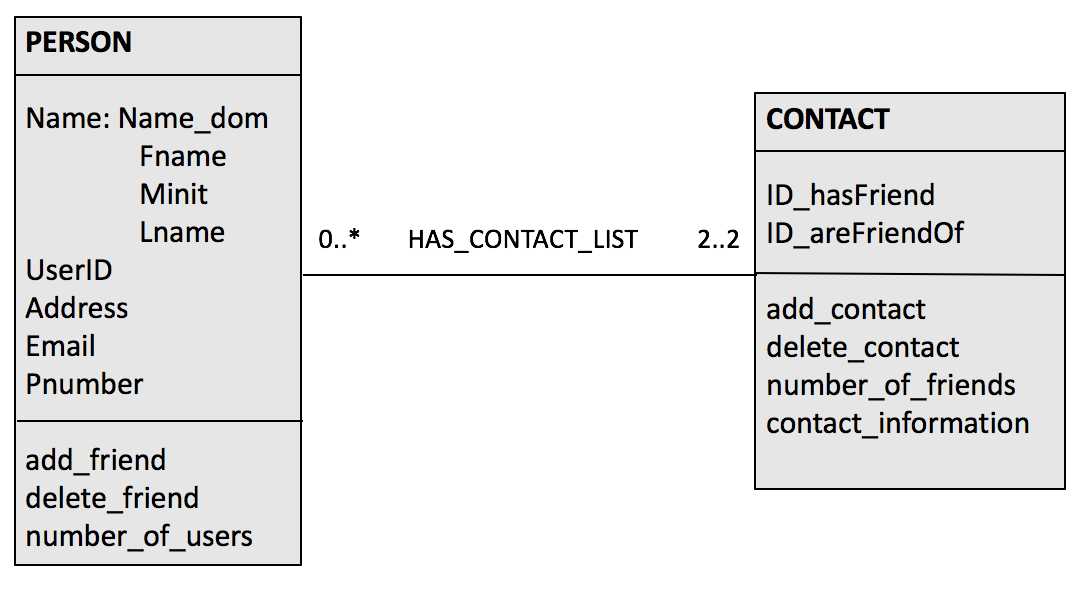
CIS 530 Extra Lab Assignment 6

Jiahui Wu #2670210

Building a social network web service system named FriendBook

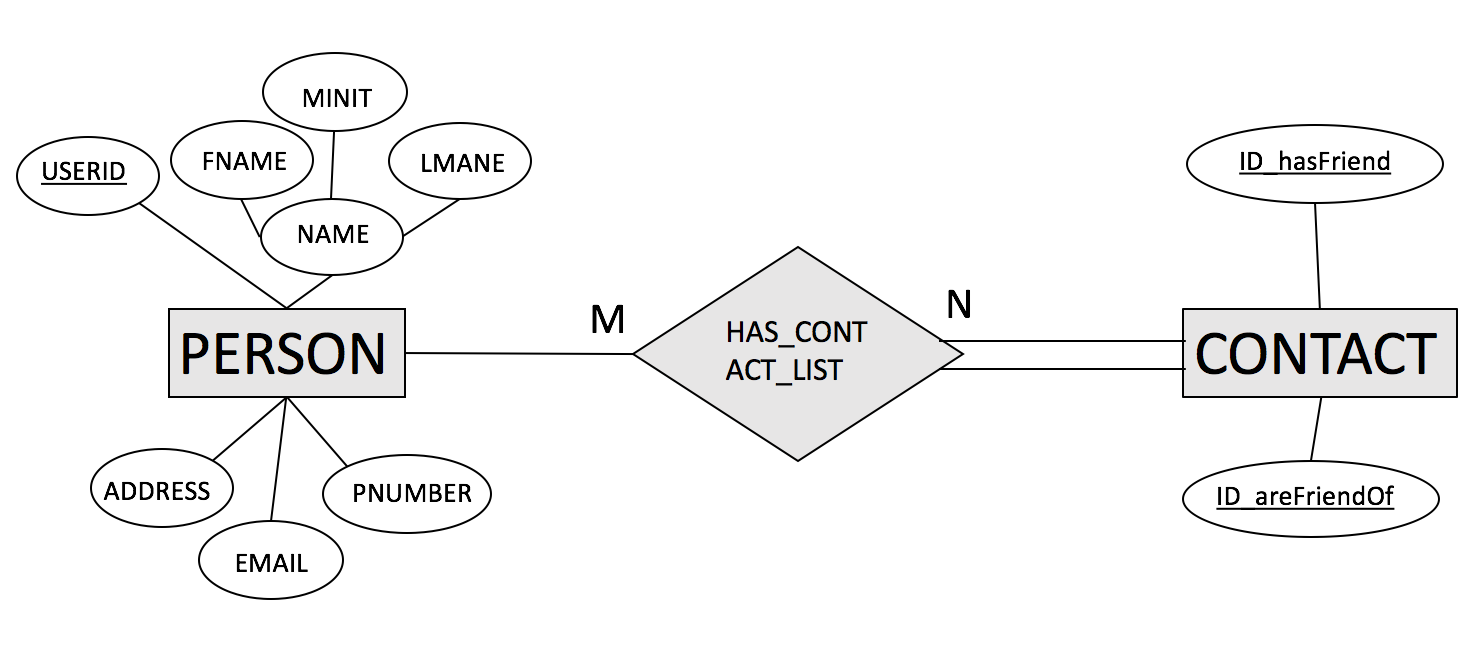
PART 1

1. The friendbook conceptual schema in UML class diagram notation.



There are two classes, PERSON and CONTACT, the attributes are listed, USERID and ID\_hasFriend, ID\_areFriendOf are in the same data field. The multiplicities of PERSON to contact is 0..\*, for each entity p in PERSON, p could participate 0 relationship instances in CONTACT(p has no friend), or p could participate no limit relationship instances (p has many friends). The multiplicities of CONTACT to PERSON is 2..2, each entity in CONTACT must participate 2 relationship instances in PERSON (each entity has two person’s userID), no more, no less.

2. E-R diagram



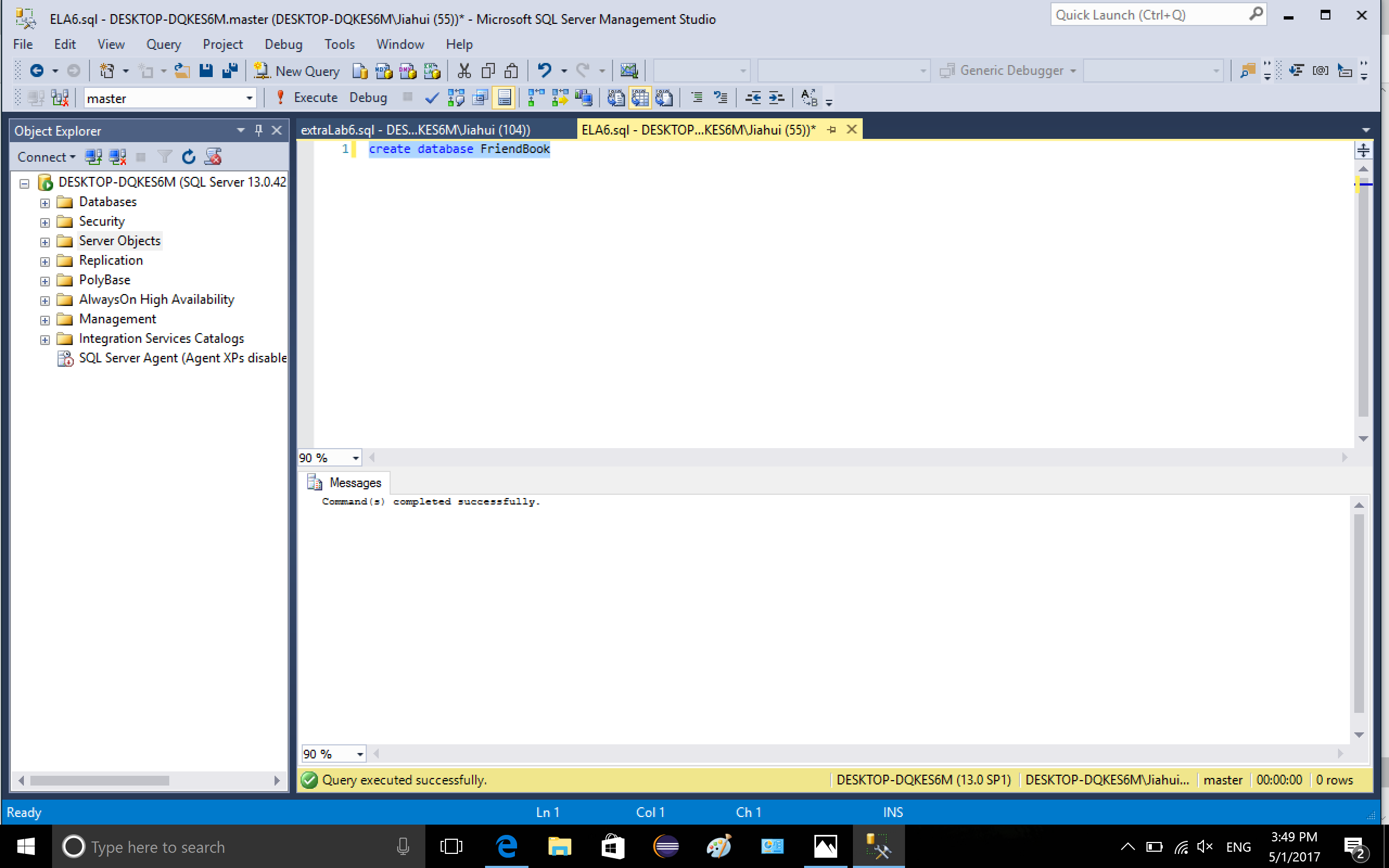
Account information is stored in table PERSON, primary key is USERID. Contact list information is stored in table CONTACT, primary key is the composition of ID\_hasFriend and ID\_areFriendOf. One account could have several contacts, but not all user have a contact list (in case a person does not have a friend), it is not total participation. Every contact list has 2 user accounts in PERSON, it is total participation. The relation between two entities PERSON and CONTACT is M:N.

PART2

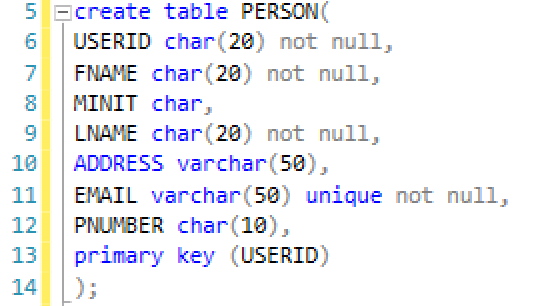
The database has two tables, one is PERSON, it is used to store the information of each user. Another is CONTACT, it has two attributes, both have the same data field as USERID in PERSON, and both are FK that reference USERID in table PERSON.

