

Statement and Confirmation of Own Work

***A signed copy of this form must be submitted with every assignment.***

***If the statement is missing your work may not be marked.***

***For Level 7 assignments, please use the separate Level 7 Candidate***

***Statement of Own Work form instead of this one.***

**Student Declaration**

I confirm the following details:

|  |  |
| --- | --- |
| **Candidate Name:** | Saw Myat Mon Hnin |
| **Candidate ID Number:** | 202698 |
| **Qualification:** | Level 5 Diploma in Computing |
| **Unit:** | Database Design and Development |
| **Centre:** | Strategy First University |
| **Word Count:** |  |
| I have read and understood both NCC Education’s Academic Misconduct Policy and the Referencing and Bibliographies document. To the best of my knowledge my work has been accurately referenced and all sources cited correctly.  I confirm that I have not exceeded the stipulated word limit by more than 10%.  I confirm that this is my own work and that I have not colluded or plagiarised any part of it. | |
| **Candidate Signature:** | **Myat** |
| **Date:** | 5/8/2024 |

Contents

[Task-1 4](#_Toc172753045)

[**Scenario of Banking Loan System** 5](#_Toc172753046)

[**Entities** 5](#_Toc172753047)

[**Transactions** 6](#_Toc172753048)

[**Abaddon’s Loan Form** 7](#_Toc172753049)

[**Limitations** 7](#_Toc172753050)

[Task-2 9](#_Toc172753051)

[**Entity Relationship Diagram** 10](#_Toc172753052)

[**Data Dictionary** 10](#_Toc172753053)

[**Registration** 10](#_Toc172753054)

[**Customer** 10](#_Toc172753055)

[**Customer Type** 11](#_Toc172753056)

[**Loan** 11](#_Toc172753057)

[**Loan Type** 11](#_Toc172753058)

[**Loan\_Rule** 12](#_Toc172753059)

[**Rule** 12](#_Toc172753060)

[**Staff** 12](#_Toc172753061)

[**Staff Type** 12](#_Toc172753062)

[**Register\_Collateral** 12](#_Toc172753063)

[**Collateral** 13](#_Toc172753064)

[**Collateral\_Type** 13](#_Toc172753065)

[Task-3 14](#_Toc172753066)

[**Normalization** 15](#_Toc172753067)

[**Optimization Table** 16](#_Toc172753068)

[**Anomalies of Normalization** 17](#_Toc172753069)

[Task-4 18](#_Toc172753070)

[**Assessment of Design** 19](#_Toc172753071)

[Task-5 20](#_Toc172753072)

[**Create Table** 21](#_Toc172753073)

[Task-6 27](#_Toc172753074)

[**Insert Table** 28](#_Toc172753075)

[Task-7 35](#_Toc172753076)

[**Select Data** 36](#_Toc172753077)

[Task-8 41](#_Toc172753078)

[**Future development of a distributed database** 42](#_Toc172753079)

[Task-9 44](#_Toc172753080)

[**Future Development of a Data Warehouse** 45](#_Toc172753081)

[References 46](#_Toc172753082)

# Task-1

## **Scenario of Banking Loan System**

Abaddon Bank, based in Yangon, Myanmar, started its journey in 2010. By 2019, the bank had expanded its services to include a loan system. Initially, all loan-related data was managed using Excel spreadsheets, which seemed practical at the time. However, as the number of loans and customers increased, this method led to numerous issues and errors, making it clear that a more reliable solution was needed. To address these challenges, Abaddon Bank decided to transition to a database system, aiming to improve data accuracy, streamline operations, and enhance security. This move was crucial for supporting the bank’s growth and ensuring better for its customers.

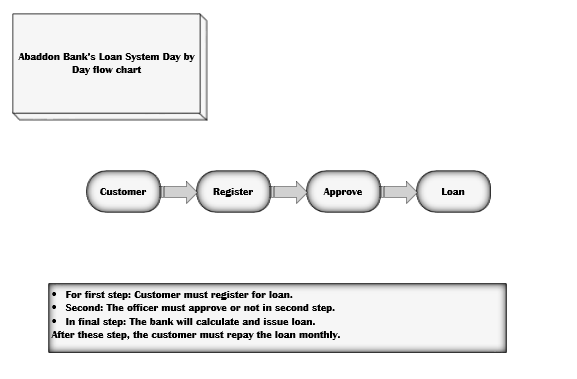


Figure 1 Day by Day flow chart

At Abaddon Bank, Customer must register their collateral for loan. The collateral has collateral type and collateral detail information. Abaddon’s staff approved or not the registration according to the rules set by the bank. After then, the bank calculates the loan type and % for loan. In the end, the bank issue the loan and customer repay the loan monthly.

## **Entities**

1. Customer
2. Customer\_Type
3. Registration
4. Loan
5. Loan\_Type
6. Loan\_Rule
7. Rule
8. Staff
9. Staff\_Type
10. Register\_Collateral
11. Collateral
12. Collateral\_Type

## **Transactions**

* Customer entity- Insert, update or delete customer data.
* Customer\_Type- Insert, update or delete customer type data.
* Registration- Insert, update or delete registration data.
* Loan- Insert, update or delete loan information.
* Loan\_Type- Update loan type.
* Loan\_Rule- Insert or delete loan rule link.
* Rule- Insert, update or delete rule.
* Staff- Insert, update or delete staff data.
* Staff\_Type- Insert, update or delete staff type data.
* Register\_Collateral- Insert or delete register collateral link.
* Collateral- Insert, update or delete collateral information.
* Collateral\_Type- Insert or update collateral type.

## **Abaddon’s Loan Form**

The Loan Form for Abaddon Bank is designed to collect all necessary information for processing loan database. It includes registration information, customer data, staff data, collateral data, rules and loan information.



Abaddon’s Loan Form

## **Limitations**

It has several limitations despite the thorough design of Abaddon Bank’s database system. While it focuses on detailed loan management that it doesn’t track or display payment transactions. This gap restricts the ability to conduct detailed financial audits or analyze payment trends. It’s impacting financial planning and the identification of payment-related issues. Additionally, the system may lack advanced user access controls, permissions and potentially allowing unauthorized data access and modifications. Addressing these limitations will be crucial for enhancing the bank.

# Task-2

## **Entity Relationship Diagram**

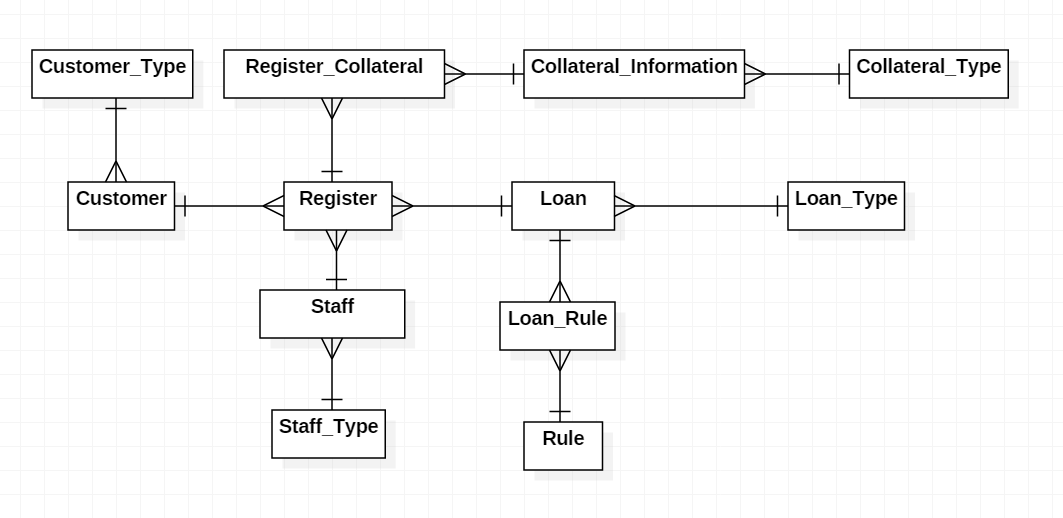


Figure 2 Registration Form

## **Data Dictionary**

### **Registration**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Datatype | Length | Constraints | Description |
| Register\_Id | Char | 10 | Primary Key, Not Null | Registration id |
| Registration\_Date | Date |  | Not Null | Registration date |
| Approve | Varchar | 20 | Not Null | Registration that approves for loan |
| Customer\_Account\_Id | Char | 10 | Foreign Key, Not Null | Customer bank account id |
| Loan\_Id | Char | 10 | Foreign Key, Not Null | Loan id |
| Staff\_Id | Char | 10 | Foreign Key, Not Null | Staff id |

### **Customer**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Datatype | Length | Constraints | Description |
| Customer\_Account\_Id | Char | 10 | Primary Key, Not Null | Customer bank account id |
| Customer\_Name | Varchar | 50 | Not Null | Customer’s name |
| Address | Varchar | 100 | Not Null | Customer’s address |
| NRC | Char | 20 | Not Null | Customer’s NRC |
| Email | Varchar | 20 | Not Null | Customer’s email |
| Customer\_Type\_Id | Char | 10 | Foreign Key, Not Null | Customer type’s id |

### **Customer Type**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Datatype | Length | Constraints | Description |
| Customer\_Type\_Id | Char | 10 | Primary Key, Not Null | Customer type’s id |
| Customer\_Type\_Description | Varchar | 50 | Not Null | Description of customer type |

### **Loan**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Datatype | Length | Constraints | Description |
| Loan\_Id | Char | 10 | Primary Key, Not Null | Loan’s id |
| Amount\_Accepted\_Percentage | int | 5 | Not Null | % for loan |
| Start\_Date | Date |  | Not Null | Loan start date |
| End\_Date | Date |  | Not Null | Loan end date |
| Status | Varchar | 20 | Not Null | Continue or complete |
| Loan\_Type\_Id | Char | 10 | Foreign Key, Not Null | Loan type’s id |

### **Loan Type**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Datatype | Length | Constraints | Description |
| Loan\_Type\_Id | Char | 10 | Primary Key, Not Null | Loan type’s id |
| Type\_Description | Varchar | 20 | Not Null | Type of Loan |

### **Loan\_Rule**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Datatype | Length | Constraints | Description |
| Loan\_Id | Char | 10 | Primary Key, Foreign Key, Not Null | Loan’s id |
| Rule\_Id | Char | 10 | Primary Key, Foreign Key, Not Null | Rule’s id |

### **Rule**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Datatype | Length | Constraints | Description |
| Rule\_Id | Char | 10 | Primary Key, Not Null | Rule’s id |
| Rule\_Description | Varchar | 100 | Not Null | Rule’s Description |

### **Staff**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Datatype | Length | Constraints | Description |
| Staff\_Id | Char | 10 | Primary Key, Not Null | Staff’s id |
| Staff\_Name | Varchar | 30 | Not Null | Staff’s name |
| Staff\_Type\_Id | Char | 10 | Foreign Key, Not Null | Staff type’s id |

### **Staff Type**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Datatype | Length | Constraints | Description |
| Staff\_Type\_Id | Char | 10 | Primary Key, Not Null | Staff type’s id |
| Staff\_Type\_Description | Varchar | 20 | Not Null | Position of staff |
| Department | Varchar | 20 | Not Null | Department that staff work |

### **Register\_Collateral**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Datatype | Length | Constraints | Description |
| Collateral\_Id | Char | 10 | Primary Key, Foreign Key, Not Null | Collateral’s id |
| Register\_Id | Char | 10 | Primary Key, Foreign Key**,** Not Null | Registration’s id |

### **Collateral**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Datatype | Length | Constraints | Description |
| Collateral\_Id | Char | 10 | Primary Key, Not Null | Collateral’s id |
| Collateral\_Description | Varchar | 30 | Not Null | Description of collateral |
| Location | Varchar | 50 | Null | Collateral’s location |
| Estimated\_Value | Dec | 15,2 | Not Null | Collateral’s value |
| Owenership\_Status | Varchar | 50 | Not Null | To record the owner |
| Collateral\_Type\_Id | Char | 10 | Foreign Key, Not Null | Collateral type’s id |

### **Collateral\_Type**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Datatype | Length | Constraints | Description |
| Collateral\_Type\_Id | Char | 10 | Primary Key, Not Null | Collateral type’s id |
| Type\_Description | Varchar | 50 | Not Null | Type of Collateral |

# Task-3

## **Normalization**

Normalization helps organize a database to reduce duplicate data and keep it accurate. For Abaddon Bank, this means more reliable and consistent information. It improves performance, making data retrieval faster and easier. By eliminating errors, normalization ensures the bank can make decisions based on accurate data (JavaTPoint, 2021).

|  |  |  |  |
| --- | --- | --- | --- |
| UNF | 1NF | 2NF | 3NF |
| Register\_Id  Registration\_Date  Approve  Customer\_Account\_Id  Customer\_Name  Address  Phone\_Number  NRC  Email  Customer\_Type\_Id  Customer\_Type\_Description  Loan\_Id  Amount\_Accepted\_Percentage  Start\_Date  End\_Date  Status  Loan\_Type\_Id  Type\_Description  Rule\_Id  Rule\_Description  Staff\_Id  Staff\_Name  Staff\_Type\_Id  Staff\_Type\_Description  Department  Collateral\_Id  Collateral\_Description  Location  Estimated\_Value  Ownership\_Status  Collateral\_Type\_Id  Type\_Description | Register\_Id  Registration\_Date  Approve  Customer\_Account\_Id  Customer\_Name  Address  Phone\_Number  NRC  Email  Customer\_Type\_Id  Customer\_Type\_Description  Loan\_Id  Amount\_Accepted\_Percentage  Start\_Date  End\_Date  Status  Loan\_Type\_Id  Type\_Description  Rule\_Id  Rule\_Description  Staff\_Id  Staff\_Name  Staff\_Type\_Id  Staff\_Type\_Description  Department  Collateral\_Id  Register\_Id\*  Collateral\_Description  Location  Estimated\_Value  Ownership\_Status  Collateral\_Type\_Id  Type\_Description | Register\_Id  Registration\_Date  Approve  Customer\_Account\_Id  Customer\_Name  Address  Phone\_Number  NRC  Email  Customer\_Type\_Id  Customer\_Type\_Description  Loan\_Id  Amount\_Accepted\_Percentage  Start\_Date  End\_Date  Status  Loan\_Type\_Id  Type\_Description  Rule\_Id  Rule\_Description  Staff\_Id  Staff\_Name  Staff\_Type\_Id  Staff\_Type\_Description  Department  Collateral\_Id  Register\_Id\*  Collateral\_Id  Collateral\_Description  Location  Estimated\_Value  Ownership\_Status  Collateral\_Type\_Id  Type\_Description | **Registration**  Register\_Id  Customer\_Account\_Id\*  Loan\_Id\*  Staff\_Id\*  Registration\_Date  Approve  **Customer**  Customer\_Account\_Id  Customer\_Type\_Id\*  Customer\_Name  Address  Phone\_Number  NRC  Email  **Customer\_Type**  Customer\_Type\_Id  Customer\_Type\_Description  **Loan**  Loan\_Id  Loan\_Type\_Id\*  Amount\_Accepted\_Percentage  Start\_Date  End\_Date  Status  **Loan\_Type**  Loan\_Type\_Id  Type\_Description  **Loan\_Rule**  Loan\_Id\*  Rule\_Id\*  **Rule**  Rule\_Id  Rule\_Description  **Staff**  Staff\_Id  Staff\_Type\_Id\*  Staff\_Name  **Staff\_Type**  Staff\_Type\_Id  Staff\_Type\_Description  Department  **Register\_Collateral**  Collateral\_Id\*  Register\_Id\*  **Collateral**  Collateral\_Id  Collateral\_Type\_Id\*  Collateral\_Description  Location  Estimated\_Value  Ownership\_Status  **Collateral\_Type**  Collateral\_Type\_Id  Type\_Description |

## **Optimization Table**

|  |  |  |
| --- | --- | --- |
| **Registration**  Register\_Id (PK)  Customer\_Account\_Id (FK)  Loan\_Id\*  Staff\_Id\*  Registration\_Date  Approve | **Customer**  Customer\_Account\_Id (PK)  Customer\_Type\_Id (FK)  Customer\_Name  Address  Phone\_Number  NRC  Email | **Customer\_Type**  Customer\_Type\_Id (PK)  Customer\_Type\_Description |
| **Loan**  Loan\_Id (PK)  Loan\_Type\_Id (FK)  Amount\_Accepted\_Percentage  Start\_Date  End\_Date  Status | **Loan\_Type**  Loan\_Type\_Id (PK)  Type\_Description | **Loan\_Rule**  Loan\_Id (PK)(FK)  Rule\_Id (PK)(FK) |
| **Rule**  Rule\_Id (PK)  Rule\_Description | **Staff**  Staff\_Id (PK)  Staff\_Type\_Id (FK)  Staff\_Name | **Staff\_Type**  Staff\_Type\_Id (PK)  Staff\_Type\_Description  Department |
| **Register\_Collateral**  Collateral\_Id (PK) (FK)  Register\_Id (PK) (FK) | **Collateral**  Collateral\_Id (PK)  Collateral\_Description  Location  Estimated\_Value  Ownership\_Status | **Collateral\_Type**  Collateral\_Type\_Id (PK)  Type\_Description |

## **Anomalies of Normalization**

Normalization solved some of Abaddon Bank’s problems. Some examples are:

1. Insert Anomaly

* Adding a new loan type without immediately linking it to a loan can create orphaned data entries.

1. Update Anomaly

* Updating “Manager” to “Senior Manager” involves modifying each staff record with this type, which can lead to inconsistencies if some rows are not updated correctly.

1. Delete Anomaly

* Removing a loan that is the only instance of a specific type (e.g., “Home Loan”) would delete the loan type description, losing information about that loan type.

# Task-4

## **Assessment of Design**

For Abaddon Bank, the logical database design was turned into a physical design using SQL scripts. These logical entities, attributes, and relationships from the design phase were translated into physical tables, columns, and constraints. This ensured that the abstract design was accurately implemented, allowing for efficient data storage and retrieval. The transition involved specifying data types, and setting up primary and foreign keys to maintain data integrity (GeeksforGeeks, 2021).

