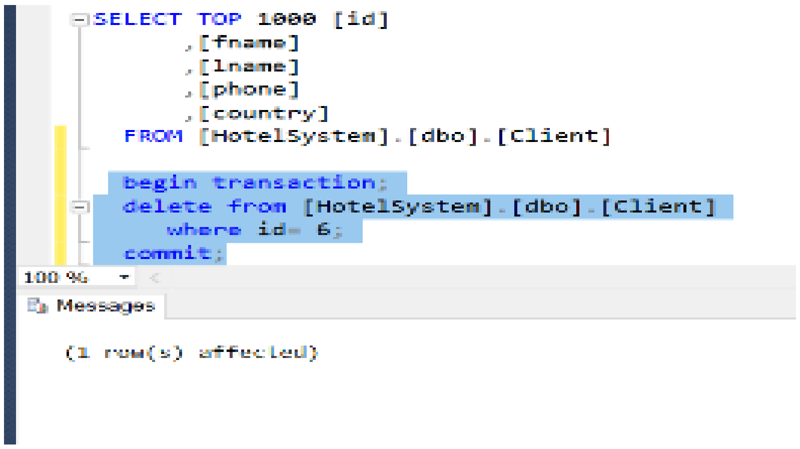
Assignment 11: Using T-SQL

Transactions (Transact-SQL):

A transaction is a single unit of work. If a transaction is successful, all of the data modifications made during the transaction are committed and become a permanent part of the database. If a transaction encounters errors and must be canceled or rolled back, then all of the data modifications are erased.

- **Auto-commit transactions**
  each individual statement is a transaction.
- **Explicit transactions**
  each transaction is explicitly started with the BEGIN TRANSACTION statement and explicitly ended with a COMMIT or ROLLBACK statement.
- **Implicit transactions**
  A new transaction is implicitly started when the prior transaction completes, but each transaction is explicitly completed with a COMMIT or ROLLBACK statement.

Using an explicit transaction:

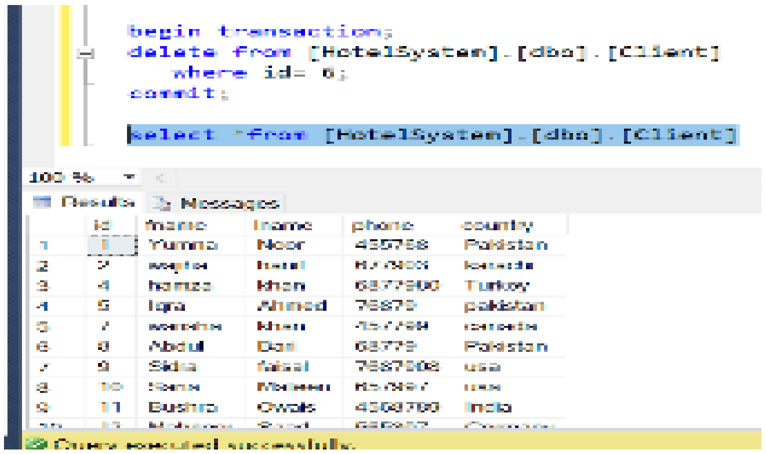


```
SELECT TOP 1000 [id]
      ,[fname]
      ,[lname]
      ,[phone]
      ,[country]
  FROM [HotelSystem].[dbo].[Client]

  begin transaction;
  delete from [HotelSystem].[dbo].[Client]
     where id= 6;
  commit;
```

100 %

Messages

```
(1 row(s) affected)
```



```
begin transaction;
delete from [HotelSystem].[dbo].[Client]
   where id= 6;
commit;

select *from [HotelSystem].[dbo].[Client]
```

