Contents

[1. INTRODUCTION 3](#_Toc195933027)

[2. DATABASE IMPLEMENTATION 3](#_Toc195933028)

[2.1. Table Creation and Referential Integrity 3](#_Toc195933029)

[2.1.1 Tables 3](#_Toc195933030)

[2.1.2. Referential Integrity Constraints 3](#_Toc195933031)

[2.2. Stored Procedures 4](#_Toc195933032)

[2.2.1. GetAvailableProvidersForService 4](#_Toc195933033)

[2.2.2. UpdateProviderRating 5](#_Toc195933034)

[2.2.3. GetTopRatedProvidersByCategory 6](#_Toc195933035)

[2.2.4. CreateBookingWithXML 7](#_Toc195933036)

[2.2.5. SearchBookingsByXMLDetails 8](#_Toc195933037)

[2.2.6. UpdateBookingStatusAndXML 9](#_Toc195933038)

[2.2.7. Additional Procedures 10](#_Toc195933039)

[2.3. Triggers 10](#_Toc195933040)

[2.3.1. tr\_CheckProviderAvailability 10](#_Toc195933041)

[2.3.2. tr\_UpdateBookingOnPayment 10](#_Toc195933042)

[2.4. Views 11](#_Toc195933043)

[2.4.1. ProviderServiceDetails 11](#_Toc195933044)

[2.4.2. ServiceBookingSummary 12](#_Toc195933045)

[3. TESTING AND RESULTS 13](#_Toc195933046)

[4. INNOVATION 15](#_Toc195933047)

[4.1. XML Data Storage for Flexible Booking Details 15](#_Toc195933048)

[4.2. Advanced Provider Selection Algorithm 15](#_Toc195933049)

[4.3. Transaction Safety with Error Handling 15](#_Toc195933050)

[4.4. Status Update History in XML 16](#_Toc195933051)

[4.5. Provider Verification System 16](#_Toc195933052)

[5. CONCLUSION 16](#_Toc195933053)

[Bibliography 17](#_Toc195933054)

# 1. INTRODUCTION

The database system development project will establish functionality for HIRE home services by linking customers to professional providers for housework tasks. This report incorporates Part A design specifications to build the technical implementation of the database system on Microsoft SQL Server.

Users can schedule services through HIRE which includes repairs and plumbing services and electrical repairs and many other types of work. A database system handles information management by efficiently controlling users as well as providers and service categories together with bookings and payments and rating in a sophisticated environment.

# 2. DATABASE IMPLEMENTATION

## 2.1. Table Creation and Referential Integrity

Our implemented database schema uses normal form design to establish 10 interconnected tables. Table design includes suitable data types together with proper constraints which creates efficient storage while maintaining data integrity through defined relationships.

## 2.1.1 Tables

The core tables in our implementation include:

|  |  |
| --- | --- |
| Table Name | Description |
| Users | Stores customer information |
| Providers | Stores service provider information |
| Provider\_Ratings | Stores aggregate ratings for providers |
| Service\_Categories | Stores available service types |
| Provider\_Categories | Maps providers to service categories they offer |
| Addresses | Stores customer addresses |
| Bookings | Stores booking information with XML details |
| Ratings | Stores individual ratings from customers |
| Payment\_Methods | Stores available payment methods |
| Payments | Stores payment information for bookings |

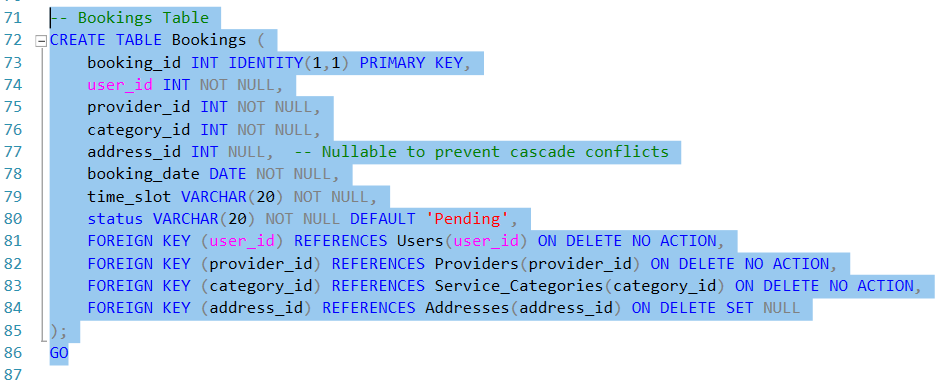
## 2.1.2. Referential Integrity Constraints

The referential integrity constraints which we establish create consistent data across the entire database. The business needs and related data structure drive our precise selection of these constraints.

