Unit 6—Masks, Roles, and Encryption

**Corey Crooks**

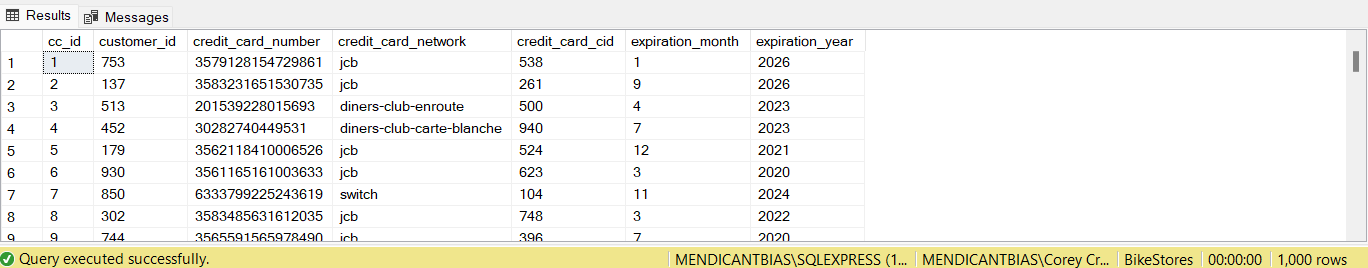
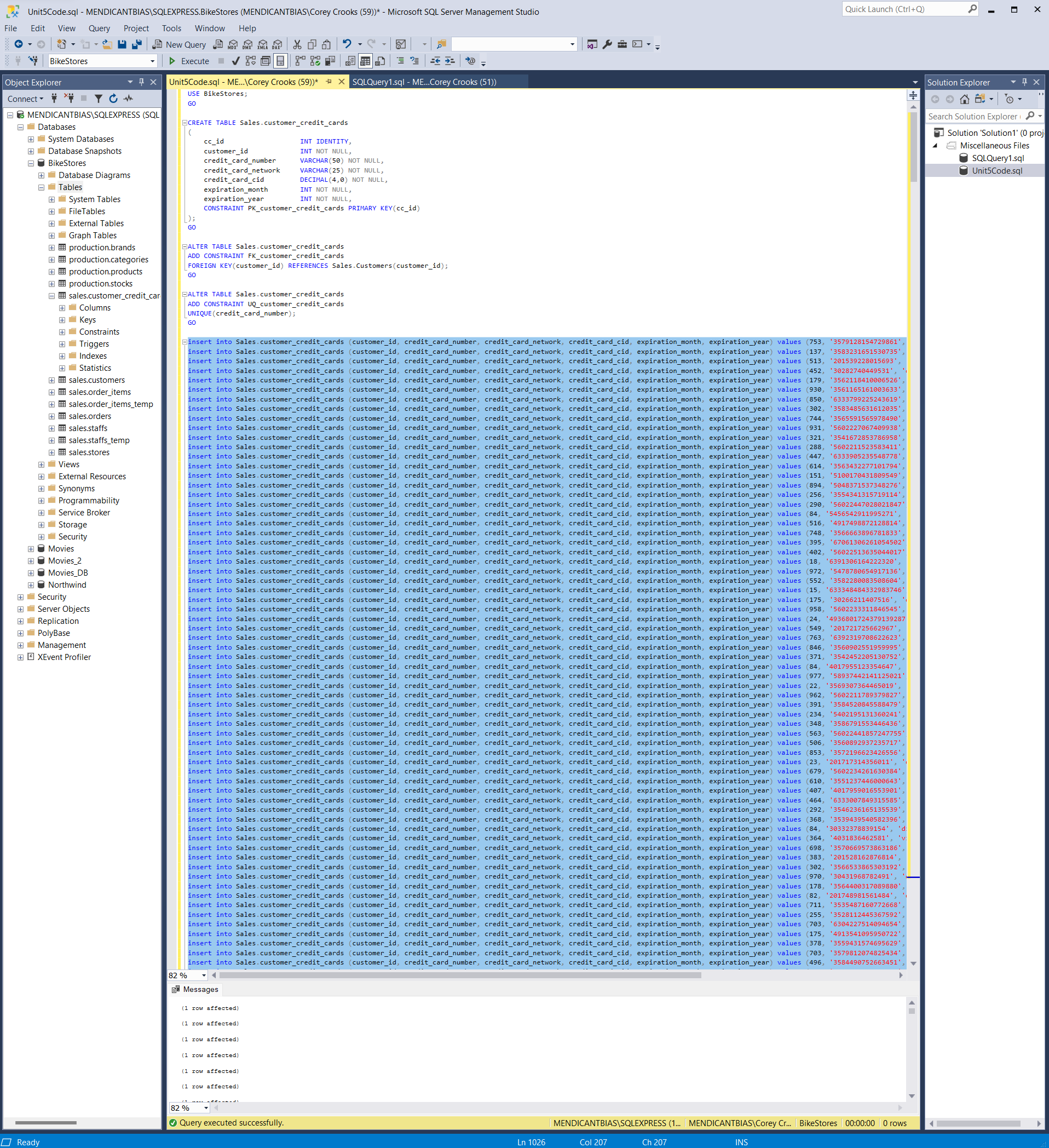
**Purdue University Global**