The tables for Abaddon Bank were designed carefully to meet of MySQL requirements. Each table had appropriate columns, data types, and constraints to matched the logical model. Primary keys were set to uniquely identify each record, and foreign keys enforced relationships between tables. The SQL script created a strong database structure, capable of supporting the bank’s data operations. The tables included Customer\_Type, Loan\_Type, Rule, Staff\_Type, Collateral\_Type, Customer, Loan, Staff, Collateral, Register, Register\_Collateral, and Loan\_Rule (www.tutorialspoint.com,n.d.).

Derived data such as data that can be calculated or inferred from existing data that was identified and represented through SQL queries and views. For example, the total outstanding loan amount for a customer could be calculate using aggregate functions. This kept the database normalized and avoided redundancy while providing necessary information through calculated results (www.tutorialspoint.com,n.d.).

De-normalization was also considered to enhance query performance and simplify data retrieval for Abaddon Bank. While the initial design followed normalization principles to maintain data integrity and avoid redundancy and some scenarios necessitated de-normalization to optimize performance. The involved combining related tables and duplicating some data such as customer details and their loan information to ensure efficient query execution. While extensive de-normalization was not needed, the approach was planned to balance performance optimization and maintaining a logical database structure tailored to the bank’s needs (SearchDataManagement, n.d.).

# Task-5

## **Create Table**

SQL scripts were developed to define the tables, columns, primary keys, foreign keys, and other database objects to create the database for Abaddon Bank. These scripts were run in the dbForge environment, using MySQL as the DBMS (Database Management System). The process started with the parent tables which do not have foreign keys to provent any foreign key constraint errors. These tables included Customer\_Type, Loan\_Type, Rule, Staff\_Type, and Collateral\_Type. After the parent tables were set up, the next step was to create the child tables. These tables included foreign keys to ensure the correct relationships between them. These child tables were Customer, Loan, Staff, Collateral, Regiser, Register\_Collateral, and Loan\_Rule.

SQL commands were used in XAMPP to create tables, columns, and set keys during database development. This allowed for direct management of the database and quick resolution of any errors. Occasionally, syntax errors occurred, like incorrect data types or missing commas, but these were fixed by checking the code.

Using SQL made it easy to organize the database effectively. It ensured the database structure was well-designed and data integrity was maintained, making it reliable for bank operations.

