

# **Project 2**

## **State MD Urgent Center Database**

Yuqing Liu

Fall2020

## **Abstract**

The following document will describe procedure about how to design, implement and test State MD Urgent Care database. I will create tables for several parts. I use Vetabelo to draw the E-R diagram and 3F forms. After I create the tables, I will test them by feeding data, create view and several complicated operations like function, stored procedures and scripts.

# I. Design

## Introduction:

This project is about making a database for an Urgent Health Center due to COVID-19. The important parts of this database are Centers , Patients and Tests. Since the relationship between Centers and patients are many-to-many and the Test is the bridge to connect them. Inside the database, I use 12 tables to build the business structure and relationship in the Description They are Center, Tests, Patients, Doctors, Payment, Invoice, HealthHistory, Report, MedicalEquipment DrugStorage, ProcedureCapacity, Insurance.

Here are my three main steps to design the table:

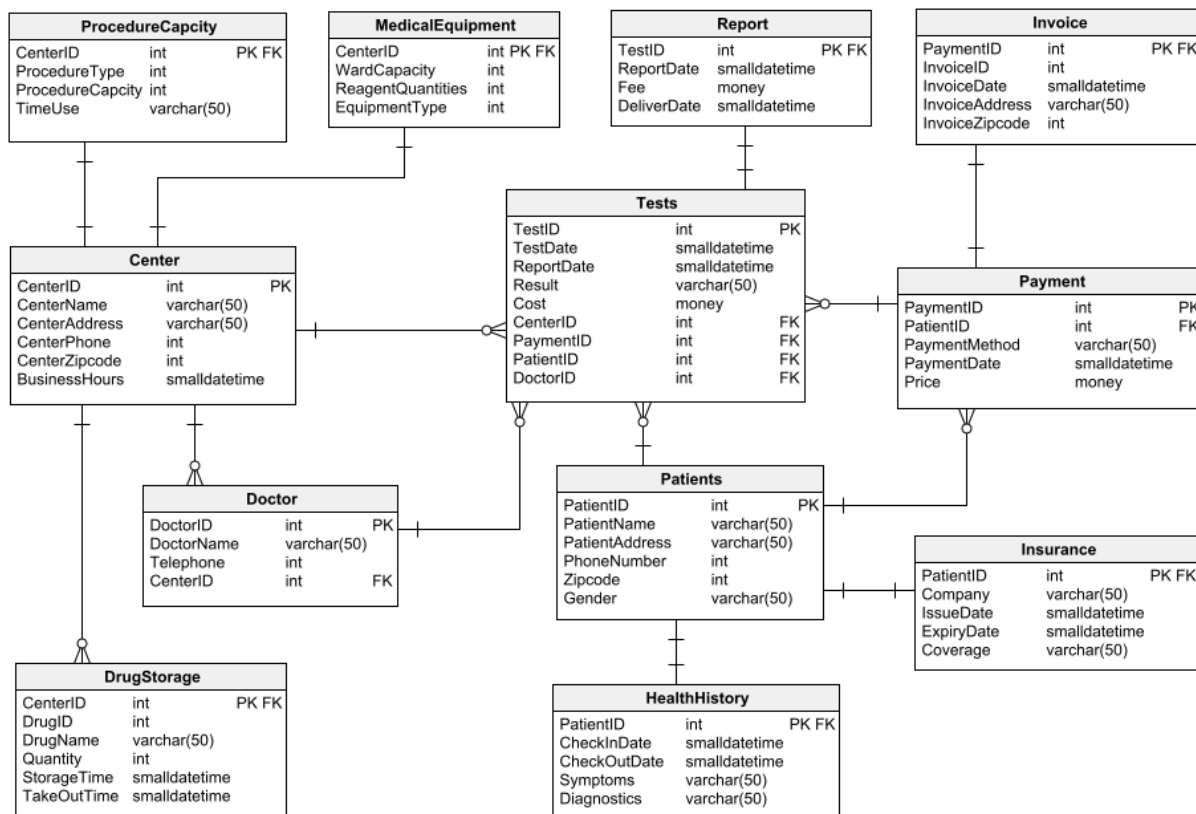
First, I make a Patients Table with 6 columns . There are PatientID, PatientName, PatientAddress, PhoneNumber, Zipcode and Gender inside this table. After that, The PatientID is primary key. It is an identity for each patients. And then I recognized the relationships between tables. Patients to Tests is one-to-many, Patients to Payment is one-to-many, Patients to Insurance and HealthHistory are one to one. Since I supposed one patient only buy one kind of healthy insurance.

And then, I make a Center Table to save the Center information. Here are also 6 columns in this table: CenterID, CenterName, CenterAddress, CenterZipcode, CenterPhone, BusinessHour. Since CenterID is identity to each center, the primary key should be CenterID. And Center to, MedicalEquipment,

ProdcedureStorage are one-to-one. Center to Doctors and DrugStorage are one to many, Center to Tests is one-to-many, since each center could test several times for patients. All the default values are not null.

Last but not the least, I create Tests table ti connect table Center and table Patients with 9 columnes. The primary key is TestID and foreign keys are PatientID, PaymentID, CenterID and DoctorID.And Tests to the table Report is one to one.

