Project - 3

Report
On
Multiuser Database Processing

Instructor: Dr. Jianchao Han

Course: CSC-553

By Team Three:

Nolan Thompson (nolanruss@gmail.com),

Janani Janardhanan (janam.nov@gmail.com),

Vaibhav Mishra (vmishra1@toromail.csudh.edu),

Jhih-tian Jhong (jjhong1@toromail.csudh.edu)

Ishu Juneja (ishu.juneja75@gmail.com),

CUSTOMER (CustomerID, Phone, Email, FirstName, LastName)

INVOICE (InvoiceNumber, CustomerID, DateIn, DateOut, Subtotal, Tax, Total)

INVOICE_ITEM (InvoiceNumber, ItemNumber, ServiceID, Quantity, UnitPrice, ExtendedPrice)

SERVICE (ServiceID, ServiceDescription, UnitPrice)

1. Managing Multiuser Databases

1-A.

***Memo**:*

After analyzing your current business needs, and the complexity of the system to be implemented, we highly recommend appointing two or more people for the task of administrating your database.

Below are some of the key responsibilities your chosen database administrator will be required to perform:

- *Manage database structure*
- *Control concurrent processing*
- *Manage processing rights and responsibilities*
- *Develop database security*
- *Provide for database recovery*
- *Manage the DBMS*
- *Maintain the data repository*

Of the current members of your business, we recommend that both owners and our company serve as the administrators for the database system. Due to the limited size of the current system, a full-time employee is not necessary to accommodate the system needs. However, it is essential that the person you select is available to repair the system when problems occur. It is also important that your administrator is a trusted and long-term member of your company. We recommend at that we train at least two people in administering the database for redundancy purposes. If training an owner on the administration of the database is unreasonable for your requirements, we recommend training your shift supervisor as the primary administrator, with our company as support.

Considering the complexity and training required for performing changes to your database system, we recommend you allow us to perform changes for concurrent processing, changes to database structure, developing security, and implementing database recovery protocols. In

addition, we will also establish the initial data repository which will be used to inform how further improvements to your system can be made.

Below, we have provided a breakdown of how each responsibility required for your database administrator applies to your business to illustrate the importance of this position.

Managing database structure:

As your business continues to grow, your needs for the database will also change. You may wish to add features to your system that allow you to track additional customer or invoice data, or to track discounts and promotions. In other cases, you may wish to alter how orders are tracked and related to invoices. These changes will require the database administrator to modify the existing database structure to accommodate the new features. Due to the general complexity of databases, it is important to have an administrator who is able to diagnose any problems that may arise with these changes to prevent a system failure.

Controlling concurrent processing:

These measures are implemented to ensure that when one user adds or modifies information in the database, it does not conflict with the actions of another user at the same time. This is done to prevent incorrect data from being stored in your system. More the number of people who need to access your database, more will be the number of transactions within the database occurring concurrently. As your needs change, the controls put in place to ensure your database is both easily accessed by those using it and that the data stored within the database is accurate will need to be updated. Adding more users and changing the information being documented will require implementation of concurrent process controls, which will require a knowledgeable individual to implement the changes.

Managing processing rights and responsibilities:

Since the responsibilities of those accessing your database will vary, for security purposes it is best that you limit the rights and responsibilities of users to their job roles. Changing order totals, customer information, and services improperly can do significant harm to your business, and for this reason it is important that you have a database administrator able to correctly implement these restrictions.

Developing database security:

As new employees arrive and others leave, it is essential that a database administrator is available to add and remove accounts and permissions to protect the integrity of your data and the information of your customers. In addition, your database must be secured from intrusions that can destroy your database or result in theft of important customer and employee data. It is essential that your administrator be knowledgeable in working with firewalls, managing logins, how to best protect the physical machine storing your data, and implementing security plans for detection and prevention of security emergencies.

Providing for database recovery:

System failures can occur for numerous reasons. Because your system is shared by multiple users, and will be relied upon for critical business functions, it is important that your system can be quickly recovered in the event of a failure. Your database administrator will need to carefully consider the type of protocol to implement for system recovery in order to protect the integrity of customer invoices and your system as a whole.

Managing the DBMS:

All computer systems encounter problems. If your system begins performing slowly, or errors occur, you need an administrator to analyze problems in order to determine whether your system needs changes to improve performance.

Maintaining the data repository:

In order to improve the efficiency and reduce the cost of changes to your database system, it is also important that your database administrator constructs and maintains an accurate repository. The data repository stores important data about your database, its tables, relationships, the applications you use to access the data, and other information. An accurate repository is essential to make informed decisions on how your system can be improved in the future, and it is important you have a database administrator to ensure it collects correct and up to date data.