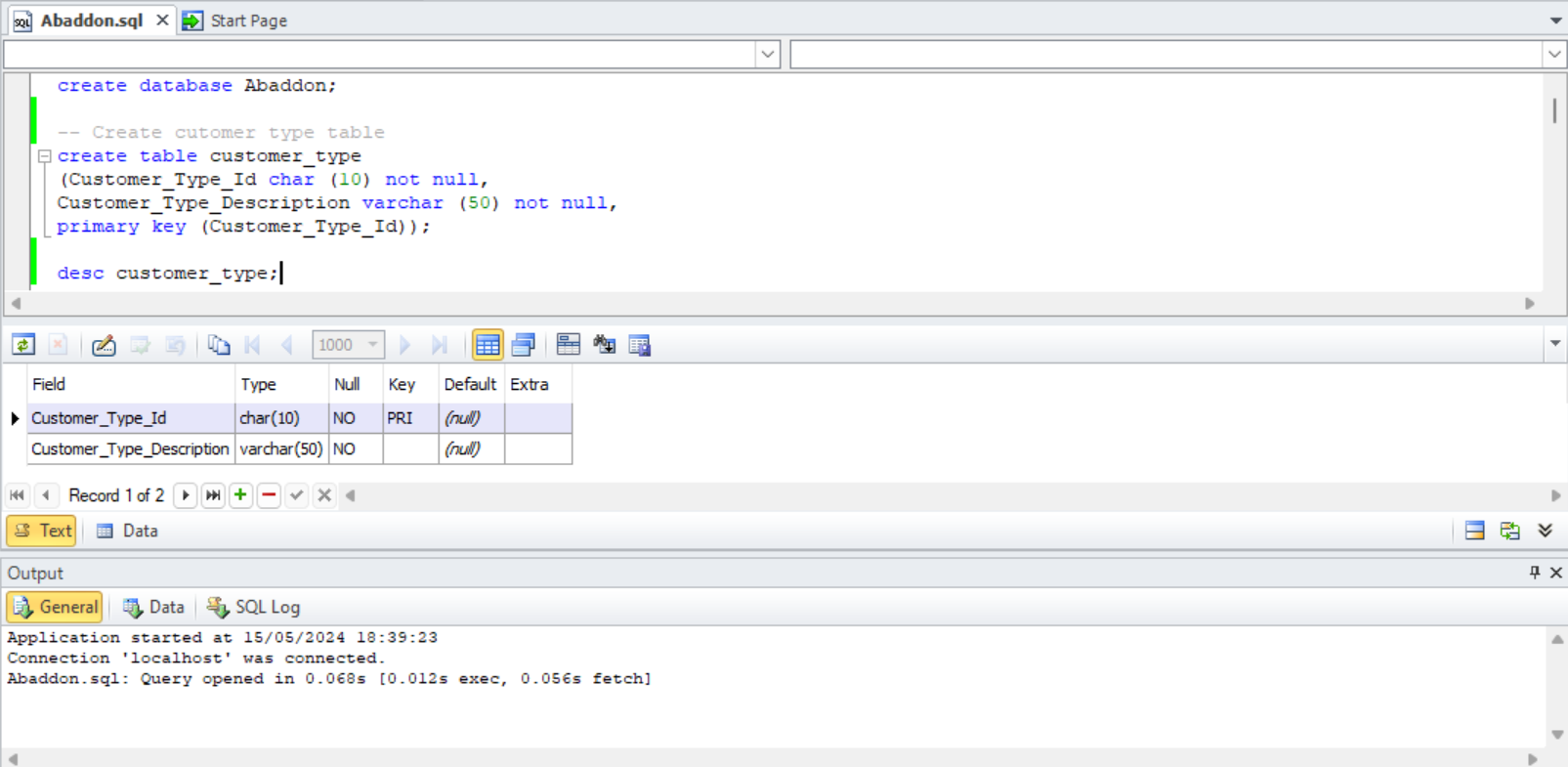


Fig 5.1 Create customer\_type table

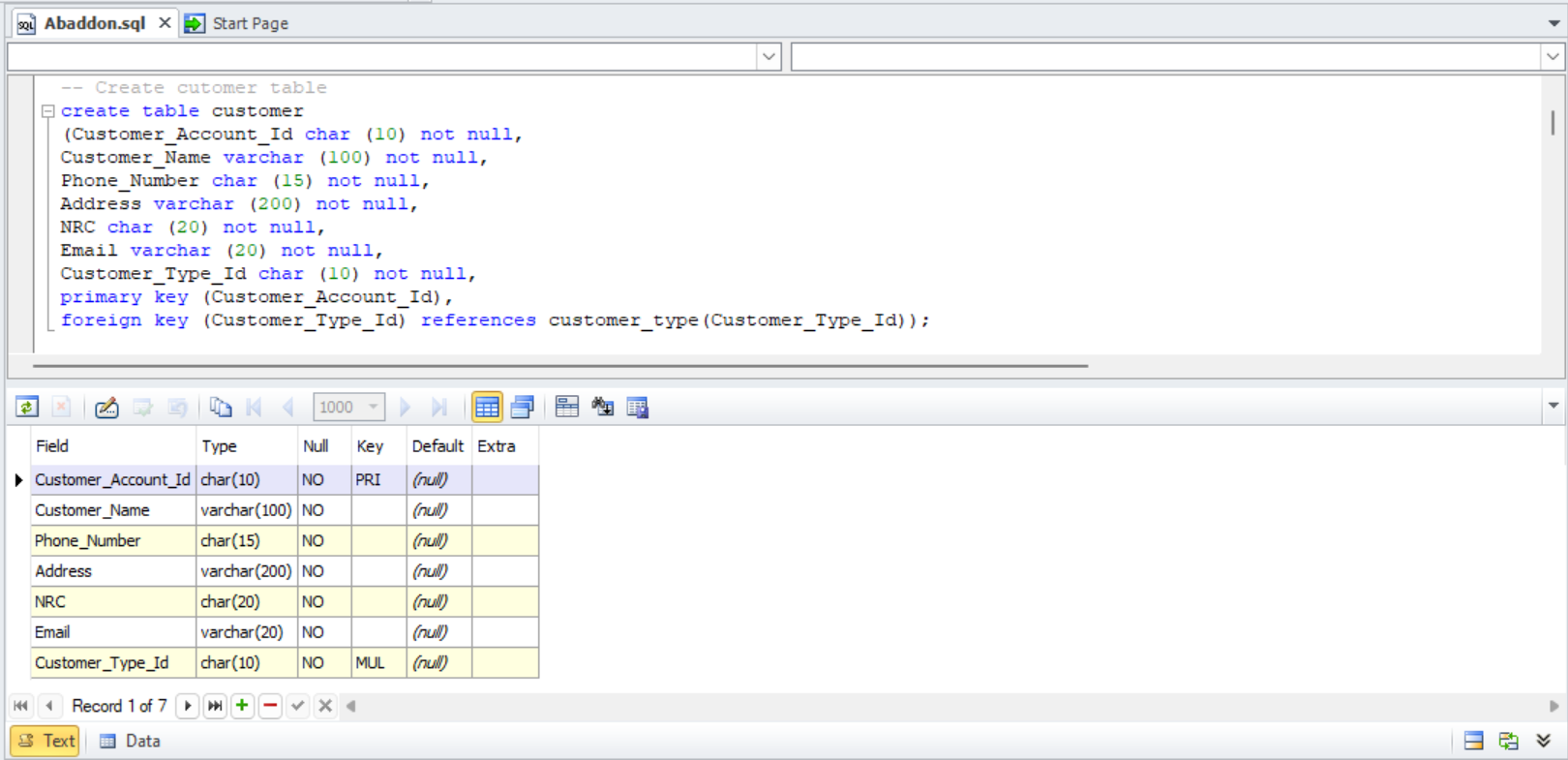


Fig 5.2 Create customer table

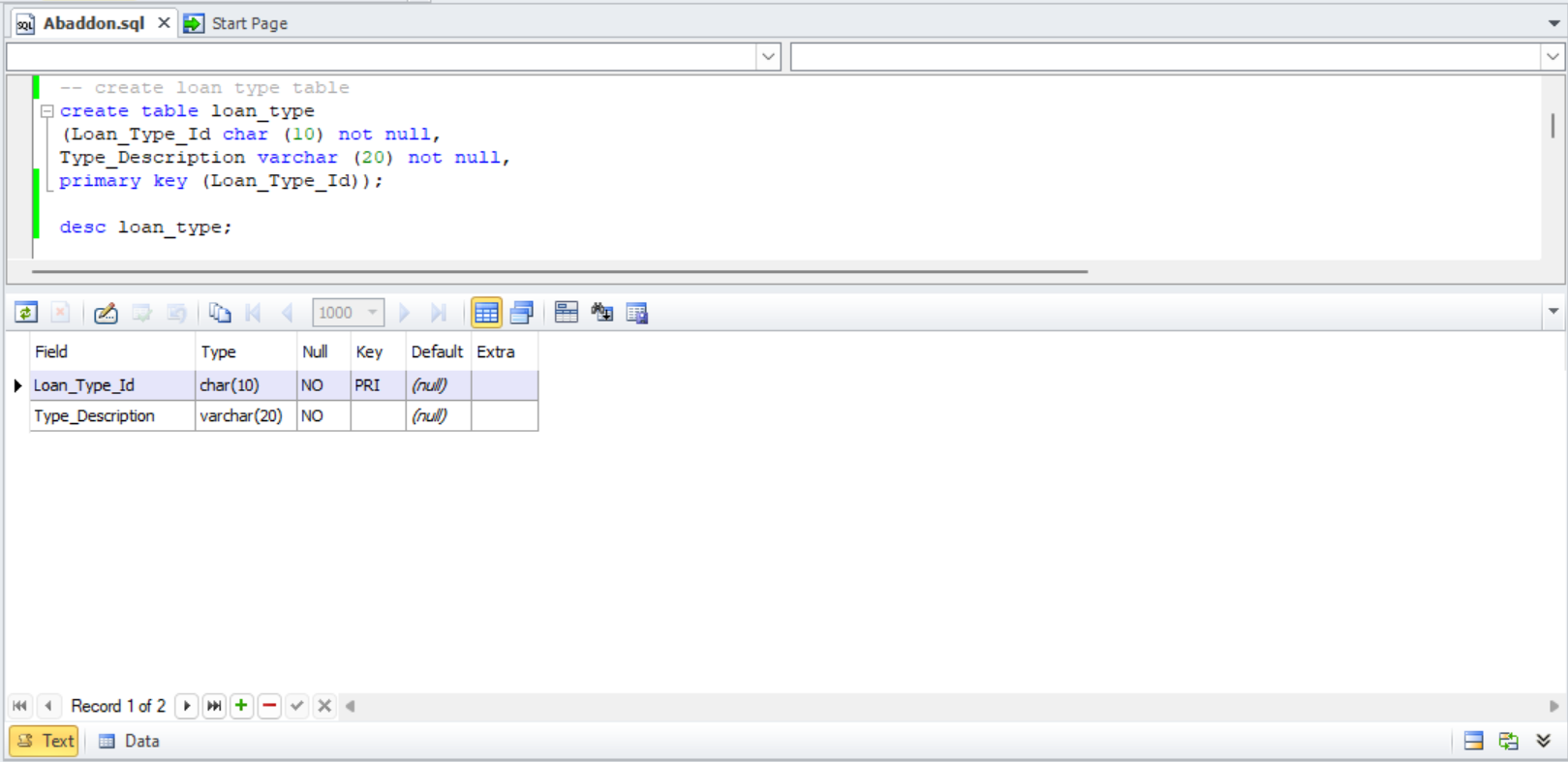


Fig 5.3 Create loan\_type table

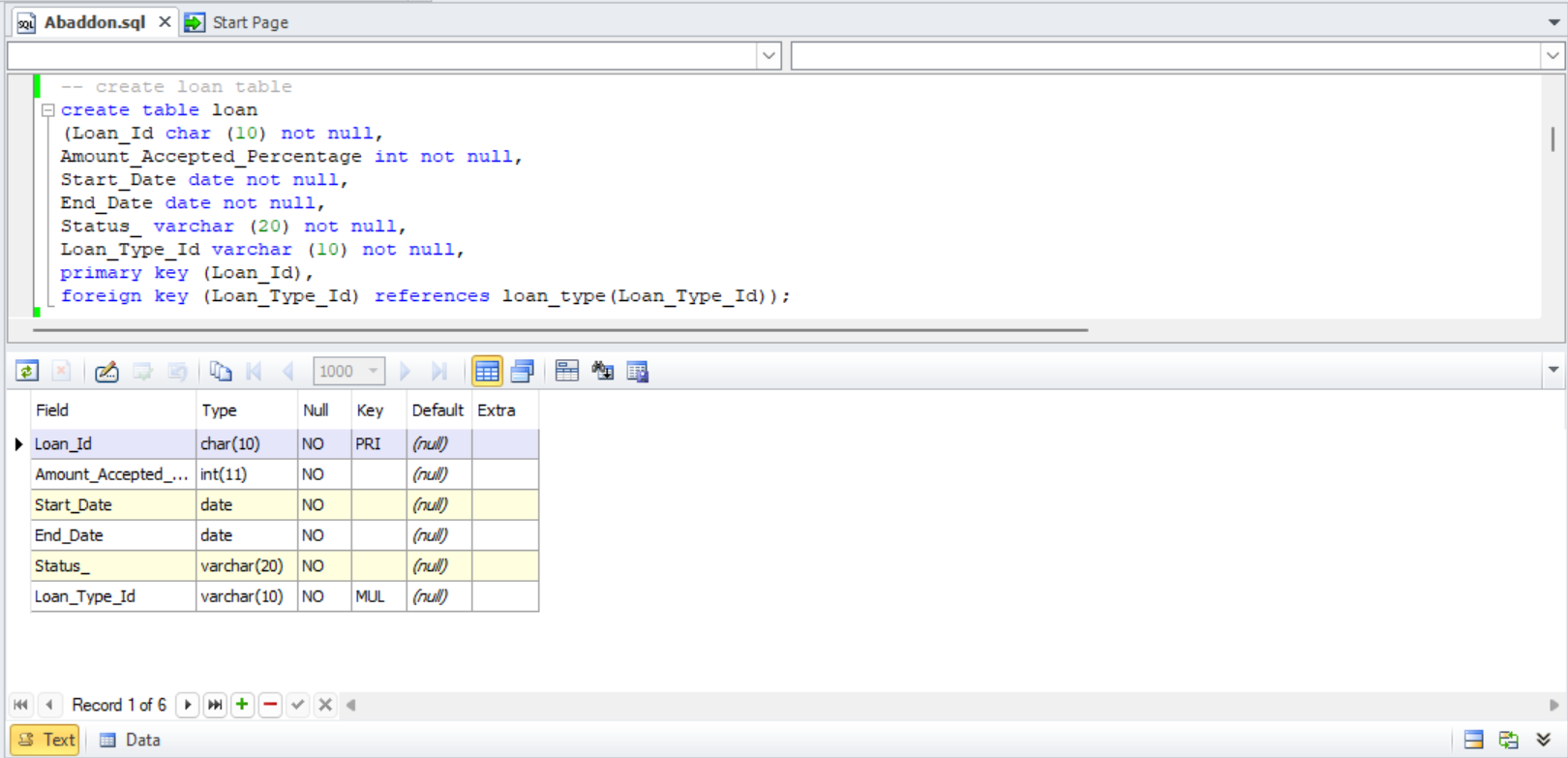


Fig 5.4 Create loan table

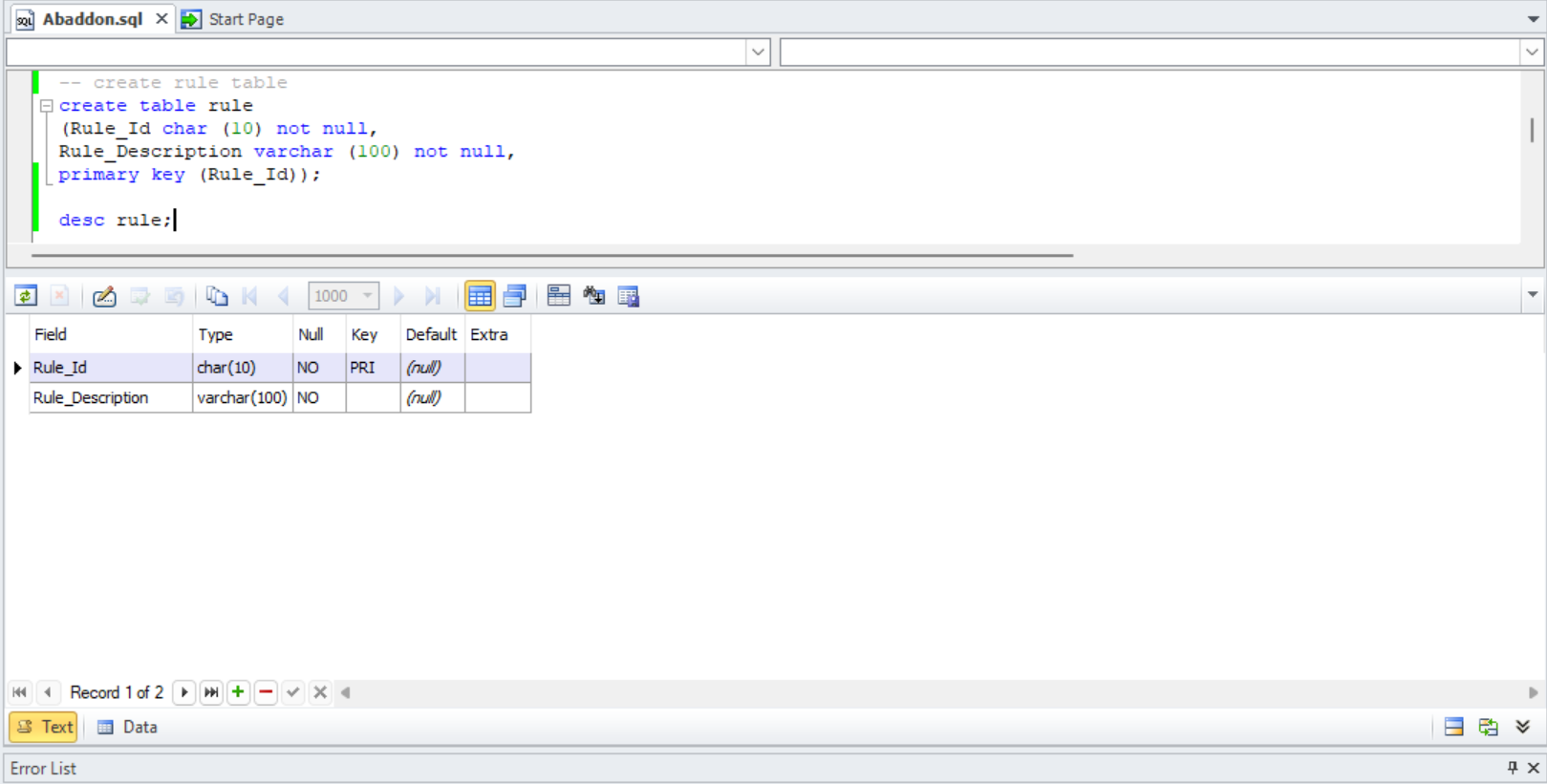


Fig 5.5 Create rule table

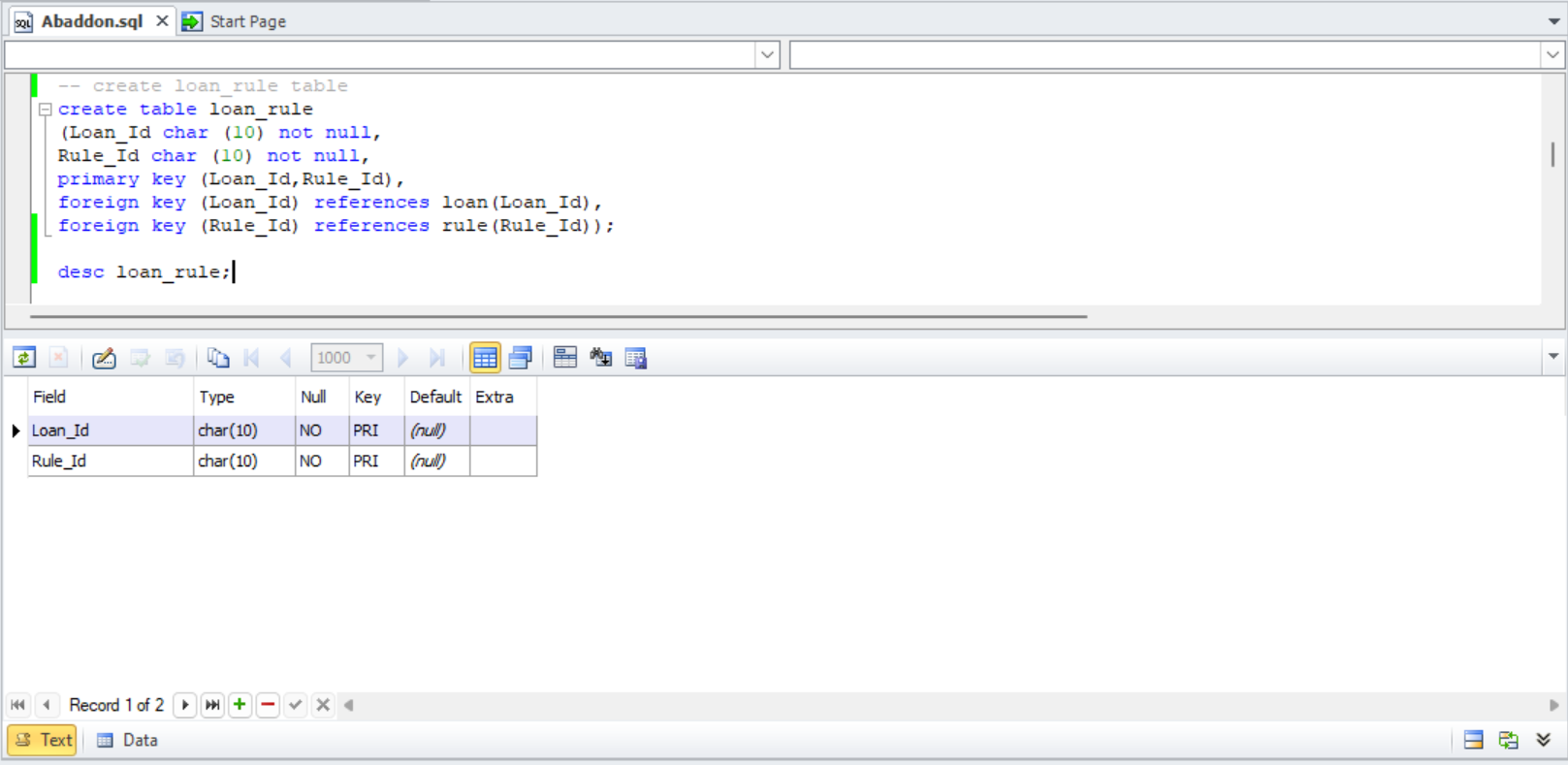


Fig 5.6 Create loan\_rule table

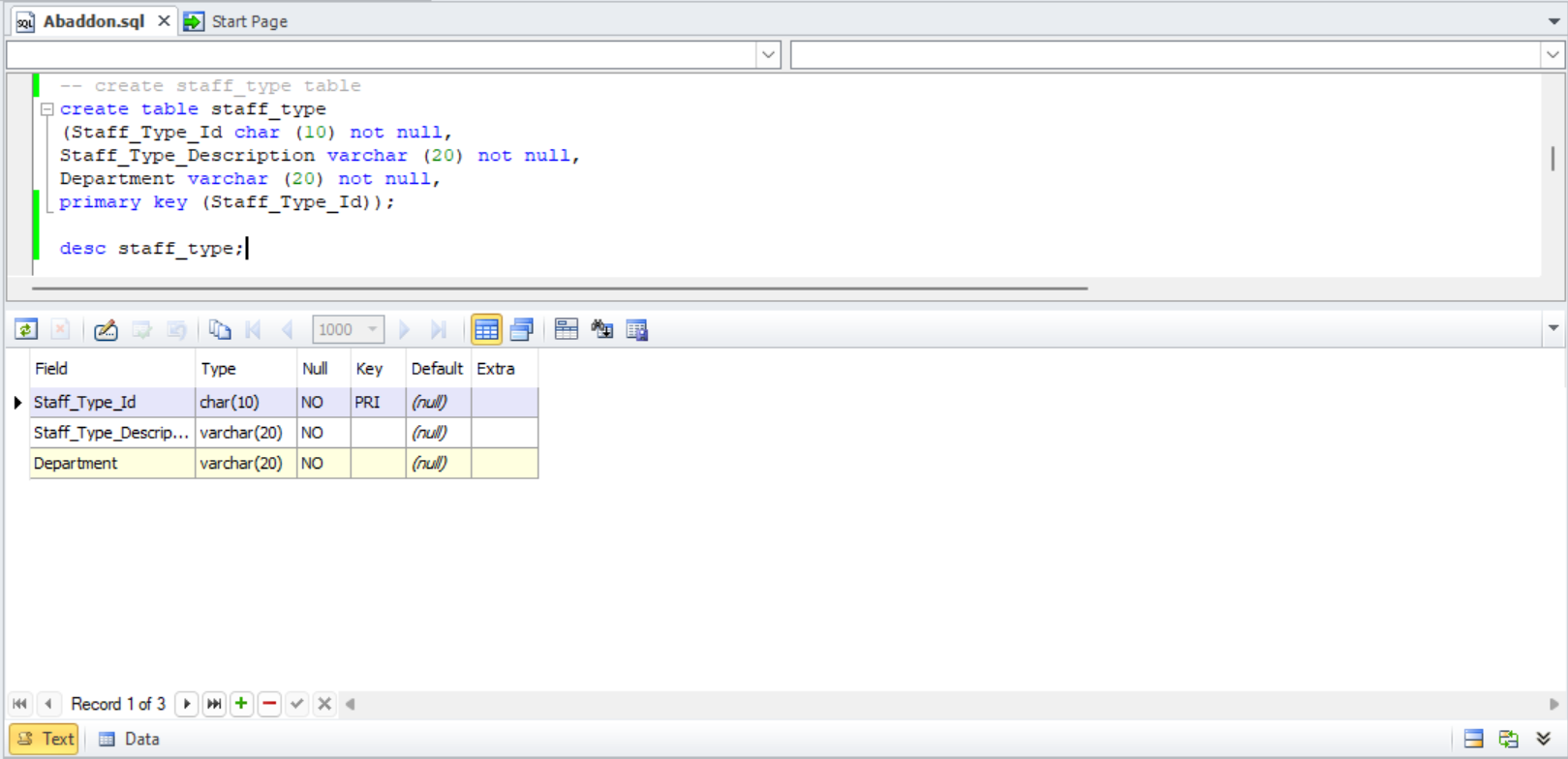


Fig 5.7 Create staff\_type table

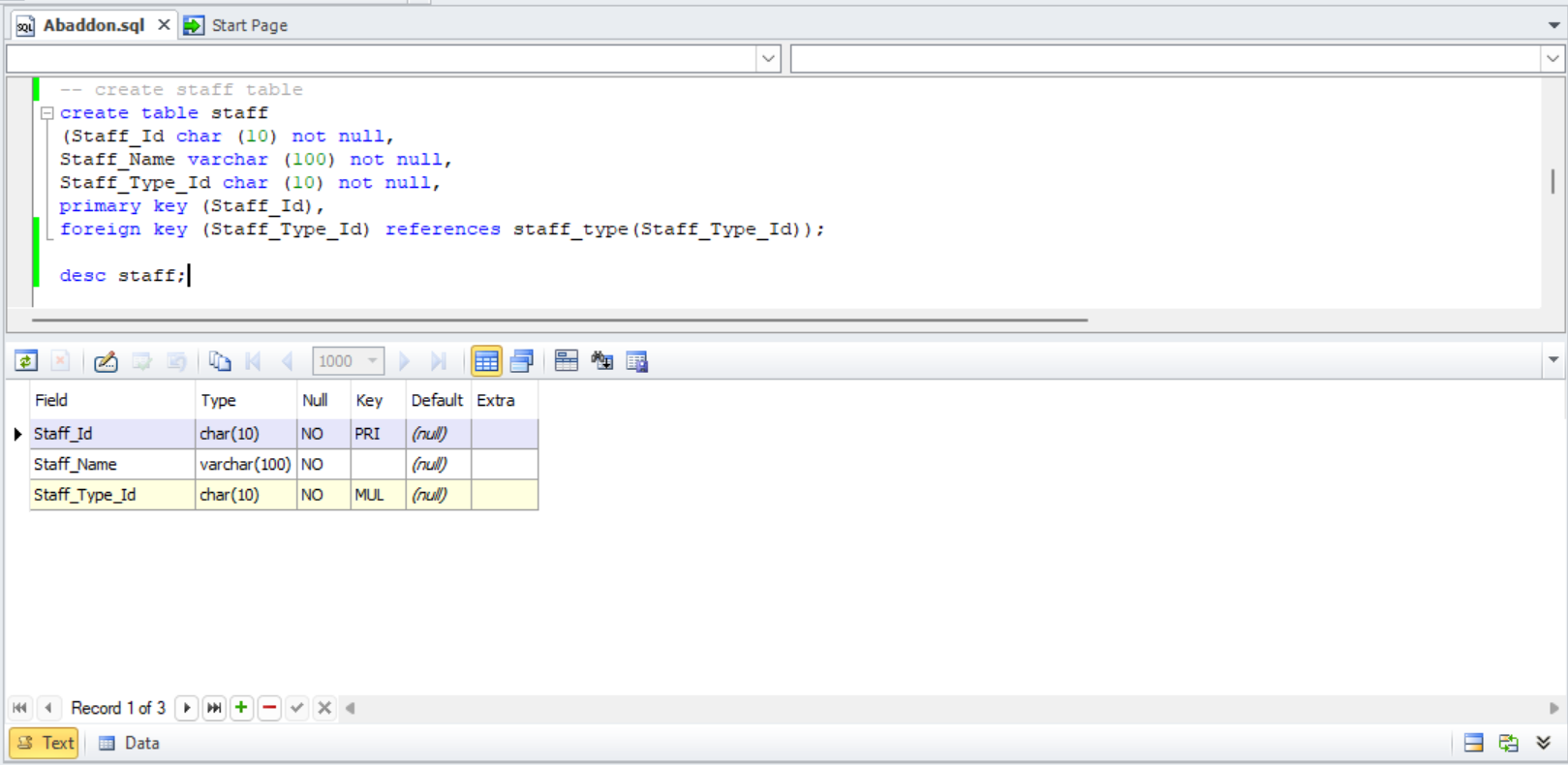


Fig 5.8 Create staff table

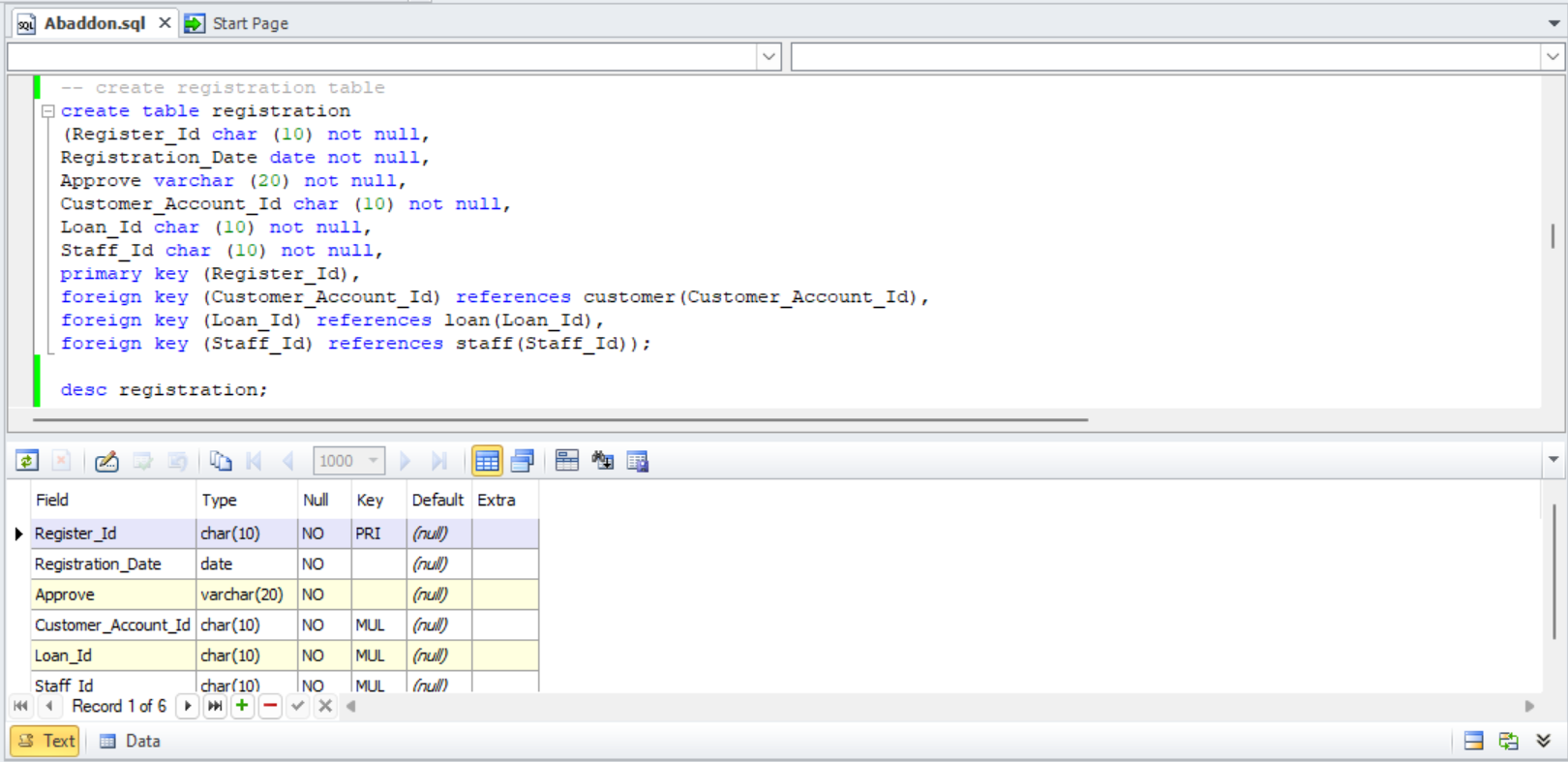


Fig 5.9 Create registration table