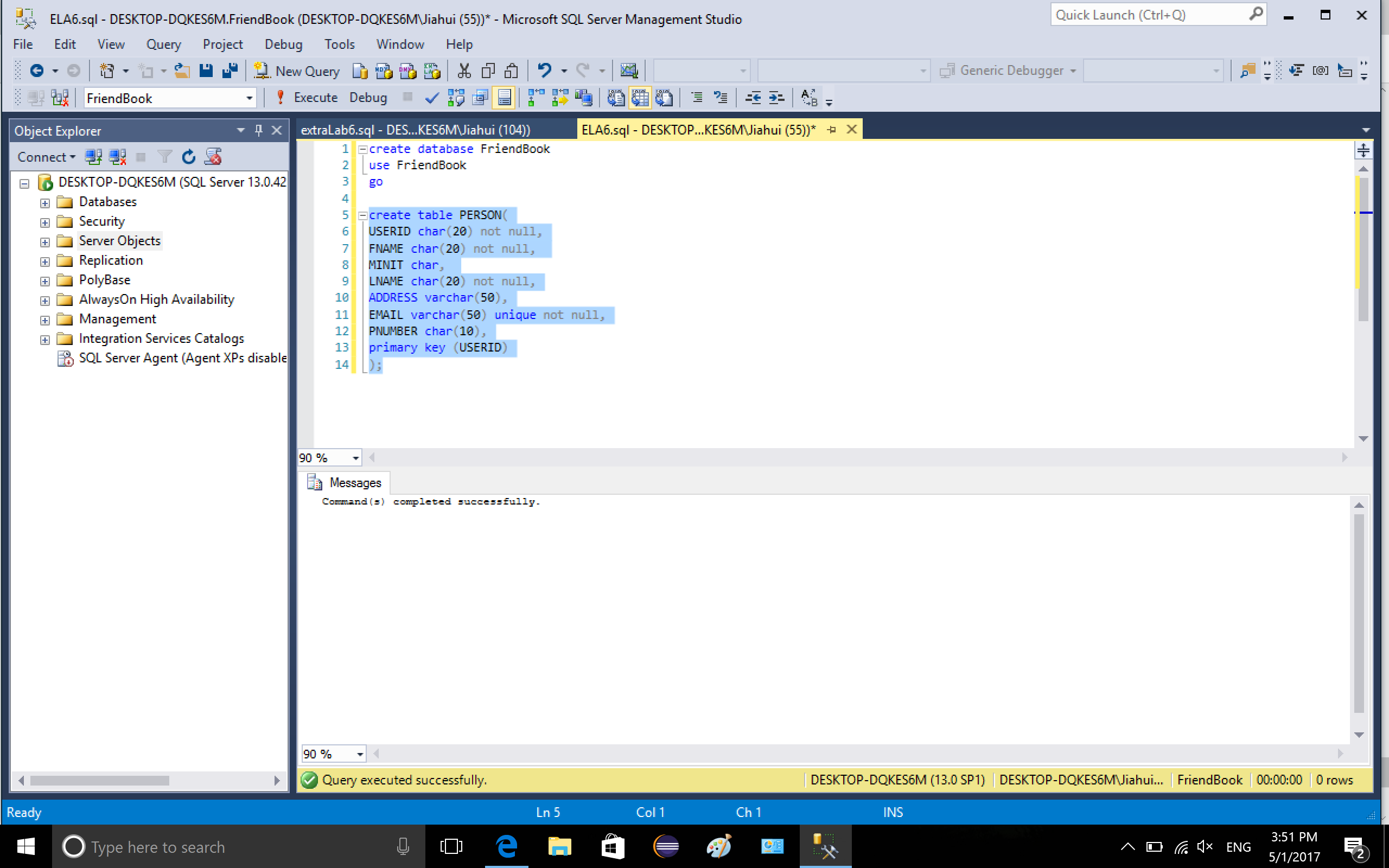
The database allows data insertion, update (USERID in immutable), retrieval (such as show a person’s friend list), deletion (such as deleting a person from both tables).

1. create database named FriendBook and use the database

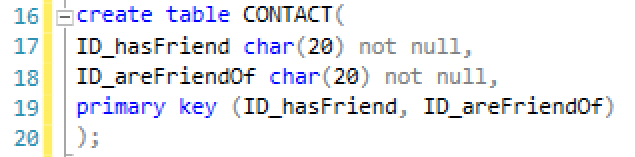


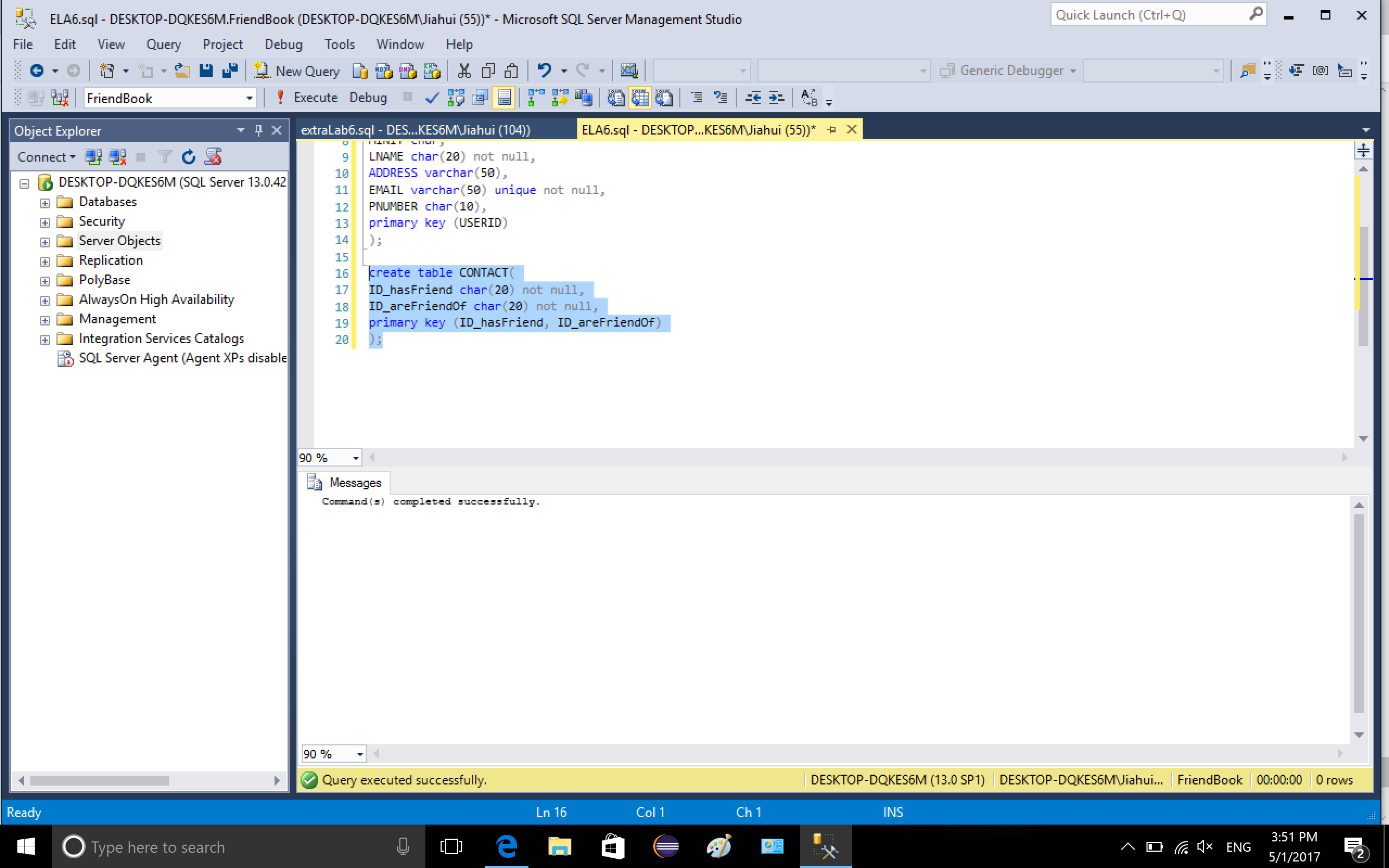
2-1. create table PERSON, the attributes are USERID, name, address, email and phone number. USERID is primary key, email address is unique.



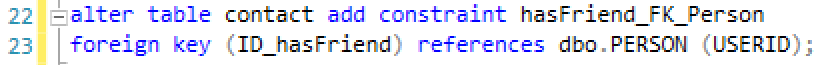


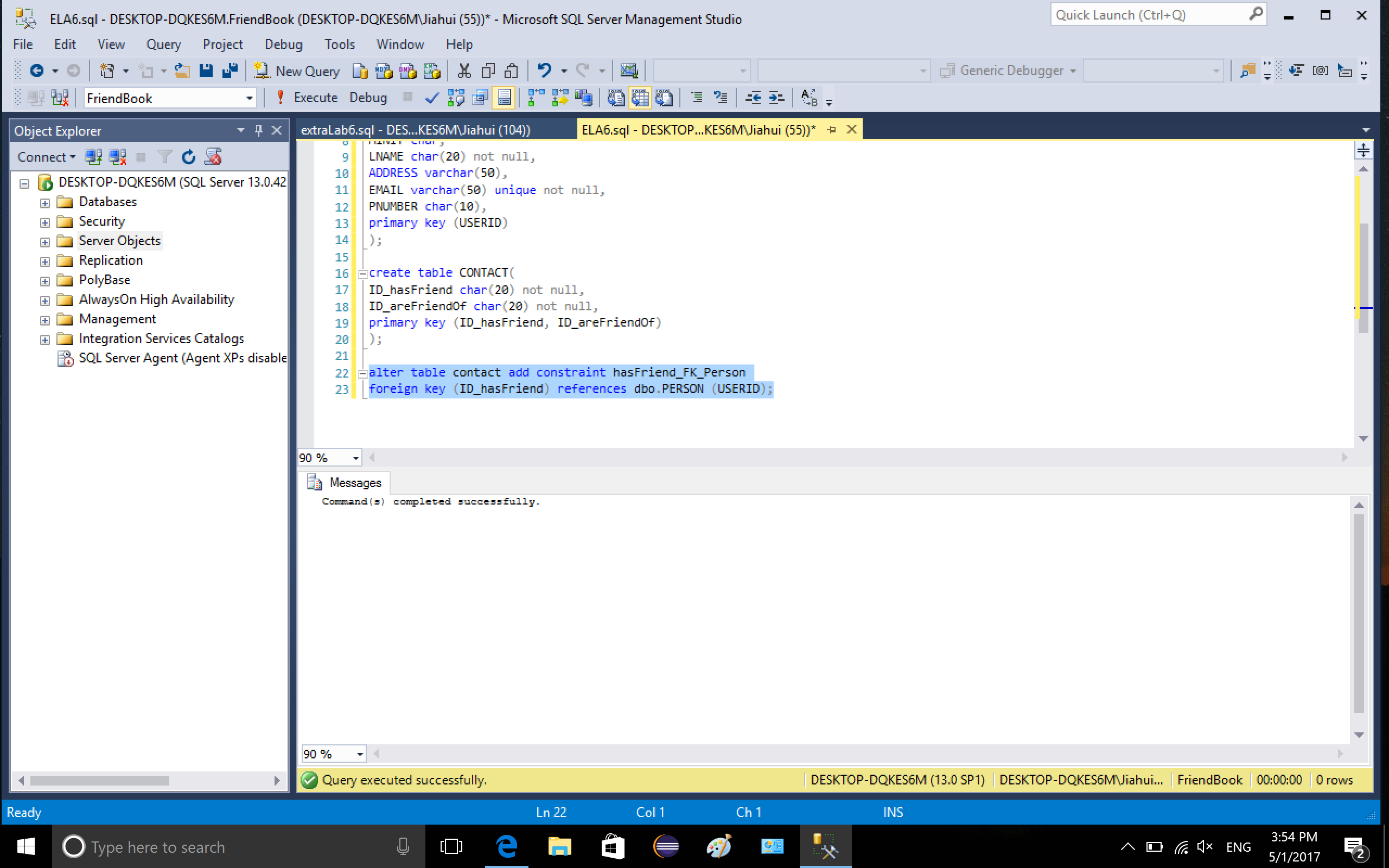
2-2. create another table named CONTACT, it has two attributes, one is ID\_hasFriend and ID\_areFriendOf, both are in the same data field as USERID in table PERSON. The table has a composite key.



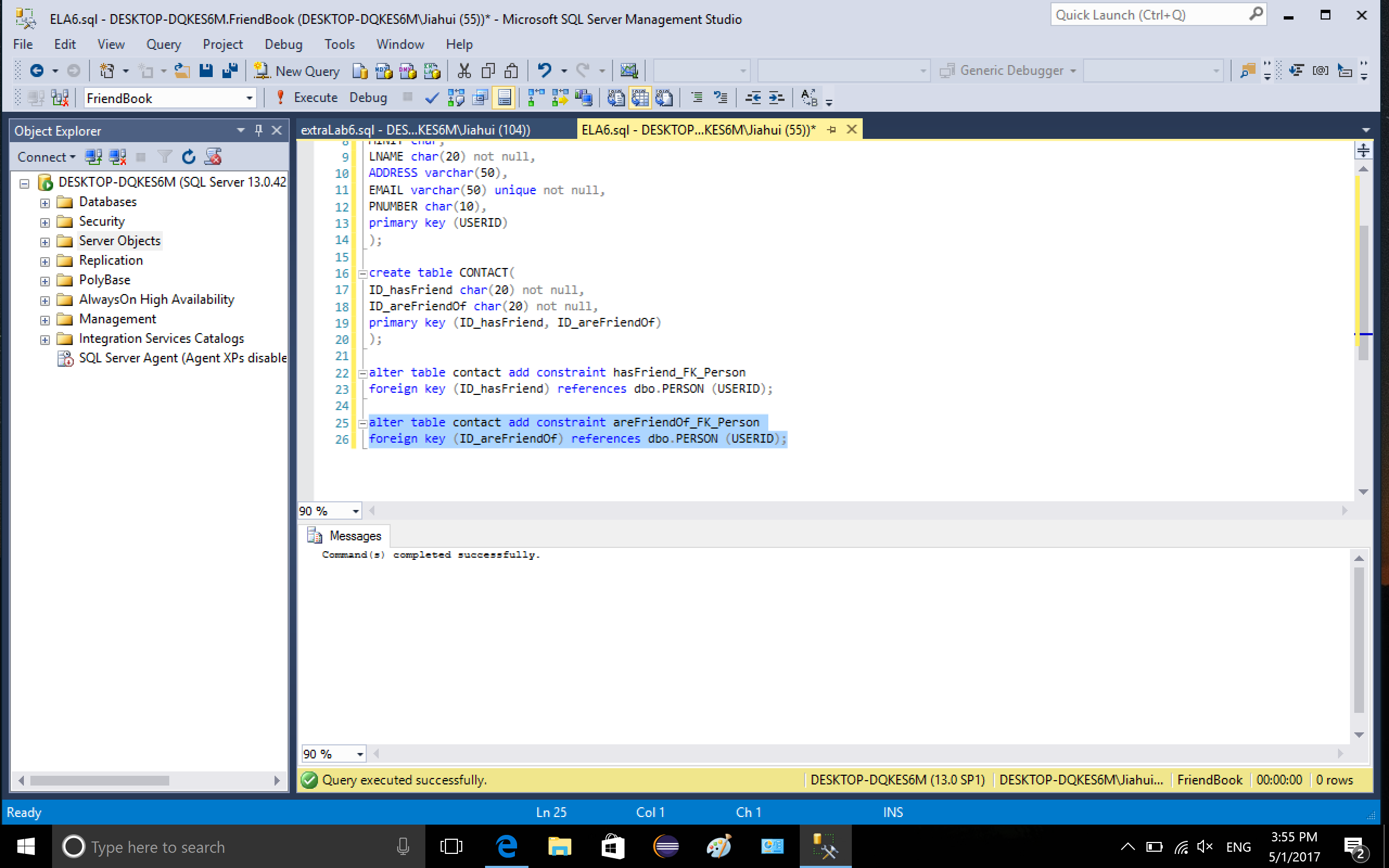
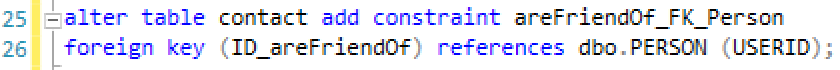