We look forward to working with you to implement and train your team on using the new system, we believe it will greatly help you to save time and money, and improve customer satisfaction as you continue to grow your business.

1-B.

User Group	Member(s)
Service personnel	Salesclerks, seamstress
Management	Shift manager, owners
System Administrator	Owners, Vendor

Figure 1: User group names and members of the Marcia's Dry Cleaning database.

User Group	CUSTOMER	INVOICE	INVOICE_ITEM	SERVICE
Sales/service personnel	Insert, change, query	Insert, query	Query	Query
Management	Insert, change, query	Insert, change, query	Insert, change, query	Insert, change, query
System Administrator	Grant rights, modify structure	Grant rights, modify structure	Grant rights, modify structure	Grant rights, modify structure

Figure 2: Security scheme showing user group permissions for each database table.

1-C.

a.

Dirty Read:
Suppose one customer orders one product online and processes to check out. The transaction is processed to charge money and reduce inventory number, but the transaction fails (roll back).
Finally, the web-site shows the quantity is not the same as the database's inventory. This situation will happen in the IVOICE_ITEM table.

Non-repeatable Read:
Suppose two customers do different transactions at the same time. Customer A adds the item to the shipping car and checks it out.
Customer B searches the same item and is shown the product's price and quantity.
After customer A finishes the order and payment, the quantity is reduced one stack in the database.
Customer B search information is not correct. Because customer B search transaction is between customer A transaction and case non-repeatable read. This situation happens in the IVOICE_ITEM table.

Phantom Read:
Suppose the owner wants to know the total number of customers in the database. When the owner searches sum in the customer table, at the same time, a new customer is added to the customer table.
The owner getting the total number of customers is not correct. The owner searches again to find the total number of customers. The number is different in both the searches. This situation happens in CUSTOMER table.

Level	Dirty Read	Unrepeatable Read	Phantom Read
Read Uncommitted	Yes	Yes	Yes
Read Committed	No	Yes	Yes
Repeatable Read	No	No	Yes
Serializable	No	No	No

Figure 3: Isolation levels and Read Permissions

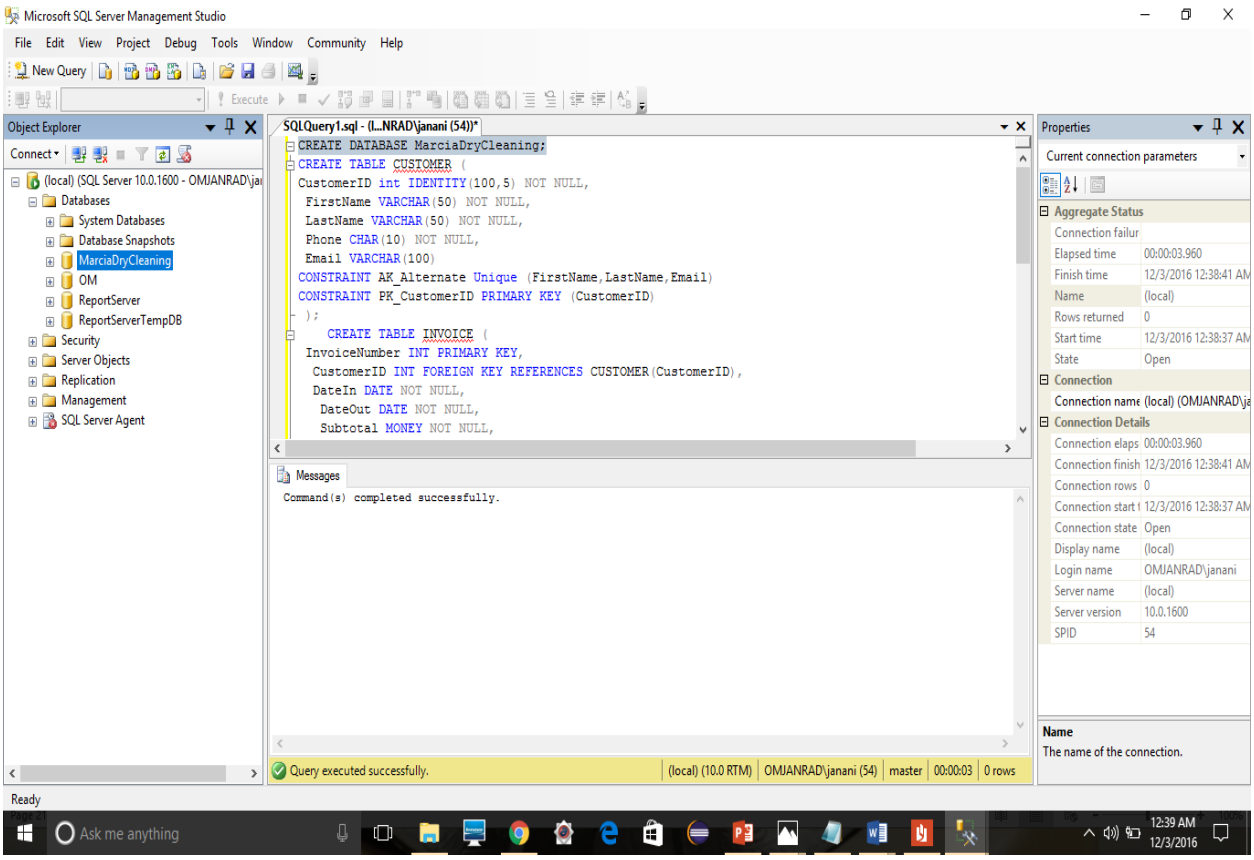
b. Procedure 1 is not performing any read operations, only insertions on the SERVICE table (Read committed Isolation Level).