The foreign key in the Provider\_Ratings table utilizes ON DELETE CASCADE to eliminate the provider ratings automatically when a provider is deleted from the Providers table. The reference makes sense since ratings lose their meaning when the provider no longer exists. We use these cascade functionalities among the tables which connect Provider\_Categories and Addresses.

The booking and payment record relationships require NO ACTION constraints for protection. The database stops booking records from removal when users, providers or service categories exist in related tables. The database implements this design to safeguard essential transaction information.

Having ON DELETE SET NULL constraints gives flexibility to specific table relationships. The DELETE EXECUTE operation protects an address\_id entry in Bookings so that a booking record remains intact during an address deletion.

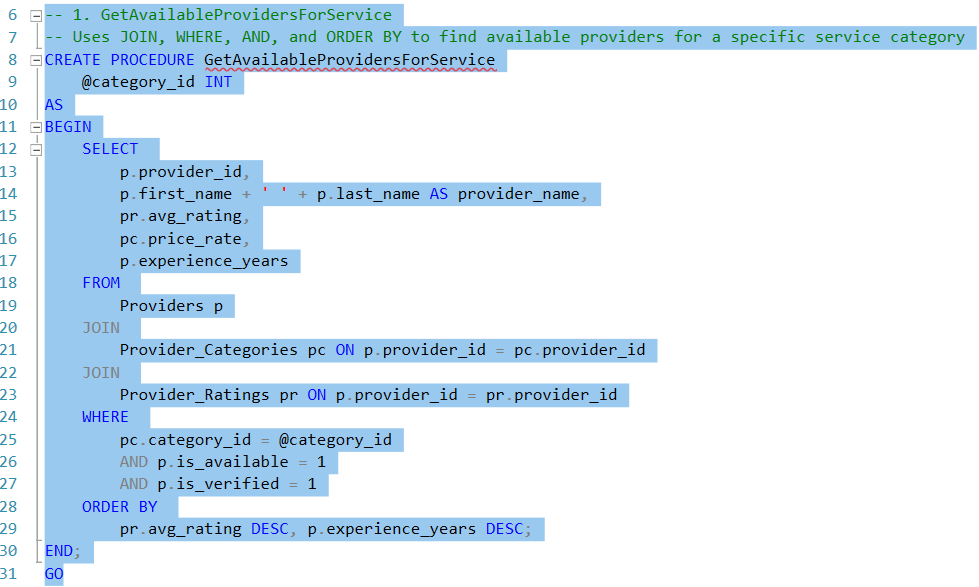


## 2.2. Stored Procedures

We implement six stored procedures to fulfill the specific requirements of the assignment, demonstrating various SQL functionalities including JOIN operations, GROUP BY with HAVING, custom functions, and comprehensive XML handling.

### 2.2.1. GetAvailableProvidersForService

The procedure supports a core platform requirement by helping users locate qualified service providers from its database. The procedure requires a category\_id parameter to provide verified providers who offer the service arranged by quality metrics.



The procedure uses multiple JOIN operations to connect three tables, filters for appropriate providers using WHERE clauses with AND conditions and applies sophisticated sorting with ORDER BY to prioritize the most qualified providers. This approach ensures customers connect with the best available service professionals.

