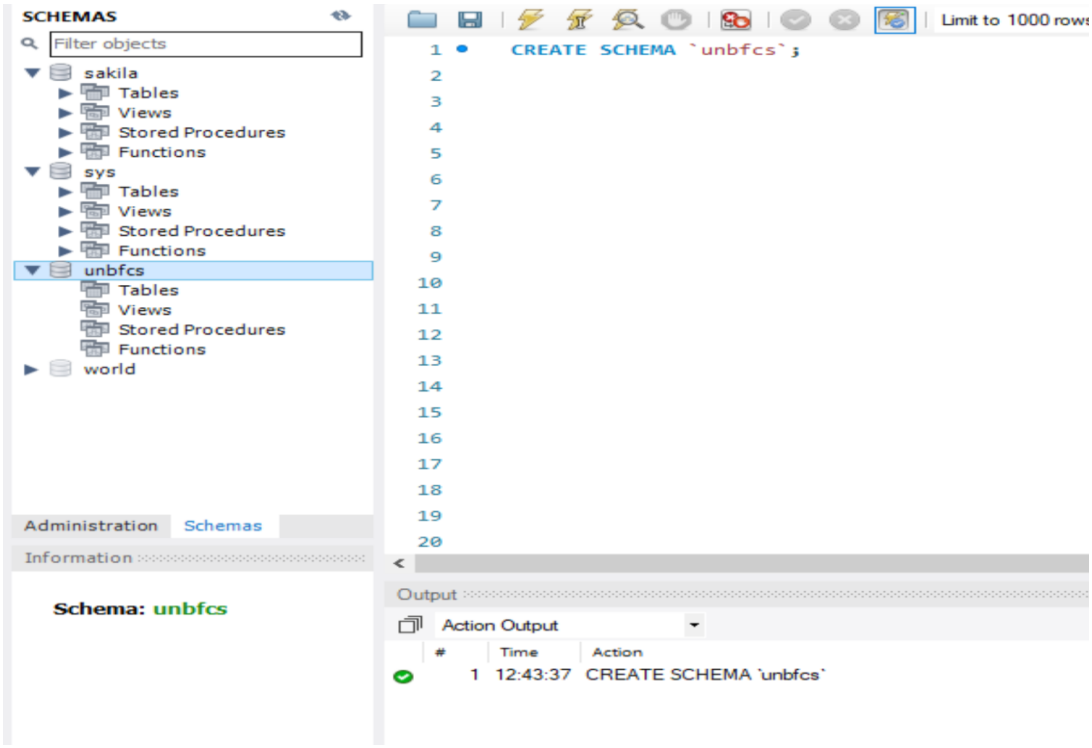
NETWORK SECURITY

Assignment 3

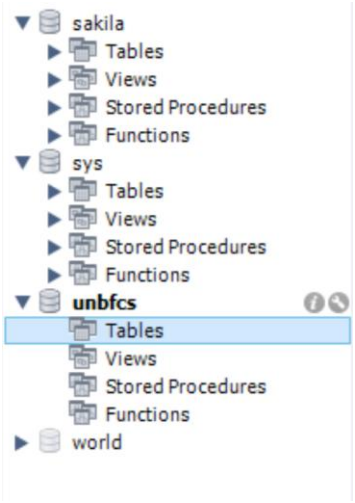
Soheil Shirvani 3720505