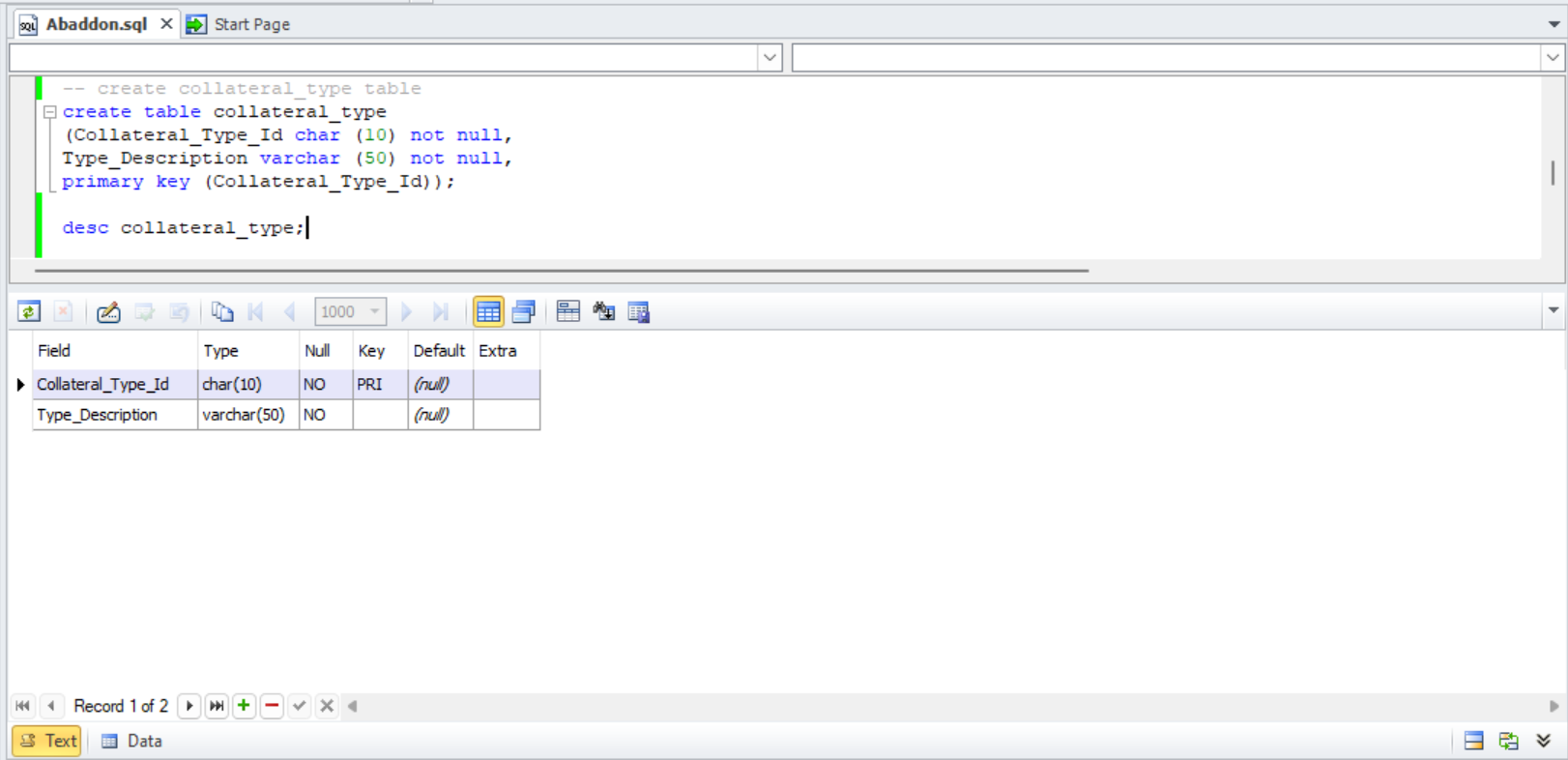


Fig 5.10 Create collateral\_type table

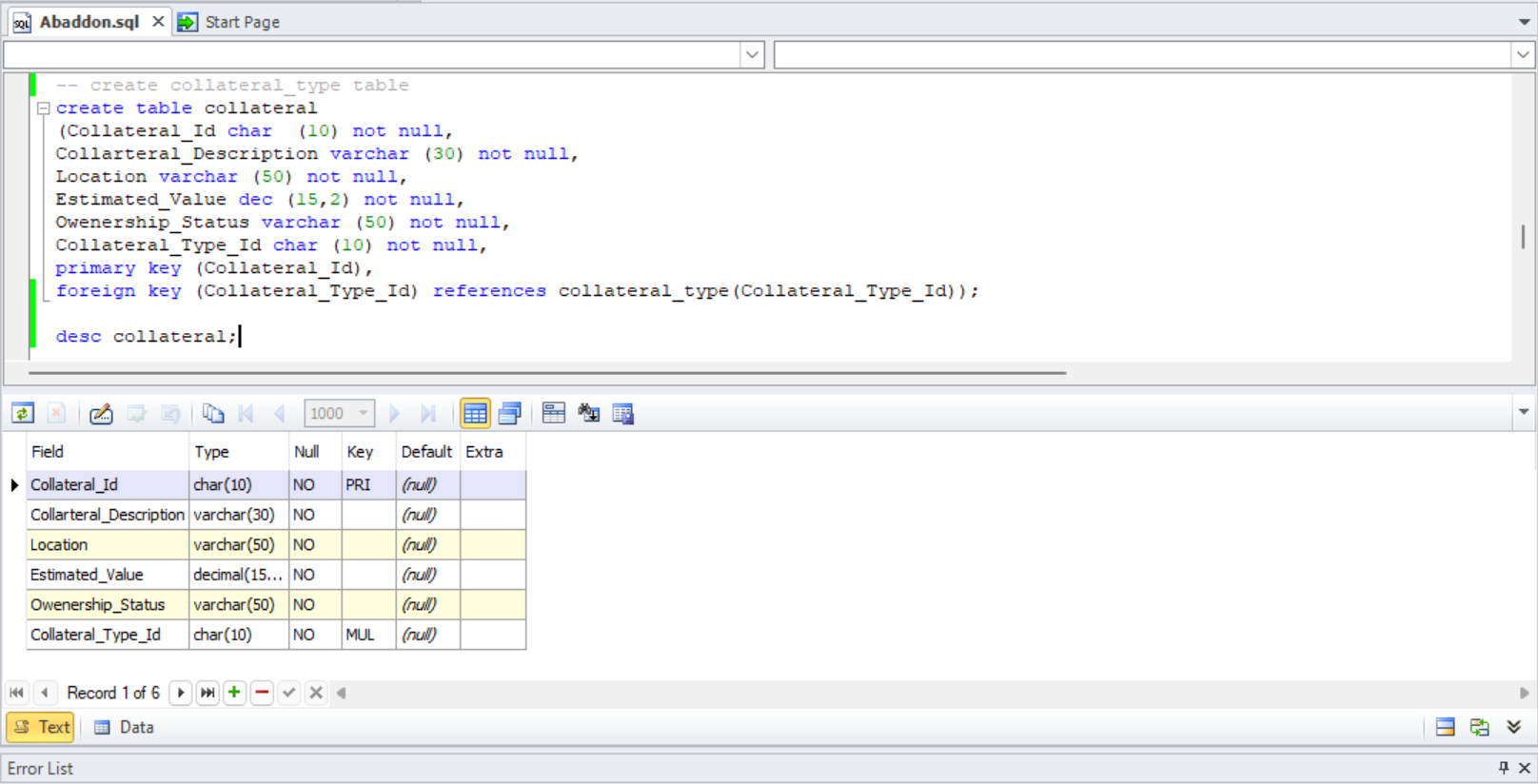


Fig 5.11 Create collateral table

# Task-6

## **Insert Table**

To insert data into the database for Abaddon Bank, a series of SQL INSERT statements were crafted to populate the tables with initial records. These statements were executed sequentially, starting with the parent tables (Customer\_Type, Loan\_Type, Rule, Staff\_Type, and Collateral\_Type) to respect foreign key constraints. Subsequently, data was inserted into the child tables (Customer, Loan, Staff, Collateral, Register, Register\_Collateral, and Loan\_Rule).



