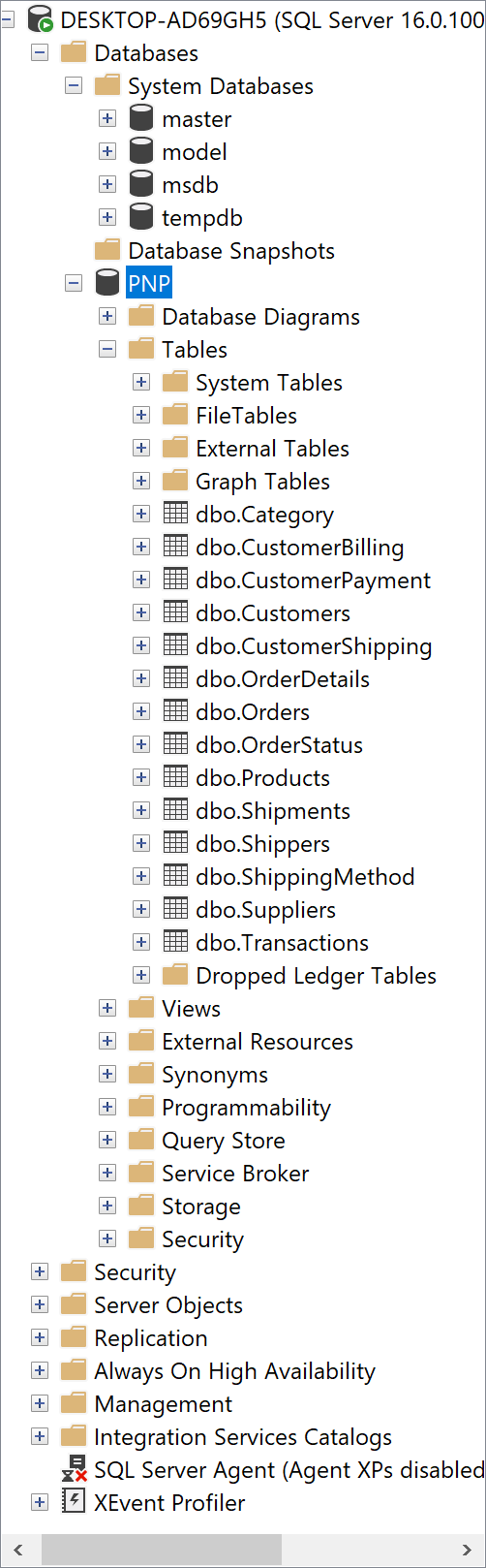
Several required fields would only populate NULL values. Finally, the tables were reviewed to ensure the schema was in third normal form (3NF), ensuring that each non-key attribute depended on the primary key and that there were no transitive dependencies. Although the values point in the right direction to each other, the error still exists. These steps helped to create a logical and efficient schema that met the business requirements and ensured data integrity in the database. The error is ongoing but will continue to be tested until it can be fixed.

Building the CREATE.sql Script

The CREATE.sql script was built to include CREATE statements needed to develop the schema in the third normal form. The script was drafted in SQL Server Management Studio Express by writing SQL code for each table, including the necessary columns, data types, and constraints, and then combined to create a master script file, as seen in Figure 3. The script was written in the order of dependency based on the relationships between the tables, starting with the creation of the database itself, then the creation of the tables.

Figure 3

Example of PNP database structure in SQL Object Explorer



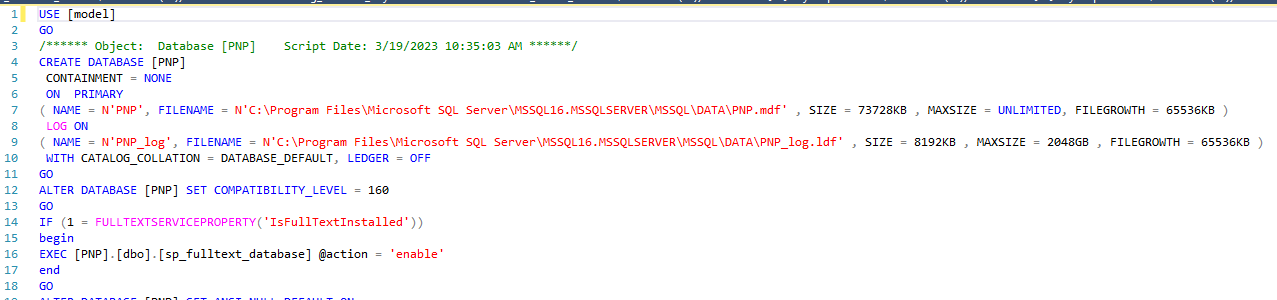
Note. This image is of the original structure before updates were discovered through testing.