**IT350 – Advanced Database Concepts**

**Jason Johnson**

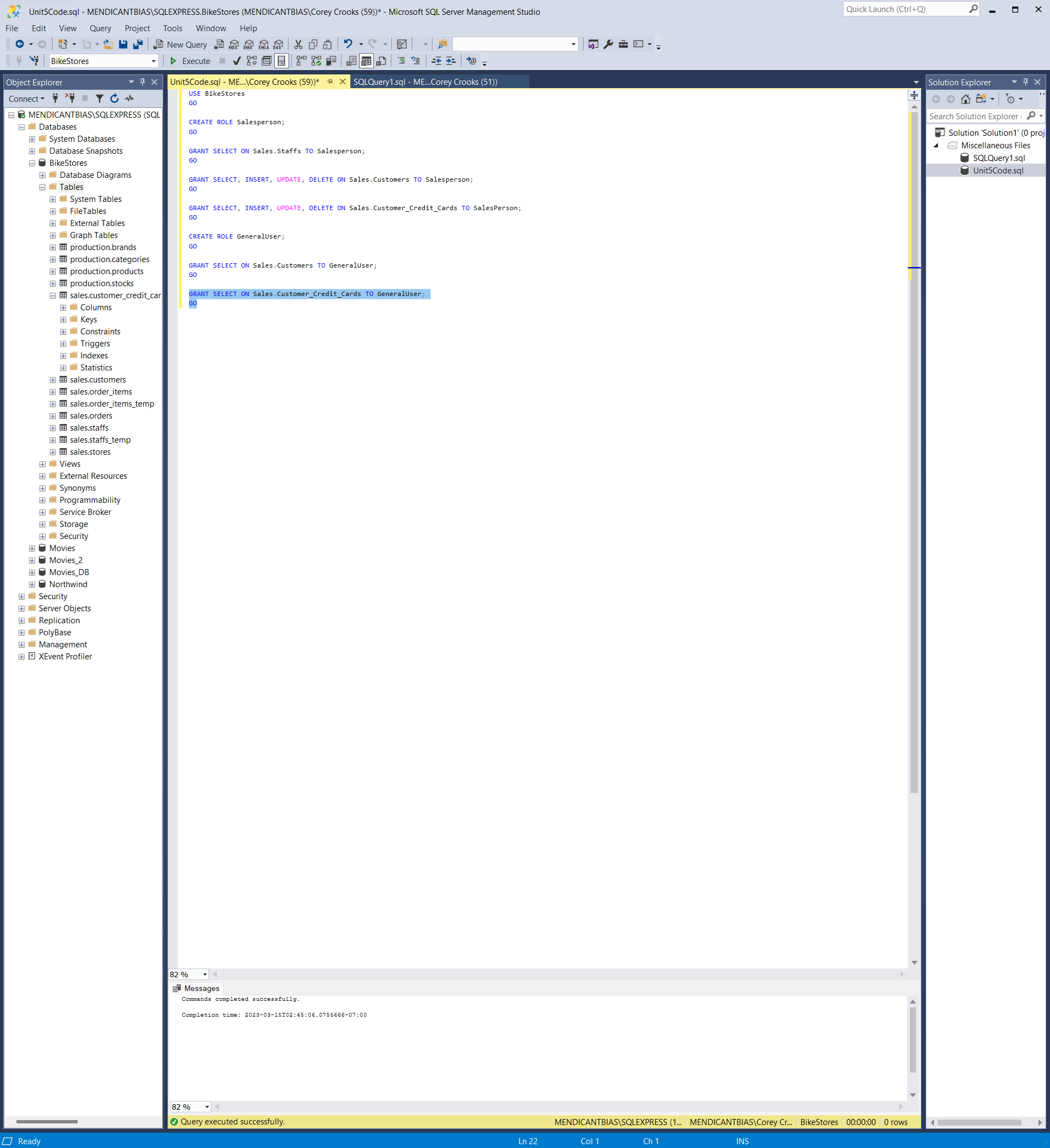
**March 8th, 2023**

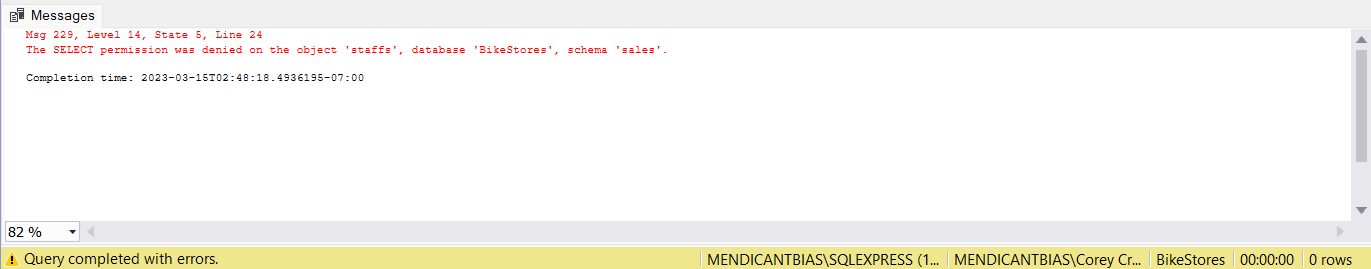
**Task 1—Table Creation**



Creation of the customer credit card table was a success. As demonstrated in the screenshot, the table has been successfully created, and all fields have been populated according to the SQL script provided in the assignment instructions. A number of issues exist with this current setup, however. These issues are primarily due to the fact that no roles or permissions currently exist with respect to this new table; this is an issue mainly due to the fact that, as it stands, anyone may pull whatever data they would like from this new table unrestricted. Due to the sensitive nature of credit card information and its role in society, it is best to add some security to this table.