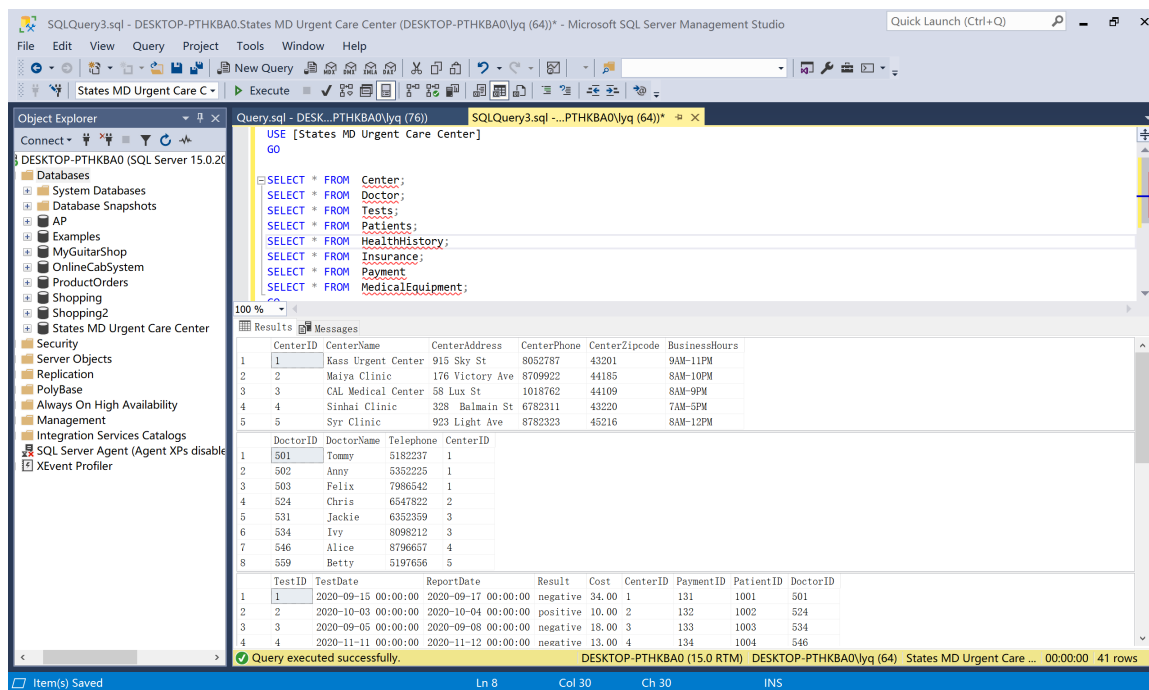
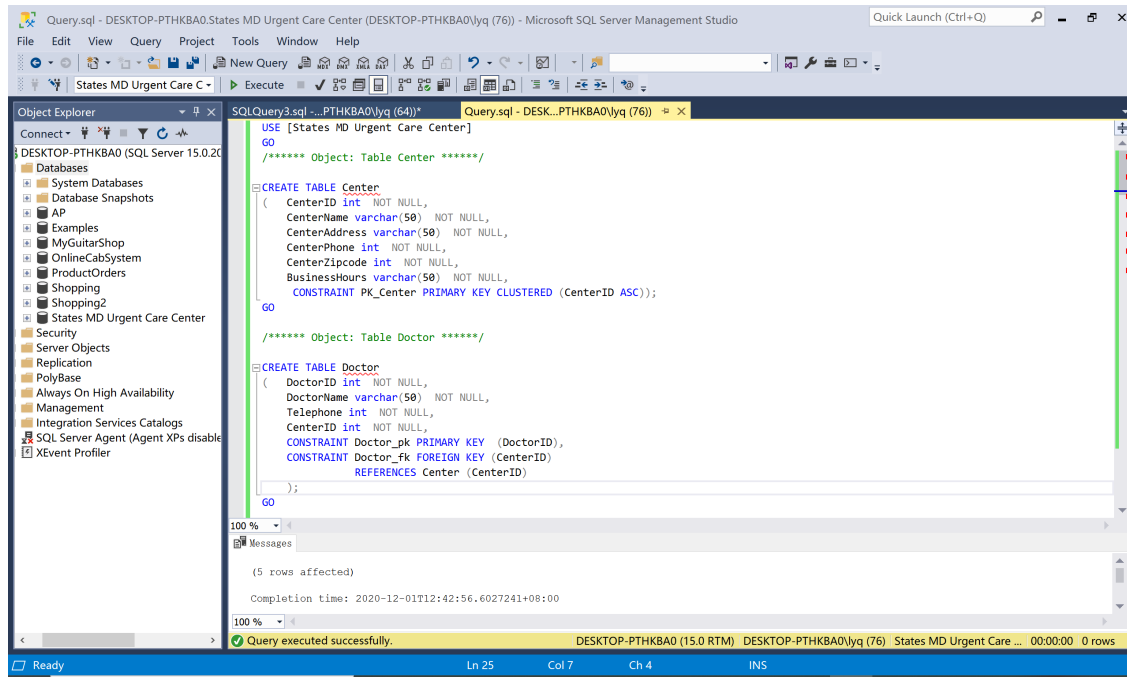
## Design Diagram:



## II. Implementation

### Screenshots:

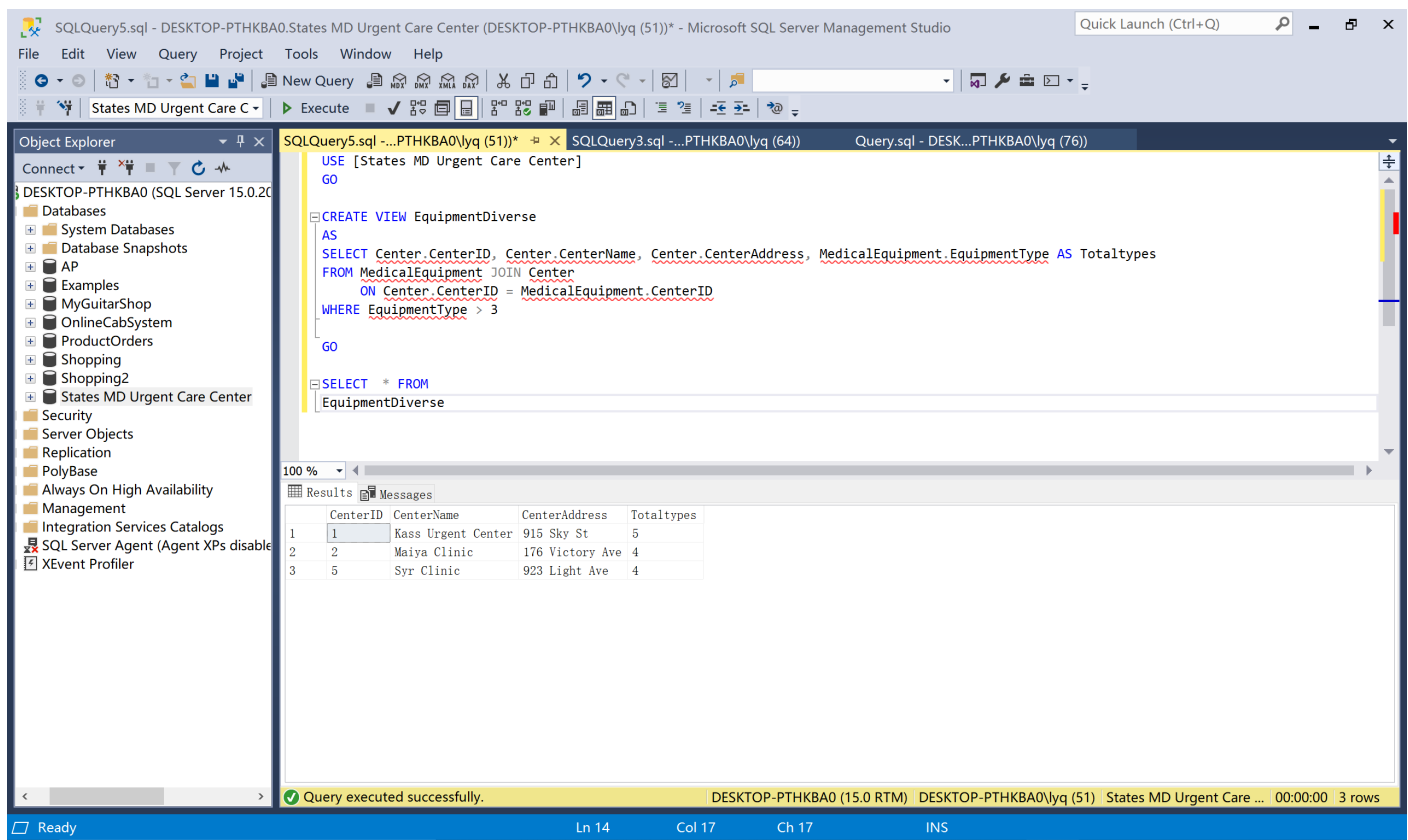
The entire code implementation could be seen as sql file



## III. Testing

### View-1

This view returns Centers with more than 3 equipment types which offers people have opportunity to choose health center with better equipments.