The CREATE.sql script began with creating the database using the CREATE DATABASE statement. The script then created each database table, including the necessary columns, data types, and constraints. The order of creation of the tables was based on their relationships, with parent tables being created before child tables.

While writing the script, one challenge was ensuring that the foreign keys were created correctly and the relationships between the tables were properly established. This required careful attention to detail to ensure that each foreign key was assigned to the correct primary key and that the data types and constraints were consistent across related columns. As previously noted, a few unnecessary relationships and columns were created in the first draft, which was deleted, and a new table was created called ‘OrderStatus.’ This table streamlined the relationship between several tables and reduced redundant data requirements. An example of the CREATE DATABASE script is seen in Figure 4.

Figure 4

Example of CREATE DATABASE script

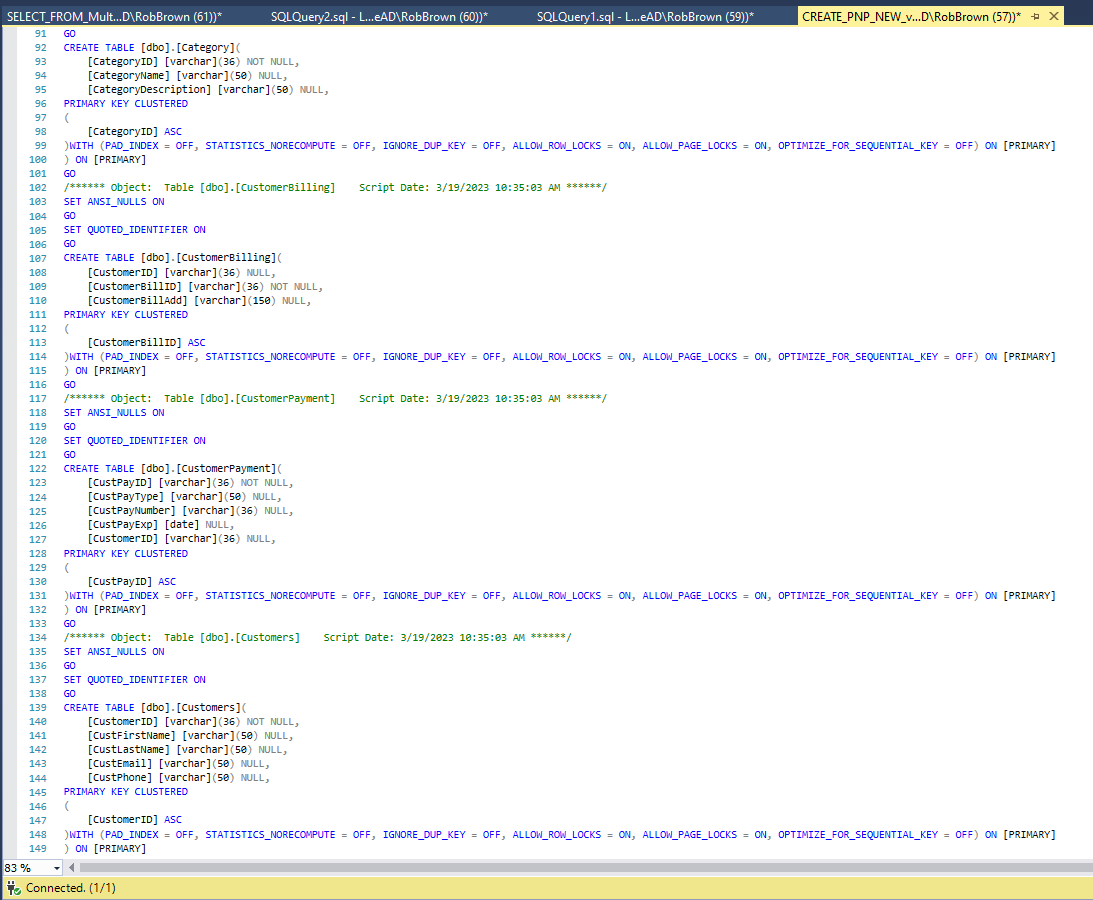


Note. Most recent version of database script.

Another consideration while writing the script was ensuring that each column had appropriate constraints to ensure data integrity, such as NOT NULL constraints and foreign key constraints. This required a thorough understanding of the business rules and relationships between the tables to ensure the constraints were applied correctly. Overall, the CREATE.sql script was designed to build the schema in the third normal form and ensure data integrity in the database, despite the issues encountered, which can be seen in another example of SQL script in Figure 5.