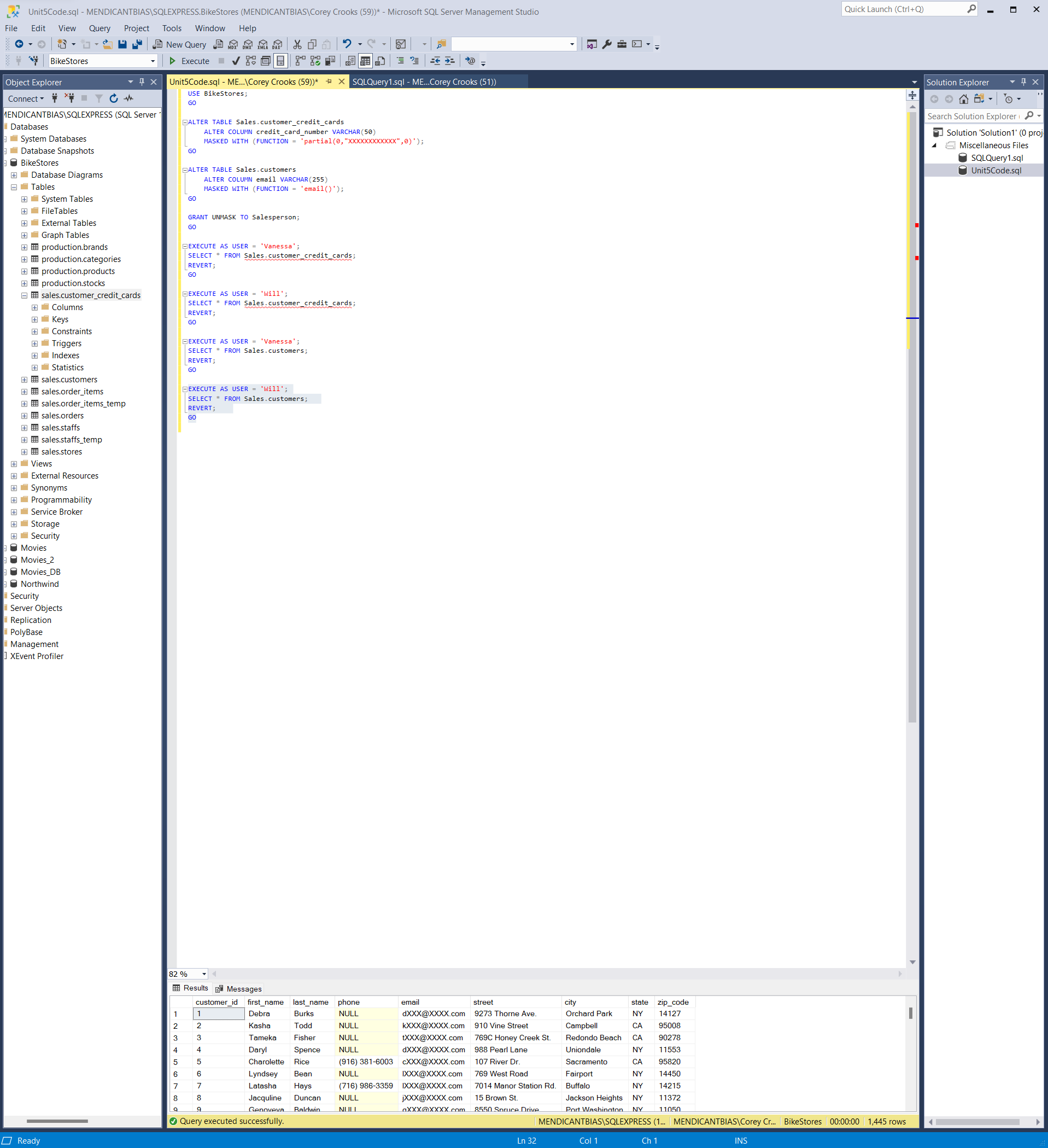
**Task 2 — Permissions and Roles**





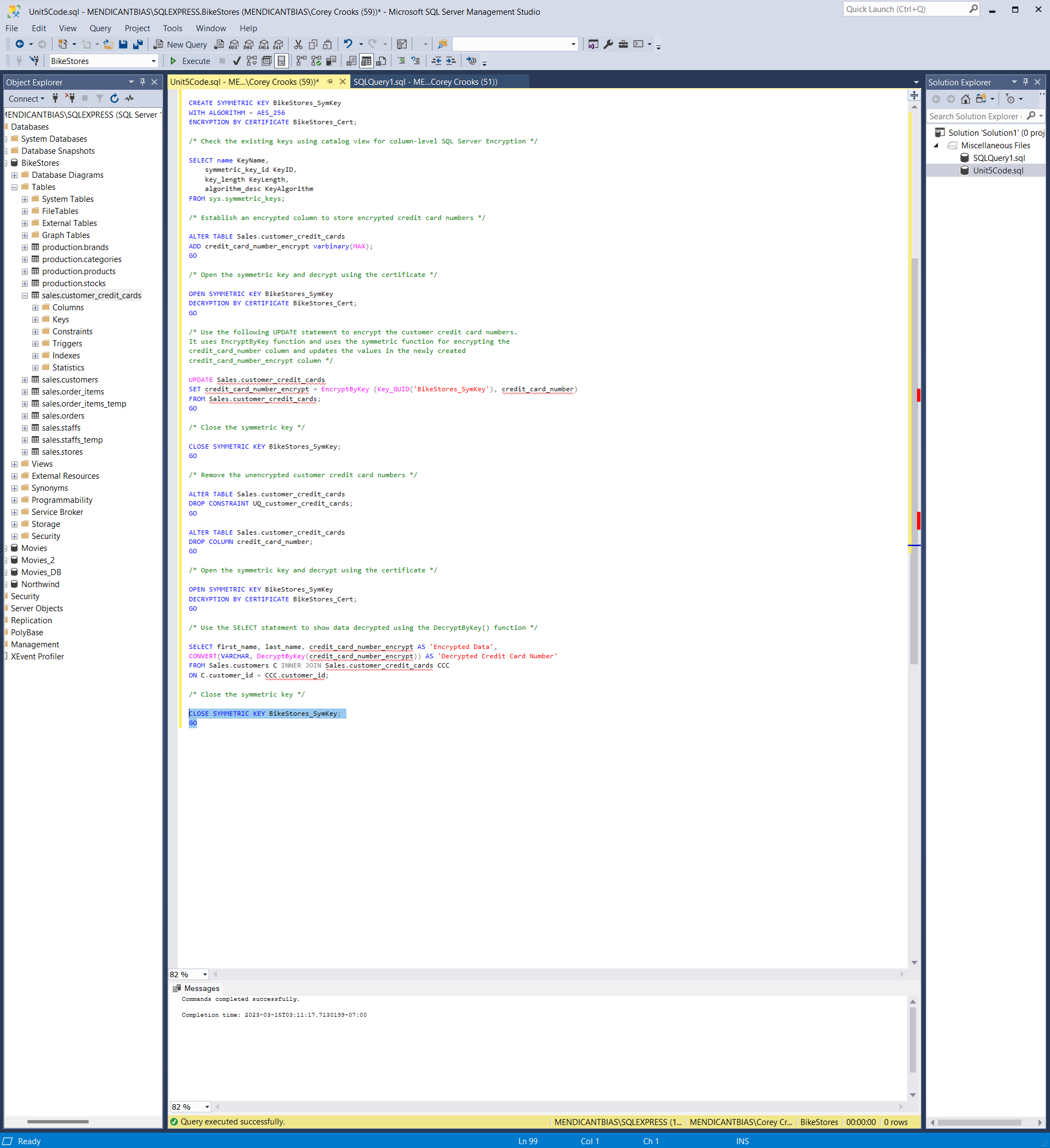
To fix the issue presented in Task 1, the database object now contains roles to assign to users, and most importantly some users to use those roles. Here we have Will and Vanessa; Will exists in the database object as a GeneralUser, while Vanessa exists as a Salesperson. Due to these roles assigned to each user, Will and Vanessa do not share the same permissions. As you can see in the second screenshot of the outputs, Will is unable to see everything from the Sales.Staffs table in the BikeStores database, as a general User would not need that information in their daily operations. This provides a layer of security within the database, and helps everyone sleep better at night.