The screenshot displays the Microsoft SQL Server Management Studio interface. The left pane shows the Object Explorer with the 'States MD Urgent Care Center' database selected. The central pane contains a SQL query that creates a view named 'EquipmentDiverse' and then selects from it. The query is as follows:

```
USE [States MD Urgent Care Center]
GO

CREATE VIEW EquipmentDiverse
AS
SELECT Center.CenterID, Center.CenterName, Center.CenterAddress, MedicalEquipment.EquipmentType AS Totaltypes
FROM MedicalEquipment JOIN Center
ON Center.CenterID = MedicalEquipment.CenterID
WHERE EquipmentType > 3
GO

SELECT * FROM
EquipmentDiverse
```

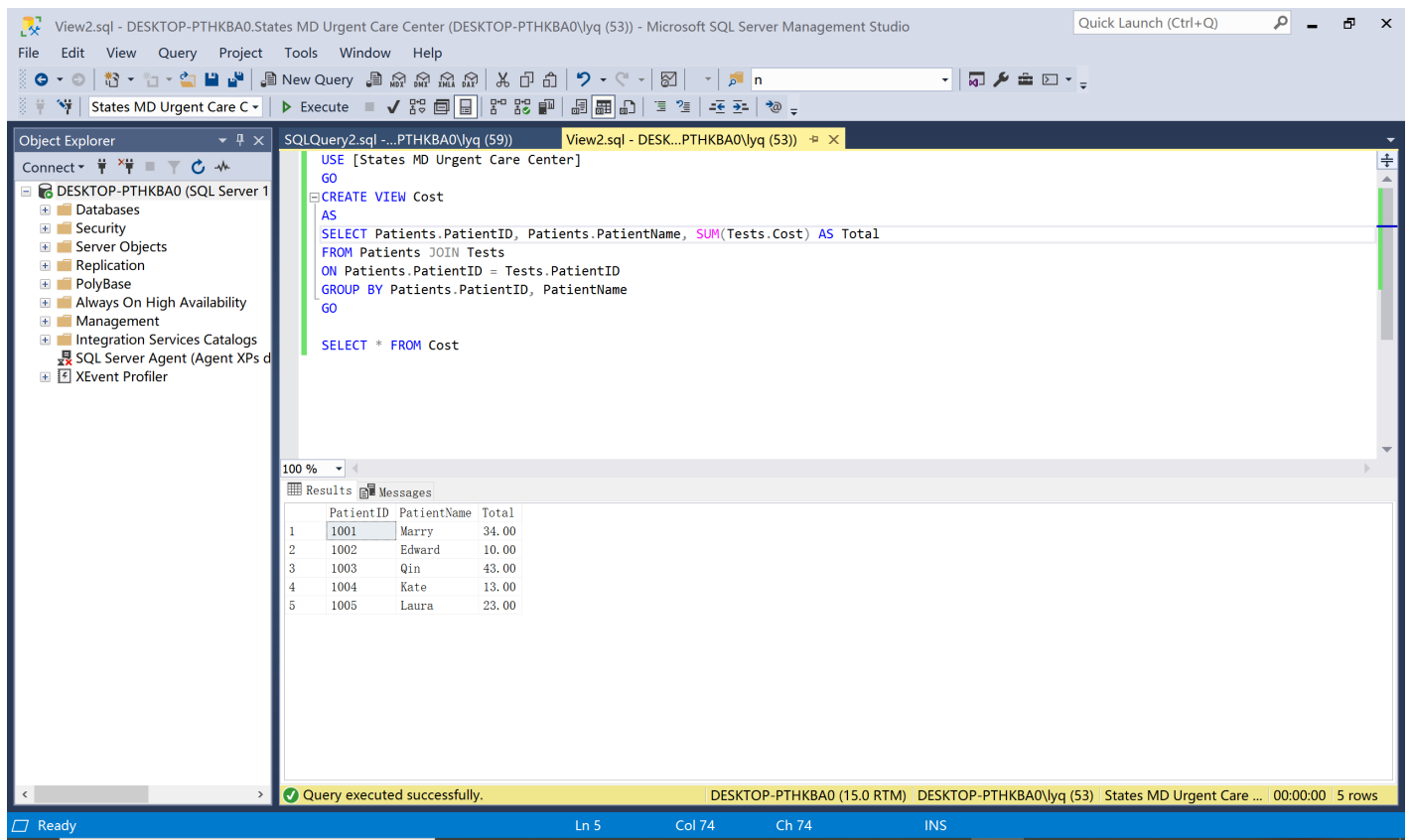
The bottom pane shows the results of the query, which are displayed in a table with 4 columns: CenterID, CenterName, CenterAddress, and Totaltypes. The table contains 3 rows of data:

CenterID	CenterName	CenterAddress	Totaltypes
1	Kass Urgent Center	915 Sky St	5
2	Maiya Clinic	176 Victory Ave	4
5	Syr Clinic	923 Light Ave	4

The status bar at the bottom indicates that the query was executed successfully, returning 3 rows.

## View-2

This view could be able to figure out the relationship between Patients and Tests, since some patients might test more than one time and the view displays the total cost for each patient.