c. Procedure 2 need to read CUSTOMER, INVOICE, INVOICE_ITEM table, also it needs insertion operation between these tables (Serializable Isolation Level). Procedure 3 is performing read and insertions on the CUSTOMER table (Serializable Isolation Level), because database don't want to get duplicate value.

2. Managing Databases

2-A.

CREATE DATABASE MarciaDryCleaning;



2-B.

```
CREATE TABLE CUSTOMER (
CustomerID int IDENTITY(100, 5) NOT NULL,
 FirstName VARCHAR(50) NOT NULL,
 LastName VARCHAR(50) NOT NULL,
 Phone CHAR(10) NOT NULL,
 Email VARCHAR(100),
CONSTRAINT AK_Alternate Unique (FirstName, LastName, Email),
CONSTRAINT PK_CustomerID PRIMARY KEY (CustomerID)
 );
CREATE TABLE INVOICE (
InvoiceNumber INT PRIMARY KEY,
 CustomerID INT FOREIGN KEY REFERENCES CUSTOMER(CustomerID),
 DateIn DATE NOT NULL,
 DateOut DATE NOT NULL,
 Subtotal MONEY NOT NULL,
 Tax MONEY NOT NULL,
 TotalAmount MONEY NOT NULL,
 CONSTRAINT fk_CustomerID
  FOREIGN KEY (CustomerID)
        REFERENCES CUSTOMER(CustomerID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

```
CREATE TABLE SERVICE(
        ServiceID INT NOT NULL PRIMARY KEY,
        ServiceDescription varchar(100) NOT NULL,
        UnitPrice MONEY NOT NULL,
        CONSTRAINT AK_Service UNIQUE (ServiceDescription)
);

ALTER TABLE SERVICE
        ADD CHECK (UnitPrice> 1.50 AND UnitPrice<10.00);

CREATE TABLE INVOICE_ITEM (
        InvoiceNumber INT NOT NULL,
        ItemNumber INT NOT NULL,
        ServiceID INT NOT NULL,
        Quantity INT DEFAULT '1',
        UnitPrice MONEY NOT NULL,
        ExtendedPrice MONEY NOT NULL,
        CONSTRAINT pk_id PRIMARY KEY(InvoiceNumber,ItemNumber),
        CONSTRAINT AK_InvoiceItem Unique (InvoiceNumber,ServiceID),
        CONSTRAINT fk_id FOREIGN KEY(InvoiceNumber)
                REFERENCES INVOICE(InvoiceNumber)
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        CONSTRAINT fk_id1 FOREIGN KEY(ServiceID)
```

```
                REFERENCES SERVICE(ServiceID)