100 %

Results | Messages

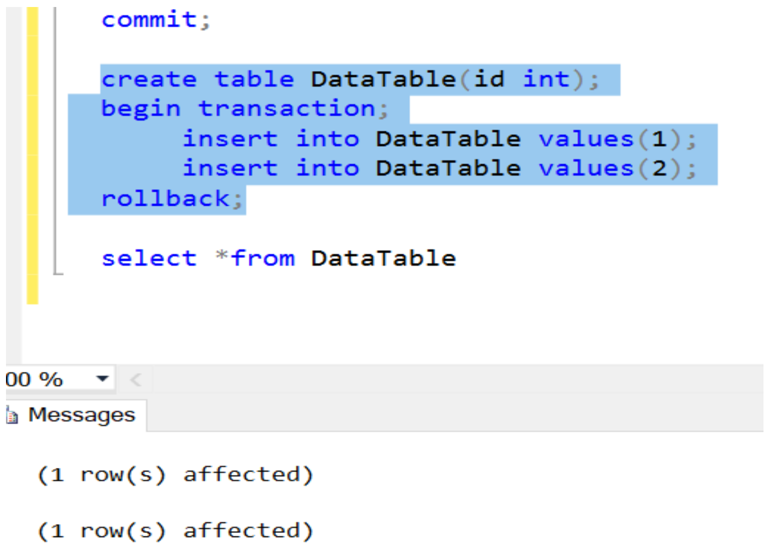
	id	fname	lname	phone	country
1	1	Yumna	Noor	435768	Pakistan
2	2	[illegible]	[illegible]	[illegible]	[illegible]
3	4	hamza	Khan	6877900	Turkey
4	5	Iqra	Ahmed	76879	pakistan
5	7	[illegible]	Khan	[illegible]	canada
6	8	Abdul	Dar	68779	Pakistan
7	9	Sidra	faisal	7687908	usa
8	10	[illegible]	[illegible]	[illegible]	usa
9	11	Bushra	Owais	4568790	India

Query executed successfully.

Rolling back a transaction:

ROLLBACK statement will roll back the INSERT statement, but the created table will exist.

Creating a table name [DataTable] then insert 2 values, then instantly rollback the table, but table exist.

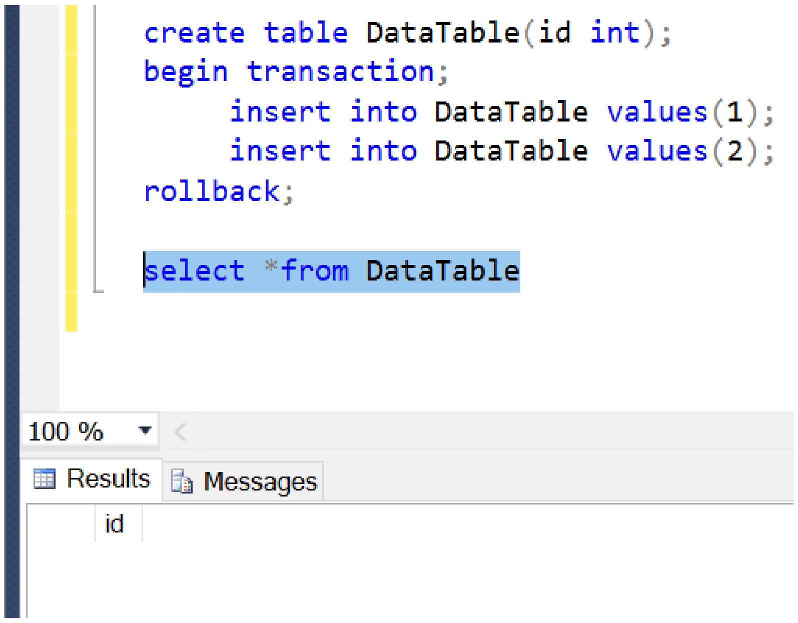


```
commit;

create table DataTable(id int);
begin transaction;
     insert into DataTable values(1);
     insert into DataTable values(2);
rollback;

select *from DataTable
```