3-1. Add constraint, define the attribute ID\_hasFriend in table CONTACT as foreign key referenced PK in table PERSON

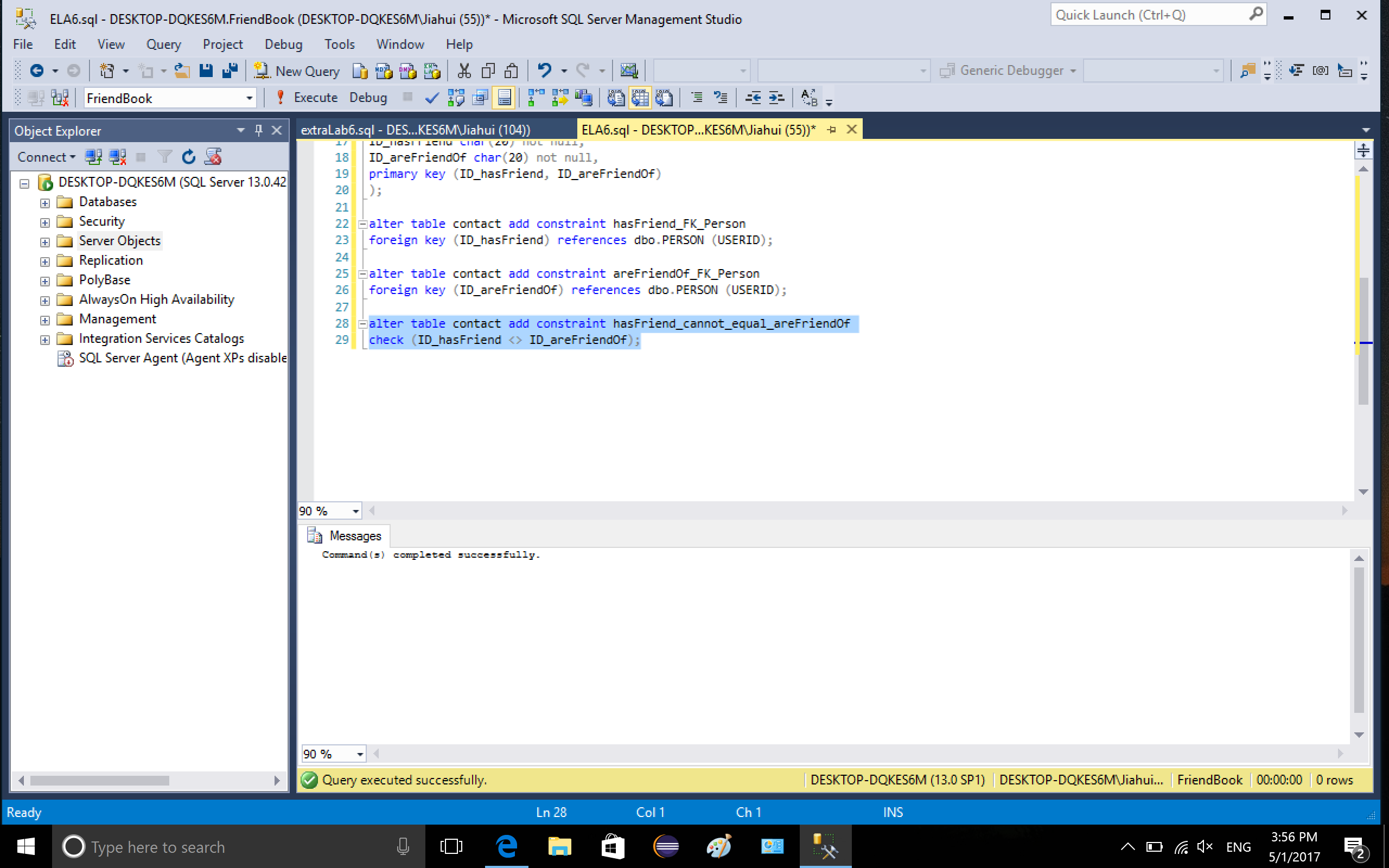
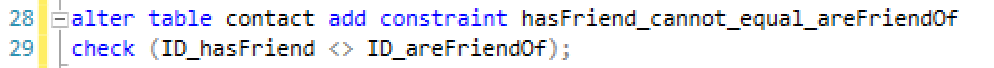




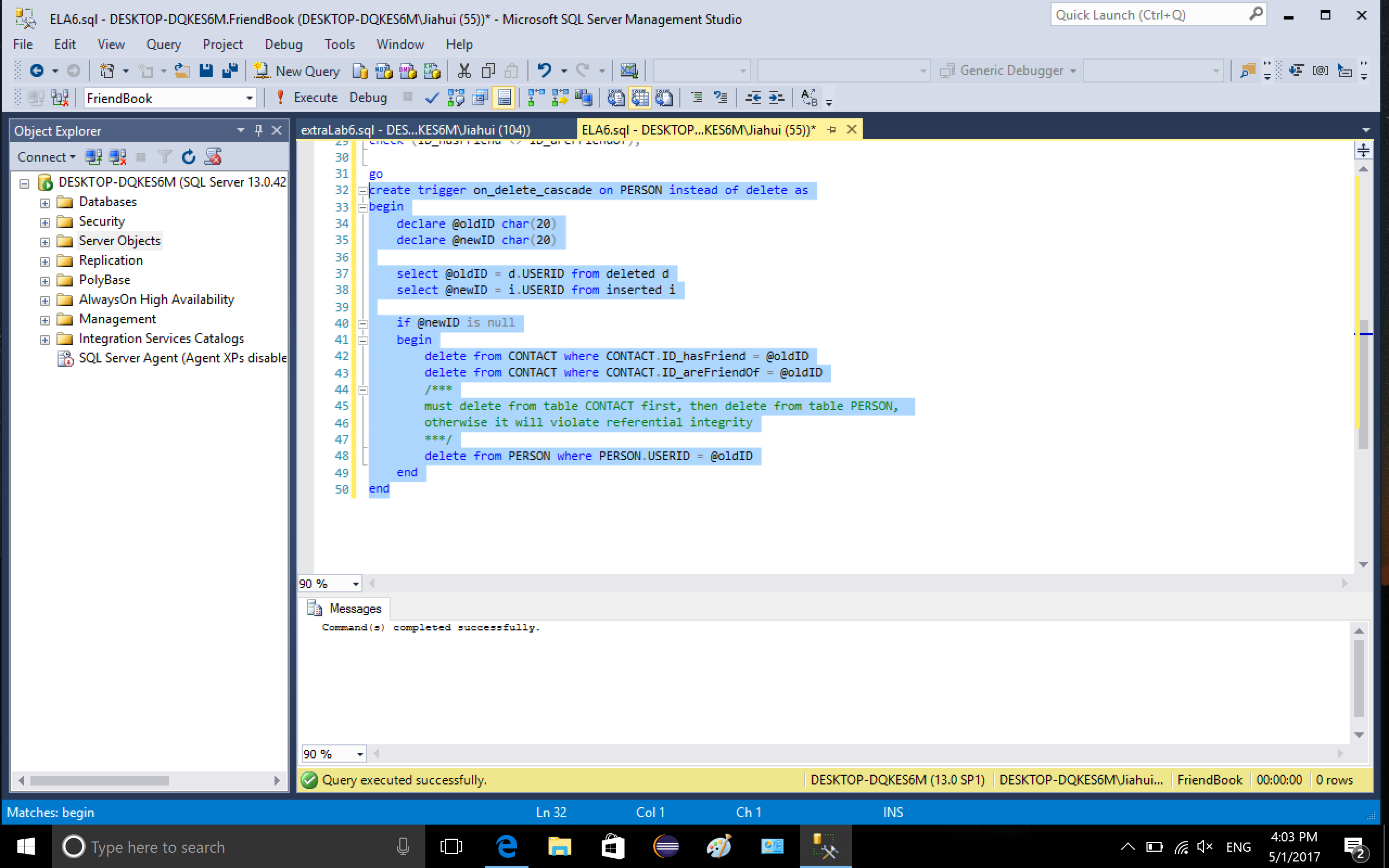
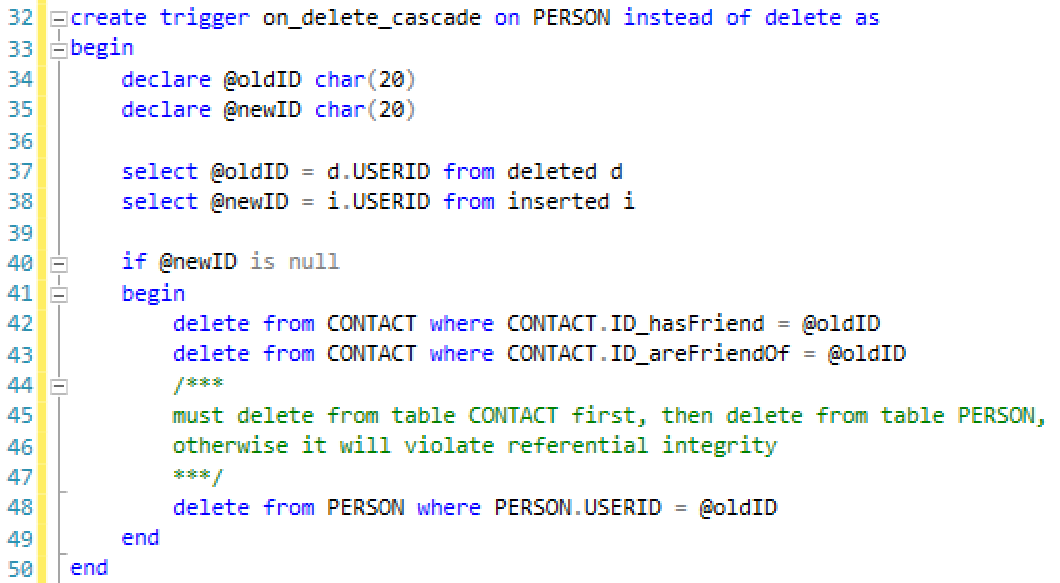
3-2. Add constraint, define the attribute ID\_areFriendOf in table CONTACT as foreign key referenced PK in table PERSON