The screenshot displays the Microsoft SQL Server Management Studio interface. The title bar indicates the current query is 'View2.sql - DESKTOP-PTHKBA0.States MD Urgent Care Center (DESKTOP-PTHKBA0\lyq (53)) - Microsoft SQL Server Management Studio'. The 'Object Explorer' on the left shows the server structure for 'DESKTOP-PTHKBA0 (SQL Server 1)'. The main query editor contains the following SQL code:

```
USE [States MD Urgent Care Center]
GO
CREATE VIEW Cost
AS
SELECT Patients.PatientID, Patients.PatientName, SUM(Tests.Cost) AS Total
FROM Patients JOIN Tests
ON Patients.PatientID = Tests.PatientID
GROUP BY Patients.PatientID, PatientName
GO
SELECT * FROM Cost
```

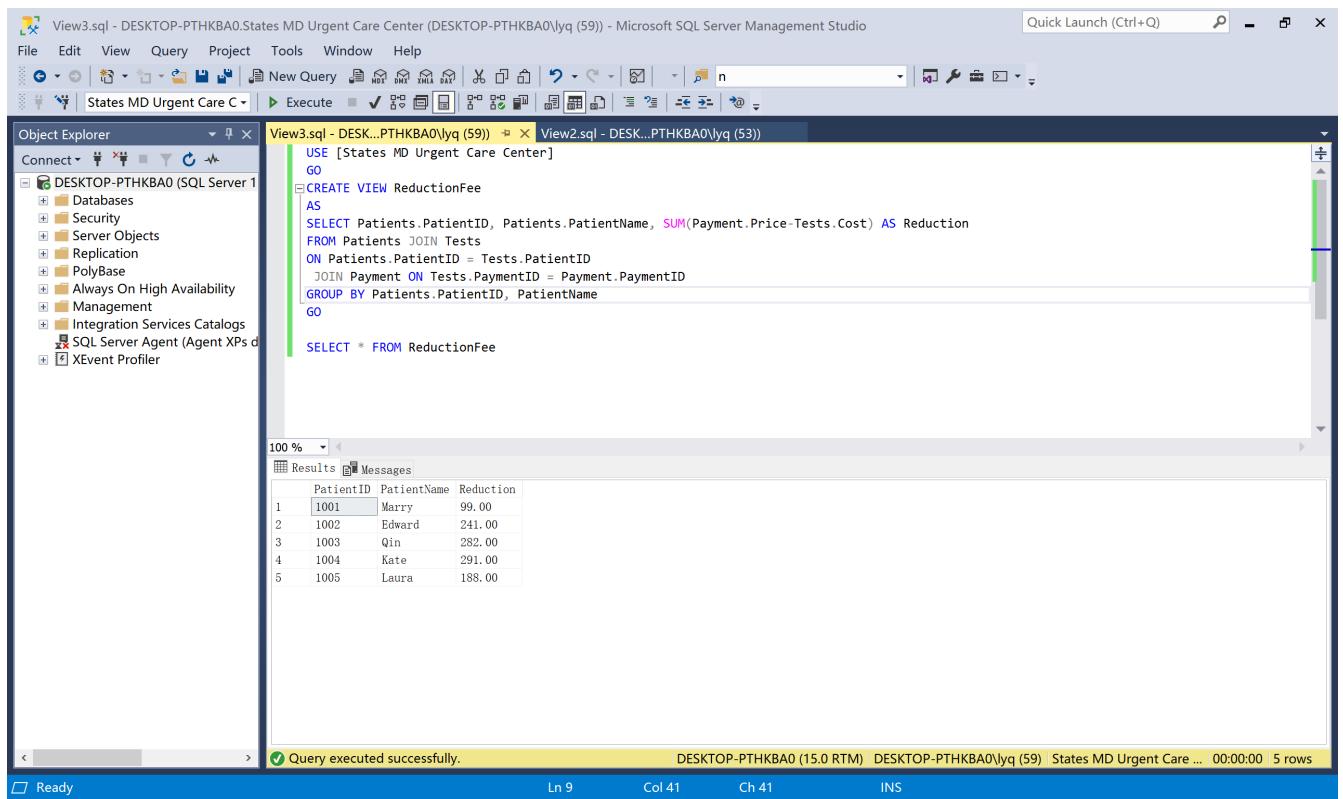
The 'Results' pane at the bottom shows the output of the query, displaying a table with 5 rows and 3 columns: PatientID, PatientName, and Total. The data is as follows:

	PatientID	PatientName	Total
1	1001	Marry	34.00
2	1002	Edward	10.00
3	1003	Qin	43.00
4	1004	Kate	13.00
5	1005	Laura	23.00

The status bar at the bottom indicates 'Query executed successfully.' and shows the execution time as '00:00:00' with '5 rows' returned.

## View-3

This view could be able to figure out the reduction fee of insurance for each patient. It could display the welfare of each patient with different insurance coverage and price to some content.



The screenshot displays the Microsoft SQL Server Management Studio interface. The main window shows a SQL query in the 'View3.sql' editor. The query is as follows:

```
USE [States MD Urgent Care Center]
GO
CREATE VIEW ReductionFee
AS
SELECT Patients.PatientID, Patients.PatientName, SUM(Payment.Price-Tests.Cost) AS Reduction
FROM Patients JOIN Tests
ON Patients.PatientID = Tests.PatientID
JOIN Payment ON Tests.PaymentID = Payment.PaymentID
GROUP BY Patients.PatientID, PatientName
GO
SELECT * FROM ReductionFee
```

The 'Object Explorer' on the left shows the database structure for 'DESKTOP-PTHKBA0 (SQL Server 11.0.5600.10)'. The 'Results' pane at the bottom shows the output of the query, which is a table with 5 rows and 3 columns: PatientID, PatientName, and Reduction.

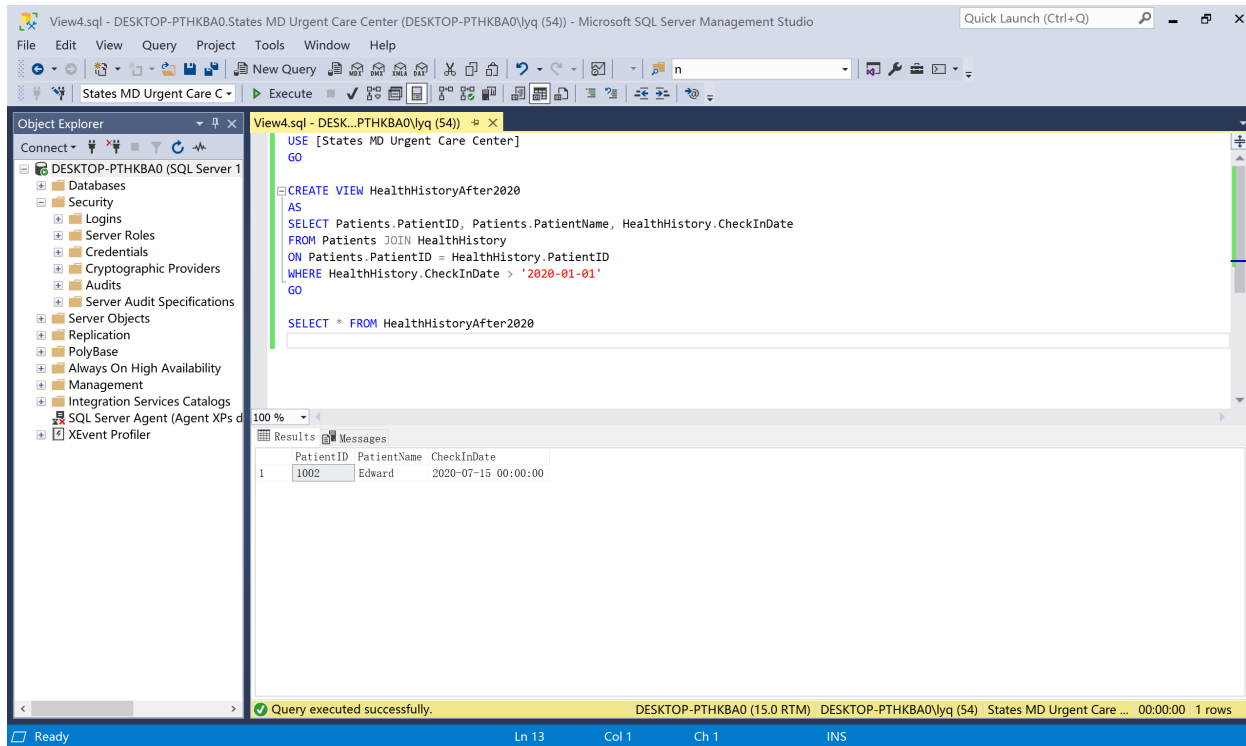
	PatientID	PatientName	Reduction
1	1001	Marry	99.00
2	1002	Edward	241.00
3	1003	Qin	282.00
4	1004	Kate	291.00
5	1005	Laura	188.00