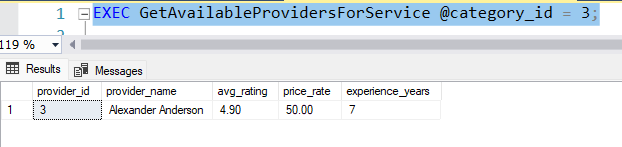
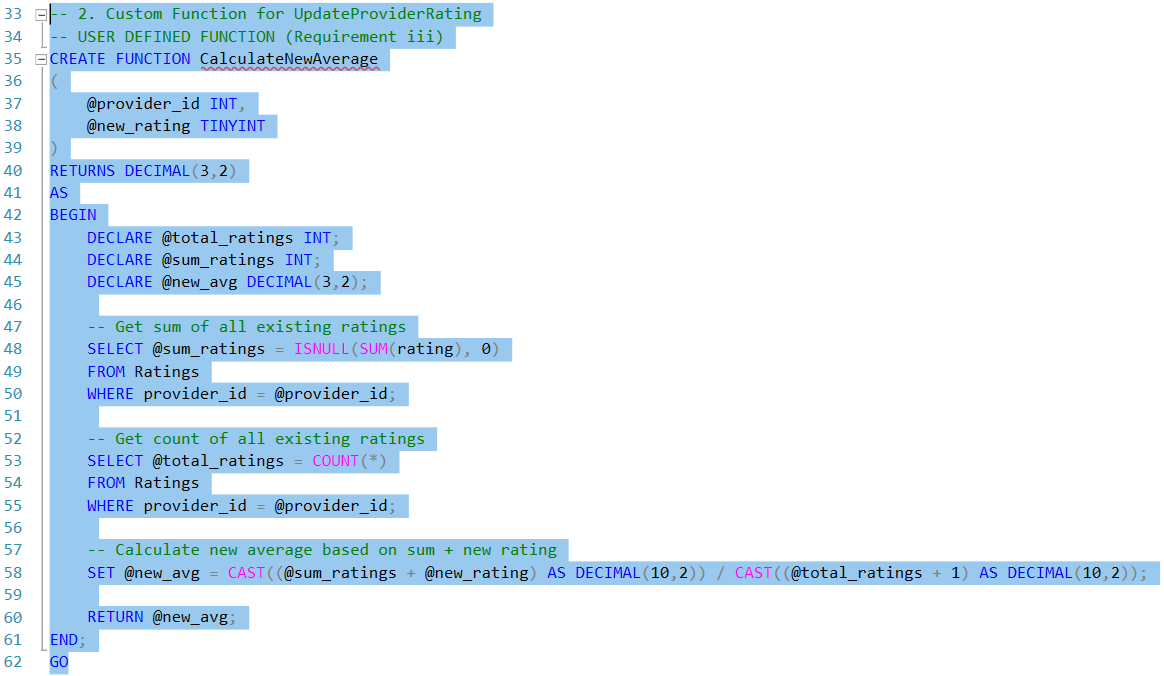


Figure: 1 Executing Stored Procedure 1

### 2.2.2. UpdateProviderRating

The customized User-Defined Functions and transaction management appear in this procedure. The system reviews provider feedback to modify their score before recalculating their rating average. The procedure combines operations with the CalculateNewAverage function to compute the updated average rating for new ratings that get submitted.

The procedure uses TRY-CATCH blocks to manage transactions as part of its error handling process. The procedure starts by validating the rating value before determining whether a rating exists for the booking to decide between updating or inserting new records and following with average score calculation. The method's complete structure maintains rating information with accurate and unified results.



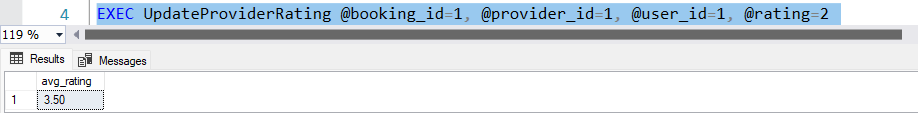
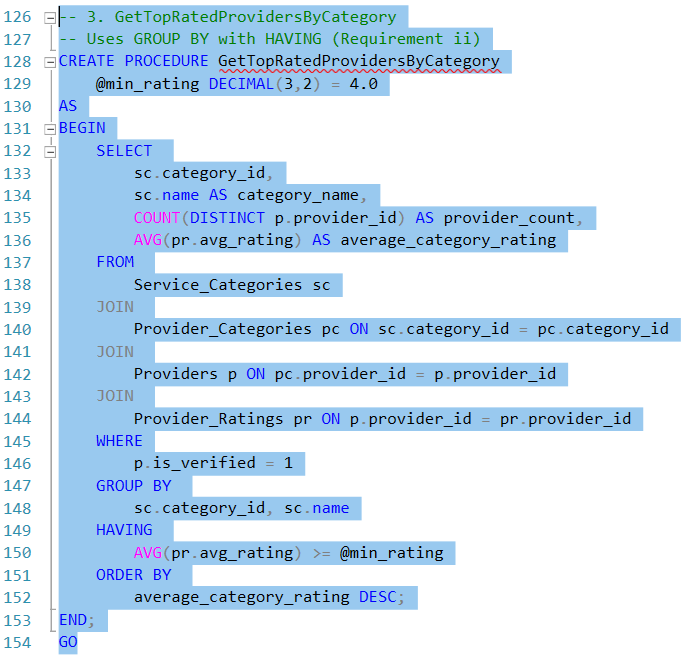