Fig 6.1 Insert customer\_type data

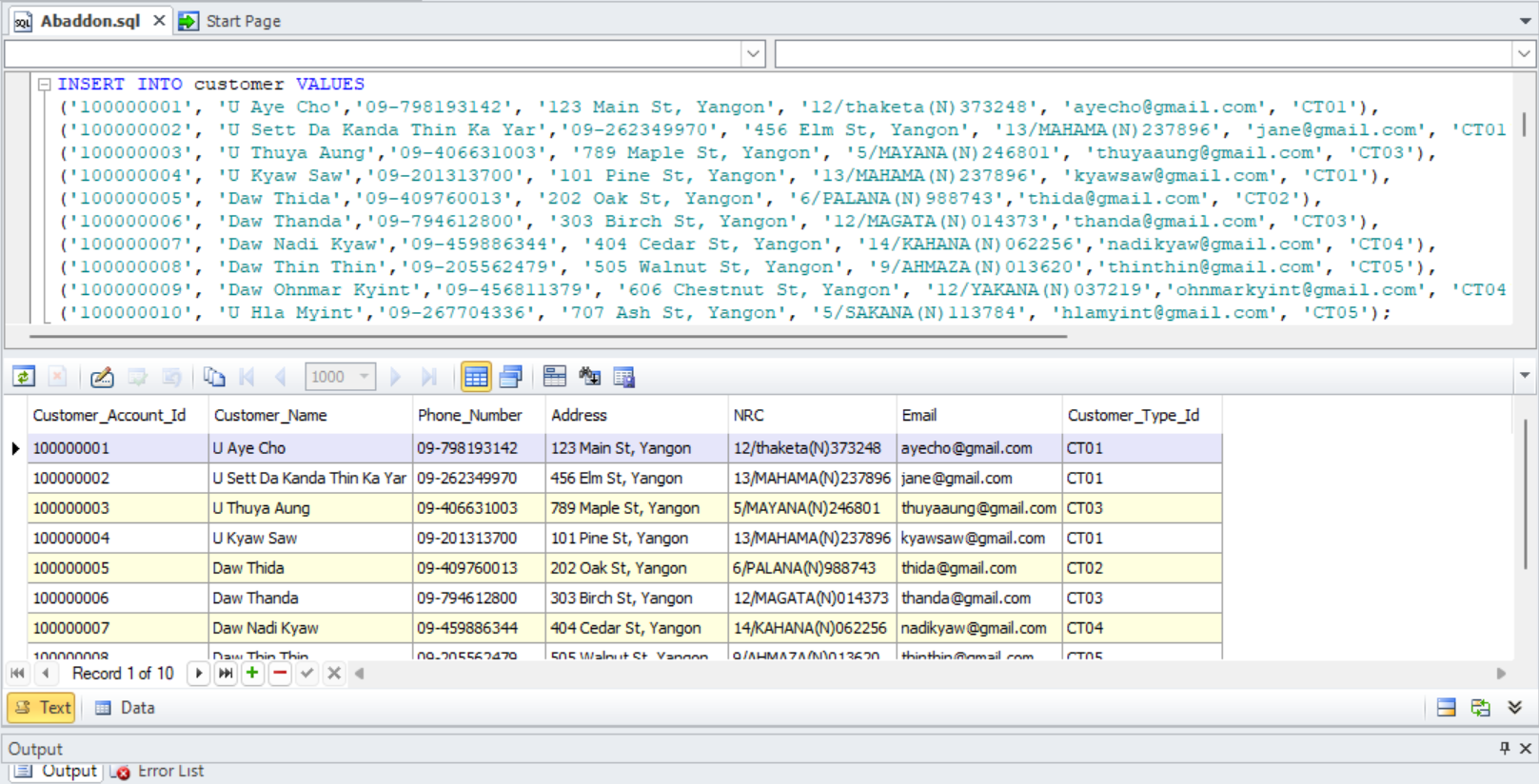


Fig 6.2 Insert customer data

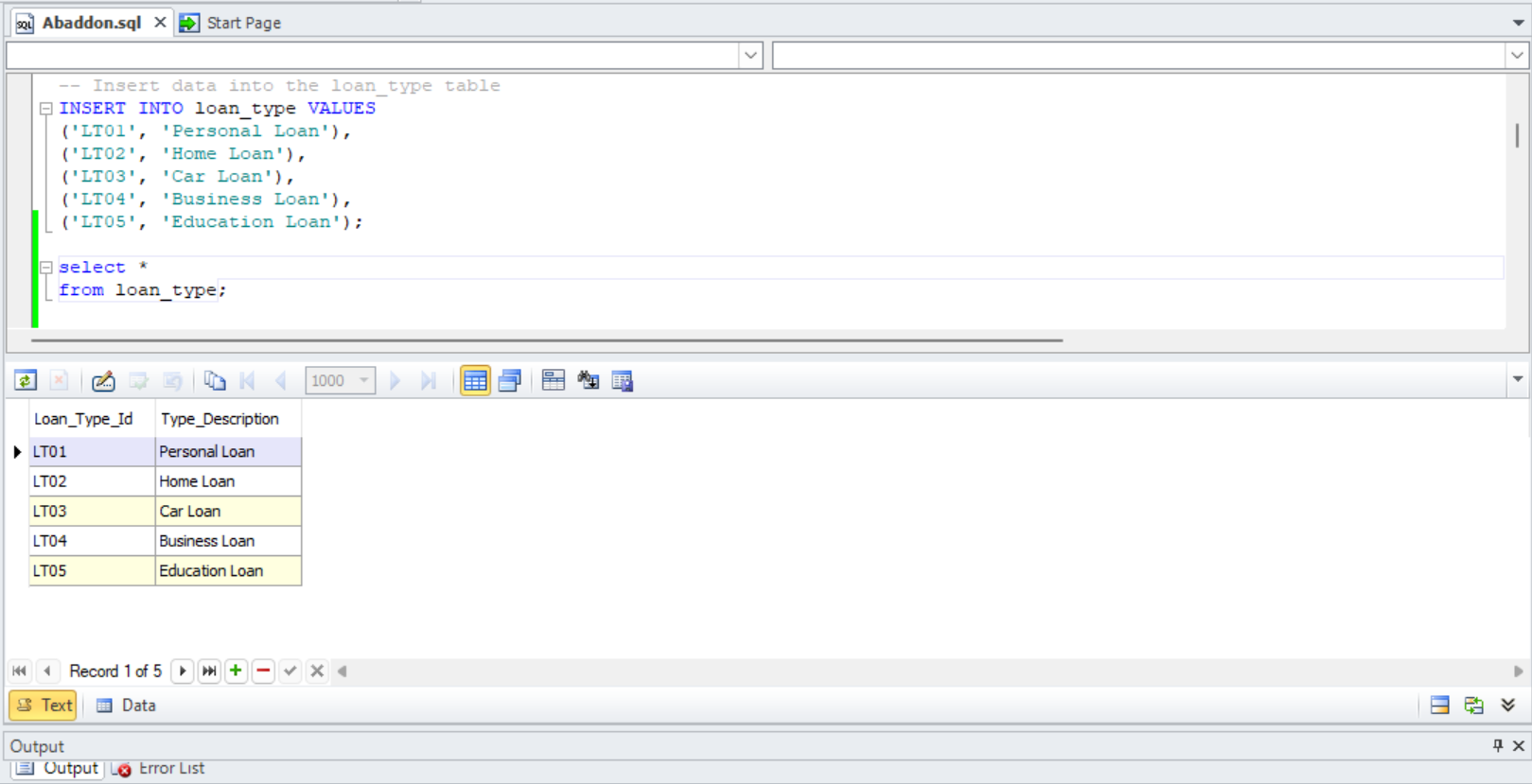


Fig 6.3 Insert loan\_type data

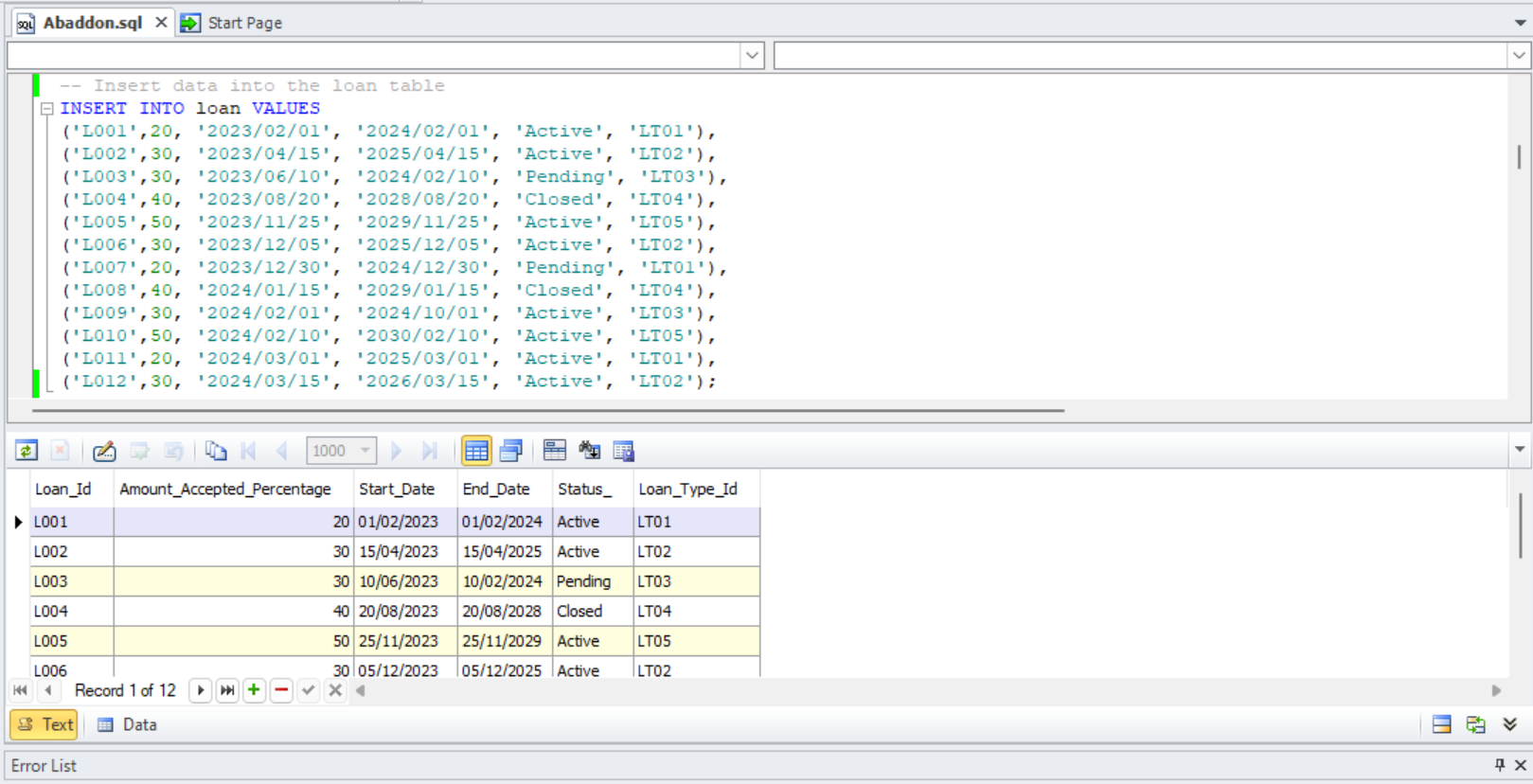


Fig 6.4 Insert loan data

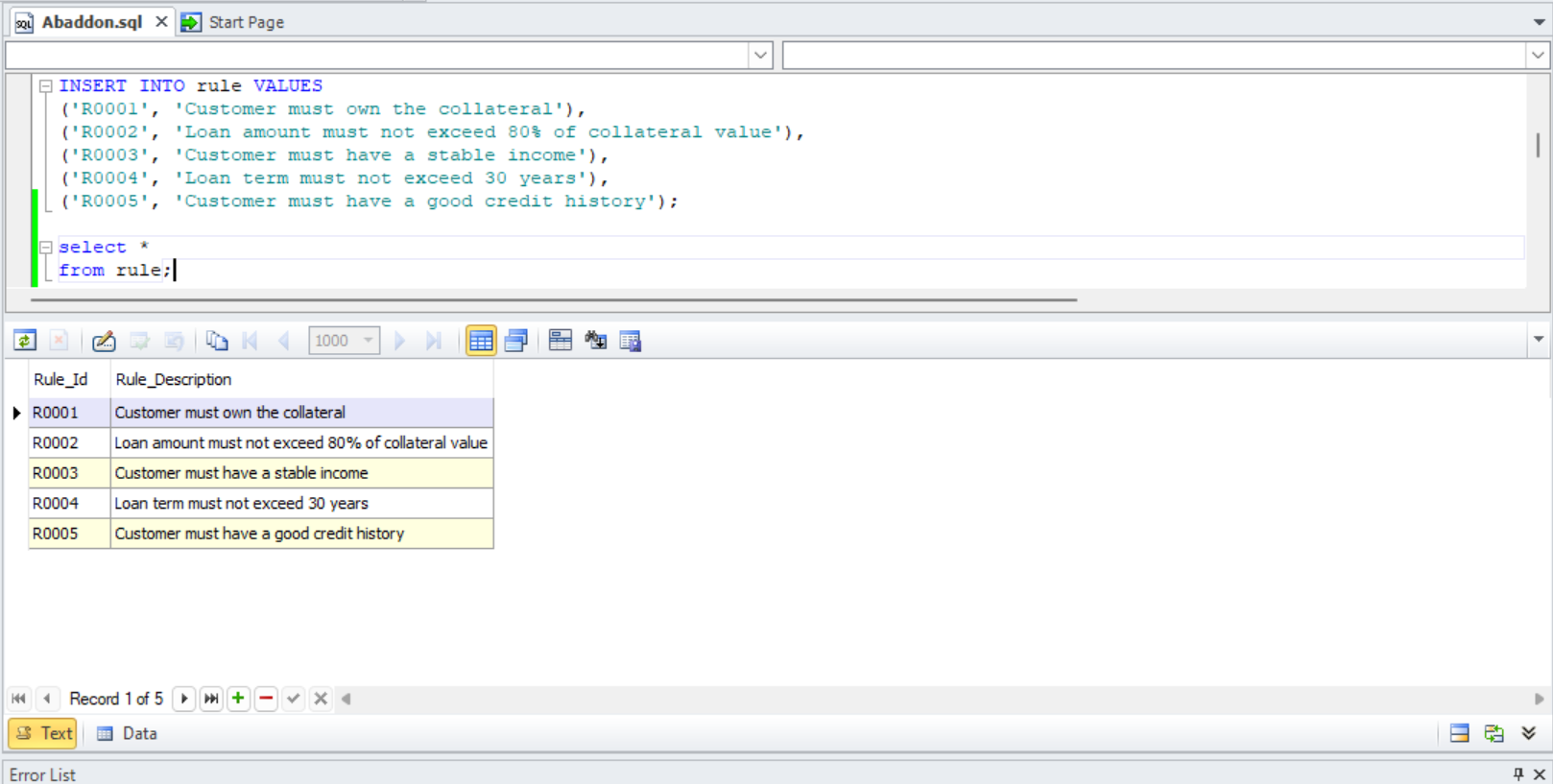


Fig 6.5 Insert rule data

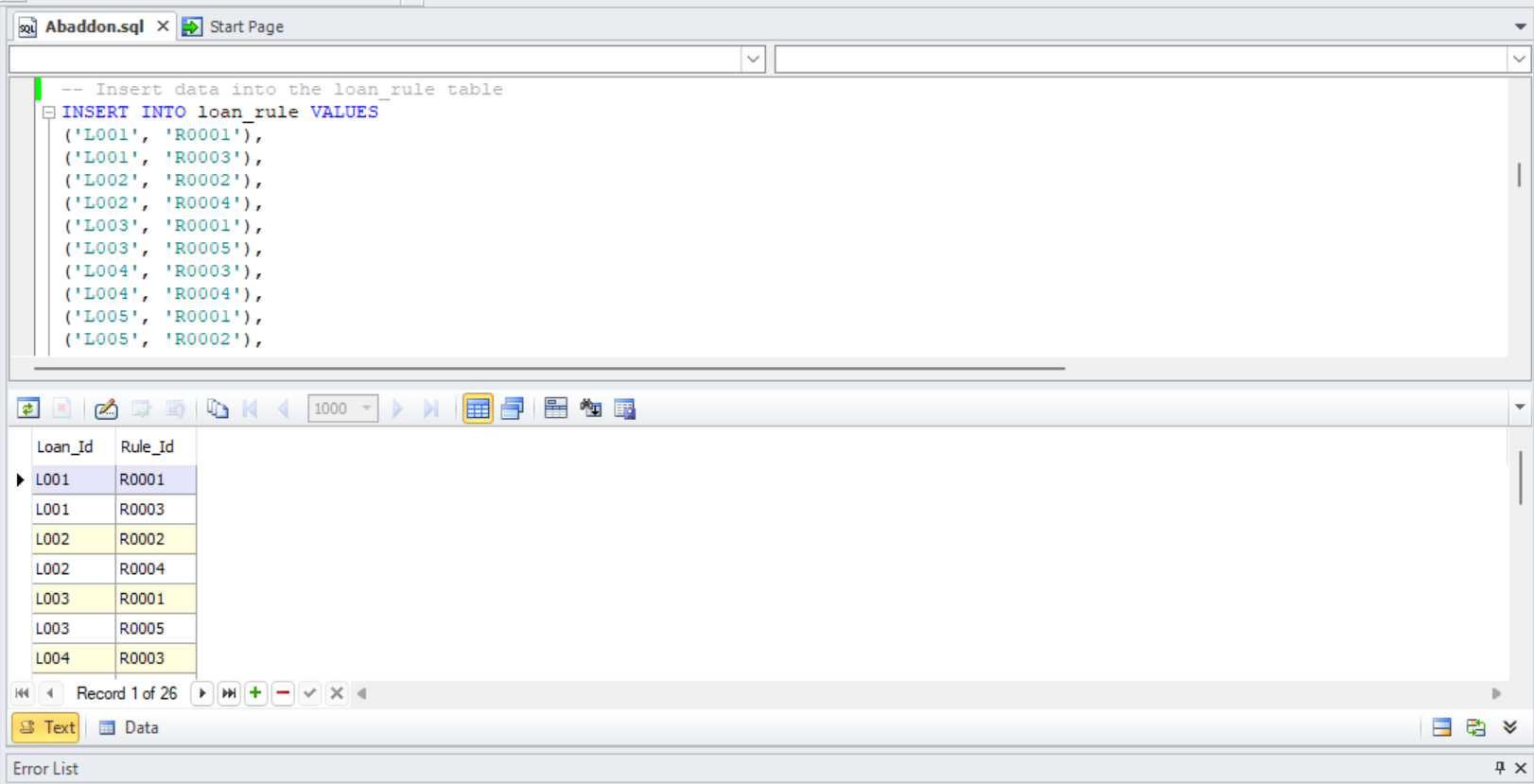


Fig 6.6 Insert loan\_rule data

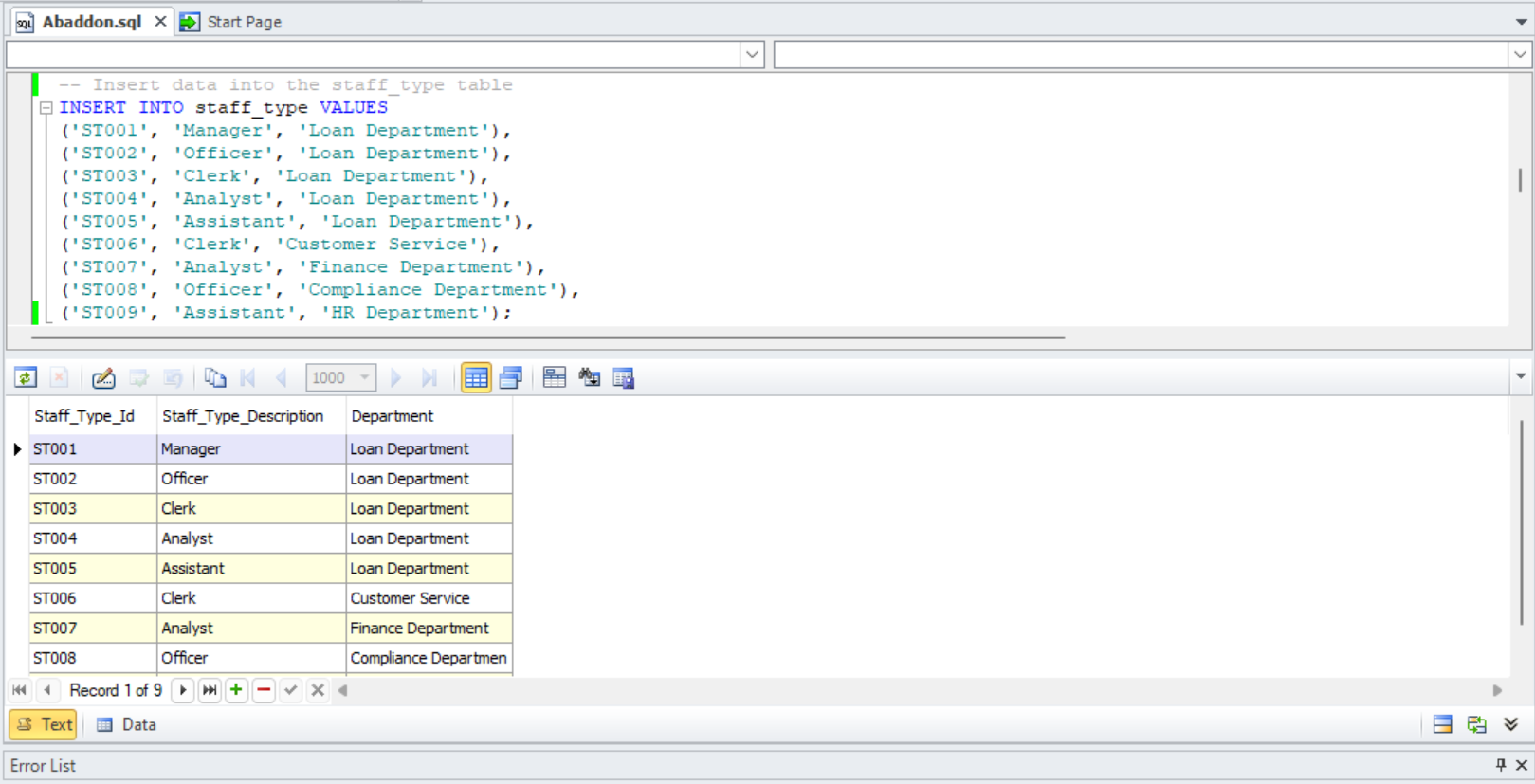


Fig 6.7 Insert staff\_type data

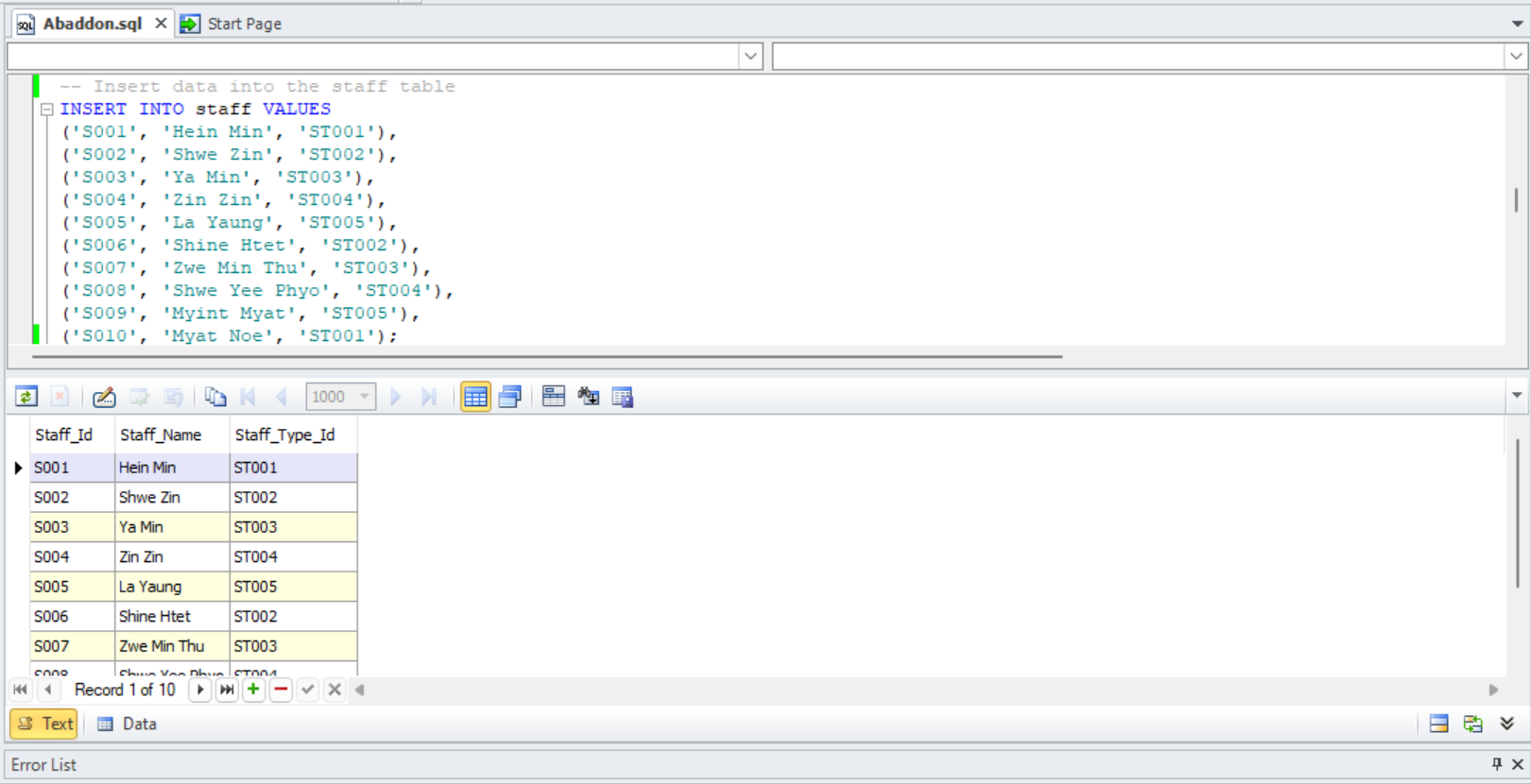


Fig 6.8 Insert staff data

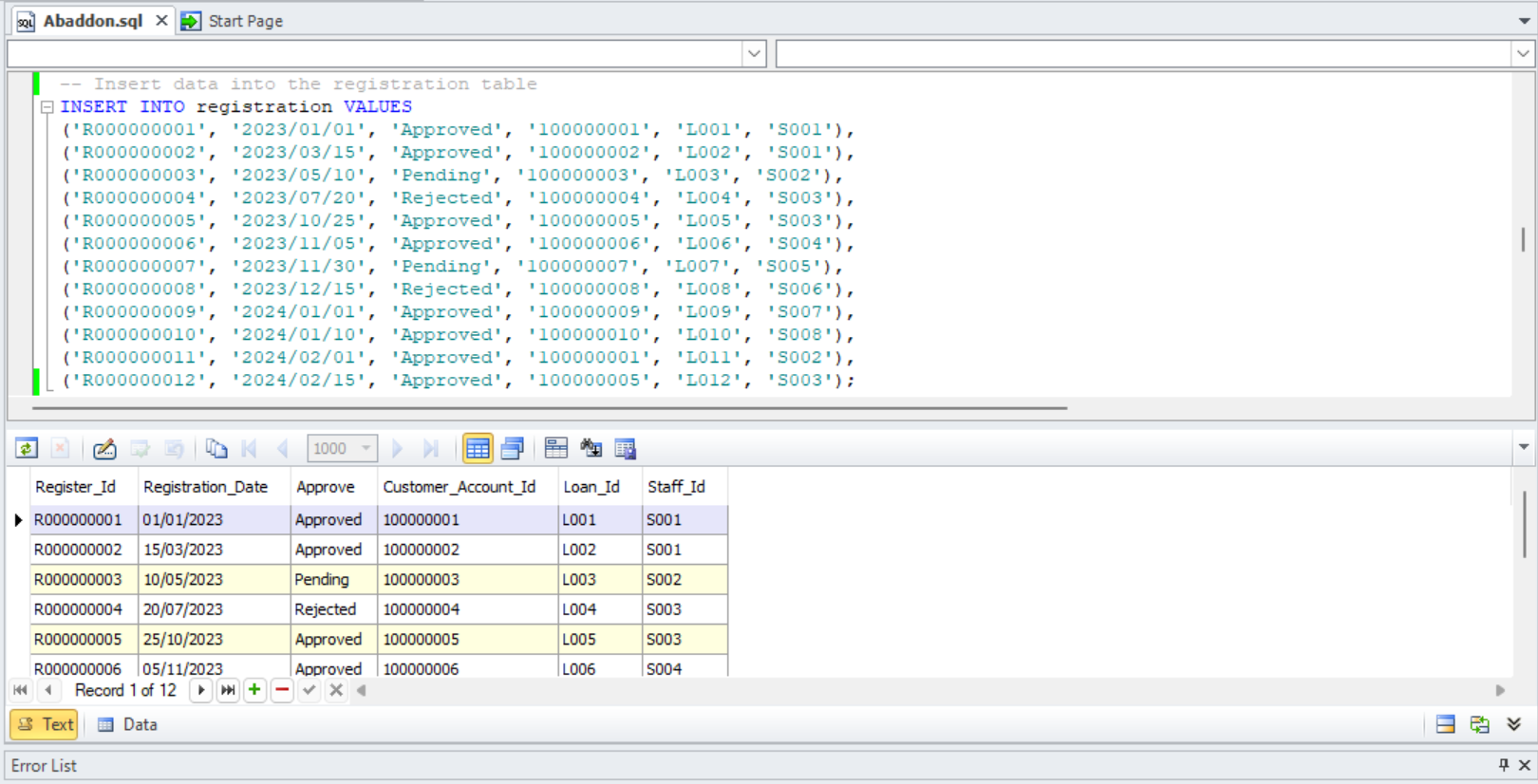


Fig 6.9 Insert registration data

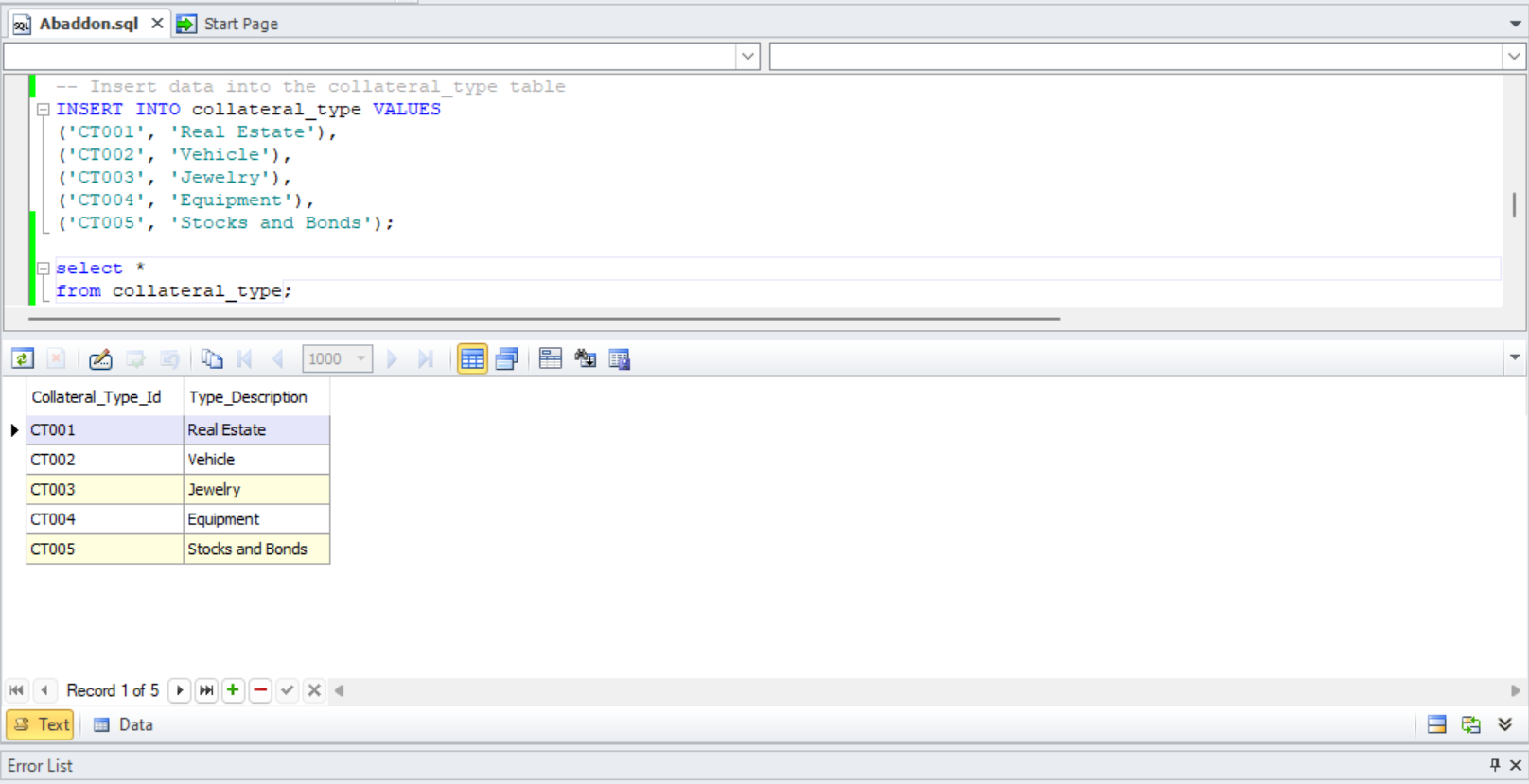


Fig 6.10 Insert collateral\_type data

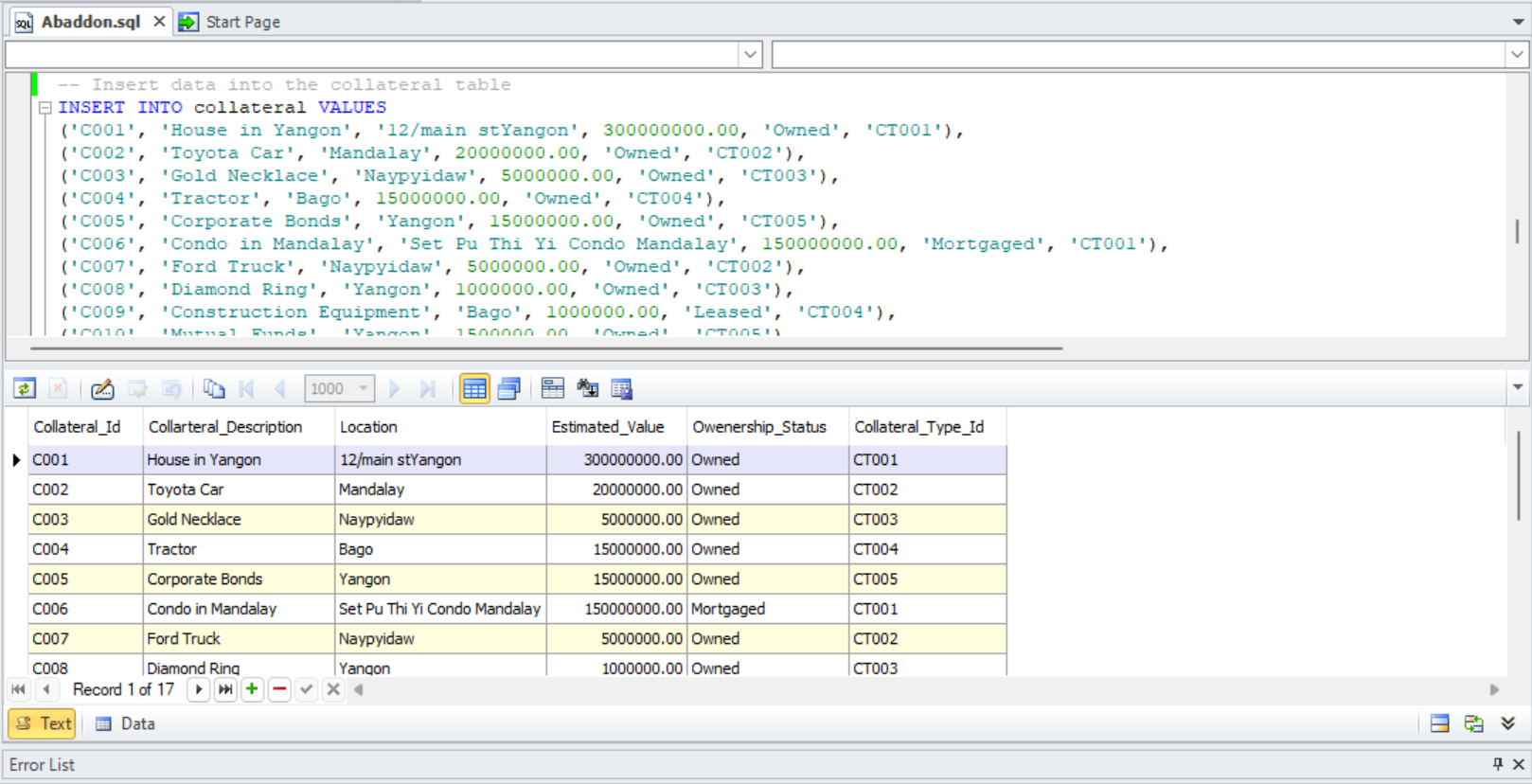


Fig 6.11 Insert collateral data

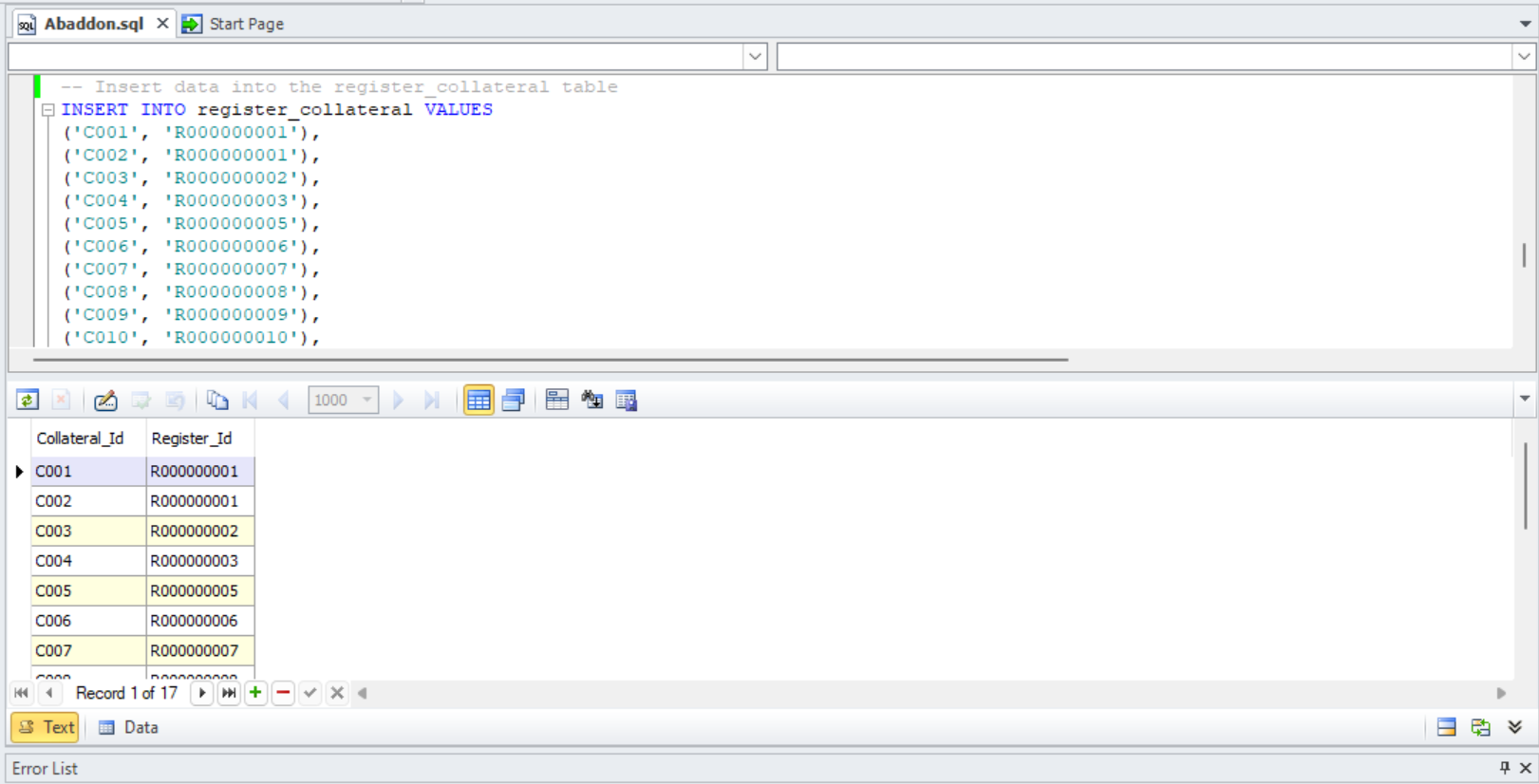


Fig 6.12 Insert register\_collateral data

# Task-7

## **Select Data**

1. The total amount that the bank loaned to each customer

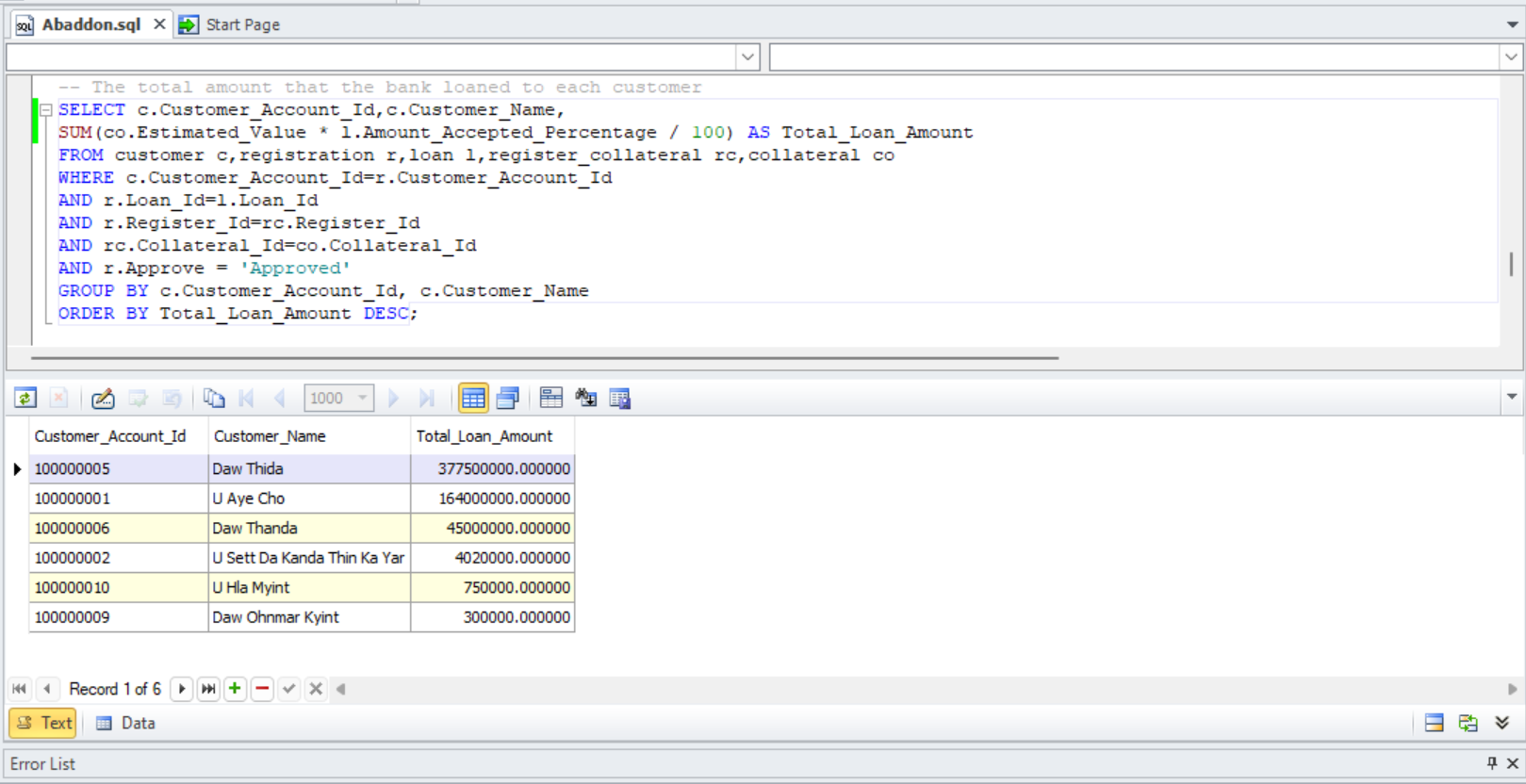


Fig 7.1 Select query 1

1. Find the staff who have handled the greatest number of approved loans and the total loan amount they managed

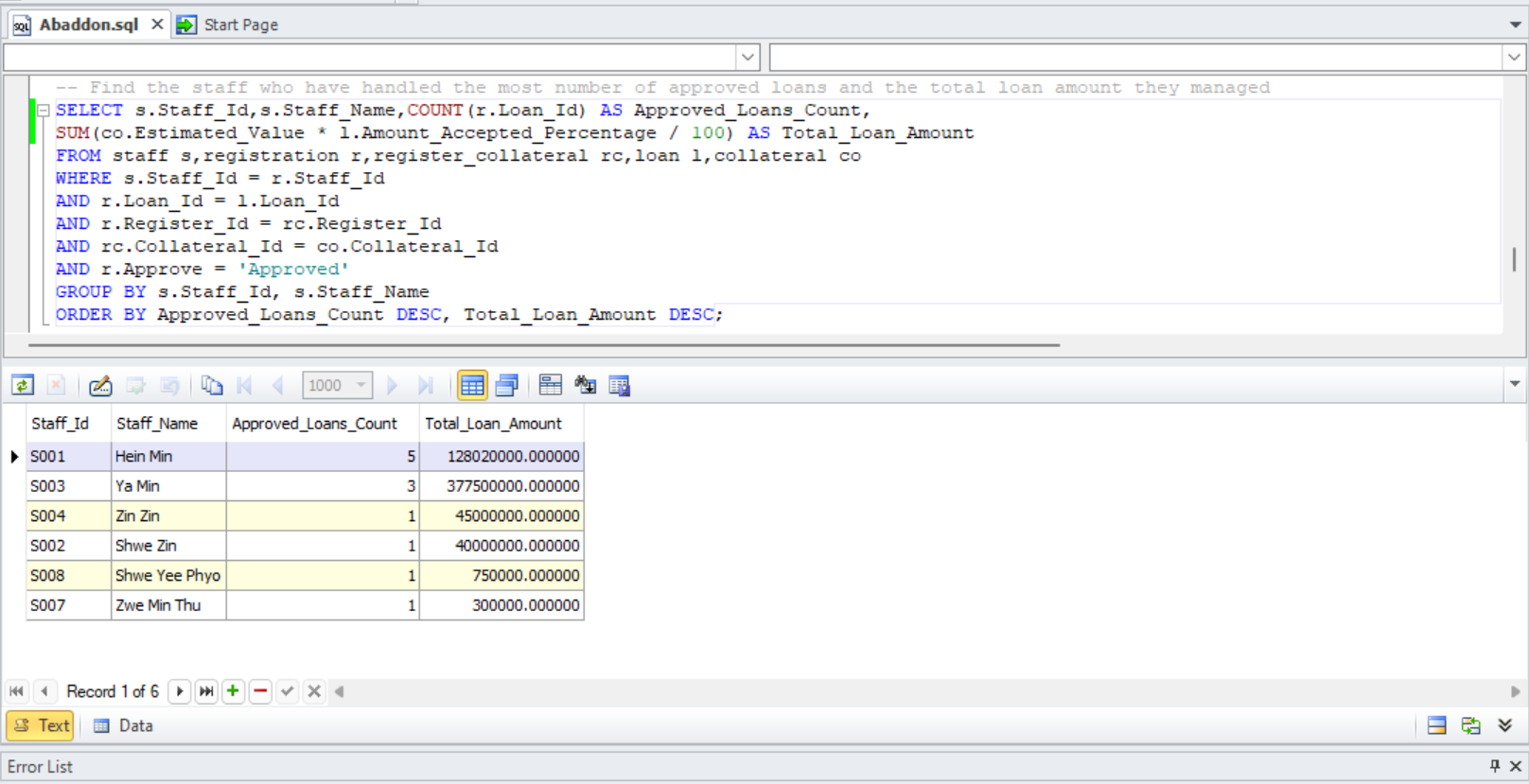


Fig 7.2 Select query 2

1. Find the loan type that has the highest total loan amount across all customers

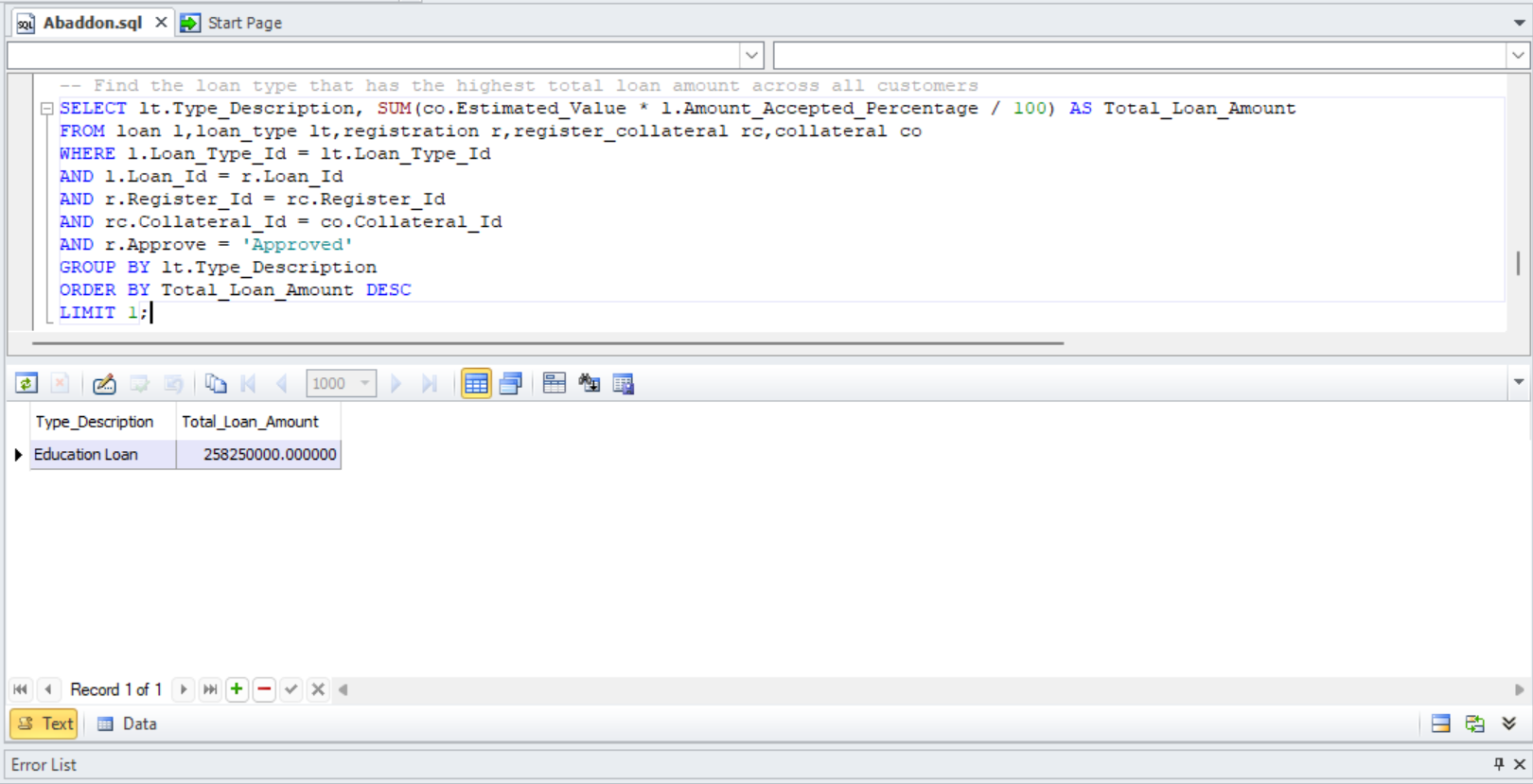


Fig 7.3 Select query 3

1. Find customers with more than one loan

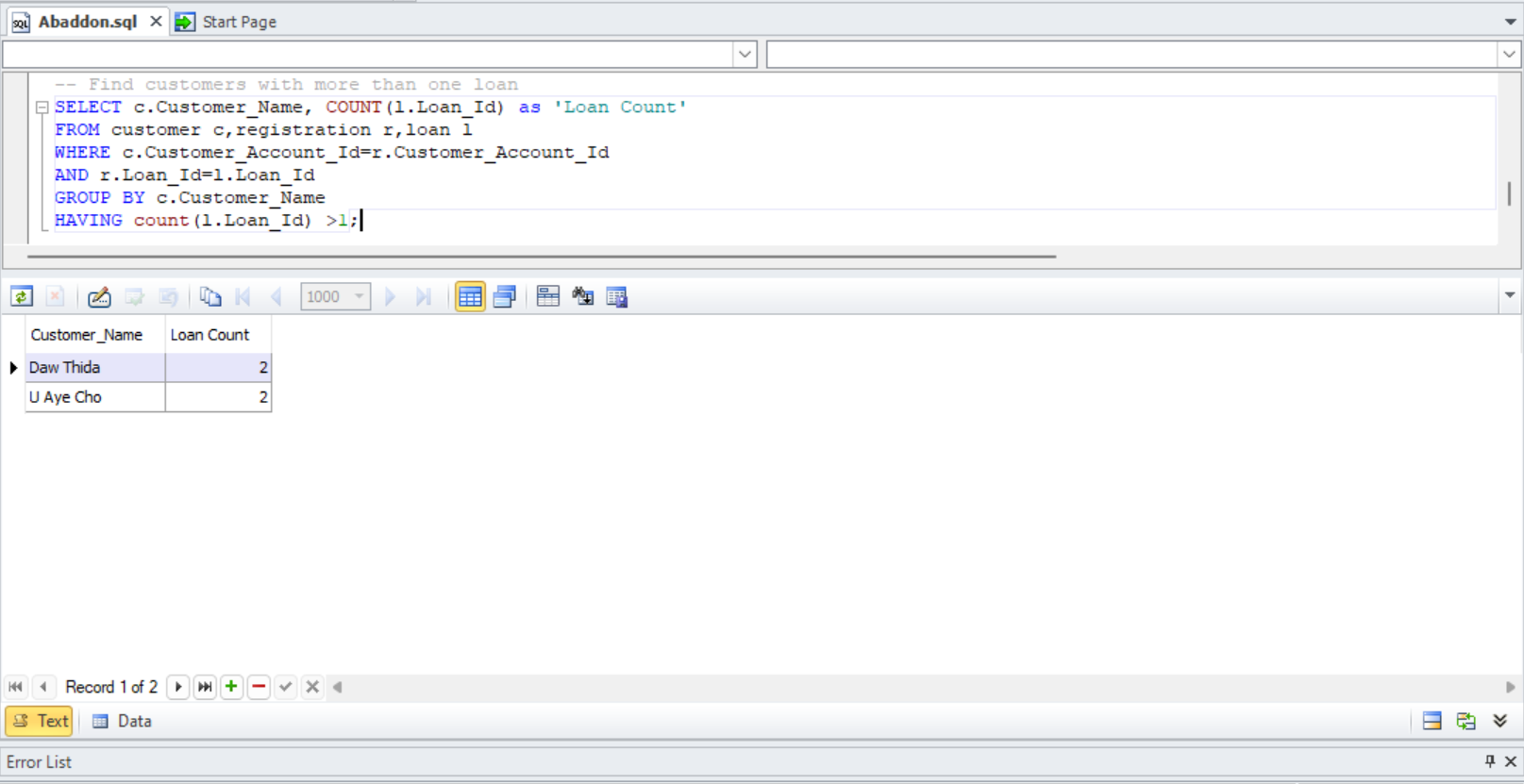


Fig 7.4 Select query 4

1. Show the total loan amount and number of loans for each customer who has taken loans of different types

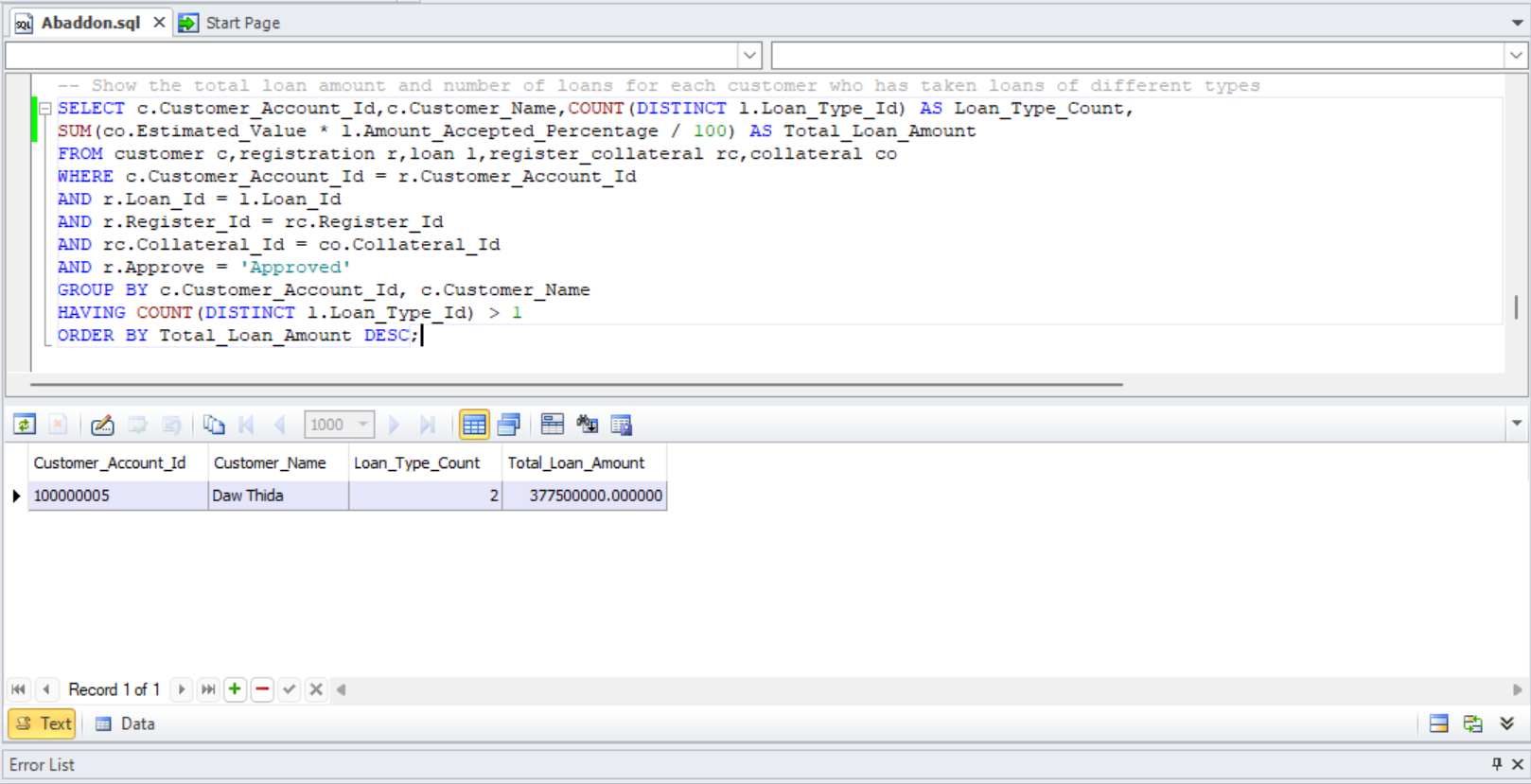


Fig 7.5 Select query 5

1. Show the customers who have been approved for the highest number of loans with a specific rule applied

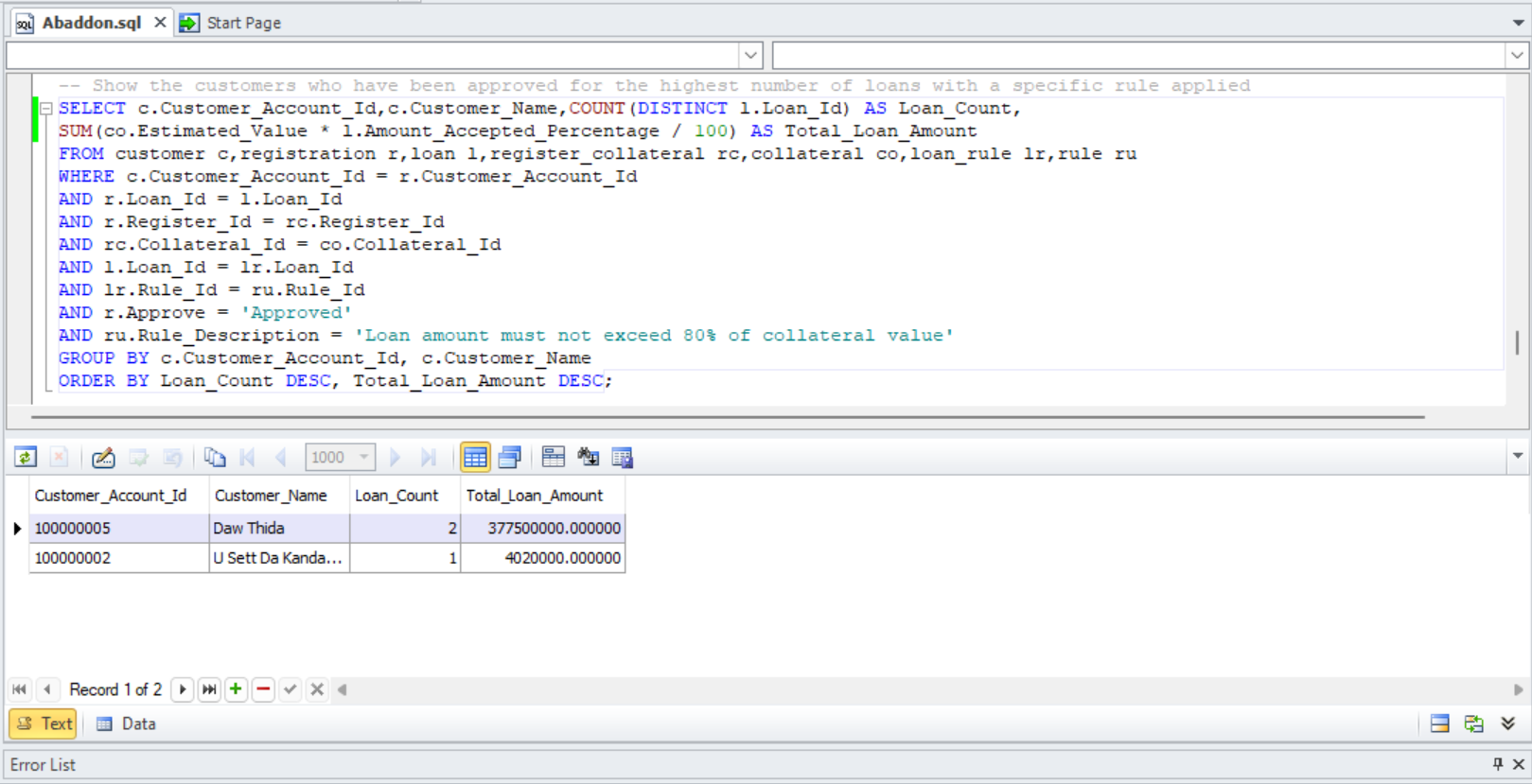


Fig 7.6 Select query 6

1. List the staff members who have approved loans and the total number of loans they approved

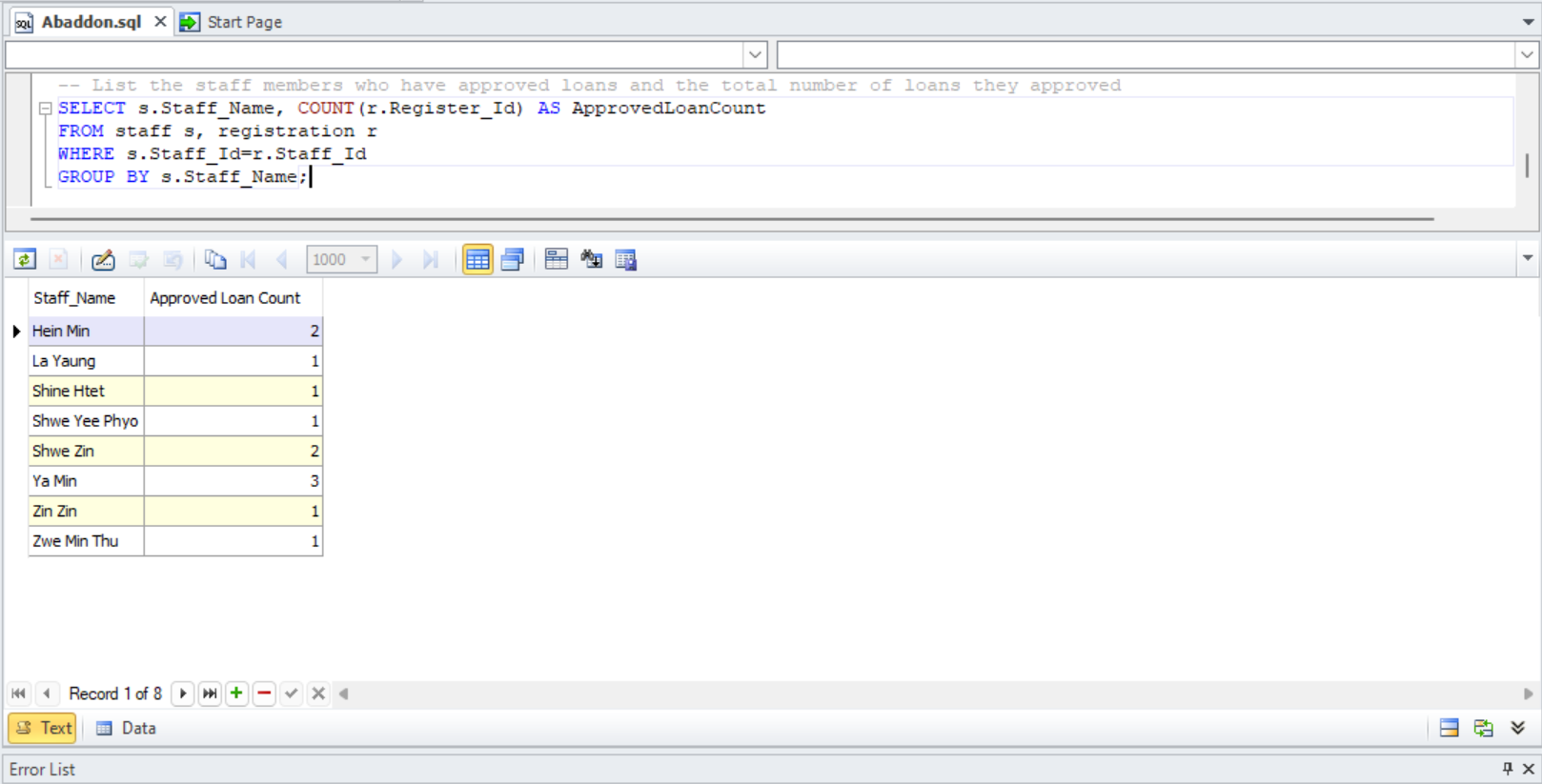


Fig 7.7 Select query 7

1. List the total value of collaterals provided by each customer