A. Data Integrity

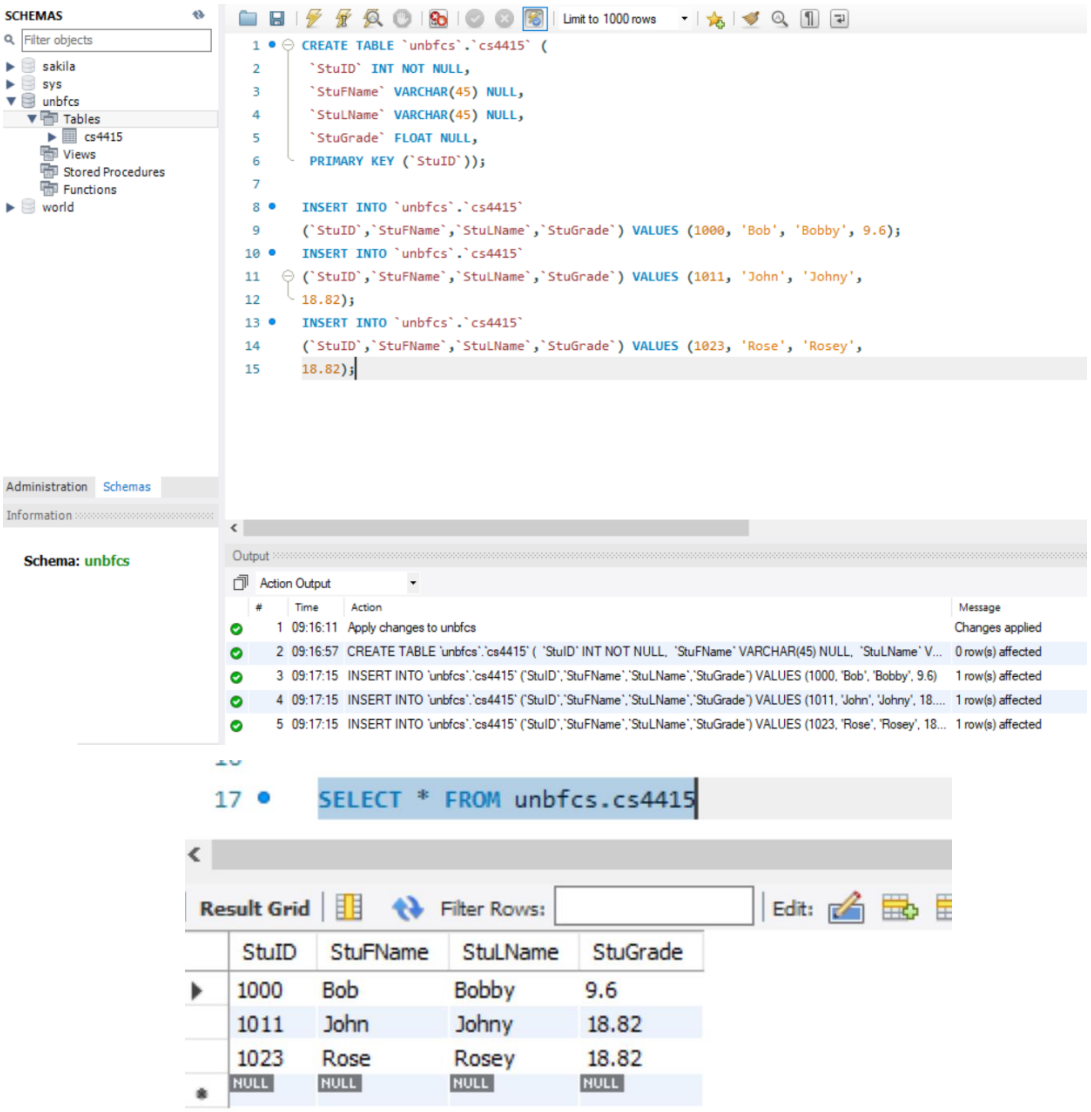
1. Create Schema for UNBFCS



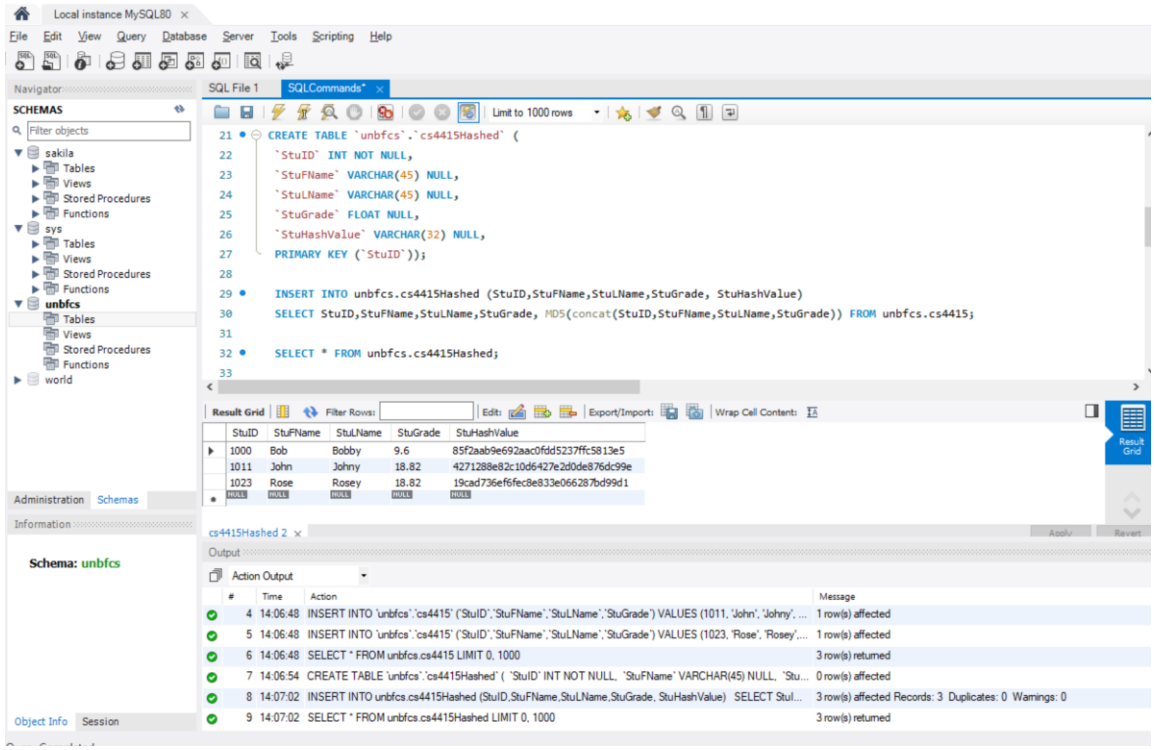