The status bar at the bottom indicates 'Query executed successfully.' and 'DESKTOP-PTHKBA0 (15.0 RTM) DESKTOP-PTHKBA0\lyq (59) States MD Urgent Care ... 00:00:00 5 rows'.



## View-4

This view could be able to figure out the patient who had past medical history after 2020. It could help the urgent center to analyze the risk of re-infection after treatment.



The screenshot displays the Microsoft SQL Server Management Studio interface. The title bar indicates the connection to 'DESKTOP-PTHKBA0\States MD Urgent Care Center (DESKTOP-PTHKBA0\lyq (54))'. The Object Explorer on the left shows the database structure, including 'DESKTOP-PTHKBA0 (SQL Server 15.0 RTM)' with folders for Databases, Security, Logins, Server Roles, Credentials, Cryptographic Providers, Audits, Server Audit Specifications, Server Objects, Replication, PolyBase, Always On High Availability, Management, Integration Services Catalogs, SQL Server Agent (Agent XPs d...), and XEvent Profiler.

The central query editor shows the following SQL script:

```
USE [States MD Urgent Care Center]
GO

CREATE VIEW HealthHistoryAfter2020
AS
SELECT Patients.PatientID, Patients.PatientName, HealthHistory.CheckInDate
FROM Patients JOIN HealthHistory
ON Patients.PatientID = HealthHistory.PatientID
WHERE HealthHistory.CheckInDate > '2020-01-01'
GO

SELECT * FROM HealthHistoryAfter2020
```

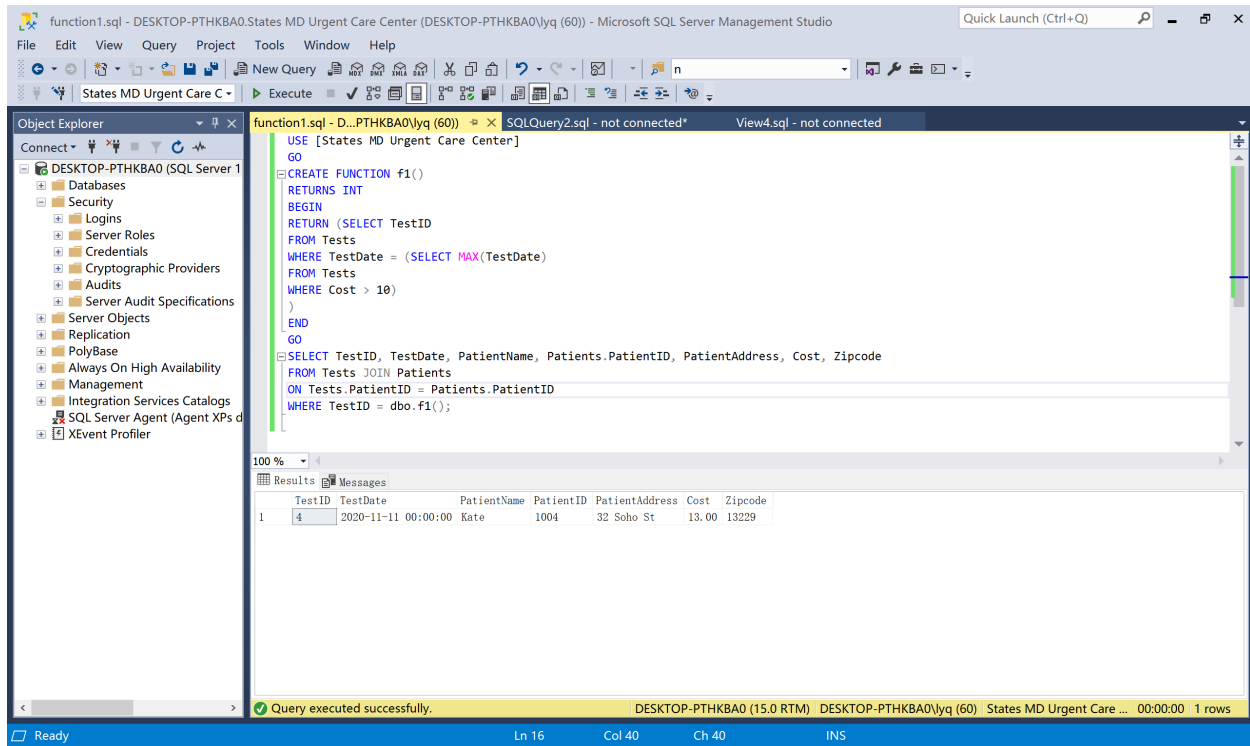
The Results pane at the bottom shows the output of the query, displaying a single row of data:

	PatientID	PatientName	CheckInDate
1	1002	Edward	2020-07-15 00:00:00

The status bar at the bottom indicates 'Query executed successfully.' and 'DESKTOP-PTHKBA0 (15.0 RTM) | DESKTOP-PTHKBA0\lyq (54) | States MD Urgent Care ... | 00:00:00 | 1 rows'.

# Function-1

1. In this function, it will return the Test information with patient information that in the most recent day. And the Cost of this test should bigger than 10 dollars .