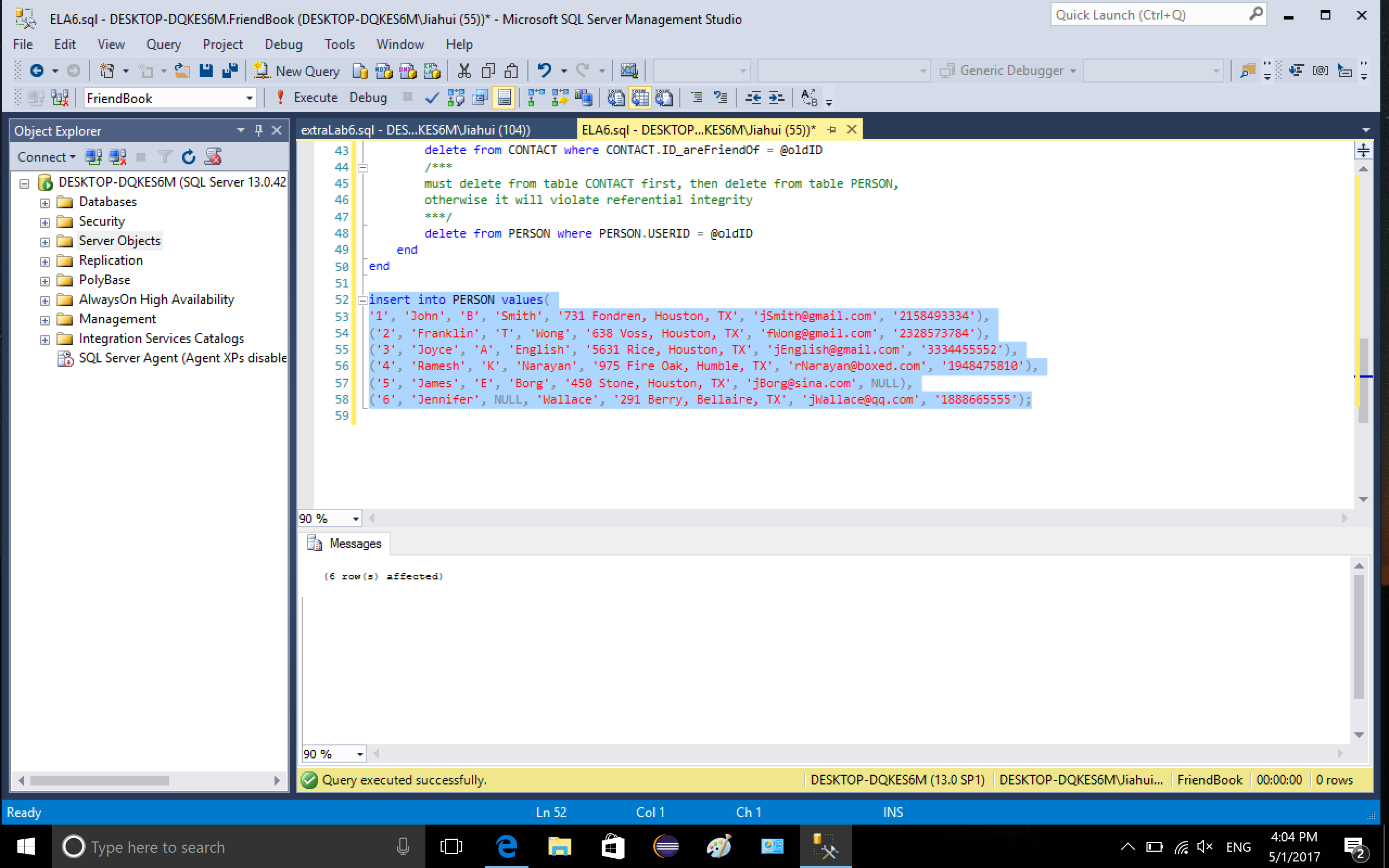
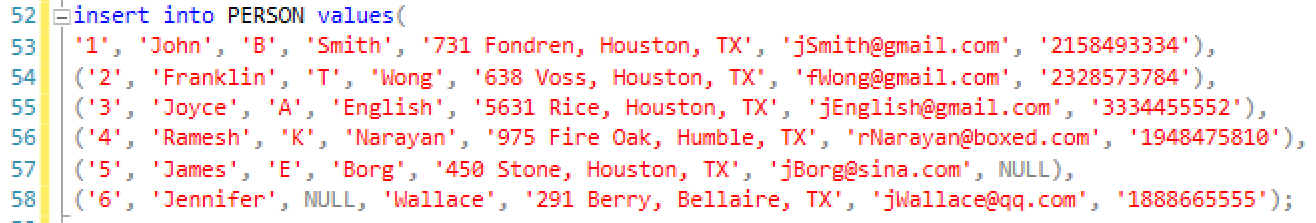
3-3. Add constraint to CONTACT, ensure that the two attributes are not equal because a person cannot add him/herself as friend



4. If we delete a tuple in table PERSON, say a person choose to close his/her account, then we want this deletion to cascade to table CONTACT, the deleted ID will be removed from CONTACT as well, whether it is in ID\_hasFriend or ID\_areFriendOf. So I create a trigger on delete cascade on person. Because of the two FK defined in the table CONTACT, if I do not create the trigger, SQL server does not allow the deletion to cascade to CONTACT table. So I use instead of delete rather than for/after delete to make sure the trigger override delete command. Two variables are declared in the trigger, we could obtain the old and new USERID during deletion. Since there is no insertion, new ID is null. We need to delete the target values from both ID\_hasFriend and ID\_areFriendsOf in table CONTACT before we could delete the target values from PERSON, such that we do not violate referential integrity.