Figure: 2 Executing Stored Procedure 2

### 2.2.3. GetTopRatedProvidersByCategory

This procedure demonstrates the use of GROUP BY with HAVING clause to filter aggregate results. It accepts a minimum rating threshold parameter and returns service categories whose providers meet or exceed this rating, helping to identify high-performing service areas.

 The procedure joins multiple tables, filters for verified providers, groups results by category and then applies the HAVING clause to filter categories based on their average rating. This provides valuable business intelligence for marketing and service expansion decisions.

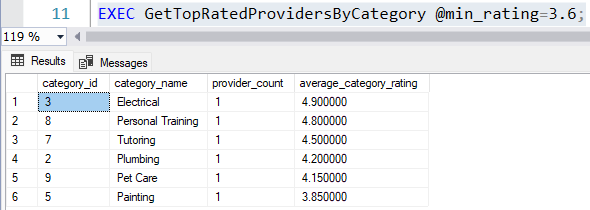
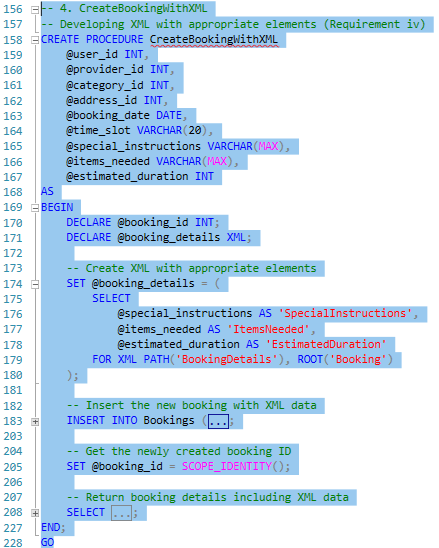


Figure: 3 Executing Stored Procedure 3

### 2.2.4. CreateBookingWithXML

This procedure demonstrates the creation of XML data from relational parameters. It accepts standard booking information plus three special parameters (special\_instructions, items\_needed, and estimated\_duration) that are stored as structured XML data.



The FOR XML PATH syntax with a ROOT element enables creation of structured XML documents as part of the procedure. Storing different booking information types becomes possible without needing table schema adjustments by the use of this method.



Figure: 4 Executing Stored Procedure 4

### 2.2.5. SearchBookingsByXMLDetails

This procedure demonstrates searching within XML data. It accepts a search term and finds bookings where either the SpecialInstructions or ItemsNeeded elements contain that term, enabling powerful free-text searching within structured XML.



The procedure executes XML .exist() method with XPath expressions to conduct searches in specific XML elements. The method unites structured database features with text-based query functionality.

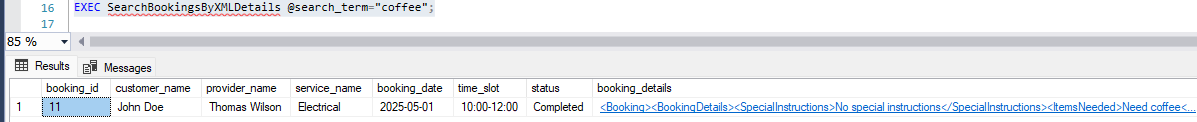


Figure: 5 Executing Stored Procedure 5

### 2.2.6. UpdateBookingStatusAndXML

The procedure shows how to edit XML data while simultaneously extracting values from XML elements. The procedure both modifies a booking status and creates a new status update entry inside the XML data for tracking changes across time.



The procedure implements the .modify() method to create a new StatusUpdate element inside the existing XML framework. The XML modifications include both data retrieval through .value() and .query() methods. The applied combination of techniques enables powerful audit tracking within each booking record.