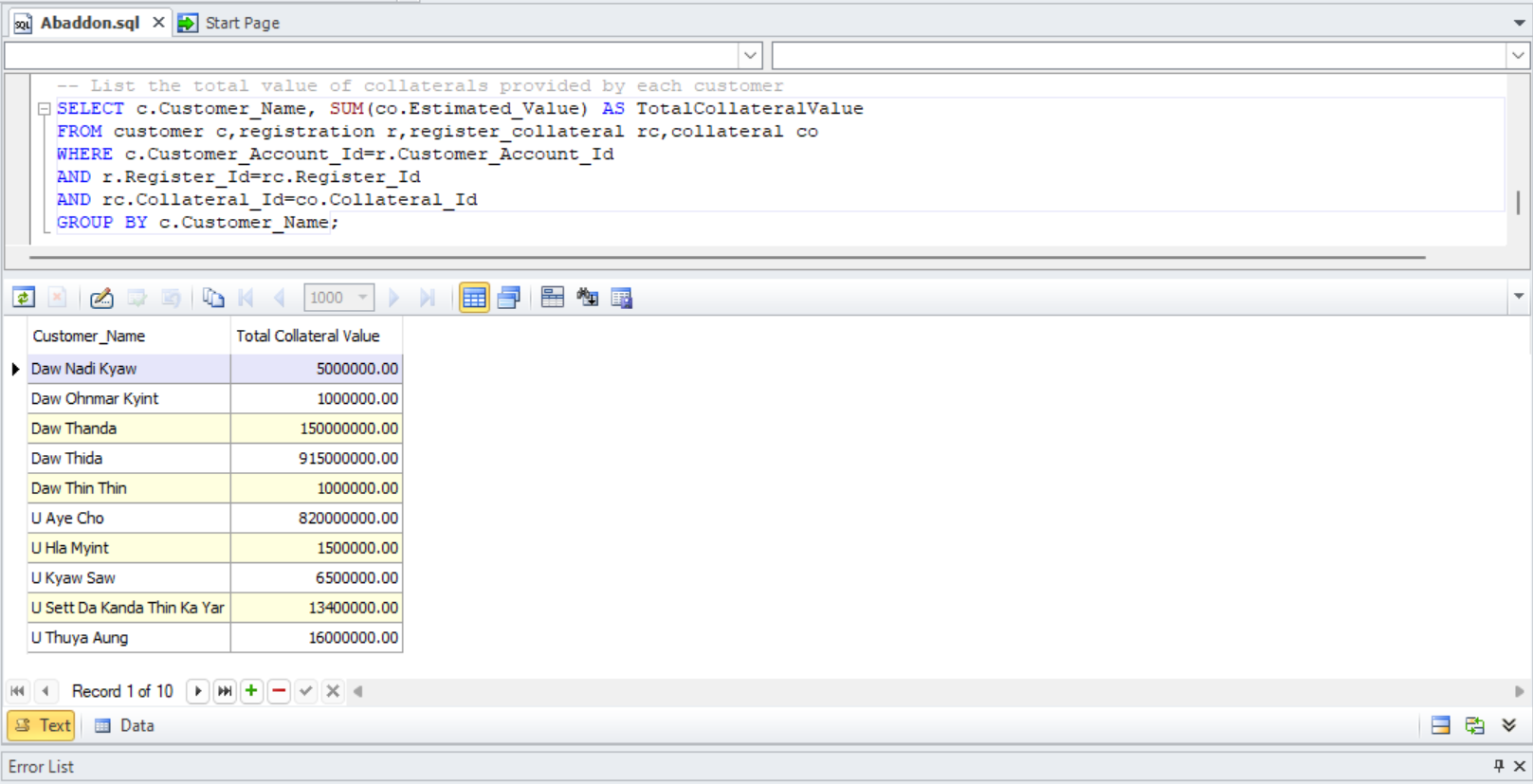


Fig 7.8 Select query 8

1. List the customers with loans ending in the current year

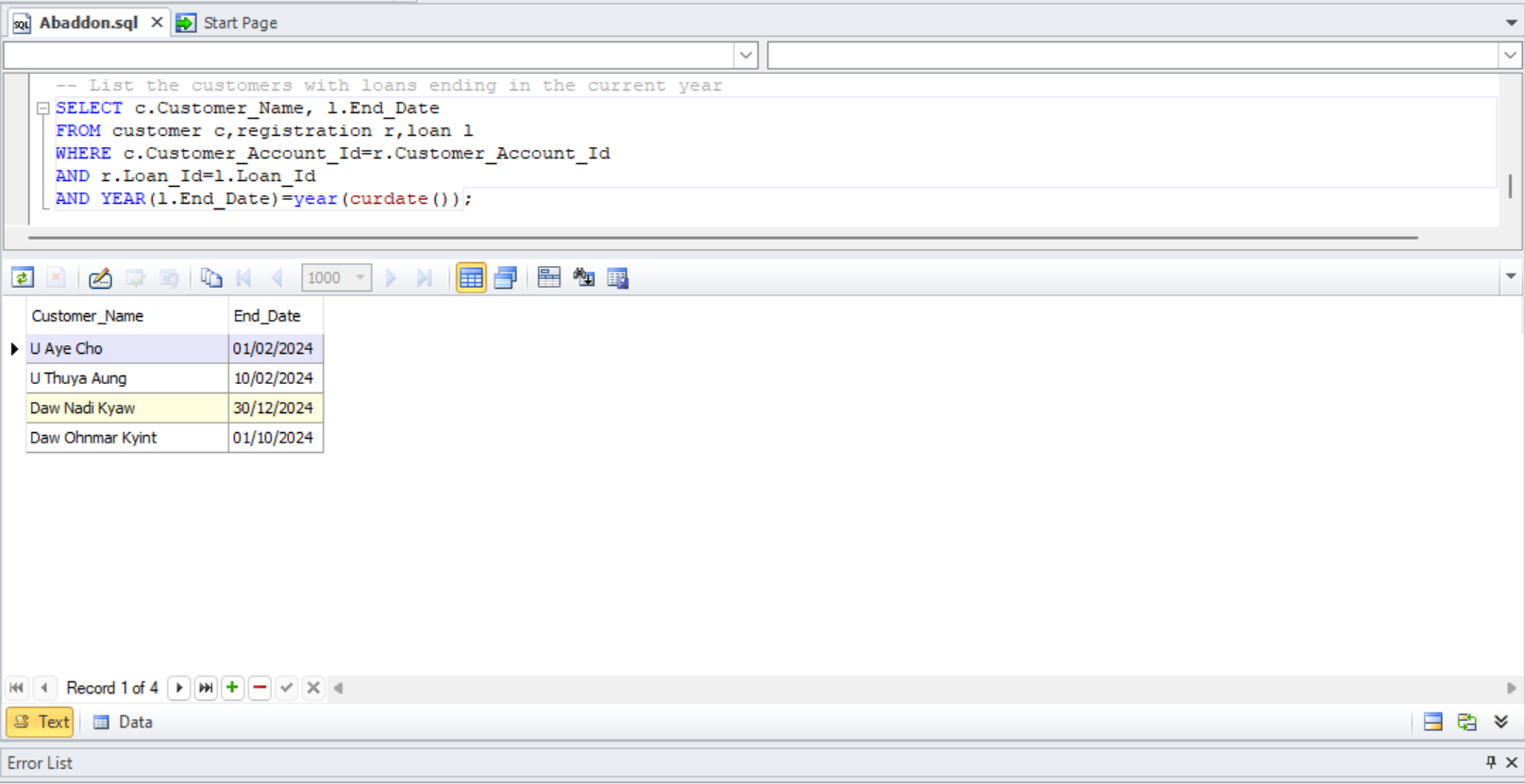


Fig 7.9 Select query 9

1. List customers who have provided collaterals valued more than the average collateral value



Fig 7.10 Select query 10

# Task-8

## **Future development of a distributed database**

In the future, Abaddon Bank might merge with other similar companies, a common strategy for business looking to expand. As Abaddon Bank grows and establishes branches worldwide, it will need a reliable way to manage all its data, such as loans, customer information, and financial transactions, across all its locations.

A distributed database system would allow each branch to handle its own information while also sharing it with a central database. This setup ensures that all branches use the same data and system. It’s keeping everything accurate and consistent across the organization. Abaddon Bank can use a homogeneous distributed database system to maintain consistency and simplify data integration.

A homogeneous distributed database system ensures all branches use the same DBMS. This simplifies data integration and maintenance. This uniformity allows for easier management of the system and reduces the complexity associated with data synchronization.

Data fragmentation is important in a distributed database. It involves breaking the database into smaller parts or fragments which can be stored at different locations. Proper fragmentation ensures data is stored near where it is most frequently used, improving access times and reducing communication costs. However, poorly managed fragmentation can lead to inefficiencies and increased complexity in query processing.

The following are advantages and disadvantages of using distributed database for Abaddon Bank:

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| Improved Data Management: Handles vast data across all branches efficiently. | Complex Setup: Required meticulous planning and expertise. |