                        ON DELETE CASCADE
                        ON UPDATE CASCADE

  );
```

Insertion of Records into the Tables:

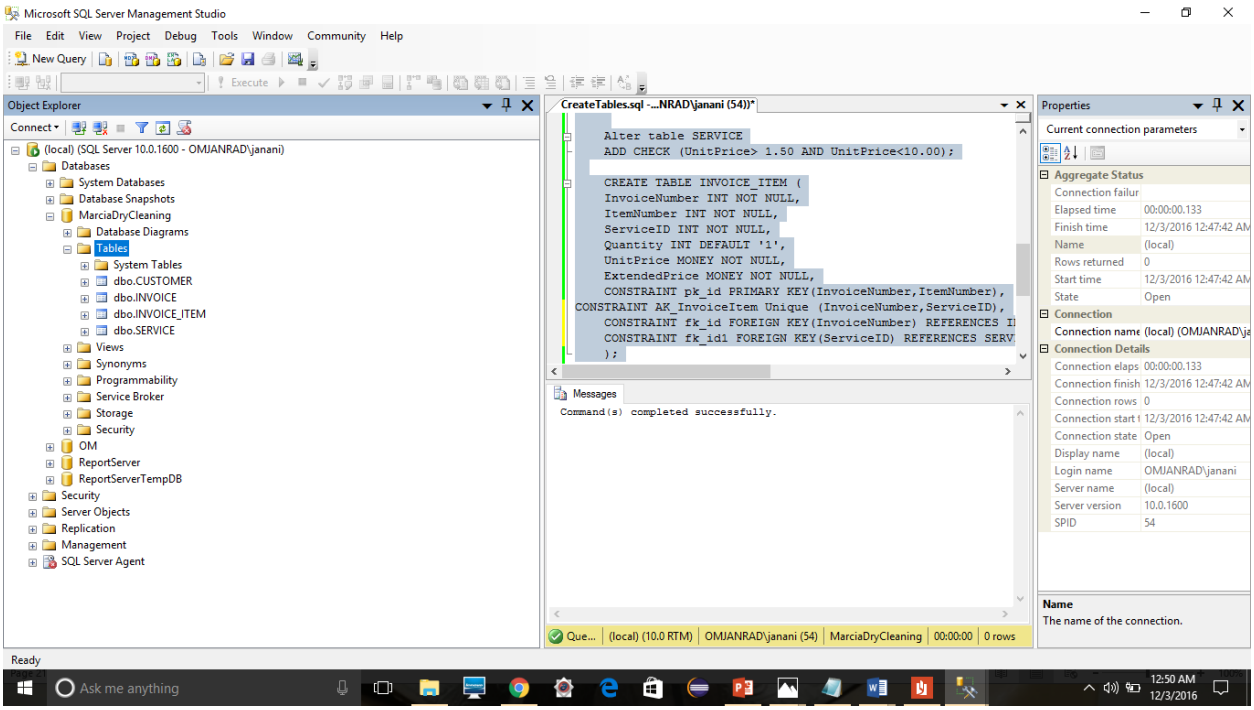
```
INSERT INTO CUSTOMER VALUES('Tom','Nelson','4522214321','tom@abc.com');

INSERT INTO CUSTOMER VALUES('Venica','Aldrin','3102426732','valdrin@abc.com');

INSERT INTO CUSTOMER VALUES('Ranjeet','Singh','5620916543','ransingh@efg.com');

INSERT INTO SERVICE VALUES ('1100','MENS SHIRT','9.99');

INSERT INTO SERVICE VALUES ('1101','WOMENS SHIRT','9.99');

INSERT INTO SERVICE VALUES ('1102','MENS PANTS','7.99');

INSERT INTO SERVICE VALUES ('1103','WOMENS PANTS','7.99');

INSERT INTO SERVICE VALUES ('1104','WOMENS DRESS','5.99');


INSERT INTO INVOICE_ITEM(InvoiceNumber, ItemNumber, ServiceID, Quantity)

        VALUES('1011','1','1101','3');

INSERT INTO INVOICE_ITEM(InvoiceNumber, ItemNumber, ServiceID, Quantity)

        VALUES('1011','2','1103','2');

INSERT INTO INVOICE_ITEM(InvoiceNumber, ItemNumber, ServiceID, Quantity)

        VALUES('1012','1','1102','2');

INSERT INTO INVOICE_ITEM(InvoiceNumber, ItemNumber, ServiceID, Quantity)

        VALUES('1013','1','1101','2');

INSERT INTO INVOICE_ITEM(InvoiceNumber, ItemNumber, ServiceID, Quantity)

        VALUES('1013','2','1100','4');