00 %

Messages

```
(1 row(s) affected)

(1 row(s) affected)
```



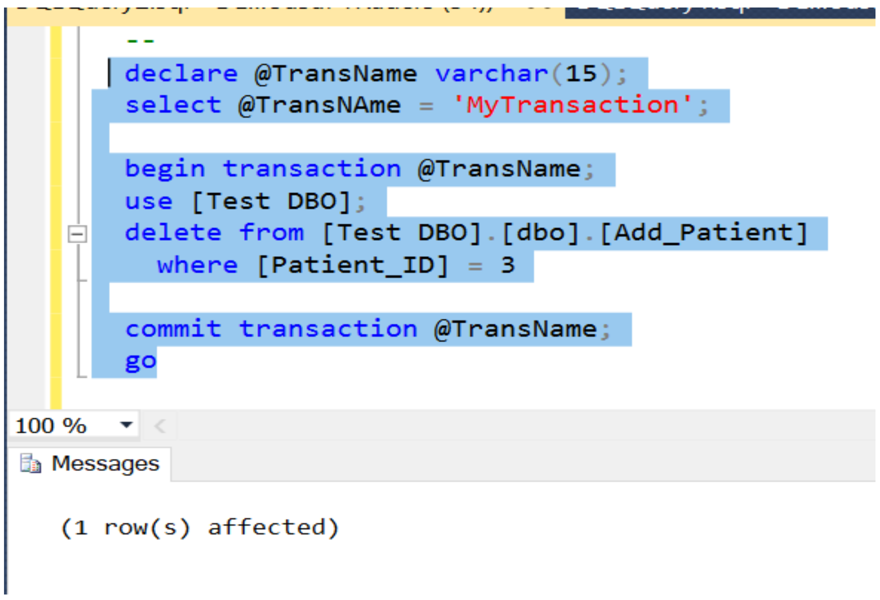
```
create table DataTable(id int);
begin transaction;
     insert into DataTable values(1);
     insert into DataTable values(2);
rollback;

select *from DataTable
```

100 %

Results | Messages

id

Naming a transaction:

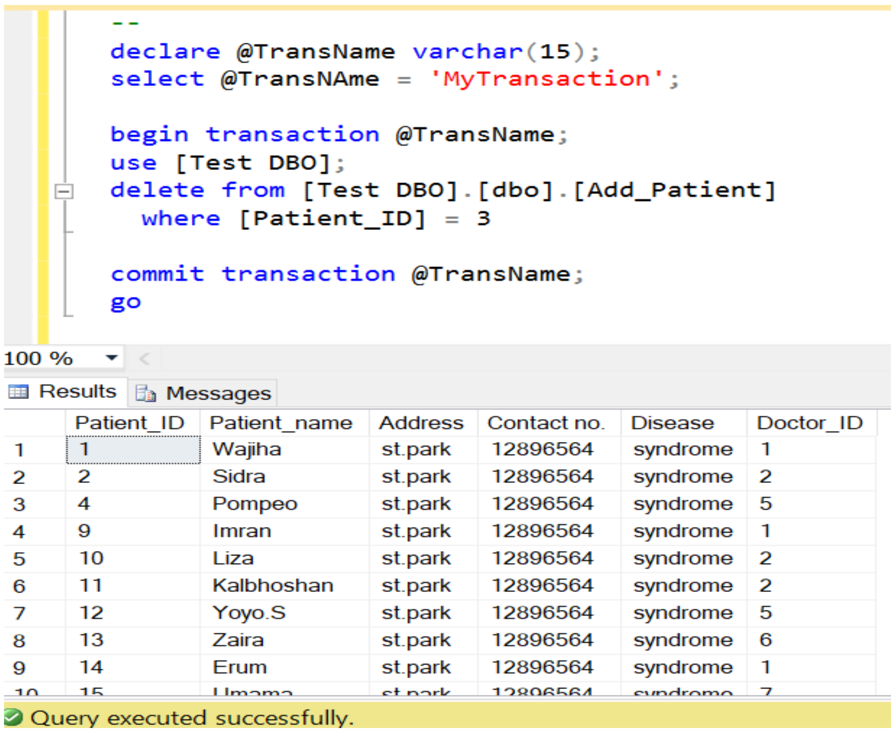


```
--
declare @TransName varchar(15);
select @TransNAme = 'MyTransaction';

begin transaction @TransName;
use [Test DBO];
delete from [Test DBO].[dbo].[Add_Patient]
  where [Patient_ID] = 3

commit transaction @TransName;
go
```

100 %

Messages

```
(1 row(s) affected)
```



```
--
declare @TransName varchar(15);
select @TransNAme = 'MyTransaction';

begin transaction @TransName;
use [Test DBO];
delete from [Test DBO].[dbo].[Add_Patient]
  where [Patient_ID] = 3

commit transaction @TransName;
go
```