Figure 5

CREATE.sql Script Example



Note. This is one of the CREATE scripts generated to mitigate the issues found with the constraints. Ultimately, this script aided in creating a nominal database example.

Creating the INSERT.sql Script

For this project, sample data was generated and added to the database using random data generation techniques and synthetic data creation methods. The data generation process was designed to test the database schema and ensure all constraints were met. Several key steps were taken during this process:

The first step was understanding the schema: An analysis of the database schema was performed, which included examining the structure, relationships, and constraints among the tables. This analysis helped identify the data type required for each table (Kroenke et al., 2022).

Second, identifying data distributions: For each attribute in the schema, appropriate data distributions were determined. This ensured that the generated data was consistent with real-world data patterns (Kroenke et al., 2022; Ramakrishnan & Gehrke, 2002).

Third was generating synthetic data: Random data generation techniques were employed to create synthetic data following the identified distributions for each attribute. It took considerable effort to find the proper coding to meet the needs of the database structure. Some SQL functions like RAND(), NEWID(), or NEWSEQUENTIALID() were used to generate random data within the database (Kroenke et al., 2022; Petkovic, 2020). The NEWID() function is also embedded in the Primary Keys of all the tables. While generating sample data, the NEWID() was left out of the column properties; however, after the sample data was populated, the functions were added.

The fourth step was to preserve relationships: The sample data generation process considered the relationships between tables, such as foreign keys. Data was first generated for parent tables, and the generated primary keys were then used while populating the child tables to maintain referential integrity (Kroenke et al., 2022).

The fifth step was validating constraints: The generated data adhered to the constraints defined in the schema, such as unique constraints, check constraints, and null/not null constraints (Kroenke et al., 2022). This helped ensure the sample data accurately tested the database schema's constraints and requirements.

In the sixth step, it was essential to vary data sizes: Different sample data sets were created to test the performance and behavior of the database schema under various load conditions (Kroenke et al., 2022). This allowed for a thorough evaluation of the schema's ability to handle different volumes of data (Petkovic, 2020).

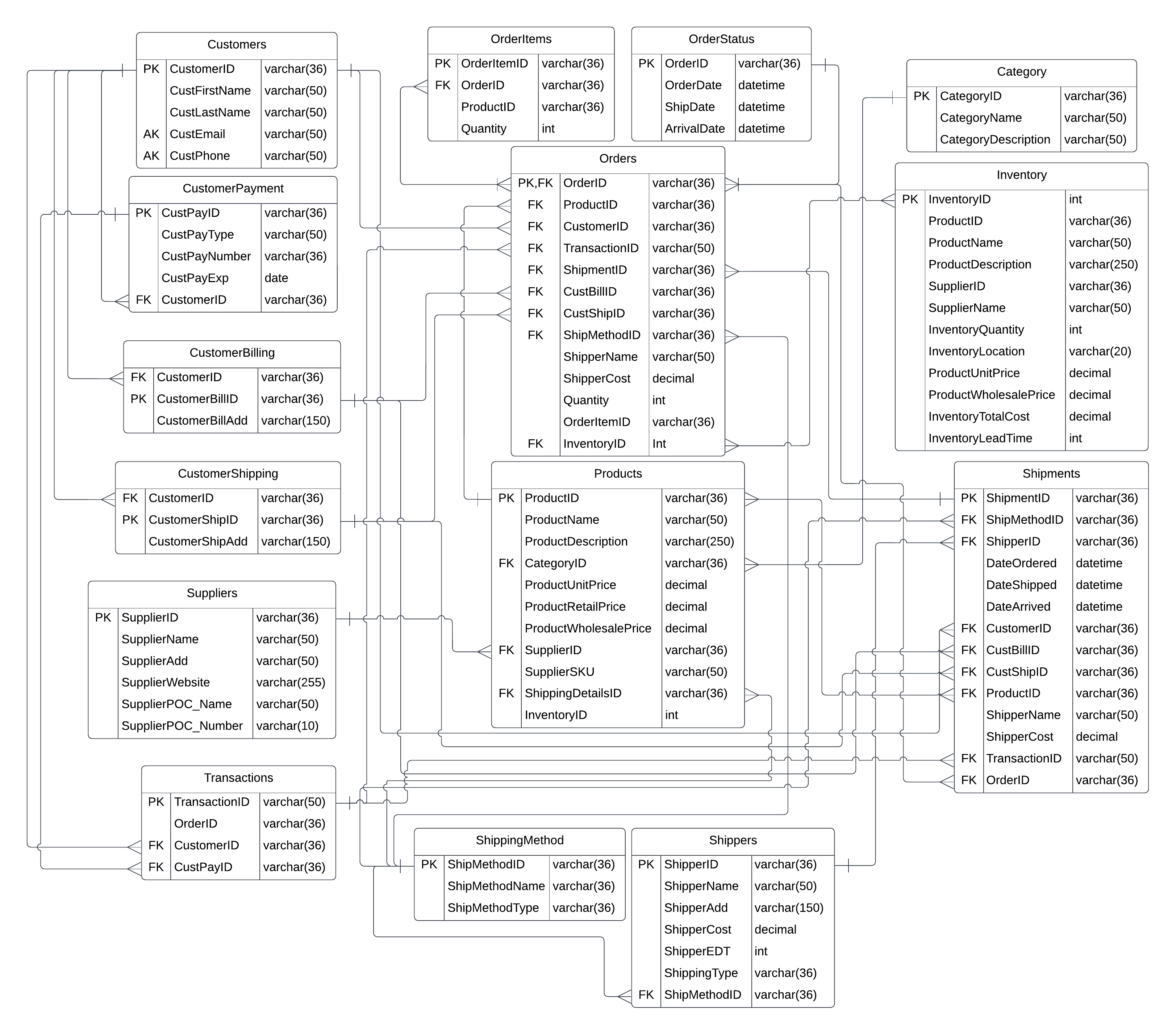
Step seven is where testing the database schema occurs. The generated sample data sets were loaded into the database schema, and test queries, data manipulation statements, and application logic that interacted with the schema were executed. This process helped to verify if the schema performed as expected and met the desired performance criteria (Petkovic, 2020).