| High Availability: Remains operational even if one part fails. | Synchronization Issues: Keeping data synchronized is challenging. |
| Faster Access: Localized storage enables quicker information access. | High Costs: Expensive to implement and maintain. |
| Scalability: Easily scales with bank growth. | Security Concerns: Ensuring data security across locations is tough. |
| Fault Tolerance: Handles failures without disrupting operations. | Fragmentation Challenges: Managing data fragmentation is difficult (www.linkedin.com,n.d.). |

# Task-9

## **Future Development of a Data Warehouse**

A data warehouse could be a game-changer for Abaddon Bank, driven by the need for comprehensive data analysis and reporting. By consolidating data from various sources into a centralized repository, the bank can significantly enhance its decision-making capabilities. Executives and managers will benefit from a unified view of data across departments, such as customer accounts, loans, and staff performance, enabling more informed strategic decisions and timely responses to market changes. Furthermore, a data warehouse improves data quality and consistency by standardizing information from disparate systems, ensuring reliable analytics and reporting. The ability to store large volumes of historical data supports trend analysis and forecasting, providing insights into past transactions and customer behavior, which are invaluable for risk management and strategic planning, by offloading analytical processing from transactional databases, the data ware house boosts operational efficiency, allowing transactional systems to handle day-to-day operations without the burden of complex queries. This results in faster transaction processing and better customer service. Additionally, a data warehouse supports regulatory compliance by maintaining a comprehensive, easily accessible record of transactions and customer