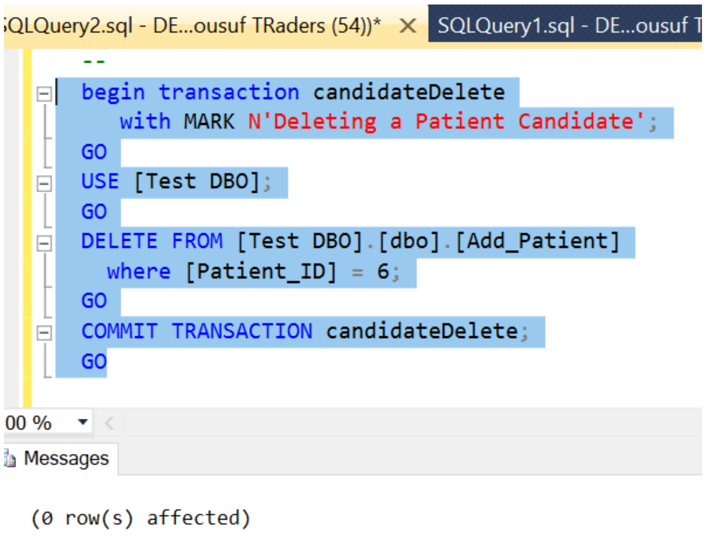
100 %

Results | Messages

	Patient_ID	Patient_name	Address	Contact no.	Disease	Doctor_ID
1	1	Wajiha	st.park	12896564	syndrome	1
2	2	Sidra	st.park	12896564	syndrome	2
3	4	Pompeo	st.park	12896564	syndrome	5
4	9	Imran	st.park	12896564	syndrome	1
5	10	Liza	st.park	12896564	syndrome	2
6	11	Kalbhoshan	st.park	12896564	syndrome	2
7	12	Yoyo.S	st.park	12896564	syndrome	5
8	13	Zaira	st.park	12896564	syndrome	6
9	14	Erum	st.park	12896564	syndrome	1
10	15	Umama	st.park	12896564	syndrome	7

Query executed successfully.

Marking a transaction:



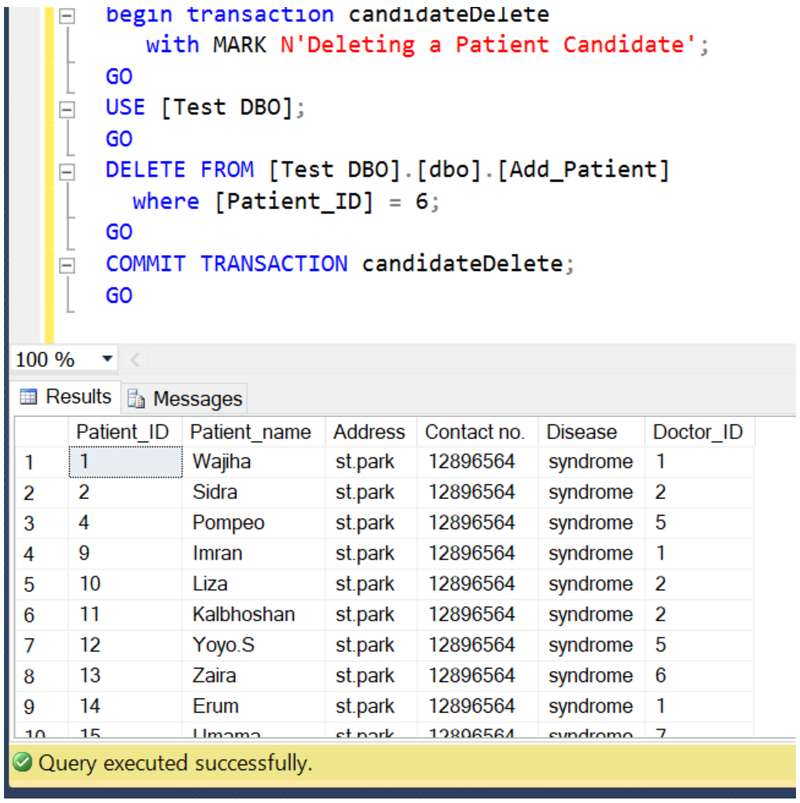
SQLQuery2.sql - DE...ousuf TRaders (54))* SQLQuery1.sql - DE...ousuf T

```
--
begin transaction candidateDelete
   with MARK N'Deleting a Patient Candidate';
GO
USE [Test DBO];
GO
DELETE FROM [Test DBO].[dbo].[Add_Patient]
  where [Patient_ID] = 6;
GO
COMMIT TRANSACTION candidateDelete;
GO
```