INSERT INTO INVOICE_ITEM(InvoiceNumber, ItemNumber, ServiceID, Quantity)
```

VALUES('1013','3','1104','2');

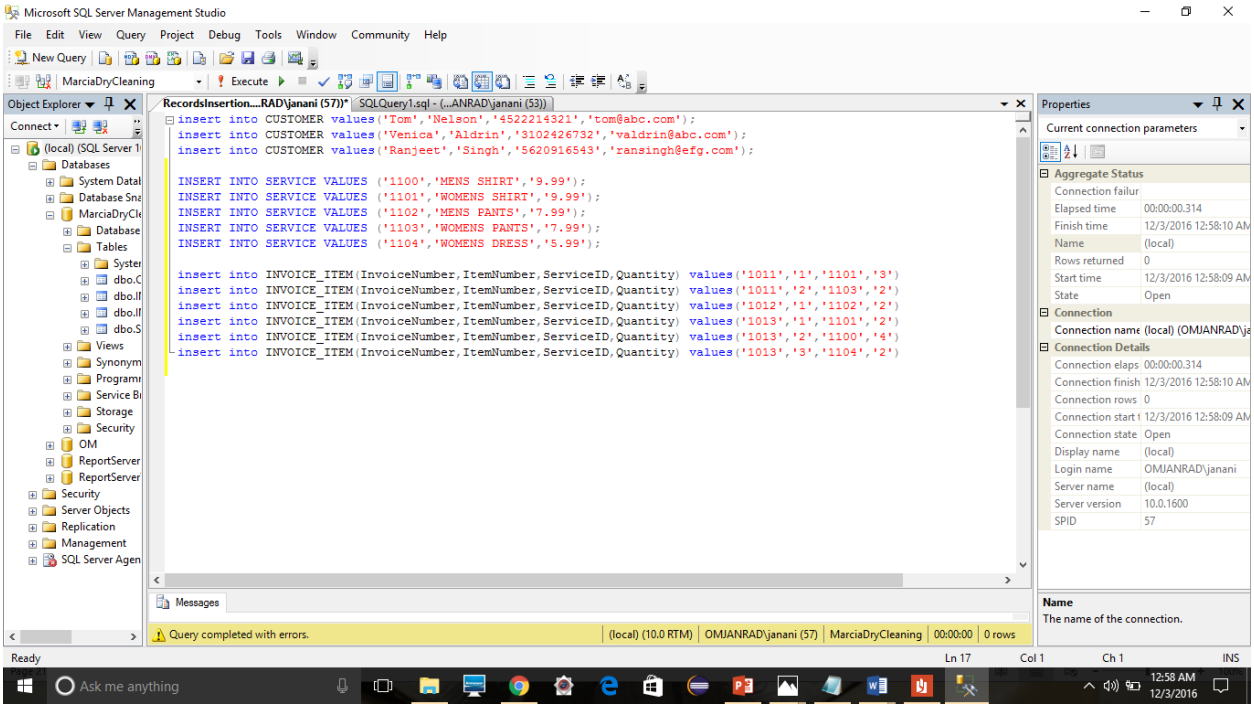
Records are inserted into INVOICE from INSERT TRIGGER OF INVOICE_ITEM

2-C.

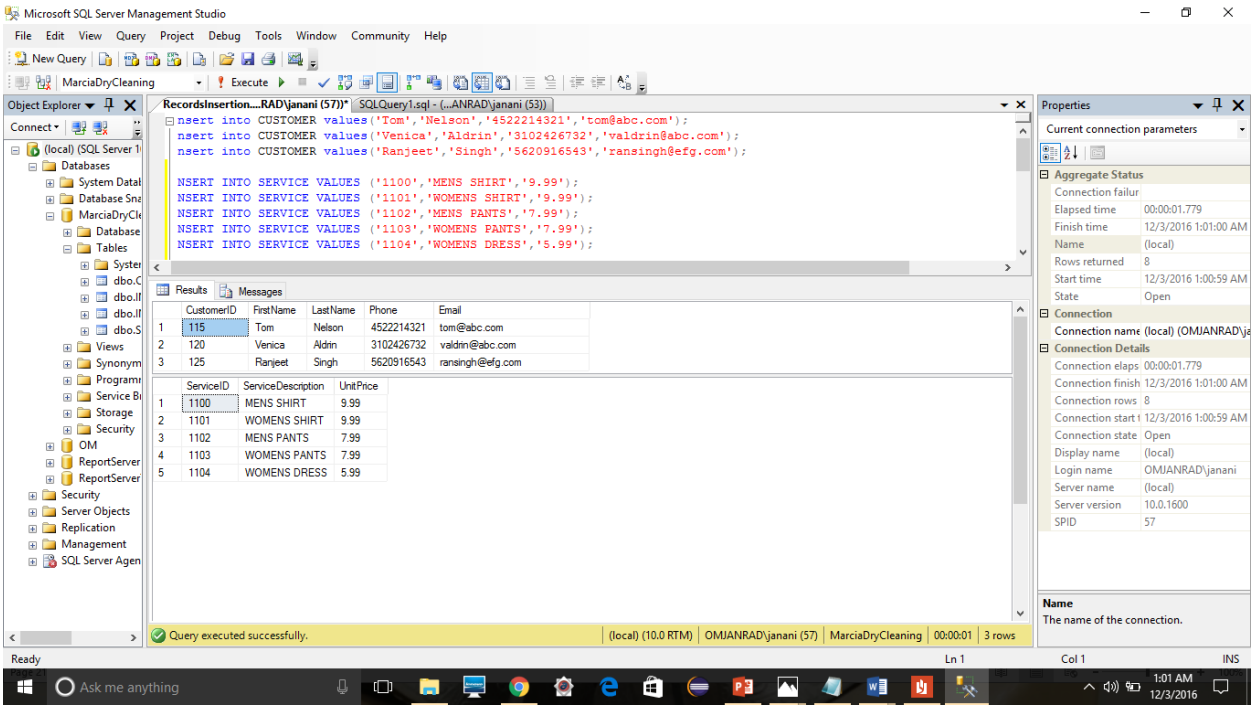