The screenshot displays the Microsoft SQL Server Management Studio interface. The left pane shows the Object Explorer with the 'DESKTOP-PTHKBA0 (SQL Server 15.0)' instance expanded. The central pane shows a T-SQL script for a function named 'f1'. The script is as follows:

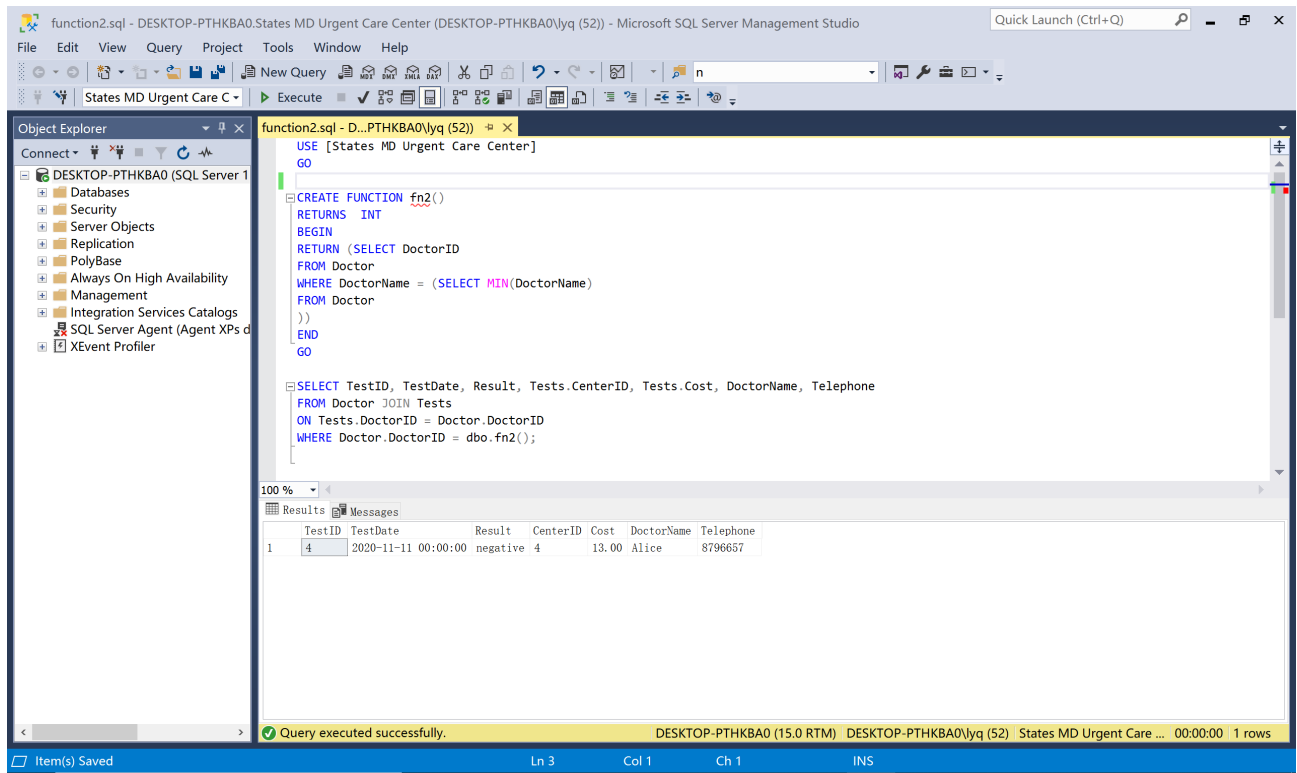
```
USE [States MD Urgent Care Center]
GO
CREATE FUNCTION f1()
RETURNS INT
BEGIN
    RETURN (SELECT TestID
    FROM Tests
    WHERE TestDate = (SELECT MAX(TestDate)
    FROM Tests
    WHERE Cost > 10)
    )
END
GO
SELECT TestID, TestDate, PatientName, Patients.PatientID, PatientAddress, Cost, Zipcode
FROM Tests JOIN Patients
ON Tests.PatientID = Patients.PatientID
WHERE TestID = dbo.f1();
```

The bottom pane shows the results of the query execution. A status bar at the bottom indicates 'Query executed successfully.' and 'DESKTOP-PTHKBA0 (15.0 RTM) DESKTOP-PTHKBA0\lyq (60) States MD Urgent Care ... 00:00:00 1 rows'.

TestID	TestDate	PatientName	PatientID	PatientAddress	Cost	Zipcode
4	2020-11-11 00:00:00	Kate	1004	32 Soho St	13.00	13229

## Function-2

In this function, it returns the Doctor with the most small doctor name 'Alice' with the information of Center and Test.



The screenshot displays the Microsoft SQL Server Management Studio interface. The main window shows a SQL query in a script editor, and the bottom pane shows the results of the query execution.

**Query Script:**

```
USE [States MD Urgent Care Center]
GO

CREATE FUNCTION fn2()
RETURNS INT
BEGIN
    RETURN (SELECT DoctorID
    FROM Doctor
    WHERE DoctorName = (SELECT MIN(DoctorName)
    FROM Doctor
    ))
END
GO

SELECT TestID, TestDate, Result, Tests.CenterID, Tests.Cost, DoctorName, Telephone
FROM Doctor JOIN Tests
ON Tests.DoctorID = Doctor.DoctorID
WHERE Doctor.DoctorID = dbo.fn2();
```