00 %

Messages

```
(0 row(s) affected)
```



```
begin transaction candidateDelete
   with MARK N'Deleting a Patient Candidate';
GO
USE [Test DBO];
GO
DELETE FROM [Test DBO].[dbo].[Add_Patient]
  where [Patient_ID] = 6;
GO
COMMIT TRANSACTION candidateDelete;
GO
```

100 %

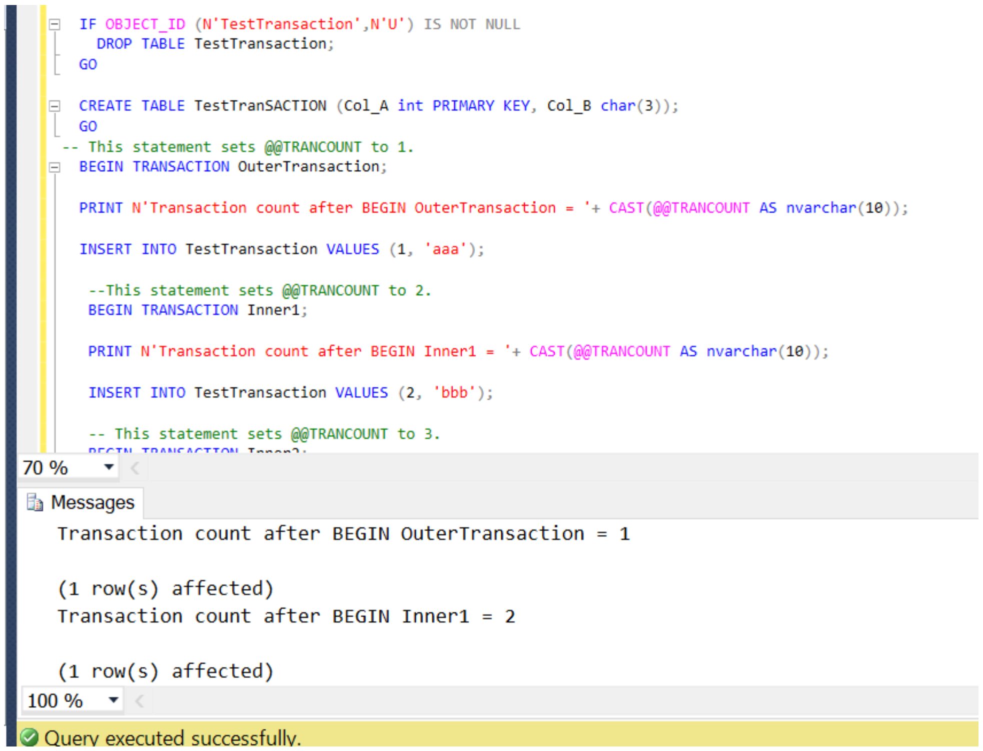
Results Messages

	Patient_ID	Patient_name	Address	Contact no.	Disease	Doctor_ID
1	1	Wajiha	st.park	12896564	syndrome	1
2	2	Sidra	st.park	12896564	syndrome	2
3	4	Pompeo	st.park	12896564	syndrome	5
4	9	Imran	st.park	12896564	syndrome	1
5	10	Liza	st.park	12896564	syndrome	2
6	11	Kalbhoshan	st.park	12896564	syndrome	2
7	12	Yoyo.S	st.park	12896564	syndrome	5
8	13	Zaira	st.park	12896564	syndrome	6
9	14	Erum	st.park	12896564	syndrome	1
10	15	Umama	st.park	12896564	syndrome	7

Query executed successfully.