2. Expand Database and Select Tables



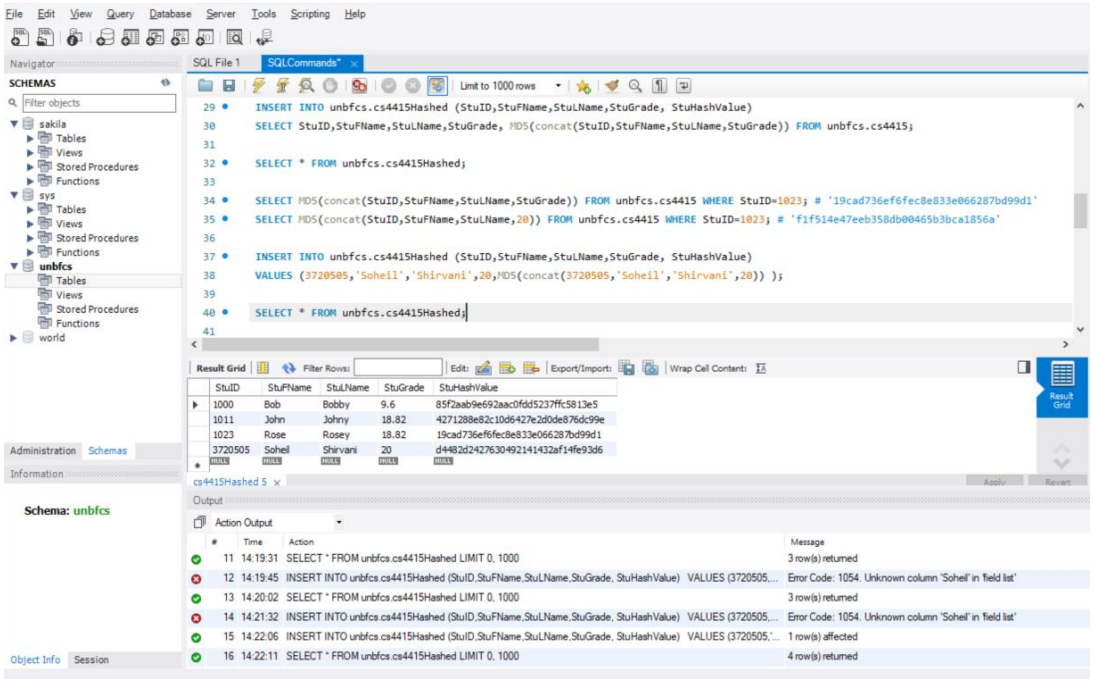
3. Create Table CS4415 with Specified Fields and Insert Values:



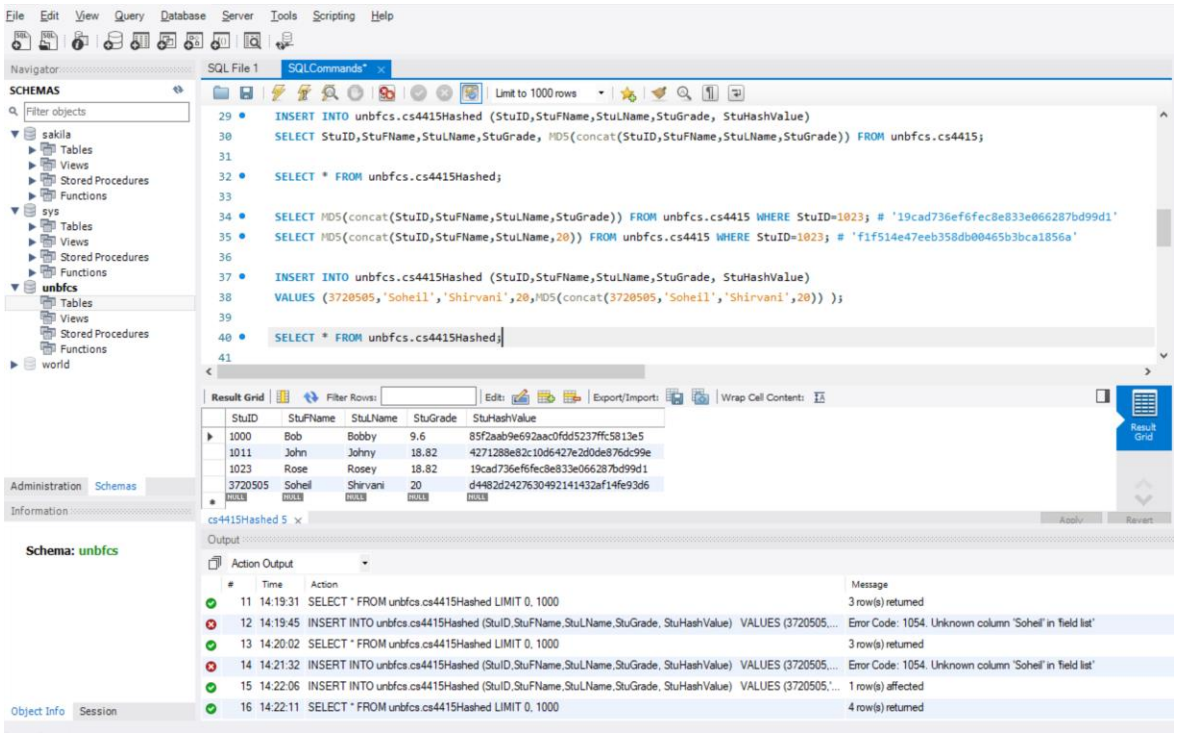
4. Create CS4415Hash Table with Addition column and Read and Insert Hash value to the new column:



Question 1: Insert (Soheil Shirvani 3720505 20) into New Table:



Question 2: Modify Grade in new Inserted Value:



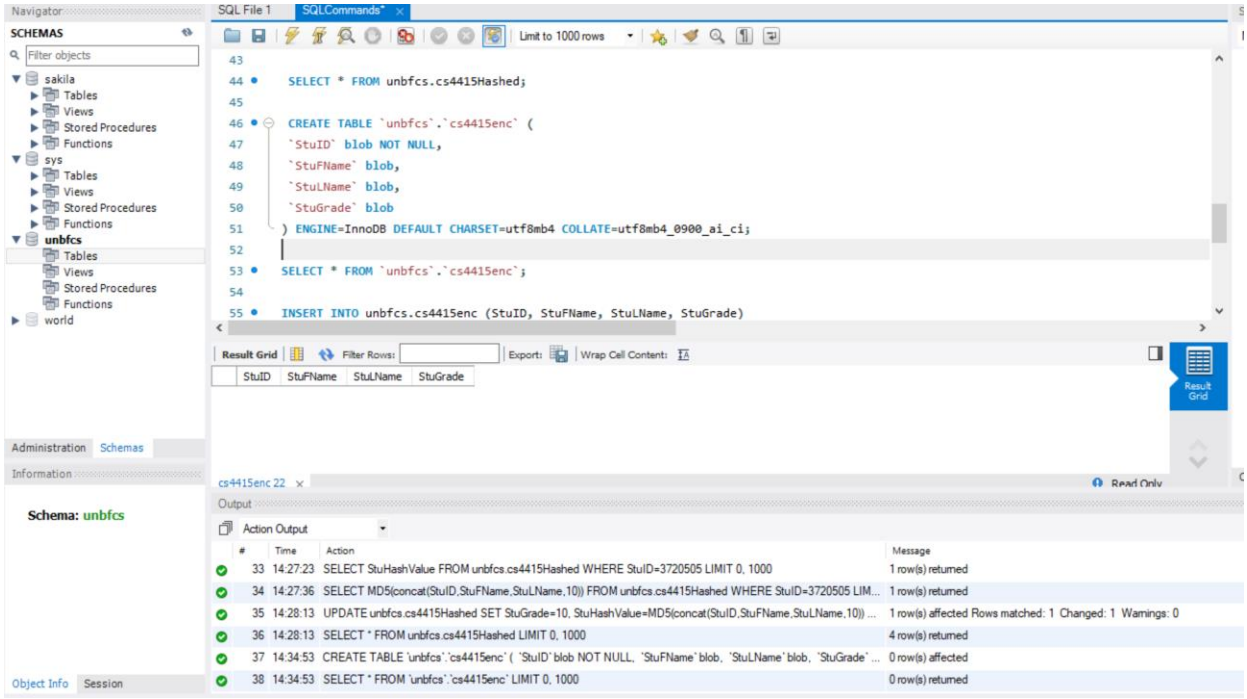
Question 3:

In this type of attack, an intruder can change each of the records and validate his changes just by updating the hash value using new values. To protect the table against such attacks, 3 important defense practices can be performed:

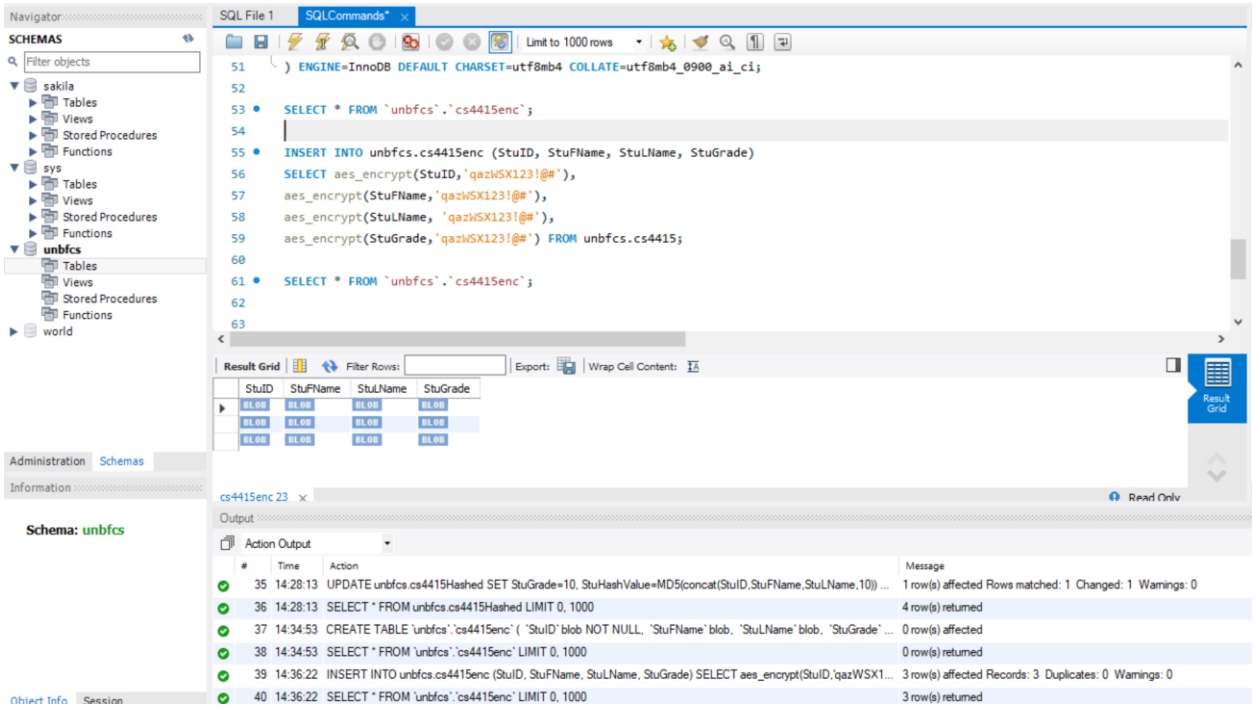
1) We can save the hash values on a separate table and ensure integrity by checking that table. This way the one who modifies a value can't change the hash value.
2) We should check if the modify value is legal and then update the hash value. This means we can't simply update the hash field of the table
3) We should consider access control. Accessing to database must be protected and given to specific people who we have acknowledged. This way if something changed in the database, digital forensics could reveal the thief identity though the logs.