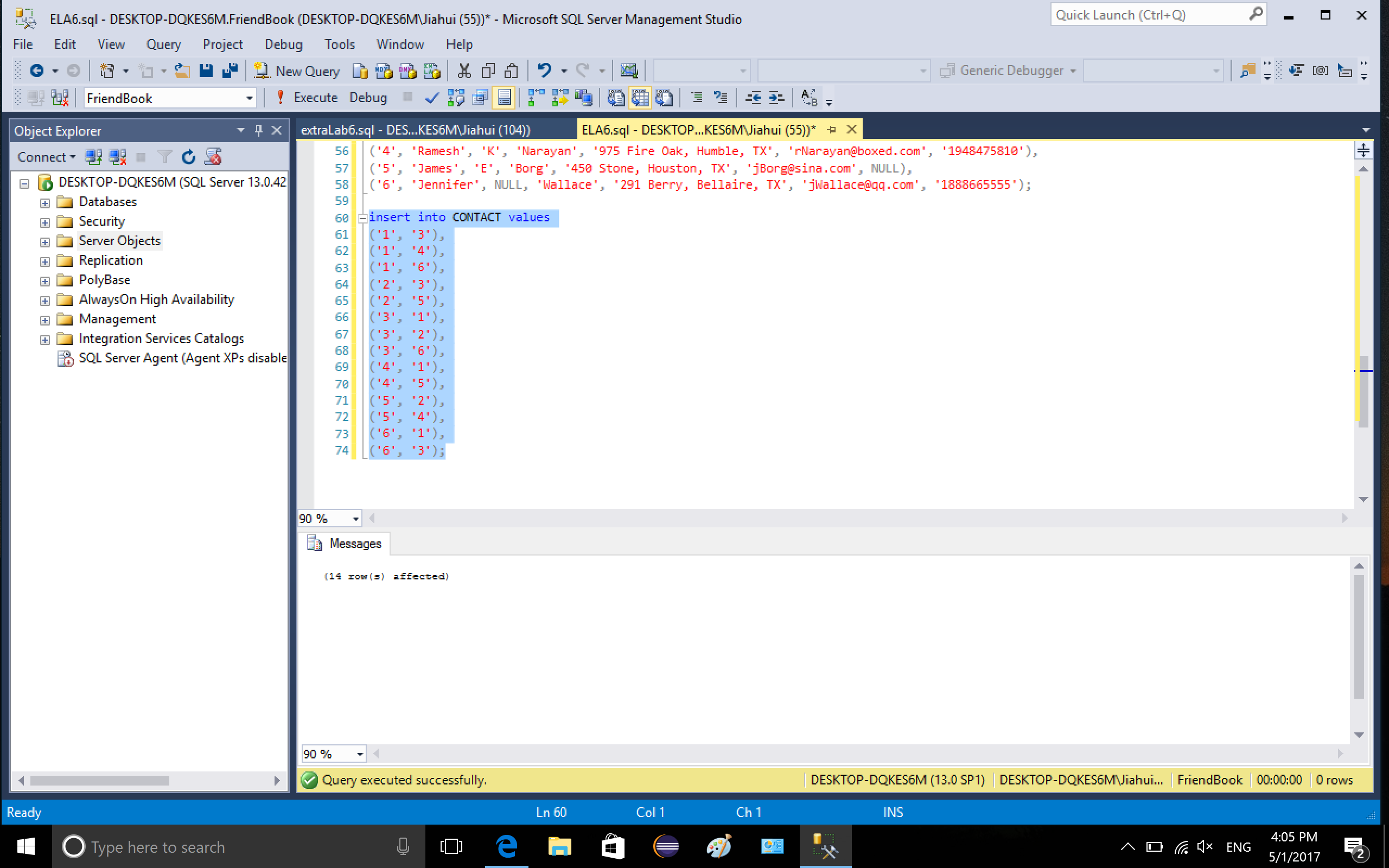


5-1. insert values to table PERSON

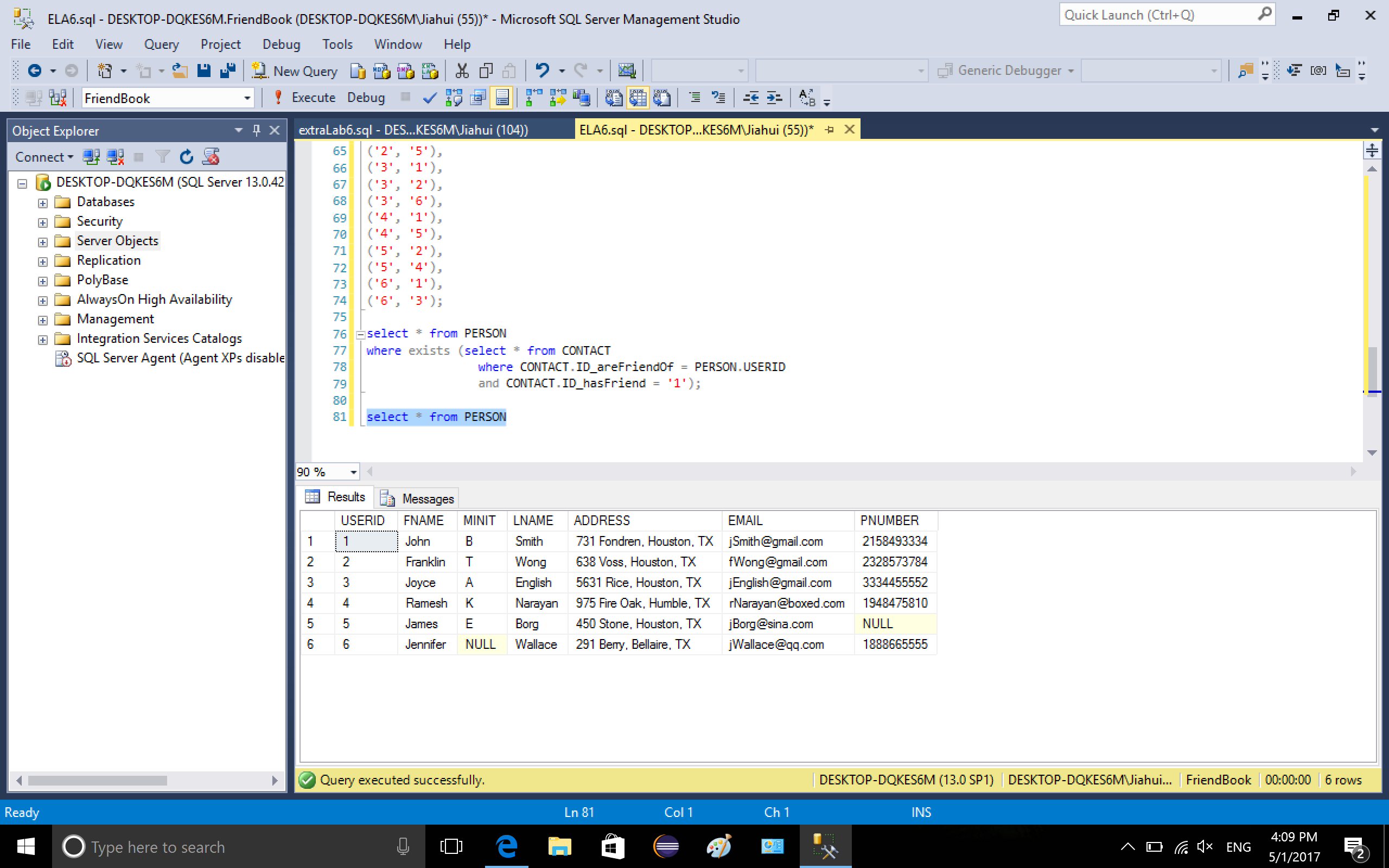


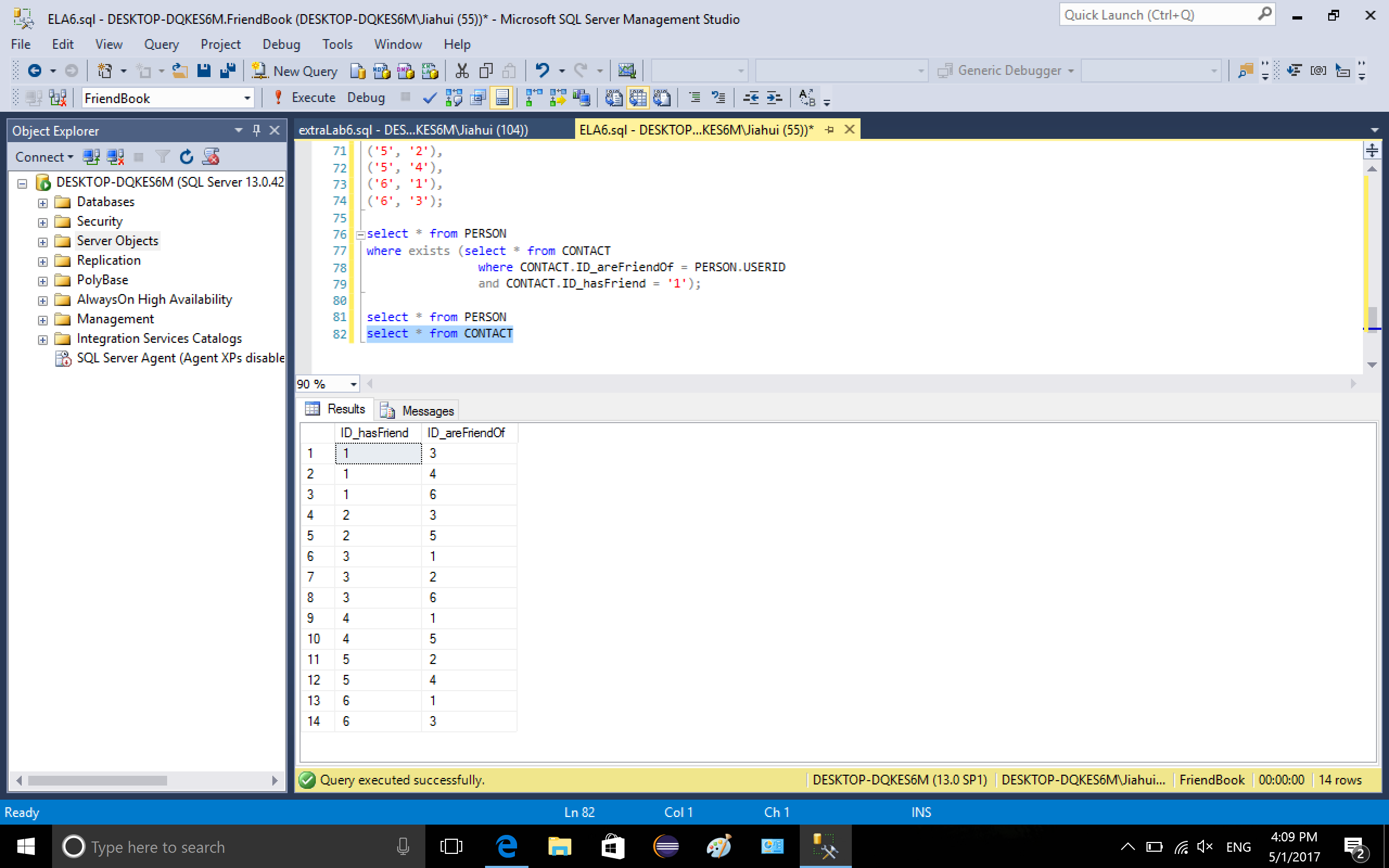
5-2. insert values to table CONTACT



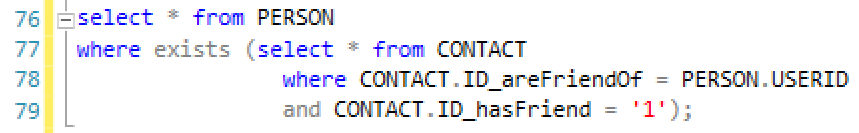


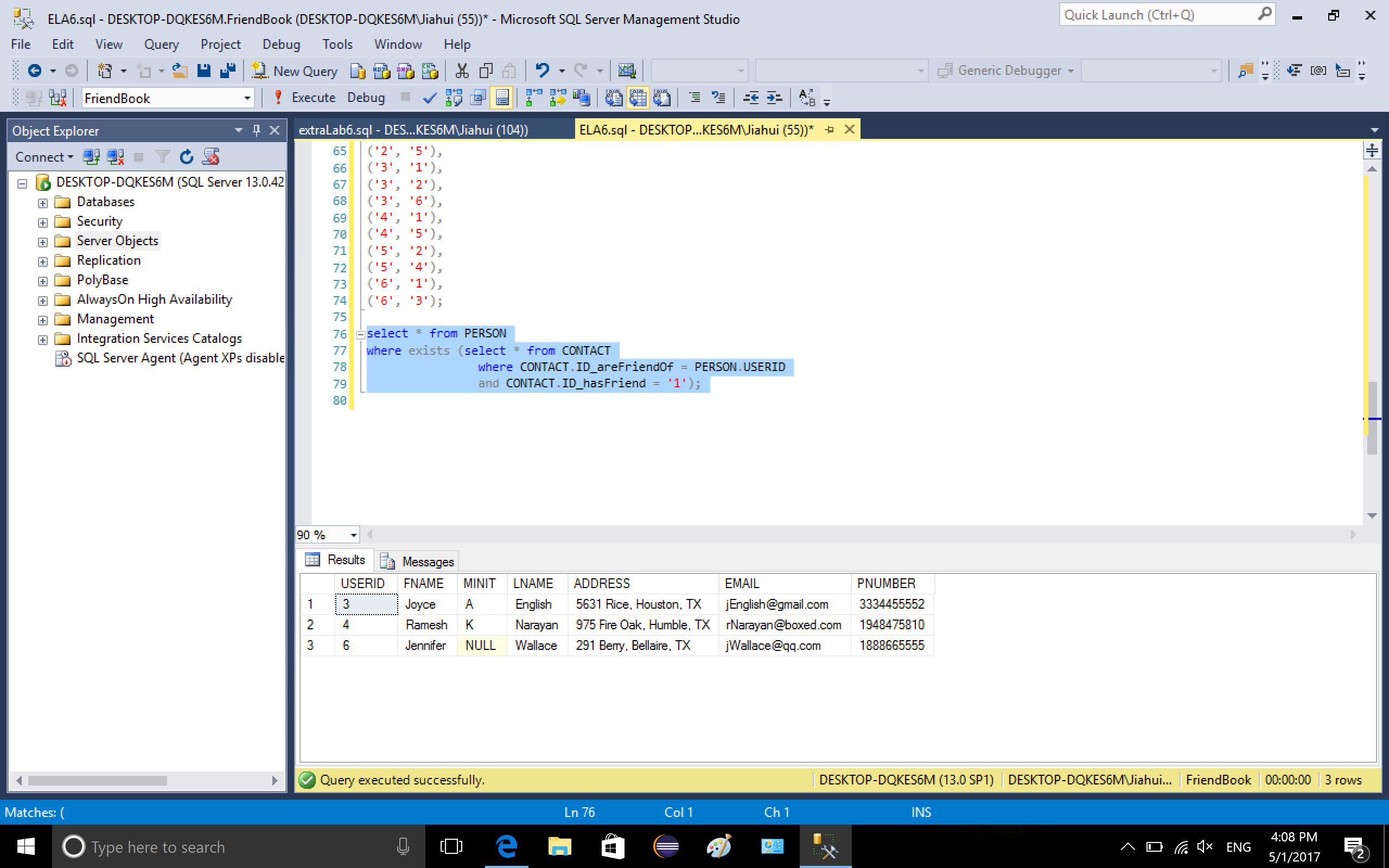
5-3. show table PERSON and CONTACT



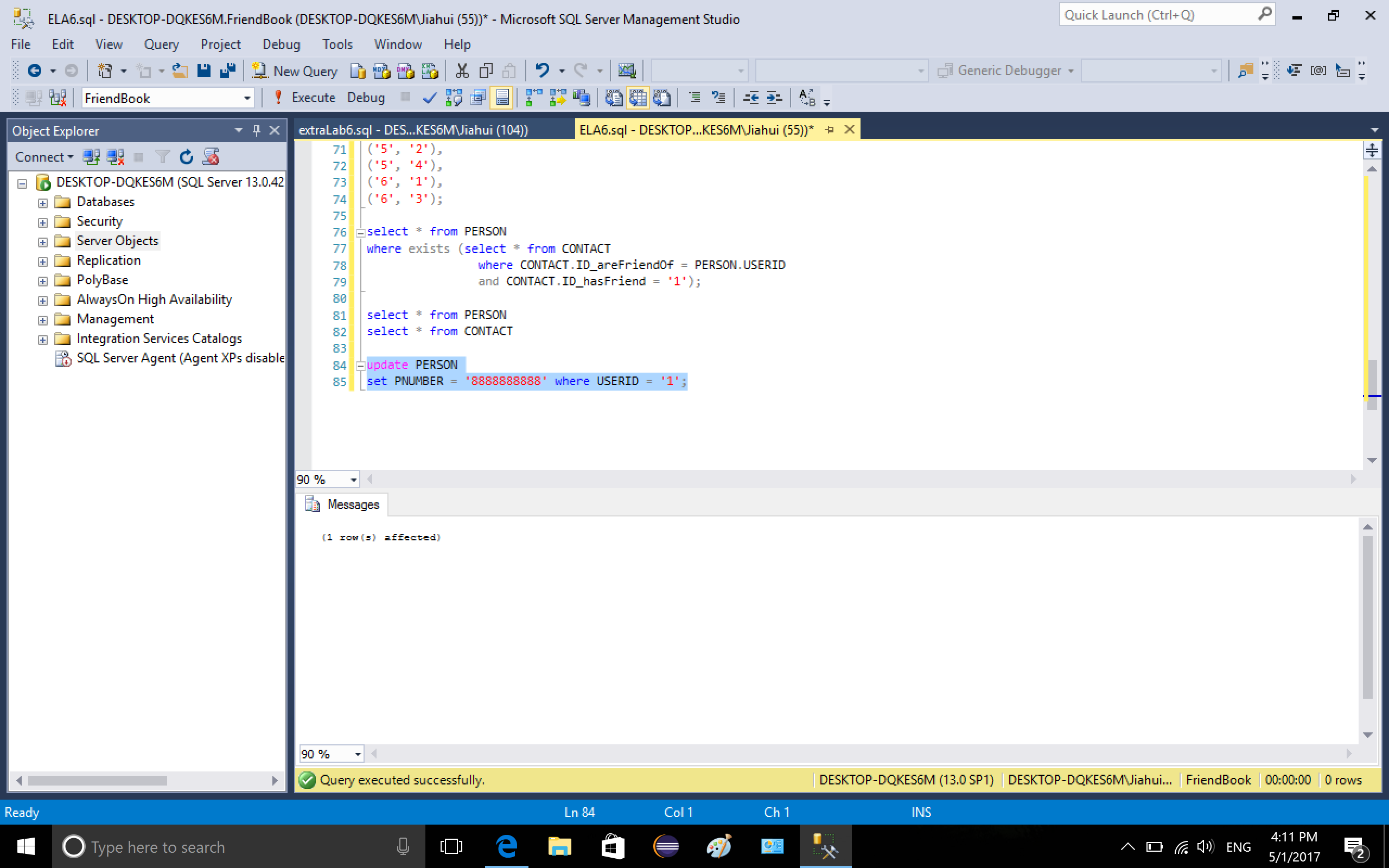


6. retrieve data, show the friend list of person whose USERID = 1. 1 has 3 friend, which are 3, 4, 6.