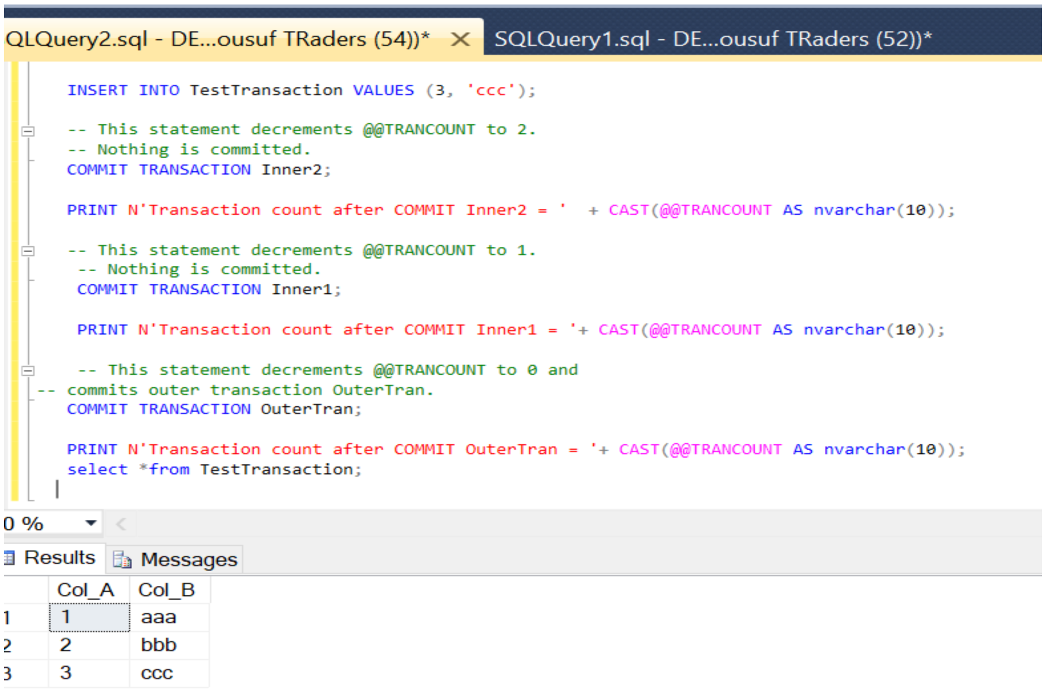
Committing a nested transaction:

The following example create a table, generates three levels of nested transactions, and then commits the nested transaction. Although each COMMIT TRANSACTION statement has a **transaction_name parameter,** there's no relationship between the COMMIT TRANSACTION and BEGIN TRANSACTION statements.



```
IF OBJECT_ID (N'TestTransaction',N'U') IS NOT NULL
  DROP TABLE TestTransaction;
GO

CREATE TABLE TestTranSACTION (Col_A int PRIMARY KEY, Col_B char(3));
GO
-- This statement sets @@TRANCOUNT to 1.
BEGIN TRANSACTION OuterTransaction;

PRINT N'Transaction count after BEGIN OuterTransaction = '+ CAST(@@TRANCOUNT AS nvarchar(10));

INSERT INTO TestTransaction VALUES (1, 'aaa');

 --This statement sets @@TRANCOUNT to 2.
 BEGIN TRANSACTION Inner1;

 PRINT N'Transaction count after BEGIN Inner1 = '+ CAST(@@TRANCOUNT AS nvarchar(10));

 INSERT INTO TestTransaction VALUES (2, 'bbb');

 -- This statement sets @@TRANCOUNT to 3.
```

```
Transaction count after BEGIN OuterTransaction = 1

(1 row(s) affected)
Transaction count after BEGIN Inner1 = 2

(1 row(s) affected)
```