B. Data Confidentiality

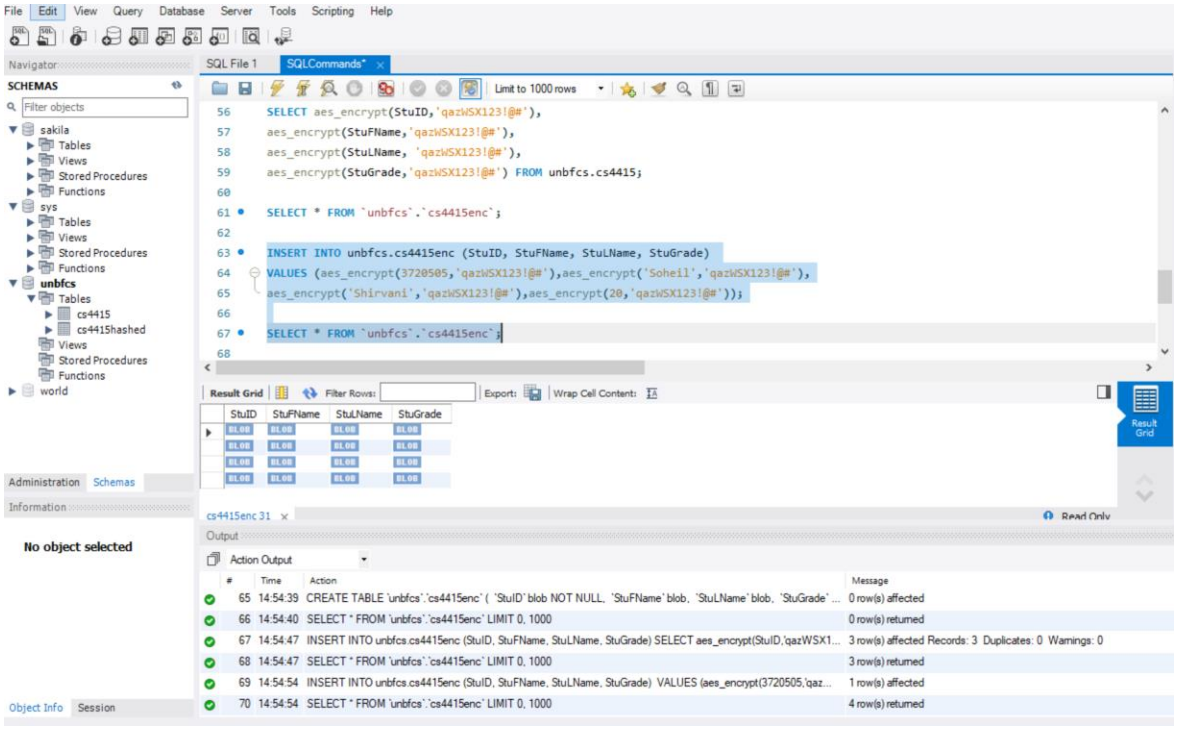
1. Create Encrypted Grade Table:



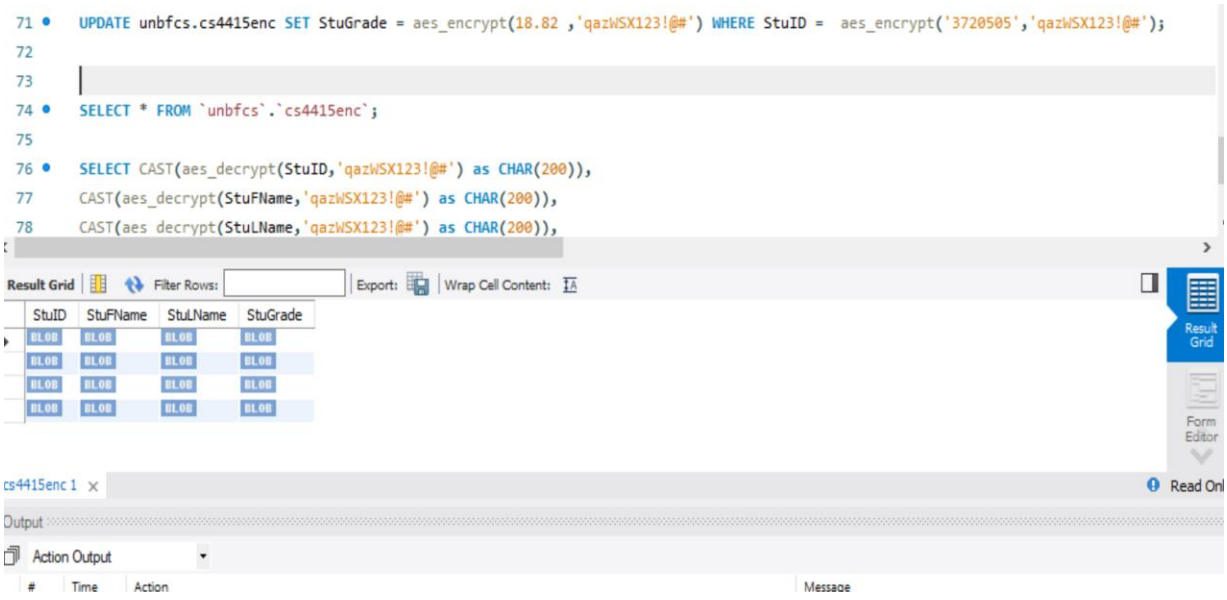
2. Read and Encrypt the Previous values into the new one:



Question 4: Insert My Encrypted Value (3720505,Soheil,Shirvani,20)



Question 5: Update my grade with encrypted value for 18.82



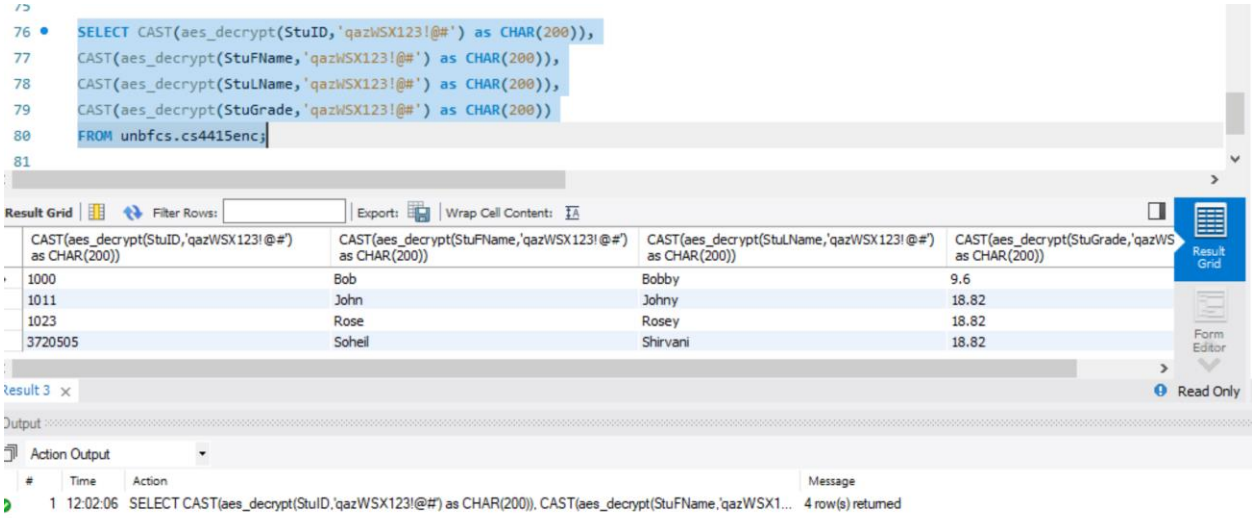
Here we can't see the result. Question 7 Shows the result after casting values to Characters.

Question 6:

The problem is if someone gets the key he/she can access the values and modify them by encrypting the value using the same key. There are 2 defnse strategies we can do.

1) We should make sure we protect the key in a safe place outside this database. This makes sure that the theif can simply access and gets the key.
2) One thing is we can sue AES CBC mode using a random vector and concatenating it with the encrypted data. So here we are enforcing the AES algorithm making sure an attacker can't break the algorithm.
3) Another problem is when an attacker access encrypted values he/she can finds similar encrypted values and get to know which users have same fields. This way attacker can compromise the privacy of the users. One solution to that is we can have different keys for each row. For this we can have a base key and add some pre-defined salt to it for each row like using users ID with the encryption key. With this solution we can make sure that we don't have similar values and if an attacker get to break the algorithm he/she has only identified one row and can't modify other rows.
4) Another solution is to define access control so if a DB administrator changed a value we can go though the logs can reveal the identity of the attacker

Question 7: Showing the plain text of the table



So we can see the update as well as all the insertions worked well.