CREATION OF TABLES:



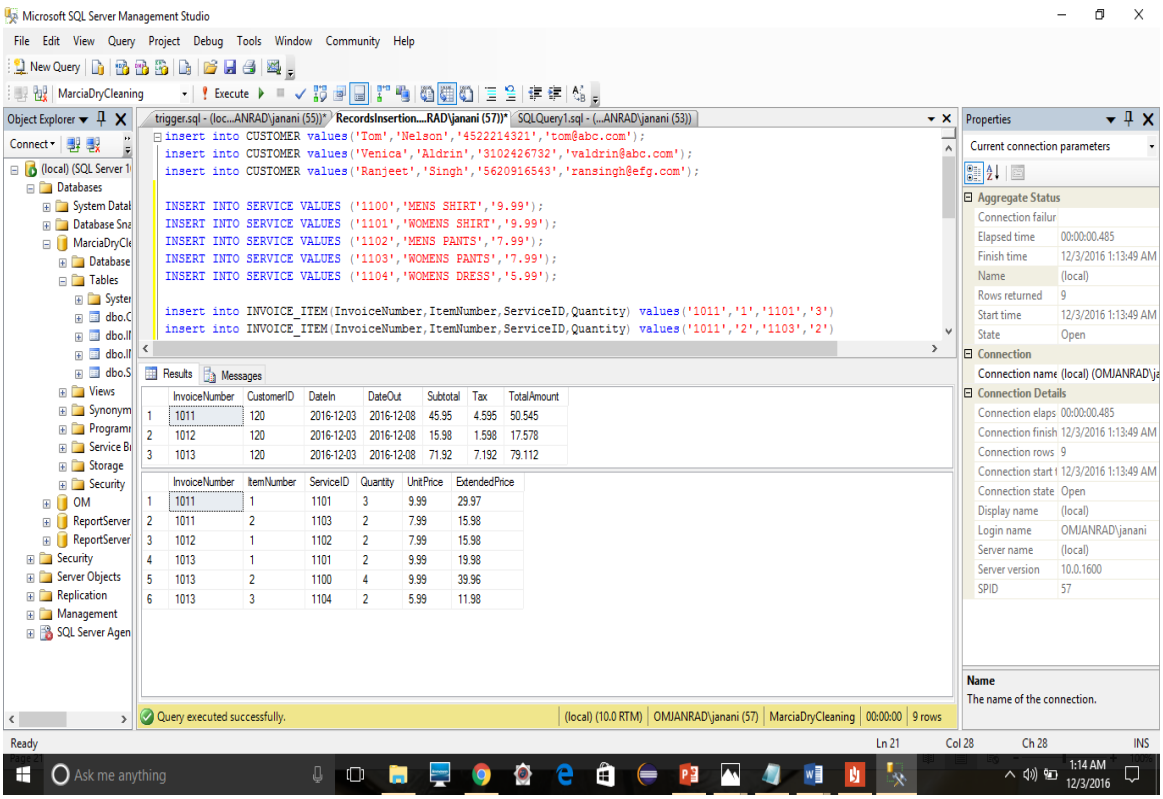
INSERTION OF RECORDS INTO THE TABLES:



CUSTOMER AND SERVICE TABLES:



INVOICE AND INVOICE_ITEM TABLES:



2-D.

IMPLEMENTATION OF TRIGGERS ON INVOICE AND INVOICE_ITEM USING SQL SERVER 2008

Insert Trigger on INVOICE_ITEM