At step eight, the sample data was successfully generated and added to the database. This ensured that the database schema was thoroughly tested and all constraints were met. This approach comprehensively evaluated the database schema's performance and behavior, validating its suitability for the intended use case (Kroenke et al., 2022).

From here, several INSERT statements were used in various techniques. In some cases, the RAND() function was utilized in a loop to iterate through the column data to generate the required information for the column specified (Petkovic, 2020). In one case, while developing the product inventory, the user inadvertently created over 9,700 products in stock at the various “Warehouse” locations across the country. The data remains in place to test the inventory formulation and triggers for the low inventory function embedded. During the creation of the follow-up ERD, seen in Figure 6, it was discovered that the relationship between ‘Inventory’ and ‘Orders’ was broken and subsequently fixed. In that case, the testing worked.

Figure 6

New ERD because of the changes

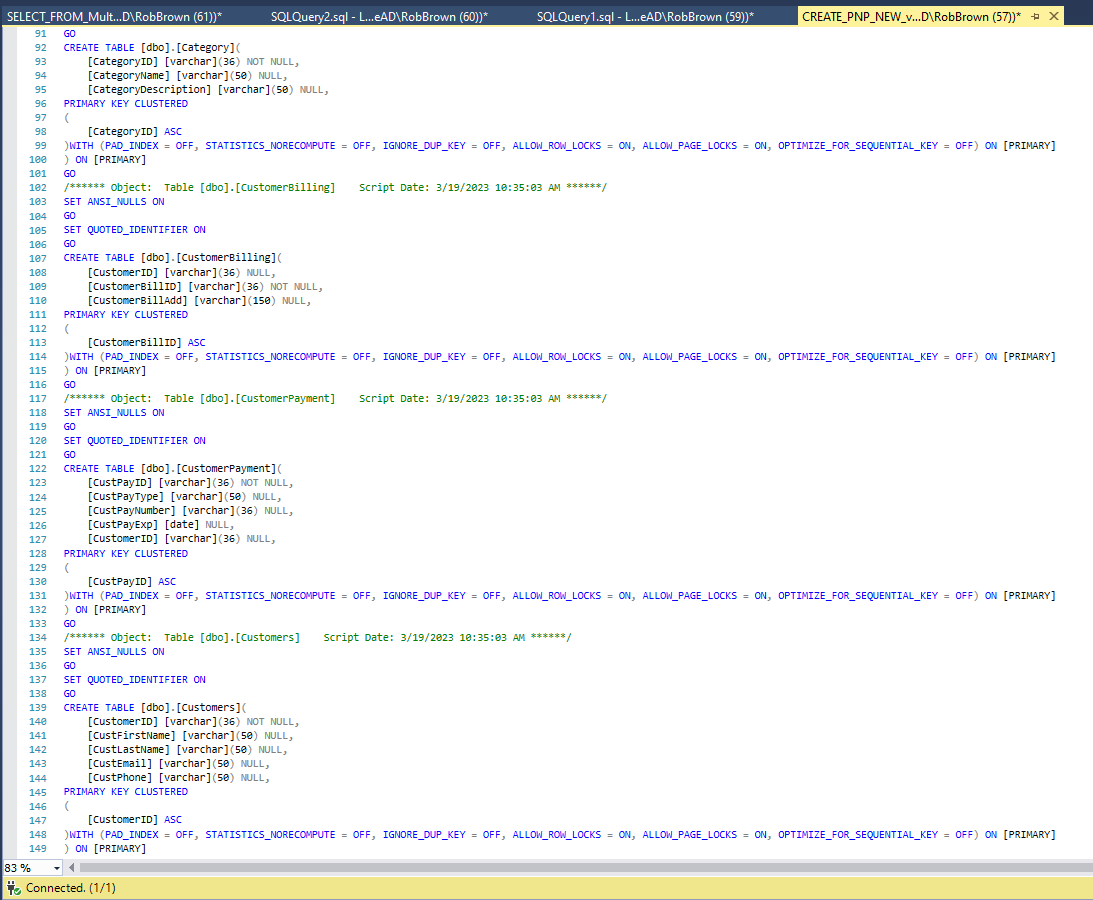


Note. New connections are created to ensure proper relational connections within the database. ERD was created using Lucidchart.

Other examples of creating sample data were done with the customer demographics needed to support payment operations in the database. To generate the sample data for the ‘CustomerBilling’ table, a loop was used using SQL that generated random data to insert into the table for the database. The quantity of data input was 1:1; in other words, for every customer, there was only one payment method. A 1:1 relationship should be adequate to test the relationship between the ‘CustomerPayment’ and ‘CustomerBilling’ tables throughout the database (Kroenke et al., 2022). First, the tables were created with appropriate relationships, as seen in Figure 7.

Figure 7

Customer Demographic tables creation example

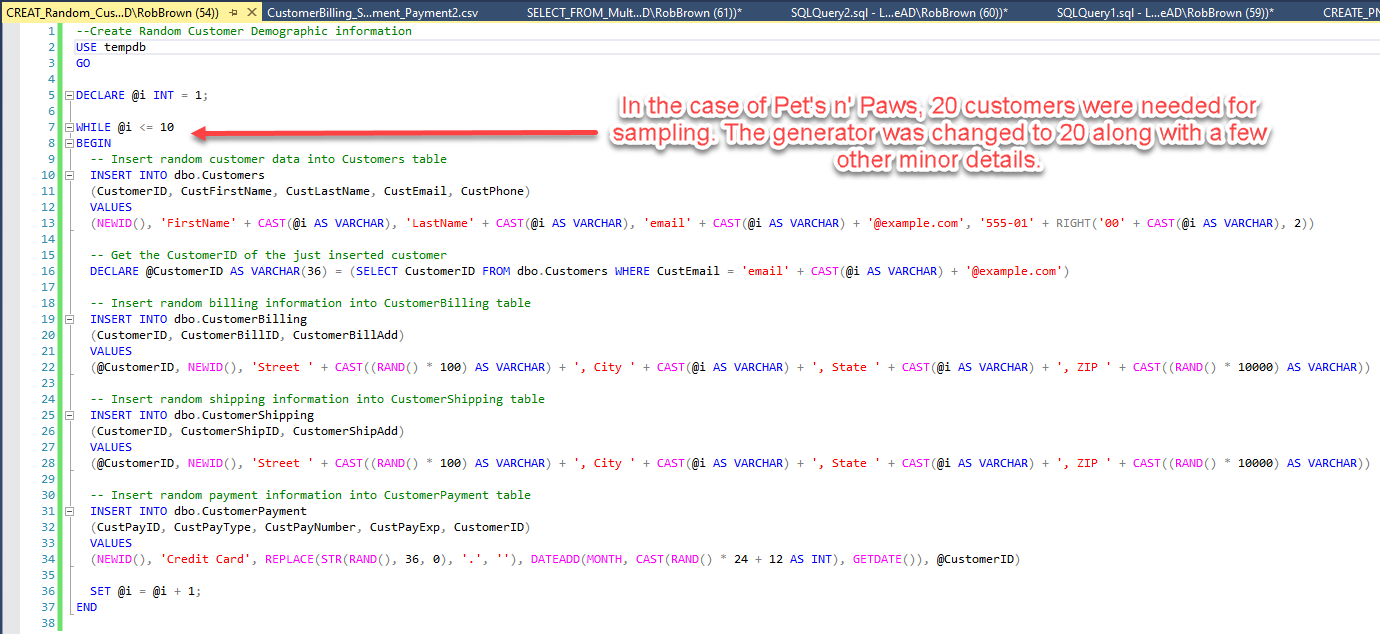


Note. Tables were created in SSMS using a simple CREATE query.