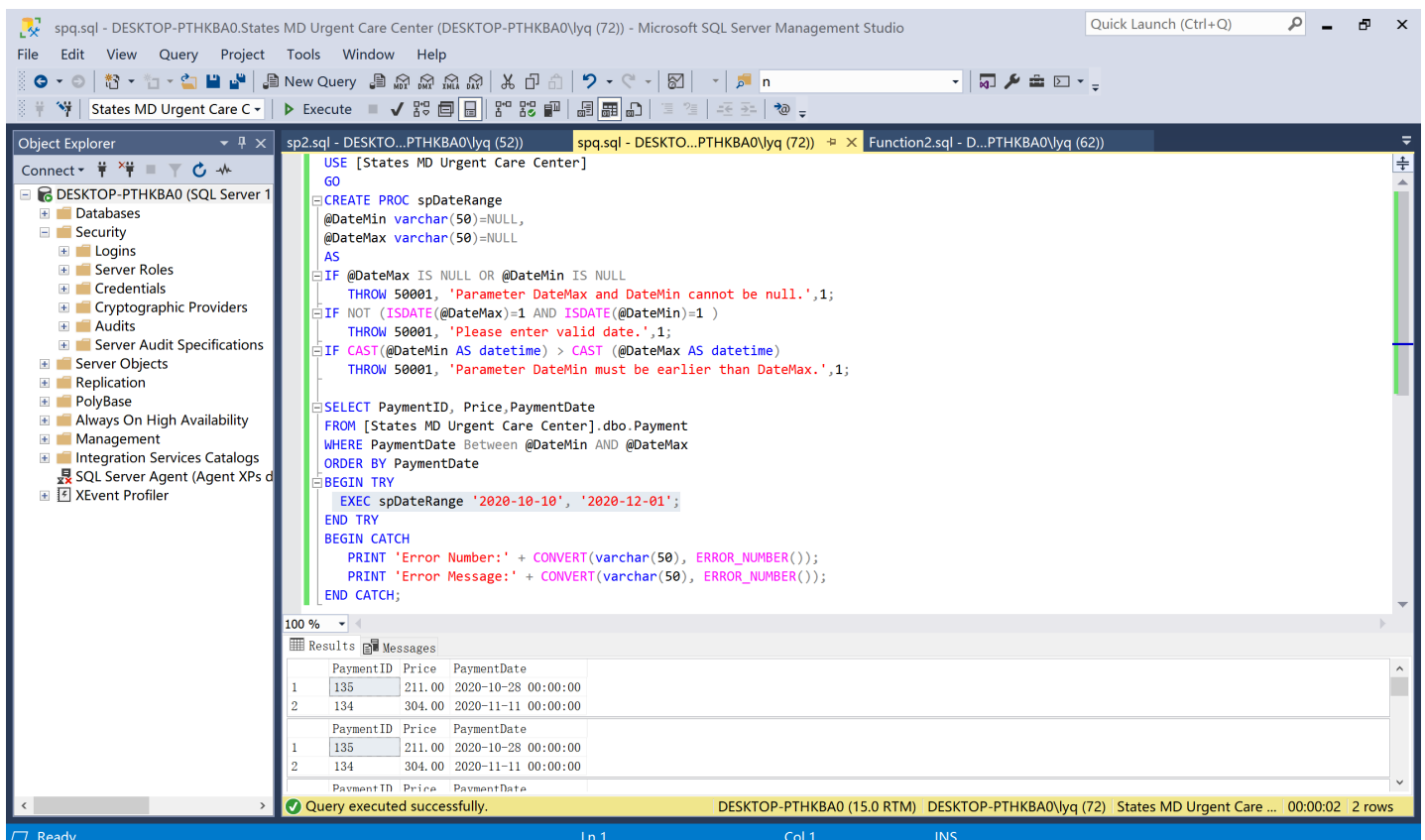
**Results:**

TestID	TestDate	Result	CenterID	Cost	DoctorName	Telephone	
1	4	2020-11-11 00:00:00	negative	4	13.00	Alice	8796657

The status bar at the bottom indicates: "Query executed successfully. DESKTOP-PTHKBA0 (15.0 RTM) DESKTOP-PTHKBA0\lyq (52) States MD Urgent Care ... 00:00:00 1 rows".

# Stored Procedures-1

This stored procedure will output the Payment of tests between 2020-10-10 and 2020-12-01. Before it output the



The screenshot displays the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the server structure for 'DESKTOP-PTHKBA0 (SQL Server 15.0 RTM)'. The main window shows the execution of a stored procedure named 'spDateRange' in the 'States MD Urgent Care Center' database. The procedure is designed to output payment data for a specific date range.

```
USE [States MD Urgent Care Center]
GO
CREATE PROC spDateRange
@DateMin varchar(50)=NULL,
@DateMax varchar(50)=NULL
AS
IF @DateMax IS NULL OR @DateMin IS NULL
THROW 50001, 'Parameter DateMax and DateMin cannot be null.',1;
IF NOT (ISDATE(@DateMax)=1 AND ISDATE(@DateMin)=1 )
THROW 50001, 'Please enter valid date.',1;
IF CAST(@DateMin AS datetime) > CAST (@DateMax AS datetime)
THROW 50001, 'Parameter DateMin must be earlier than DateMax.',1;
SELECT PaymentID, Price,PaymentDate
FROM [States MD Urgent Care Center].dbo.Payment
WHERE PaymentDate Between @DateMin AND @DateMax
ORDER BY PaymentDate
BEGIN TRY
EXEC spDateRange '2020-10-10', '2020-12-01';
END TRY
BEGIN CATCH
PRINT 'Error Number:' + CONVERT(varchar(50), ERROR_NUMBER());
PRINT 'Error Message:' + CONVERT(varchar(50), ERROR_MESSAGE());
END CATCH;
```

The Results pane shows the output of the stored procedure, displaying two rows of payment data:

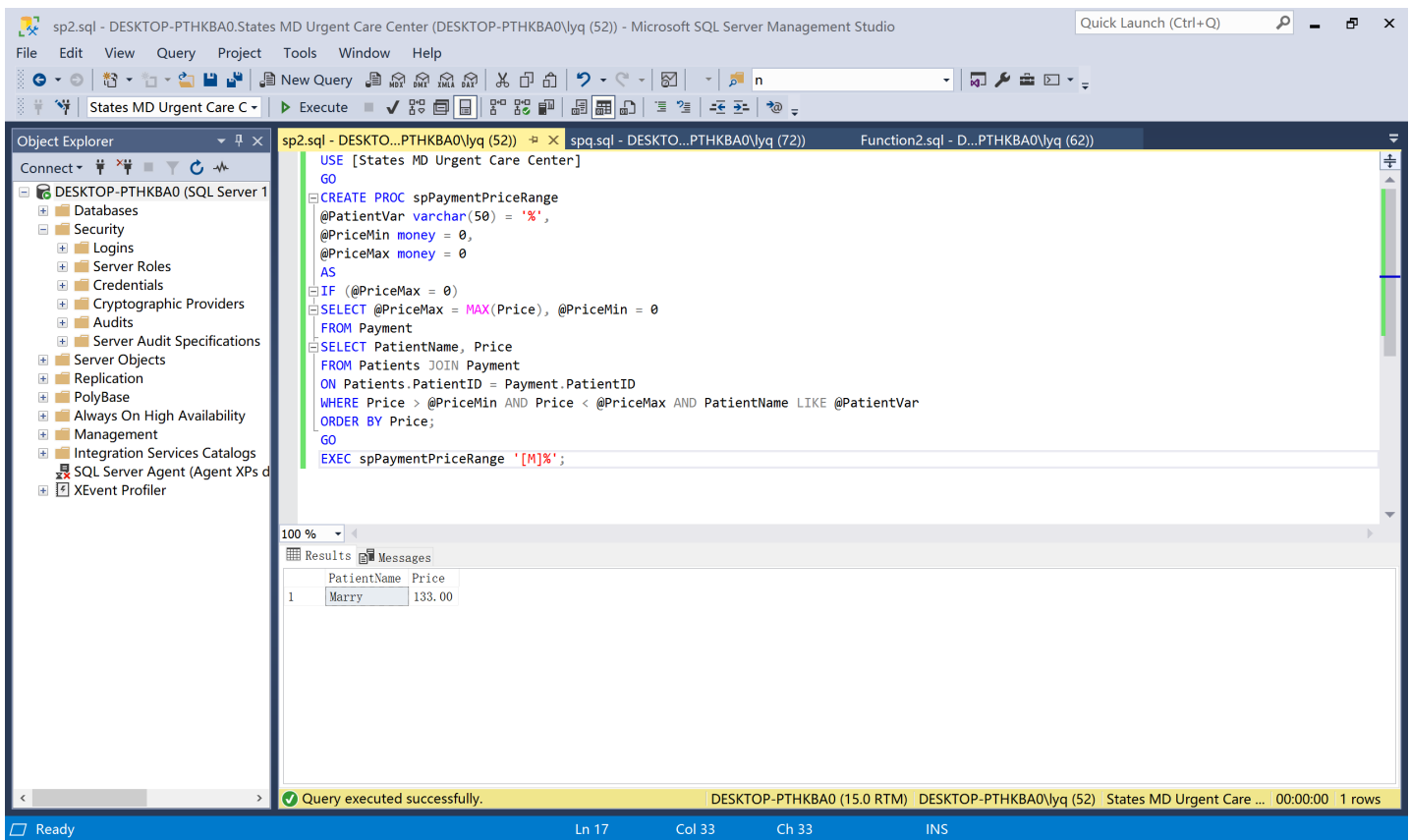
PaymentID	Price	PaymentDate
135	211.00	2020-10-28 00:00:00
134	304.00	2020-11-11 00:00:00