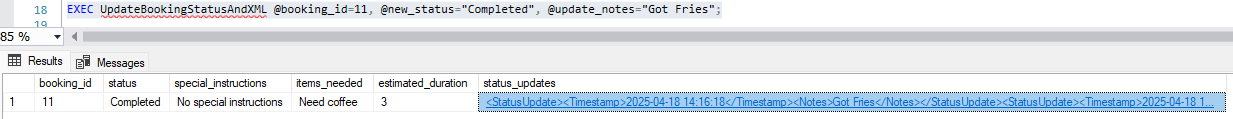


Figure: 6 Executing Stored Procedure 6

### 2.2.7. Additional Procedures

Our system contains two additional procedures referred to as GetUserBookingHistory that retrieves thorough user booking records while DeleteUserAccountAndData provides secure account deletion with transactional safeguards.

The GetUserBookingHistory procedure accomplishes its task by carrying out multiple JOIN operations to merge data from seven tables into one complete result set. Users can view their complete booking history because the system combines all necessary information such as provider profiles and payment information and user ratings.

The DeleteUserAccountAndData procedure shows advanced transaction techniques while performing explicit error handling. The procedure adopts a confirmation code security system together with a structured data removal order that follows foreign key relationships.

These procedures serve to finish our implementation by delivering necessary features which benefit platform administrators and end users.

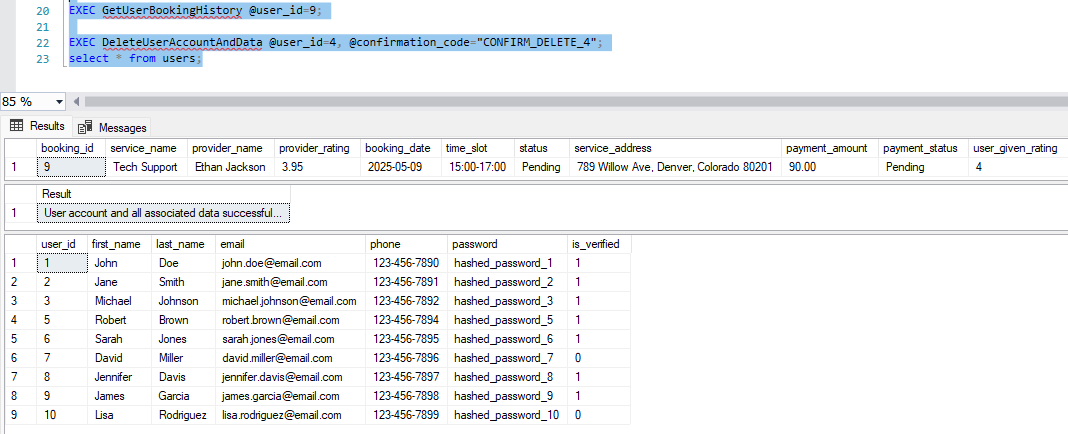


Figure: 7 Executing Stored Procedure 7 & 8

## 2.3. Triggers

Our database implementation includes two triggers that enforce critical business rules at the database level.

### 2.3.1. tr\_CheckProviderAvailability

The Bookings table contains this INSTEAD OF INSERT trigger that maintains the booking rule which allows customers to reserve providers who documented both availability and verification status. The booking creation attempt interception by the trigger verifies provider status before allowing valid reservations to continue.

### 2.3.2. tr\_UpdateBookingOnPayment

The Payments table trigger functions as an AFTER INSERT, UPDATE trigger that executes automatic booking confirmation after successful payment processing. The payment status change to 'Completed' activates the trigger to transform 'Pending' bookings into 'Confirmed' reservations through an automatic process.

## 2.4. Views

We created two views to simplify complex data access patterns and provide tailored data representations for different entities using this system.

### 2.4.1. ProviderServiceDetails

The ProviderServiceDetails view consolidates information from four different tables (Providers, Provider\_Categories, Service\_Categories, and Provider\_Ratings) to create a comprehensive provider profile in a single, easy to query virtual table.



The view simplifies provider searches by implementing complex join operations and provider verification filters within its design. The view contains readable provider name information through computed columns and showcases all necessary features for provider selection.