To generate the sample data for the ‘CustomerBilling’ table, a loop was used using SQL that generated random data to insert into the table for the database, see Figure 8. This process generated random data for the columns in the format required. Reviewing the generated data is sufficiently diverse enough to test the relationships and schema.

Figure 8

Customer Demographic Random Sample Data Set

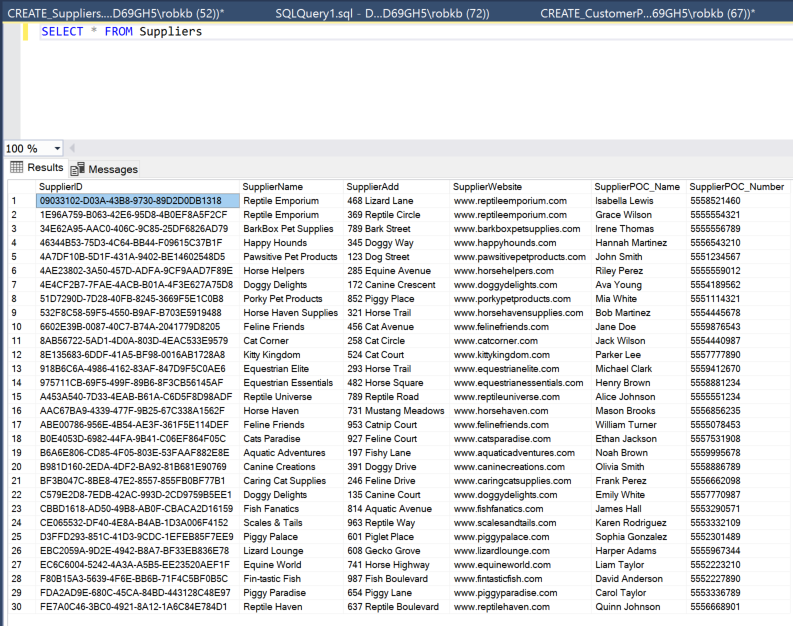


Note. This base code was modified to meet the project’s needs.

Overall, the sample data generated to test the database schema thoroughly was comprehensive and robust enough to handle various scenarios. The sample data for the ‘CustomerBilling’ can be seen in Figure 9.

Figure 9

Overview of Sample Data from Suppliers

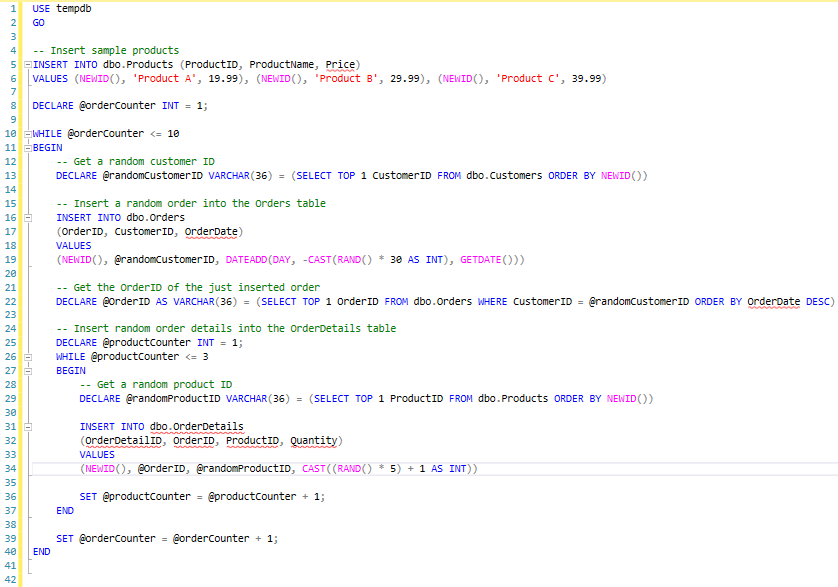


Note. Example of Suppliers Sample Data Set.

In this example shown in Figure 10, the script generates random data for the ‘OrderID,’ ‘CustomerID’, and ‘OrderDate.’ The scripts were designed to be flexible so that different values could generate different sample data sets. This allowed various scenarios to be tested and ensured the database was robust enough to handle different data types.