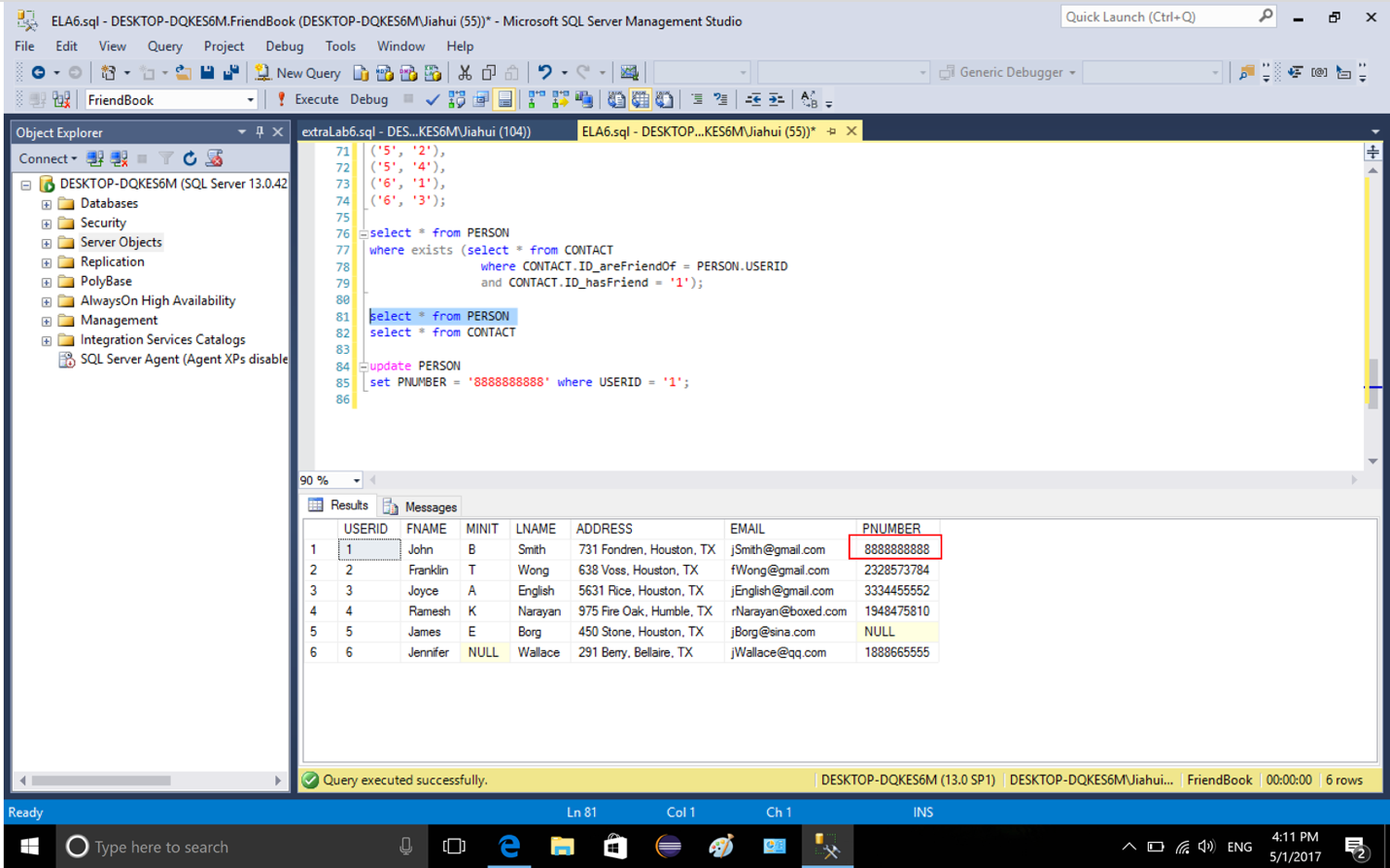




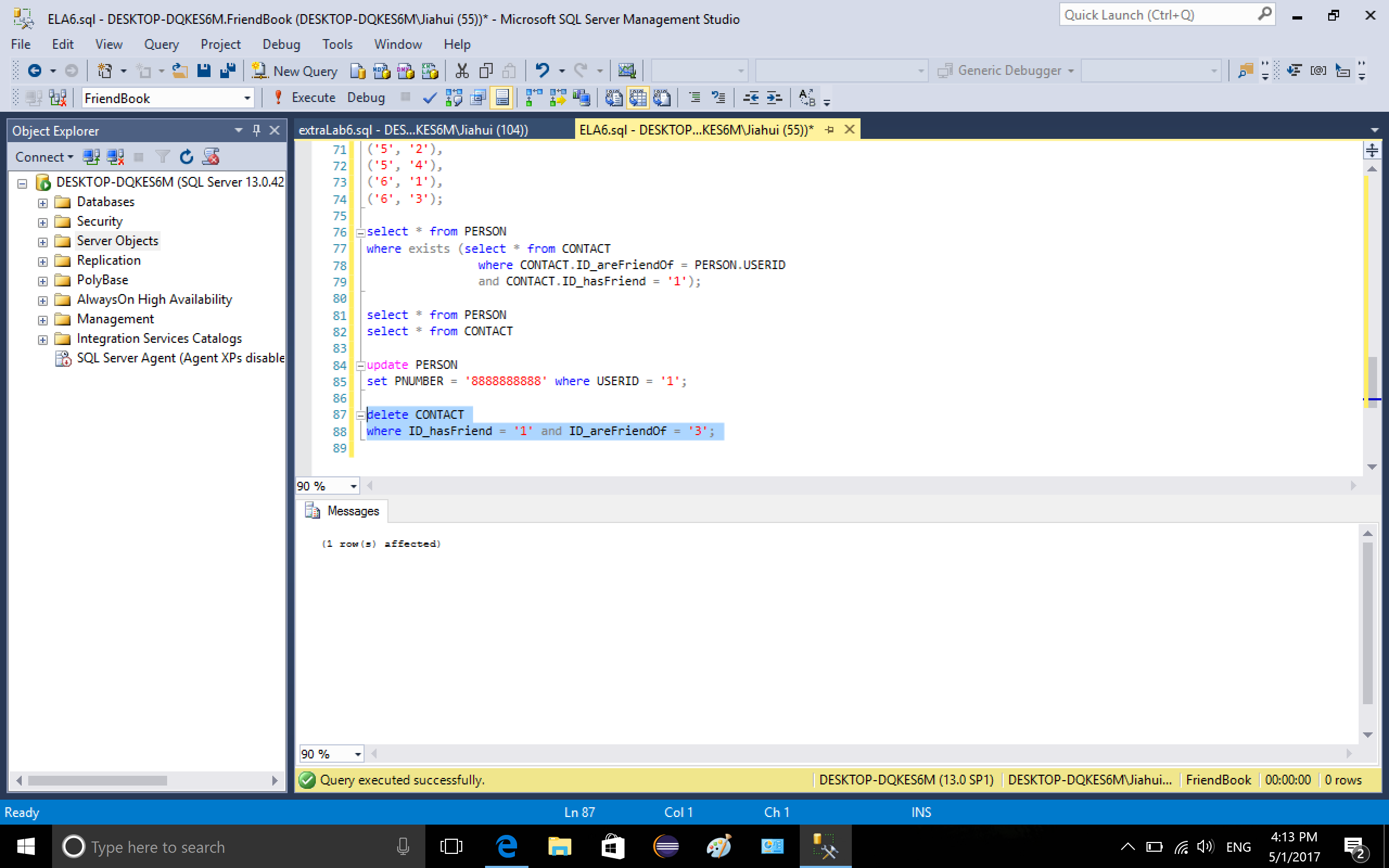
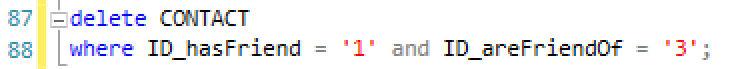
7. update a tuple in PERSON.



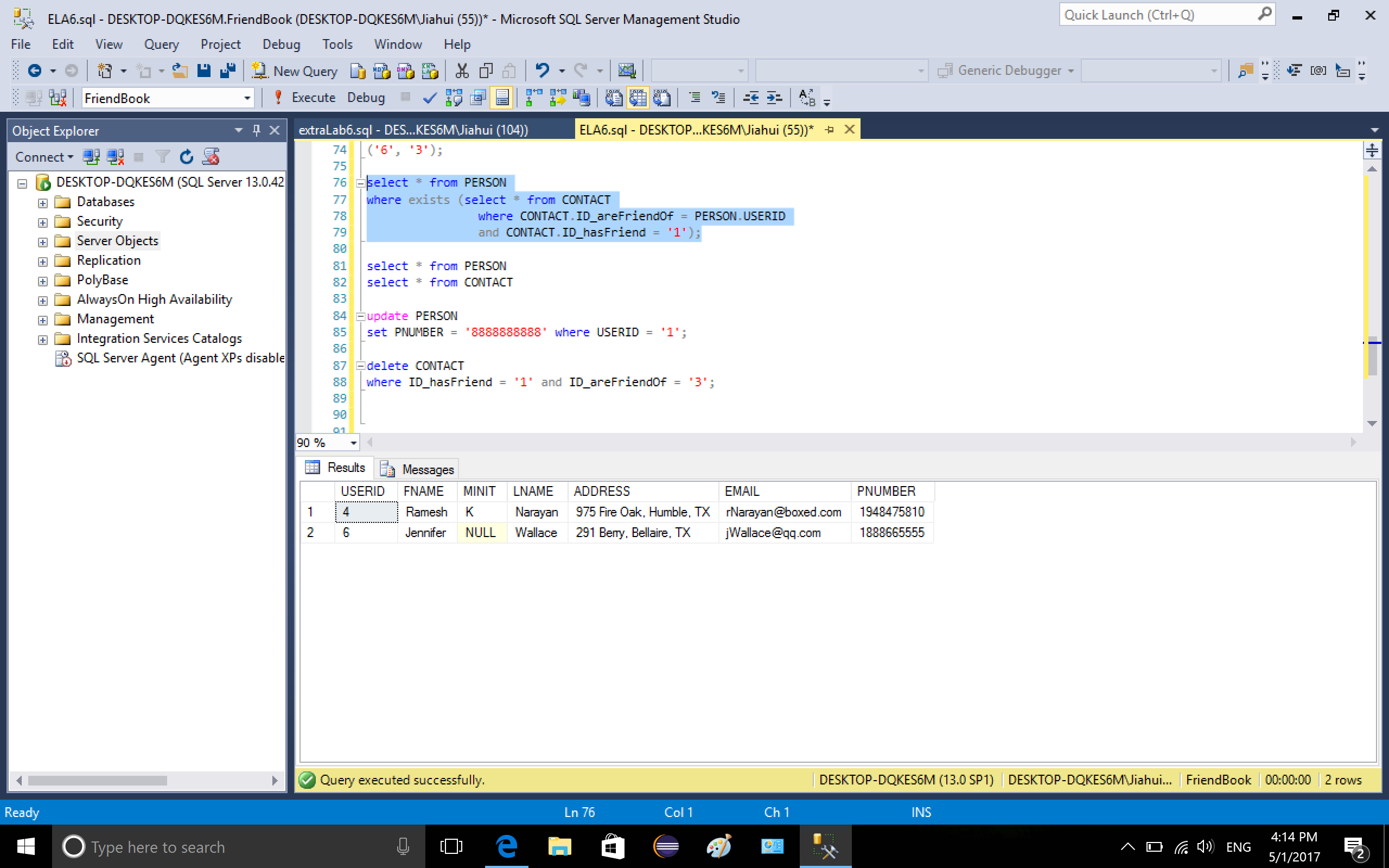
The phone number of USERID = 1 was updated



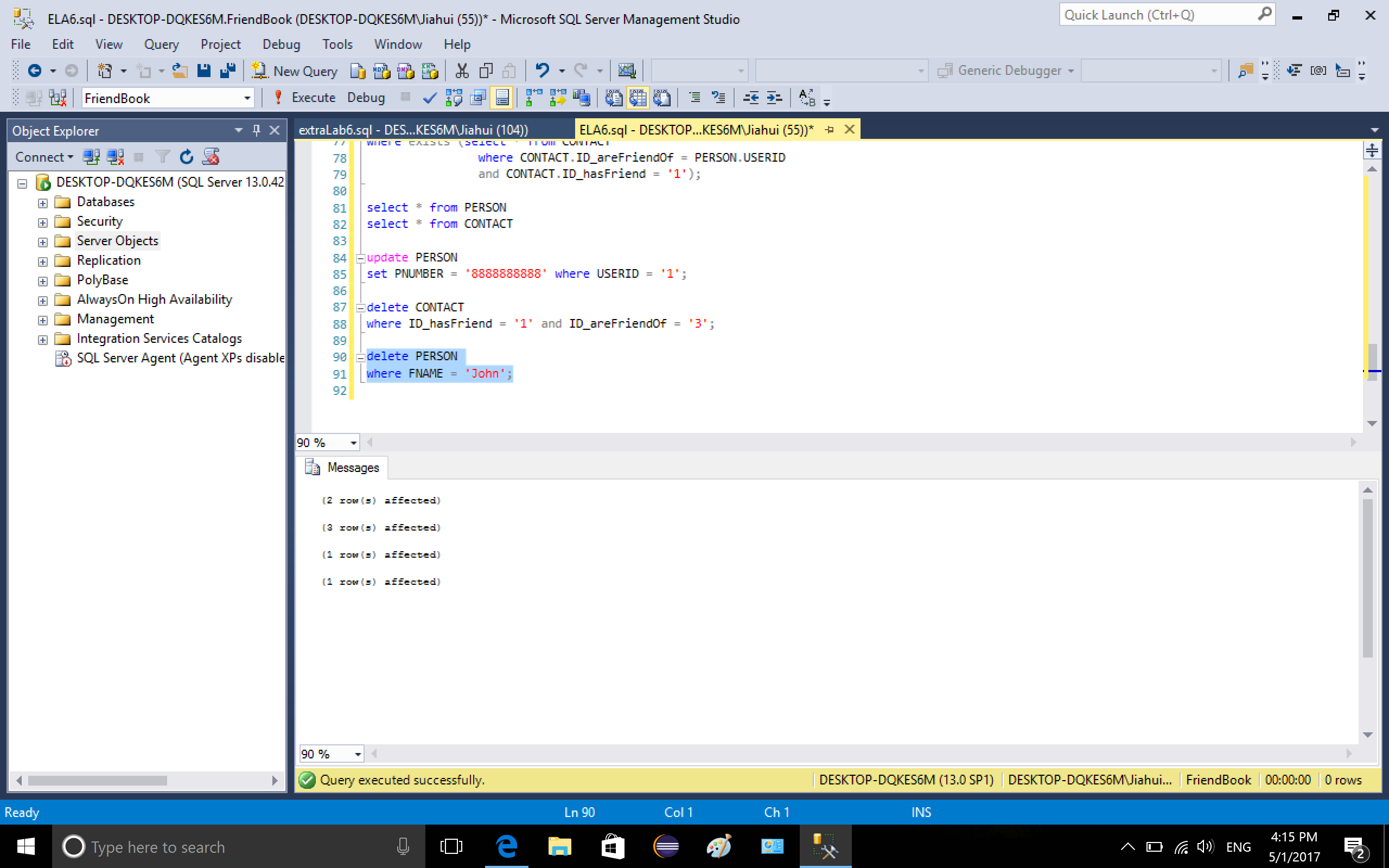
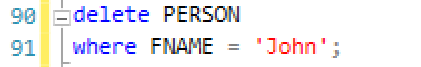
8. Delete a tuple from table CONTACT, specify the two values.