A simple query like the one below can instantly retrieve qualified plumbers sorted by rating:

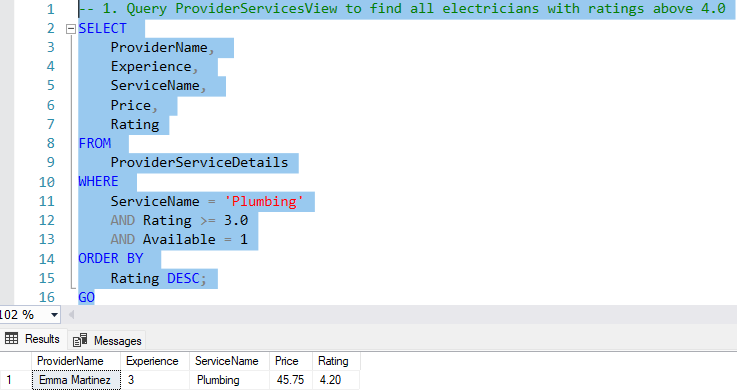
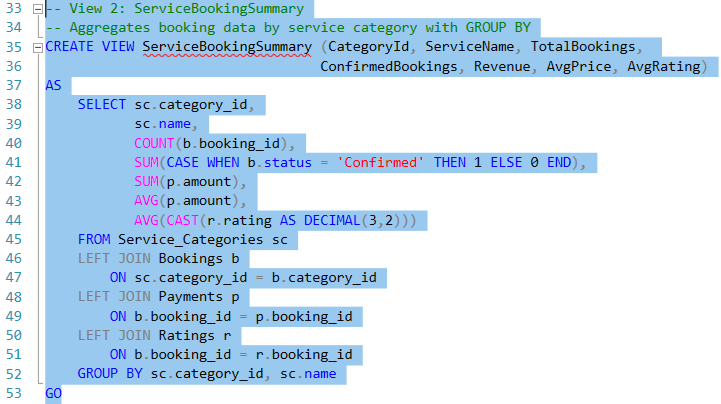


Figure: 8 Executing and Demonstrating View 1

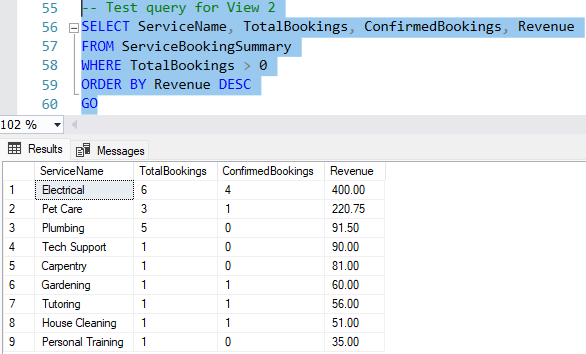
### 2.4.2. ServiceBookingSummary

The ServiceBookingSummary view delivers business intelligence needs through its functionality that combines service booking data by category. Total bookings and confirmed bookings along with revenue and average price and average rating represent the primary performance metrics computed by this view.



The view shows complex SQL usage through LEFT JOINs that retrieves all service categories and CASE expressions for conditional counting together with type casting and multiple aggregate functions and GROUP BY for accurate decimal calculations.

Business analysts can use this view to quickly identify profitable service categories:



These views improve system usability through standardized pre-made query patterns that handle typical data access requirements. Their function in application development involves simplification through encapsulated SQL complexity along with an interface maintenance capability even when tables transform.

# 3. TESTING AND RESULTS

Our testing process combines full-scale procedures to validate how each database segment works while confirming that essential business guidelines remain active. The tests verify different conditions of table structures and stored procedures and triggers and views.

Testing for tables focuses on validating the creation of tables that includes proper columns, data types alongside appropriate constraints. We specifically verify referential integrity constraints by conducting tests on boundary conditions that involve attempting to delete records linked to dependents while monitoring the behaviors of ON DELETE CASCADE, NO ACTION and SET NULL.

Our stored procedure testing includes executing each procedure with different parameter combinations:

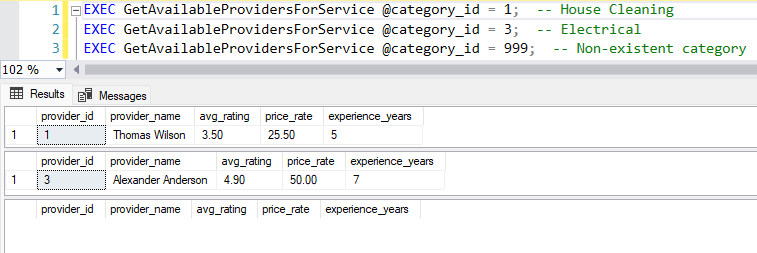


Figure: 9 Testing a Stored Procedure

Our trigger testing is particularly thorough since these components enforce critical business rules. For the tr\_CheckProviderAvailability trigger, we designed tests to verify all validation paths.