Figure 10

Random Sample Data Generation Script



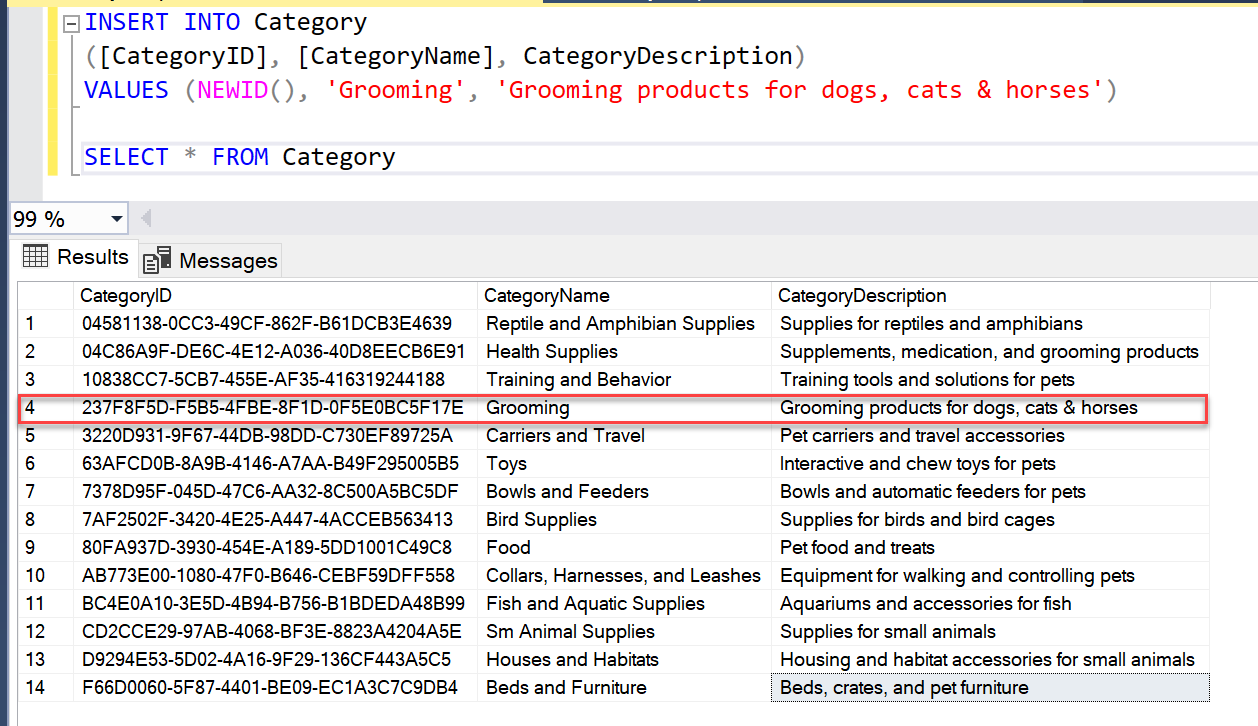
Note. This base code was modified to meet the project’s needs.

Testing the Database Schema

With the sample data sets in place, the team tested the database schema by running various test queries and data manipulation statements. These tests included SELECT, INSERT, UPDATE, and DELETE operations and the execution of application logic that interacted with the schema. This comprehensive testing approach allowed the team to evaluate the performance, correctness, and behavior of the database schema under various conditions, as seen in Figures 11 and 12, respectively.