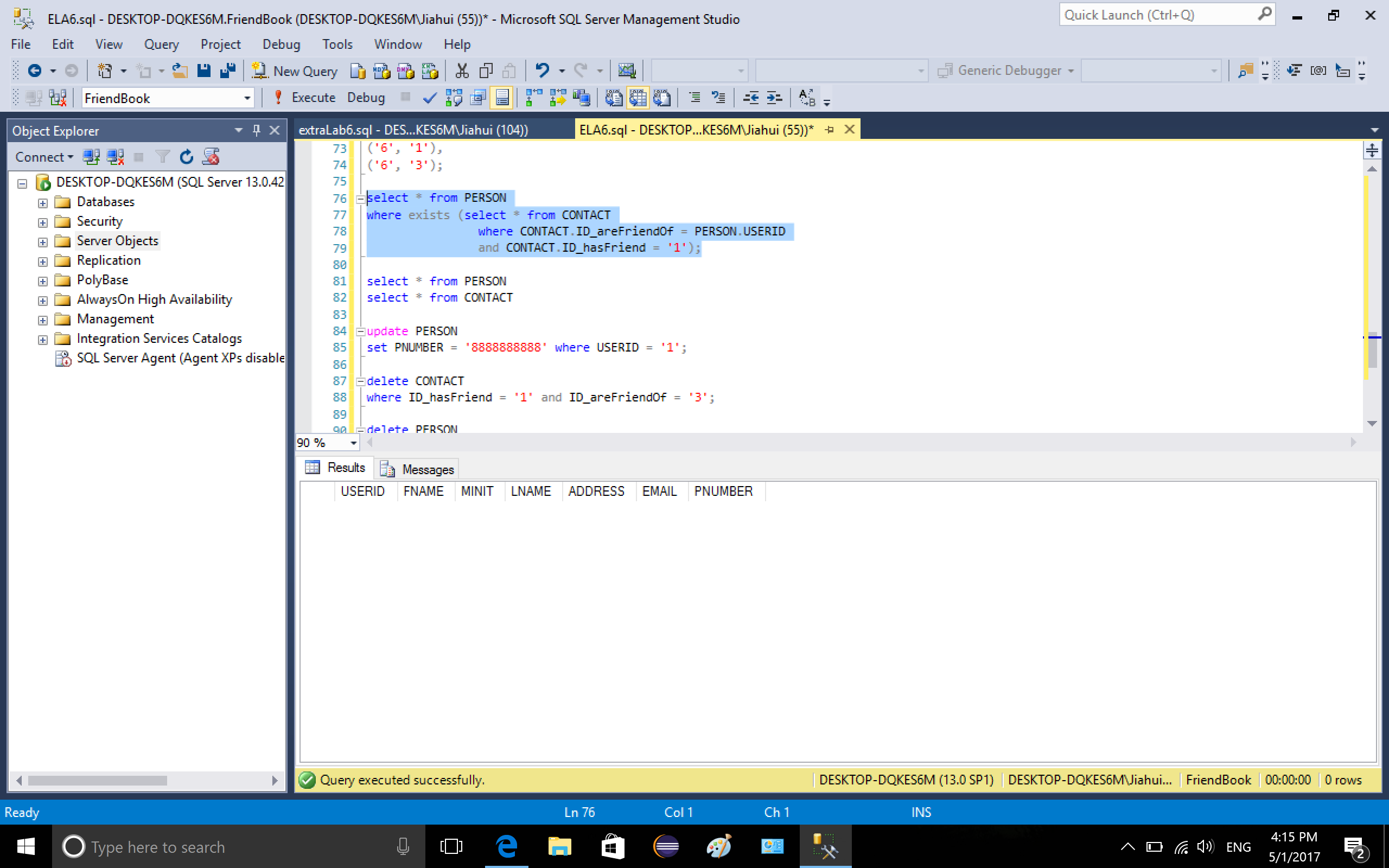
show the friend list of 1 (USERID) again, now he has friend 4 and 6



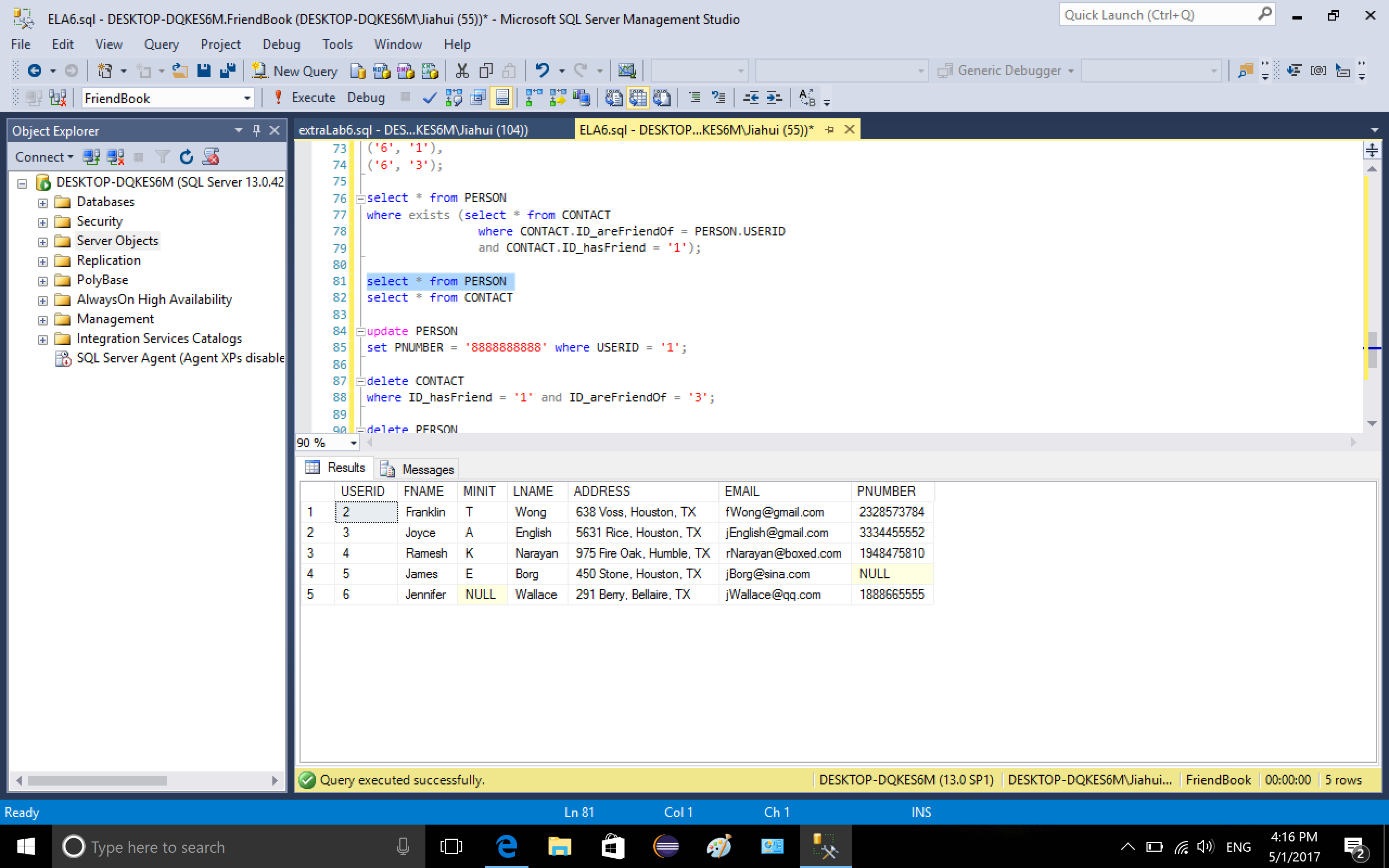
9. delete a person named John from table PERSON, whose ID is 1.



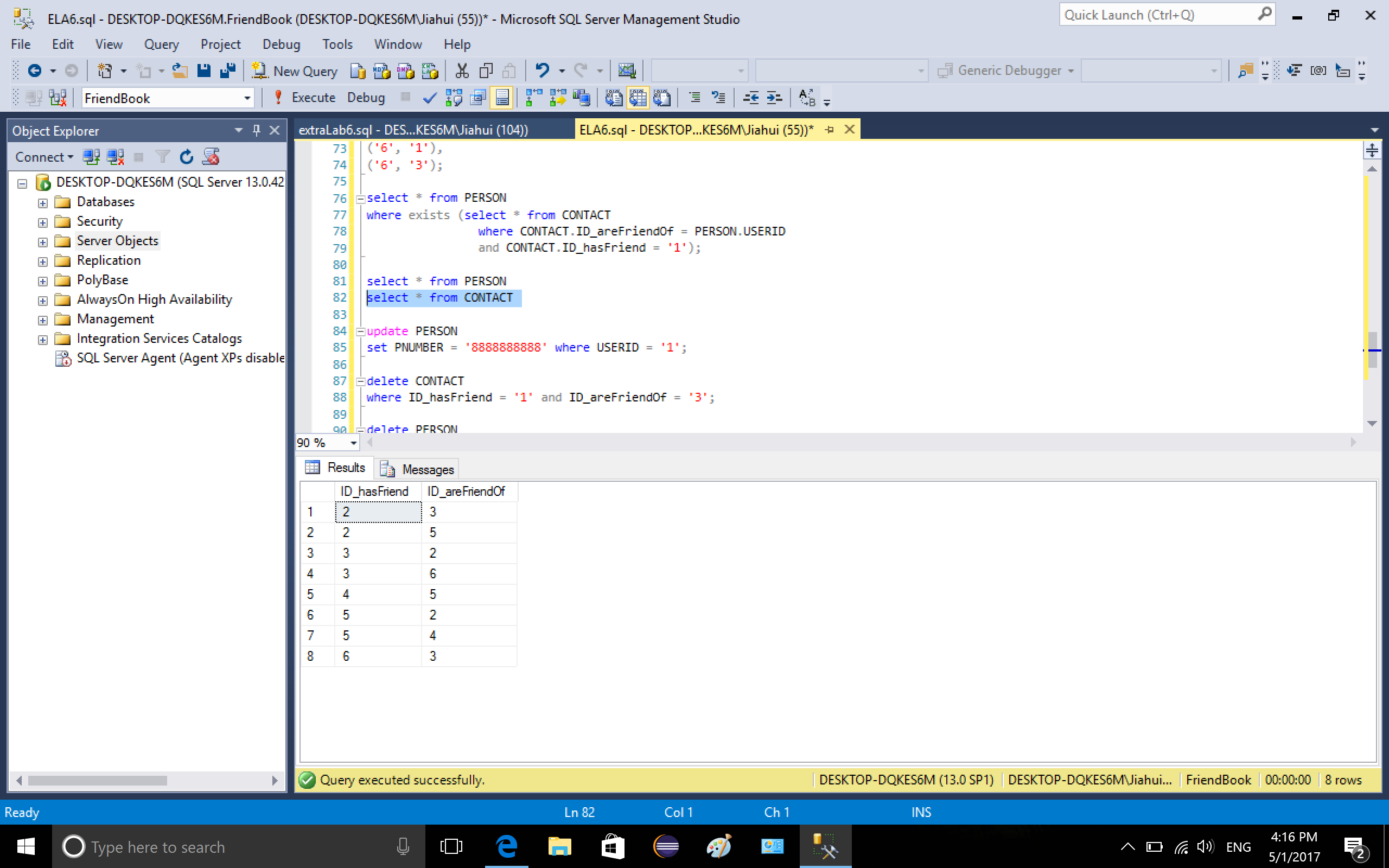
show person 1’s friend list again, it is empty because the person has been deleted.