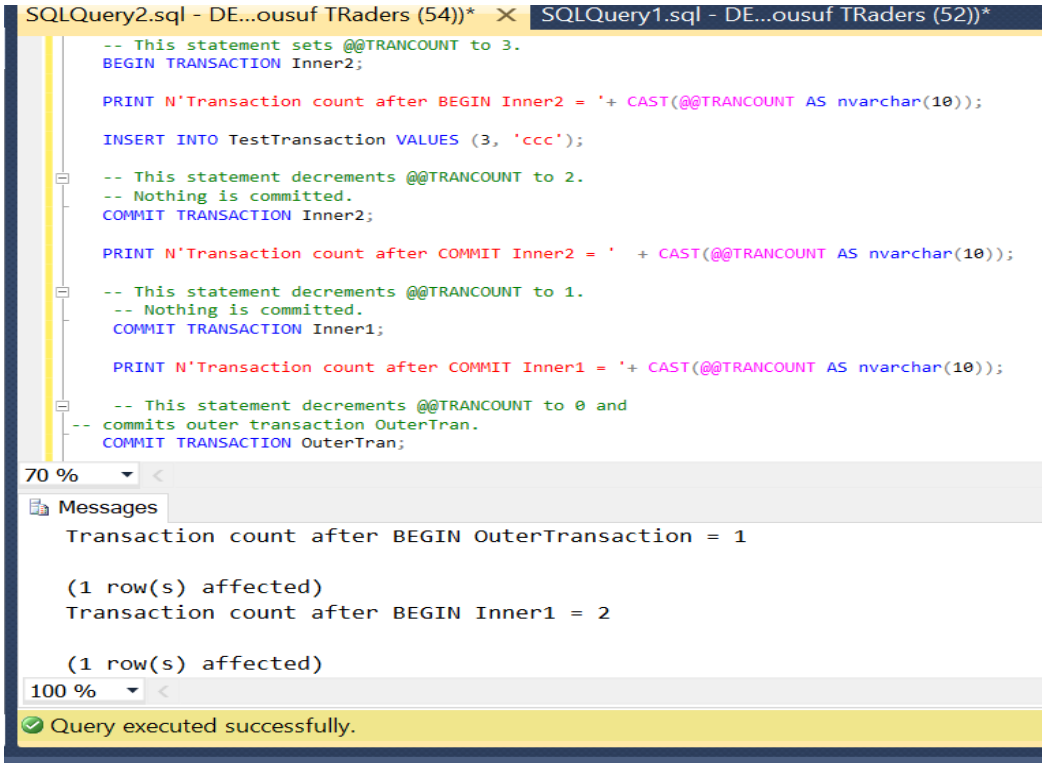
QLQuery2.sql - DE...ousuf TRaders (54))*   SQLQuery1.sql - DE...ousuf TRaders (52))*

```sql
INSERT INTO TestTransaction VALUES (3, 'ccc');

-- This statement decrements @@TRANCOUNT to 2.
-- Nothing is committed.
COMMIT TRANSACTION Inner2;

PRINT N'Transaction count after COMMIT Inner2 = '  + CAST(@@TRANCOUNT AS nvarchar(10));

-- This statement decrements @@TRANCOUNT to 1.
 -- Nothing is committed.
 COMMIT TRANSACTION Inner1;

 PRINT N'Transaction count after COMMIT Inner1 = '+ CAST(@@TRANCOUNT AS nvarchar(10));

 -- This statement decrements @@TRANCOUNT to 0 and
-- commits outer transaction OuterTran.
COMMIT TRANSACTION OuterTran;

PRINT N'Transaction count after COMMIT OuterTran = '+ CAST(@@TRANCOUNT AS nvarchar(10));
select *from TestTransaction;
```

0 %

Results   Messages

	Col_A	Col_B
1	1	aaa
2	2	bbb
3	3	ccc



SQLQuery2.sql - DE...ousuf TRaders (54))*   SQLQuery1.sql - DE...ousuf TRaders (52))*

```sql
-- This statement sets @@TRANCOUNT to 3.
BEGIN TRANSACTION Inner2;

PRINT N'Transaction count after BEGIN Inner2 = '+ CAST(@@TRANCOUNT AS nvarchar(10));

INSERT INTO TestTransaction VALUES (3, 'ccc');

-- This statement decrements @@TRANCOUNT to 2.
-- Nothing is committed.
COMMIT TRANSACTION Inner2;

PRINT N'Transaction count after COMMIT Inner2 = '  + CAST(@@TRANCOUNT AS nvarchar(10));

-- This statement decrements @@TRANCOUNT to 1.
 -- Nothing is committed.
 COMMIT TRANSACTION Inner1;

 PRINT N'Transaction count after COMMIT Inner1 = '+ CAST(@@TRANCOUNT AS nvarchar(10));

 -- This statement decrements @@TRANCOUNT to 0 and
-- commits outer transaction OuterTran.
COMMIT TRANSACTION OuterTran;
```

70 %

Messages

```
Transaction count after BEGIN OuterTransaction = 1

(1 row(s) affected)
Transaction count after BEGIN Inner1 = 2

(1 row(s) affected)
```

100 %

Query executed successfully.