Figure 11

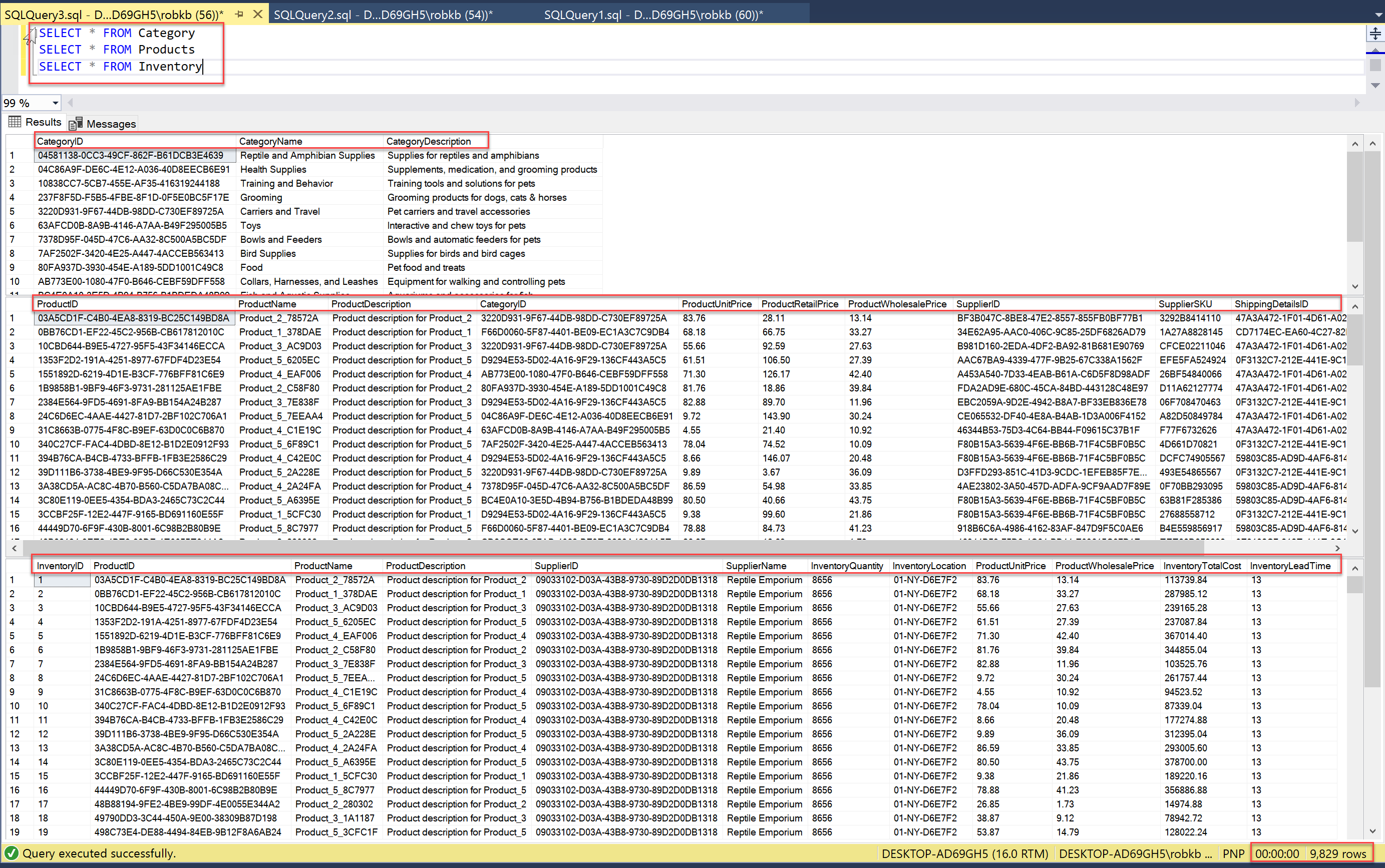
Testing INSERT Function



Note. Inserting random sample data into the Category Table.

Figure 12

Pulling multiple tables of Sample Data



Note. Sample Data from the Category, Products, and Inventory Tables.

Upon discovering any issues during the testing process, the team iterated and refined the schema and sample data generation process. By repeating the testing process until the desired performance and behavior were achieved, the team ensured that the final database schema was robust, efficient, and capable of handling the intended use case. This diligent approach to schema testing, guided by the eight key steps outlined earlier, has successfully developed the database system.

Conclusion

Despite major system configuration issues, the database development project was successfully carried out by following a systematic approach that included designing the schema, generating synthetic sample data sets, and thoroughly testing the database schema. The team leveraged SQL Server functions like RAND(), NEWID(), and NEWSEQUENTIALID() to create realistic and representative sample data. This data was then used to evaluate the performance, correctness, and behavior of the schemas under various conditions, ensuring their robustness and efficiency.

By iterating and refining the schema and sample data generation process, the team was able to address any issues discovered during testing, resulting in a high-quality and reliable database system. This rigorous approach to schema testing and development has been instrumental in ensuring that the final database solution meets the intended use case and can effectively support managing information related to customers, orders, inventory, and more.

References

Bagui, S., & Earp, R. (2011). Database design using entity-relationship diagrams (foundations of database design) (2nd ed.). Auerbach Publications.

Kroenke, D. M., Auer, D. J., Vandenberg, S. L., & Yoder, R. C. (2022). Database concepts (10th ed.) [Kindle]. Pearson Education (US).

Petkovic, D. (2020, January). Microsoft sql server 2019: A beginner's guide, seventh edition, 7th edition. O’Reilly Online Learning. <https://learning.oreilly.com/library/view/microsoft-sql-server/9781260458886/?sso_link=yes%26sso_link_from=northcentral-university>

Ramakrishnan, R., & Gehrke, J. (2002). Database management systems, 3rd edition (3rd ed.). McGraw-Hill.