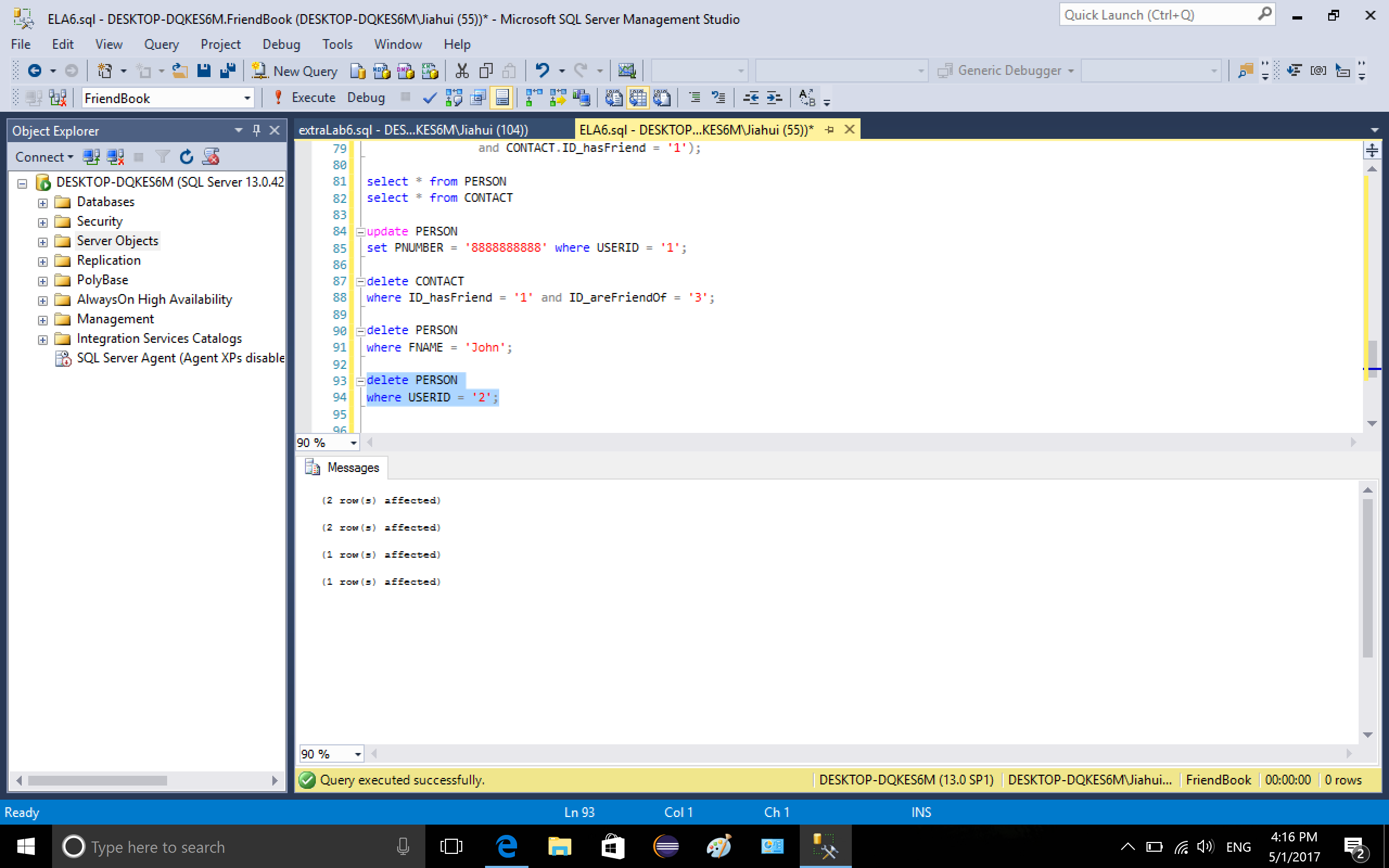
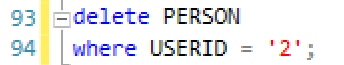
Show table PERSON, person 1 named John has been deleted



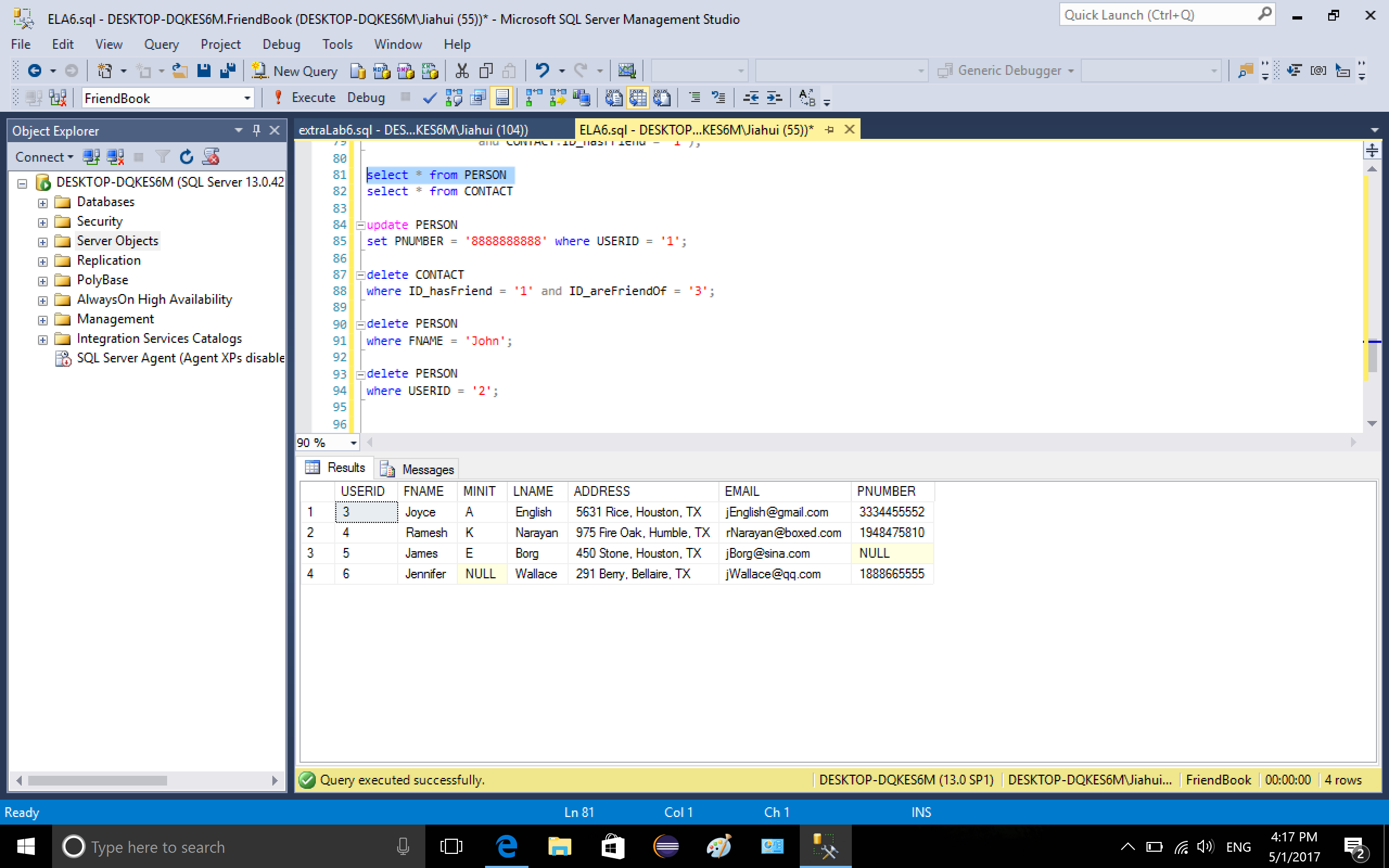
show table CONTACT, person 1 is not there, he has no friend, is not a friend of anyone, because it has been deleted from table PERSON.



10. Same as 9, delete person whose USERID is 2.



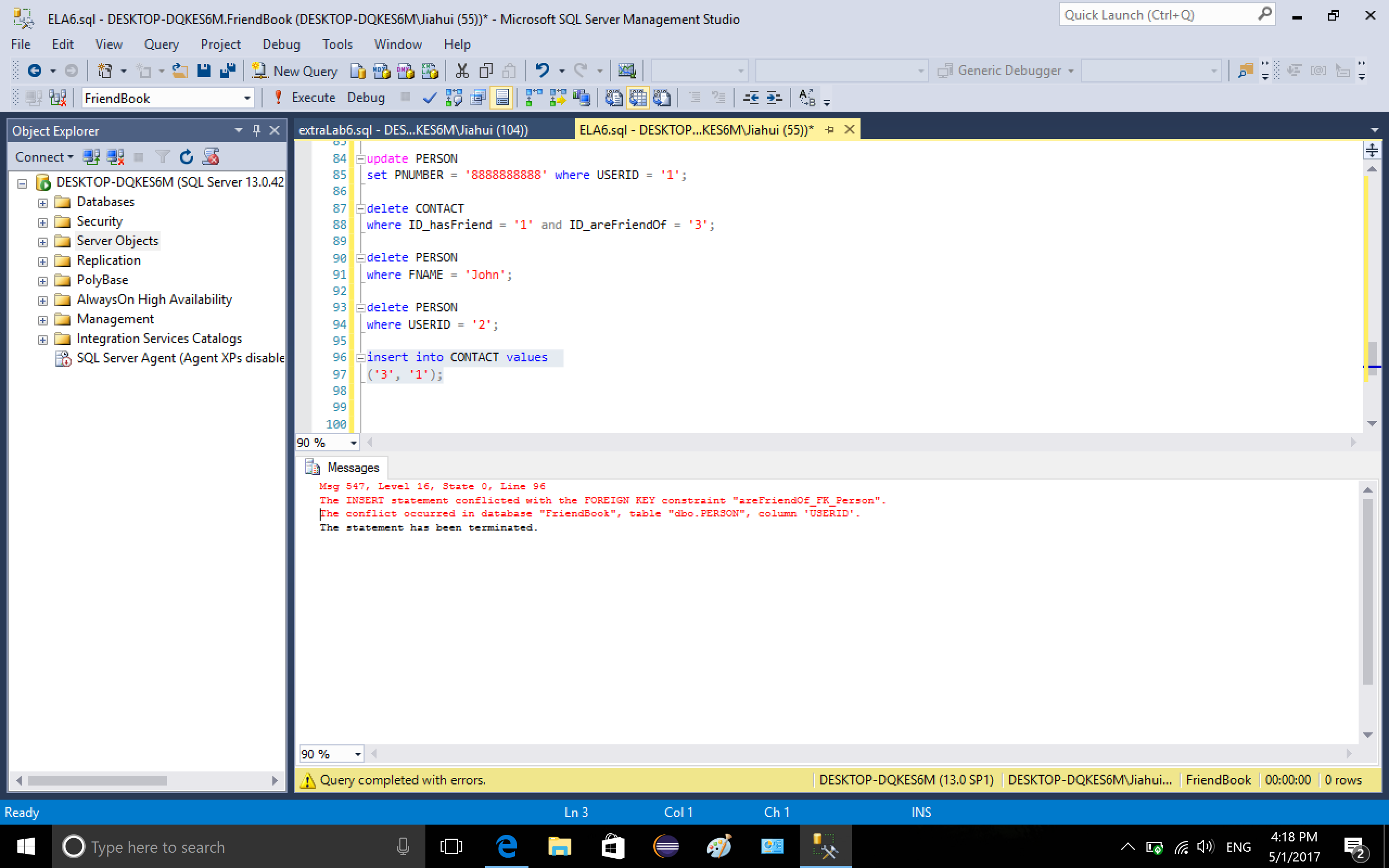
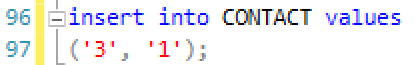
2 is deleted from table PERSON



2 is deleted from table CONTACT



11. insert value into table CONTACT, the value is not value because person 1 has been deleted. The insertion violated FK referential integrity.



12. insert values into CONTACT, the insertion is rejected because two values are the same, it violates the constraint defined before that the two attributes cannot be the same in CONTACT.

../ELA6/25-2.png