```
CREATE TRIGGER pricecalculation ON INVOICE_ITEM

INSTEAD OF INSERT

AS

DECLARE @ExtPrice Money

DECLARE @ordernum int

DECLARE @itemnum int

DECLARE @UPrice Money

DECLARE @id int

DECLARE @Quantity int
```

```
declare @subtotal MONEY
declare @parentorder int
	set @ordernum= (select InvoiceNumber from inserted i);
	set @id= (select ServiceID from inserted i);
	set @Quantity= (select Quantity from inserted i);
	set @itemnum= (select ItemNumber from inserted i);
	set @UPrice= (select UnitPrice from SERVICE
				where SERVICE.ServiceID=@id);
	set @ExtPrice= @UPrice * @Quantity;
	set @parentorder=(select count(*) from INVOICE
				where InvoiceNumber=@ordernum);
	if @parentorder=0
		begin
			INSERT INTO INVOICE
				values (@ordernum,0,'1/1/1','1/1/1',0,0,0);
		end
	INSERT INTO INVOICE_ITEM
	 values(@ordernum, @itemnum, @id, @Quantity, @UPrice, @ExtPrice);
		set @subtotal=(select Subtotal from INVOICE
			where InvoiceNumber=@ordernum);
	update INVOICE set Subtotal=@subtotal+@ExtPrice
		where InvoiceNumber=@ordernum;
	update INVOICE set Tax= INVOICE.Subtotal * 0.1
		where InvoiceNumber=@ordernum;
	update INVOICE set TotalAmount = INVOICE.Subtotal + INVOICE.Tax
		where InvoiceNumber=@ordernum;
GO
```

Insert Trigger on Invoice

```
CREATE TRIGGER insertion_rules ON INVOICE
INSTEAD OF INSERT
AS
declare @ordercount int
declare @ordernum int
declare @custid int
declare @datein DATE
declare @dateout DATE
set @custid=(select CustomerID from inserted i);
set @ordernum=(select InvoiceNumber from inserted i);
if @custid = 0
        begin
        set @custid=RAND()*(130-110)+110
                if @custid%5 <> 0
                begin
                        set @custid=@custid-(@custid%5)
                end
        set @datein=getdate();
        set @dateout=getdate() + 5 ;
        INSERT INTO INVOICE values (@ordernum,@custid,@datein,@dateout,0,0,0);
        end
else
        begin
        set @ordercount=(select count(*)
                                        from INVOICE_ITEM
                                        where InvoiceNumber=@ordernum);
        if @ordercount=0
```

```
        Begin
        RAISERROR('Manual Insertion of an INVOICE prohibited. At least one record must exist in INVOICE_ITEM table to auto-generate the invoice for given InvoiceNumber', 16, 1);
        end
    else
        Begin
        RAISERROR('Data already exists for given InvoiceNumber. Use UPDATE to make any changes', 16, 1);
        end
    end
GO
```

IMPLEMENTATION OF DELETE TRIGGER

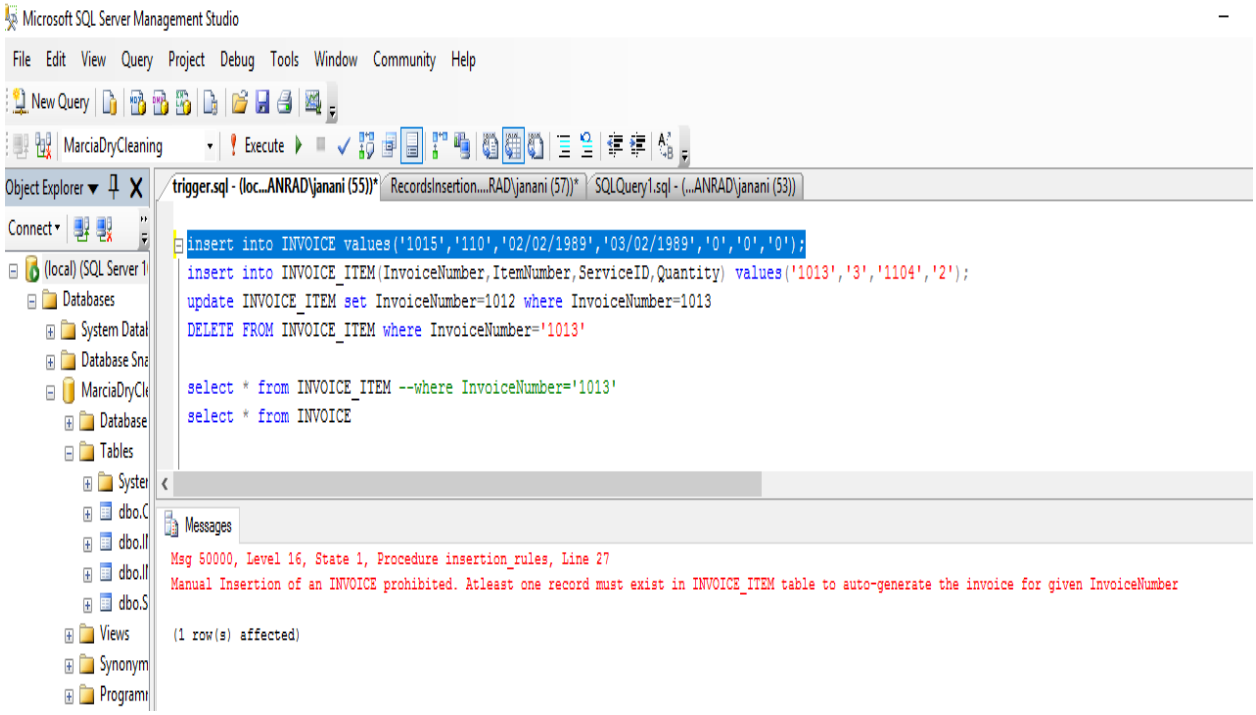
```
CREATE TRIGGER child_deletion ON INVOICE_ITEM
AFTER DELETE AS
    declare @ordercount int
    declare @ordernum int
    declare @subtotal int
    declare @tax int
    declare @amount int