interactions, facilitating quick responses to regulatory inquiries. Enhanced customer insights from consolidated data across various touchpoints enable personalized services and targeted marketing campaigns, improving customer satisfaction. The scalability and performance of a data warehouse ensure that as Abaddon Bank grows, it can handle increasing data loads and analytical demands effectively. Lastly, leveraging advanced analytics and data mining within a data warehouse helps the bank uncover hidden patterns and correlations, supporting initiatives like fraud detection, credit scoring, and customer segmentation. In essence, a data warehouse would provide Abaddon Bank with the tools necessary to drive strategic growth, operational efficiency, and enhanced customer experiences (www.quest-global.com, 2022).

# References

1. JavaTPoint (2011).DBMS Normalization – javatpoint. [online] www.javatpoint.com. Available at: <https://www.javatpoint.com/dbms-normalization>. (Accepted date- 13/6/2024)
2. GeeksforGeeks. (2021). Logical Database. [online] Available at: <https://www.geeksforgeeks.org/logical-database/>. (Accepted date- 17/6/2024)
3. www.tutorialspoint.com. (n.d.). Physical Database Design and Tuning. [online] Available at: <https://www.tutorialspoint.com/physical-database-design-and-tuning>. (Accepted date- 18/6/2024)
4. SearchDataManagement. (n.d.). What is denormalization and how does it work? [online] Available at: <https://www.techtarget.com/searchdatamanagement/definition/denormalization>. (Accepted date- 19/6/2024)
5. www.linkedin.com.(n.d.). Distributed Database Market Size, Future Trends: Evaluating Share, Trends, and Emerging Growth for 2024-2031. [online] Available at: <https://www.linkedin.com/pulse/distributed-database-market-size-future-rscsf>. (Accepted date- 24/6/2024)
6. www.quest-global.com. (2022). Future Trends in Data Warehousing | Quest Global. [online] Available at: <https://www.quest-global.com/blogs/future-trends-in-data-warehousing/>. (Accepted date- 29/6/2024)