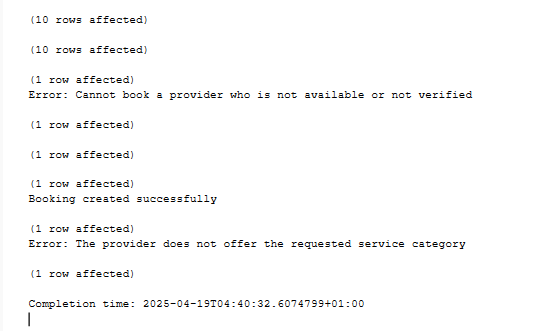


Figure: 10 Output for testing trigger 1

For our views, we executed various queries to verify they return the expected results under different filtering and sorting conditions,

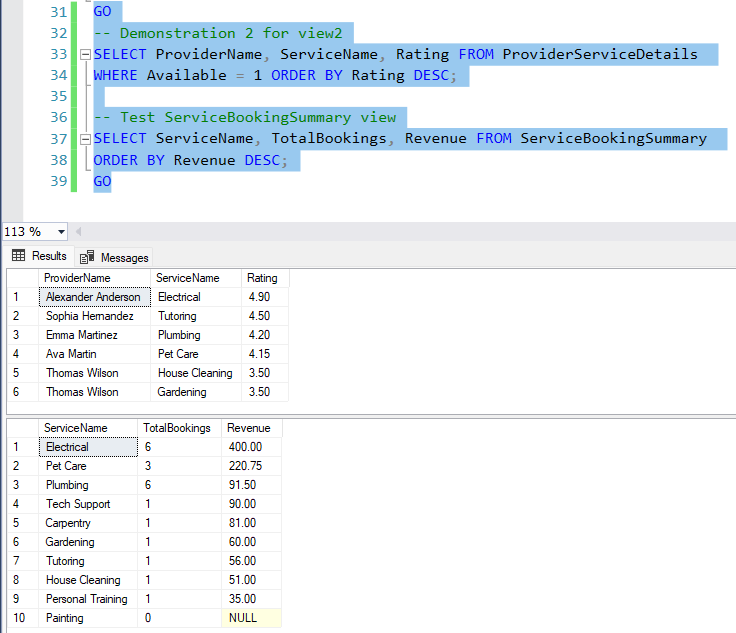


Figure: 11 Testing View 2 features

The conducted tests verified correct operation and proper enforcement of business rules throughout the HIRE platform database. The tests operate on standard operating paths in addition to testing system edge conditions to confirm database appropriate handling of different scenarios.

# 4. INNOVATION

The implementation includes multiple innovative features which boost the functionality and usability features of the database system. The new features within the database system specifically resolve home service sector challenges by offering distinct advantages to its multiple user groups.

## 4.1. XML Data Storage for Flexible Booking Details

Our application stores variable booking information with SQL Server's XML data type into a structured format that can be easily queried. Our system stores arrangeable data in a solitary XML column instead of developing specific service-type tables or numerous dedicated columns.

The platform provides advantages to entire groups of service partakers. New services become easily manageable for developers through their existing table structures. Database managers decrease the amount of time required for maintenance. Service booking details delivered to customers are designed to meet their requirements. Every service provider gets detailed task directives through the system. The support staff possesses quick access to search through booking information for solving problems.

## 4.2. Advanced Provider Selection Algorithm

The GetAvailableProvidersForService procedure employs a complex decision-making process which combines provider assessment criteria of experience level and ratings together while evaluating verification status.

This system delivers enhanced service quality to all platform members. The system connects users with the most suitable service dealers. More visibility becomes available to professional providers through both their rating system and their accumulated experience. The system achieves better customer satisfaction since it delivers improved matching ability. Novice service providers comprehend the methods which boost their market exposure.

## 4.3. Transaction Safety with Error Handling

Payment processing and account management together with rating updates utilize full transaction management because of sophisticated error handling features enabled by TRY-CATCH blocks that implement BEGIN TRANSACTION / COMMIT / ROLLBACK patterns.