    set @ordernum=(select distinct InvoiceNumber
                    from deleted d);
    print @ordernum
    set @ordercount=(select count(*) from INVOICE_ITEM
                        where InvoiceNumber=@ordernum);
    print @ordercount
    if @ordercount=0
```

```
        begin
            print @ordernum
            DELETE FROM INVOICE where InvoiceNumber=@ordernum
        End
    else
        begin
            set @subtotal=(select Subtotal
                                from INVOICE
                                where InvoiceNumber=@ordernum)- (select
                                ExtendedPrice
                                from INVOICE_ITEM
                                where InvoiceNumber=@ordernum);
            set @tax = @subtotal*0.1
            set @amount = @subtotal+@tax
            update INVOICE
            set Subtotal=@subtotal,Tax=@tax,TotalAmount=@amount
            where InvoiceNumber=@ordernum;
        End
GO
```

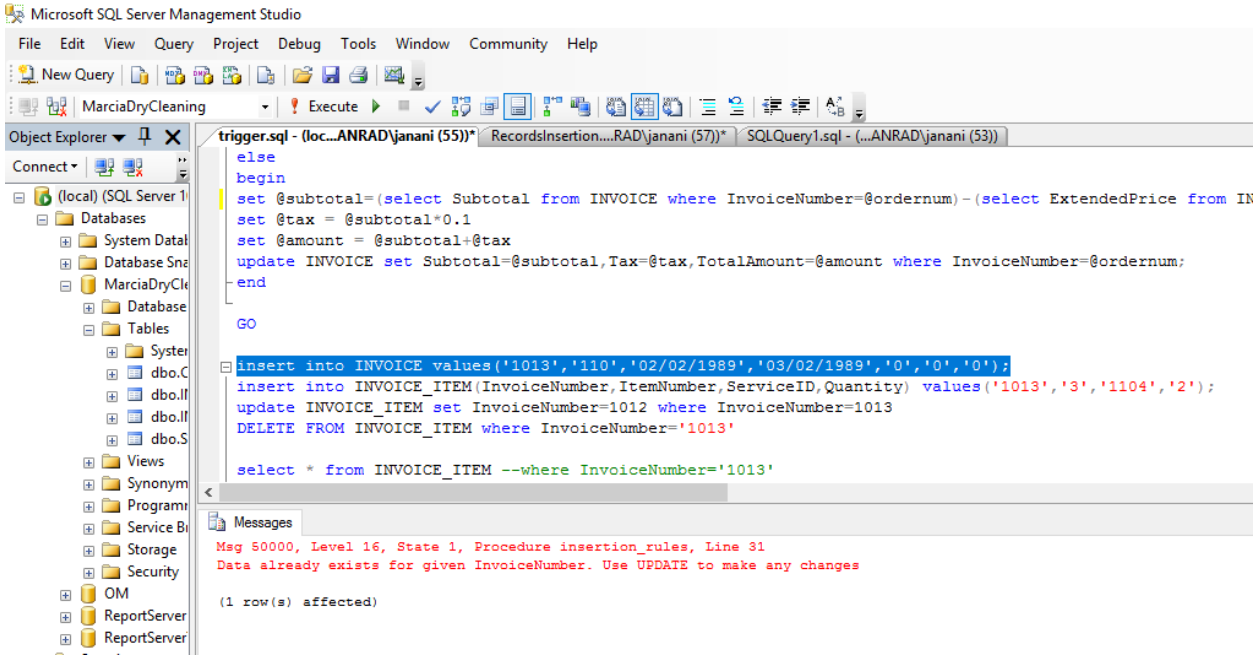
2-E.

Upon executing insert into statement for INVOICE from Query Window, the error is displayed as below

Manual Insertion of an INVOICE prohibited. At least one record must exist in INVOICE_ITEM table to auto-generate the invoice for given InvoiceNumber



Upon attempting to insert an existing Invoice which was already created from INSERTION TRIGGER OF INVOICE_ITEM on INVOICE TABLE



Upon deletion of INVOICE_ITEM, corresponding record on INVOICE gets deleted