**Task 3 — Masking**



In the first few lines of our SQL Query, we specify to use the BikeStores database specifically, as we do not want to manipulate data in the wrong database object. Moving down to the next code block, we alter the Sales.customer\_credit\_cards table (specifically the Credit\_Card\_Number column) by creating a mask that will hide the data found within that column for unauthorized users. The next SQL block applies the same property on the email column of Sales.customers. The next statement is relatively simple in that it grants the ability to unmask data to the individuals with the SalesPerson role. Here on out, we run a number of select statements as specific users to see the results. You can see the result of the all select as Will from the Sales.Customers table. In this, you will see that the emails are masked by an “x” with all but the first letter, and extension.

**Task 4 — Encryption**



The SQL statements from this section deal with data encryption. The first block creates a password for encryption that will check authentication for users. Next, we are creating and setting data for a symmetric key to be used in the encryption process for our data. Next, we create a certificate to grant access to data encryption per-column in our database. We then verify our new symmetric key certificate using sys.certificates. The code then establishes and checks a symmetric key to be used in data encryption that uses only one key to lock and unlock data during the exchange. The code gets into the heart of the exchange then by altering the Sales.Customer\_credit\_cards table by creating a column to store the encrypted values in our database. The symmetric key is then opened to ensure the data is “unlocked” for encryption. Then, the credit cards are taken from their original column, encrypted, and then dumped into the credit\_cards\_encrypt column in our credit cards table, after which the symmetric key closes. The unencrypted data for credit cards is then dropped from the database to ensure that only the encrypted data remains. The symmetric key is then opened for decryption, the data in the encrypted column is then decrypted using the prior-specified algorithm, and a SELECT statement is used to show that data. When that executes, the symmetric key is closed.

This database schema uses symmetric encryption, though asymmetric encryption is also available. The two differ in the way data is exchanged. With symmetric encryption as seen above, only one key is used for both parties to encrypt and decrypt the data being sent in traffic (University of Maryland, Baltimore County, n.d.). With asymmetric encryption, two distinct keys are used (University of Maryland, Baltimore County, n.d.). One key locks (or encrypts) the data, while another distinct key is used to unlock (or decrypt) data. While neither method is generally ‘better’ than the other, both share an equal amount of strengths and weaknesses. One weakness of symmetric keys would be their vulnerability (V., 2012). If a single key is stolen along with a packet of data, the attacker would have everything they need to decrypt that data since there are now two potential points of failure that is equally as devastating. With asymmetric keys, the attacker would need access to the specific key designated to decrypting data in order to turn that encrypted data into anything useable—halving the vulnerability associated with the encryption. That security enhancement does come with disadvantages, however (Miller, 2016). As keys become more diverse and specific, one needs to take much more care to ensure keys are maintained. This is because Asymmetric keys require much more organization to keep running efficiently, as the required key count has been doubled for the sender and receiver.

# **References**

Miller, B. (2016, August 6). *8 Pros and Cons of Asymmetric Encryption*. Retrieved from GreenGarage: https://greengarageblog.org/8-pros-and-cons-of-asymmetric-encryption

University of Maryland, Baltimore County. (n.d.). *Encryption*. Retrieved from UMBC.edu: https://redirect.cs.umbc.edu/~wyvern/ta/encryption

V., D. (2012, September 5). *Advantages and Disadvantages of Symmetric Cryptography*. Retrieved from 1000Projects.org: https://1000projects.org/advantages-and-disadvantages-of-symmetric-cryptography.html