System reliability increases for every user because of error handling measures within the system. Database managers encounter less data-related troubles in their work operations. The booking process contains fewer errors for customers who use the system. The system provides better conditions to locate and handle issues making it streamlined for support team members. The business obtains more precise report data through this system. The operational environment that developers encounter allows them to work using a predictable system platform.

## 4.4. Status Update History in XML

The UpdateBookingStatusAndXML procedure stores complete status change history inside booking records to establish an integrated audit process that eliminates the need for an extra history table.

When users can monitor status alterations they begin to trust the service platform. The system displays in real time all updates about service status. When handling questions support staff members possess all information available regarding these questions. Service providers receive documented evidence of all service timeline activities. Booking history becomes simple to review because all updated statuses are concentrated within one centralized location.

## 4.5. Provider Verification System

We have established a three-tier verification process that includes flag systems along with referencing documents and automated checkpoint checks for verifying professionals who can provide services.

This feature enables users to trust every section of the platform. Customers have peace of mind that they receive qualified providers through the system. The verified providers do not need to battle unverified services since customers trust the verified status. Through verification standards the platform establishes a reputation as a service quality provider. Service quality remains consistent throughout all service categories because of verification standards.

# 5. CONCLUSION

Our development of the HIRE database system satisfies all project objectives through its implementation of advanced relational database principles. The database schema shows proper interconnectedness between entities to form a strong base for platform abilities.

The stored procedures demonstrate different SQL features including XML management for booking data which allows flexible storage of custom booking contents. The triggers implement vital business constraints directly through the database system and the views offer simplified processes to access complex data structures for multiple user scenarios.

The special features our designs implement focus on solving problems found in residential service applications to make the database system more usable. The combination of exhaustive testing demonstrates that each component meets expectations during various operating scenarios.

Our solution shows both operational and theoretical knowledge of complex database concepts while delivering an effective solution to a business organization.

# BIBLIOGRAPHY

Bertwistle, A. (2023) MS SQL Server XML Data Type Tutorial, YouTube. Available at: https://www.youtube.com/watch?v=Yx4AK7VKJOQ (Accessed: 10 April 2025).

Chapple, M. (2024) 'Using Transactions in SQL Server Stored Procedures', Database Journal, 12 February. Available at: https://www.databasejournal.com/features/mssql/using-transactions-in-sql-server-stored-procedures.html (Accessed: 5 April 2025).

Dezső, B. (2024) 'Implementing Referential Integrity in SQL Server Databases', SQLShack, 18 January. Available at: https://www.sqlshack.com/implementing-referential-integrity-in-sql-server/ (Accessed: 3 April 2025).

Ghodke, S. (2024) Advanced SQL Server Triggers, YouTube. Available at: https://www.youtube.com/watch?v=kr4vOvXRQDo (Accessed: 8 April 2025).

Jones, P. (2024) 'SQL Server Views: Best Practices and Performance Optimization', MSSQLTips, 22 March. Available at: https://www.mssqltips.com/sqlservertip/7521/sql-server-views-best-practices/ (Accessed: 7 April 2025).

Microsoft (2025) 'XML Data Type and Columns (SQL Server)', Microsoft Learn. Available at: https://learn.microsoft.com/en-us/sql/relational-databases/xml/xml-data-type-and-columns-sql-server (Accessed: 6 April 2025).

Microsoft (2024) 'Handle Database Triggers with SQL Server', Microsoft Learn. Available at: https://learn.microsoft.com/en-us/sql/relational-databases/triggers/create-triggers (Accessed: 7 April 2025).

Singh, R. (2024) SQL Server Stored Procedures: Complete Guide, YouTube. Available at: https://www.youtube.com/watch?v=fjNsRV4zFdc (Accessed: 4 April 2025).

Thangaraj, J. and Nascimento, L. (2025) Advanced Databases Module Guide. Dublin Business School.

Wilson, T. (2024) 'Implementing Error Handling in SQL Server', SQLServerCentral, 14 February. Available at: https://www.sqlservercentral.com/articles/implementing-error-handling-in-sql-server (Accessed: 6 April 2025).

Young, S. (2025) Database Implementation for Home Service Applications, YouTube. Available at: https://www.youtube.com/watch?v=YtMR8h2VQLk (Accessed: 2 April 2025).