The status bar at the bottom indicates that the query was executed successfully, returning 2 rows.

table, it will validate the parameter. If everything goes fine, it will return the value we want to the console.

## Stored Procedures-2

In this problem, the SP will calculate the Price range for some patients with specific PatientName. It will return the DriverName, PaymentPrice to show that the price range of patients that their name includes some substring.



The screenshot displays the Microsoft SQL Server Management Studio interface. The main window shows a SQL query for a stored procedure named `spPaymentPriceRange`. The query uses a `USE` statement to connect to the `States MD Urgent Care Center` database. It then defines the procedure with parameters `@PatientVar` (varchar(50)), `@PriceMin` (money), and `@PriceMax` (money). The procedure logic includes an `IF` statement to set `@PriceMax` to the maximum price if it's zero, followed by a `SELECT` statement that joins the `Patients` and `Payment` tables. The `WHERE` clause filters for patients whose names contain the substring `@PatientVar` and whose prices are within the specified range. The results are ordered by price. Finally, the procedure is executed with the parameter `'Marry'`.

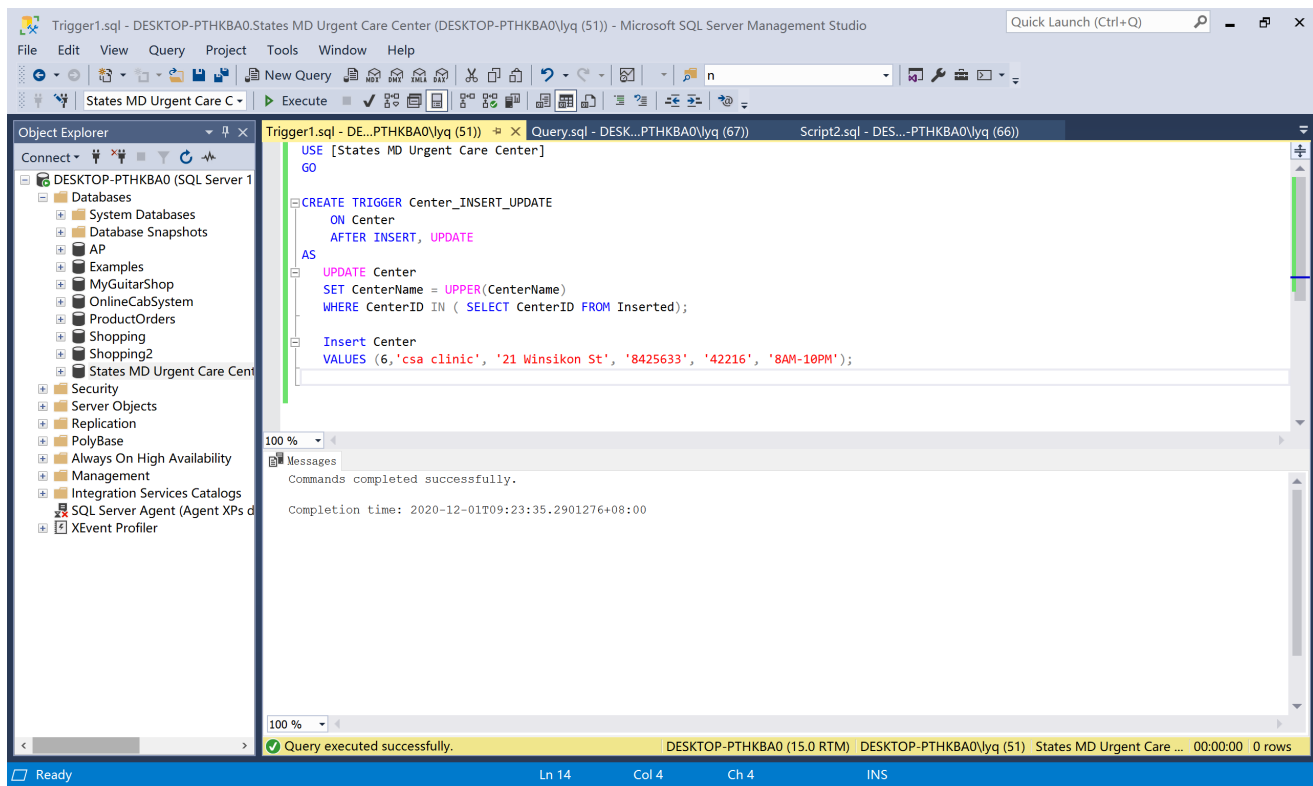
The bottom of the window shows the execution results in a table:

	PatientName	Price
1	Marry	133.00

The status bar at the bottom indicates that the query was executed successfully.

# Triggers

This trigger could be able to corrects Center name to Upper after insert or update the table which help people to modify information of database.



# Transaction

This transaction reflect a change of insurance company and PatientID for a patient whose ID number was used to be 1005 and now is 1008 with a new insurance company named OMT.

The screenshot displays the Microsoft SQL Server Management Studio interface. The main window shows a SQL query being executed in the 'Transaction.sql' file. The query is as follows:

```
USE [States MD Urgent Care Center]
GO

IF EXISTS ( SELECT * FROM sys.triggers
            WHERE object_id = OBJECT_ID ('Insurance_UPDATE_company'))
DROP TRIGGER [dbo].[Insurance_UPDATE_company];
GO

BEGIN TRAN
UPDATE Insurance
SET PatientID = 1008
WHERE PatientID = 1005;

DELETE Insurance
WHERE PatientID = 1005;

UPDATE Insurance
SET Company = 'OMT'
WHERE PatientID = 1008;
COMMIT TRAN;

SELECT * FROM Insurance WHERE PatientID = 1008
```

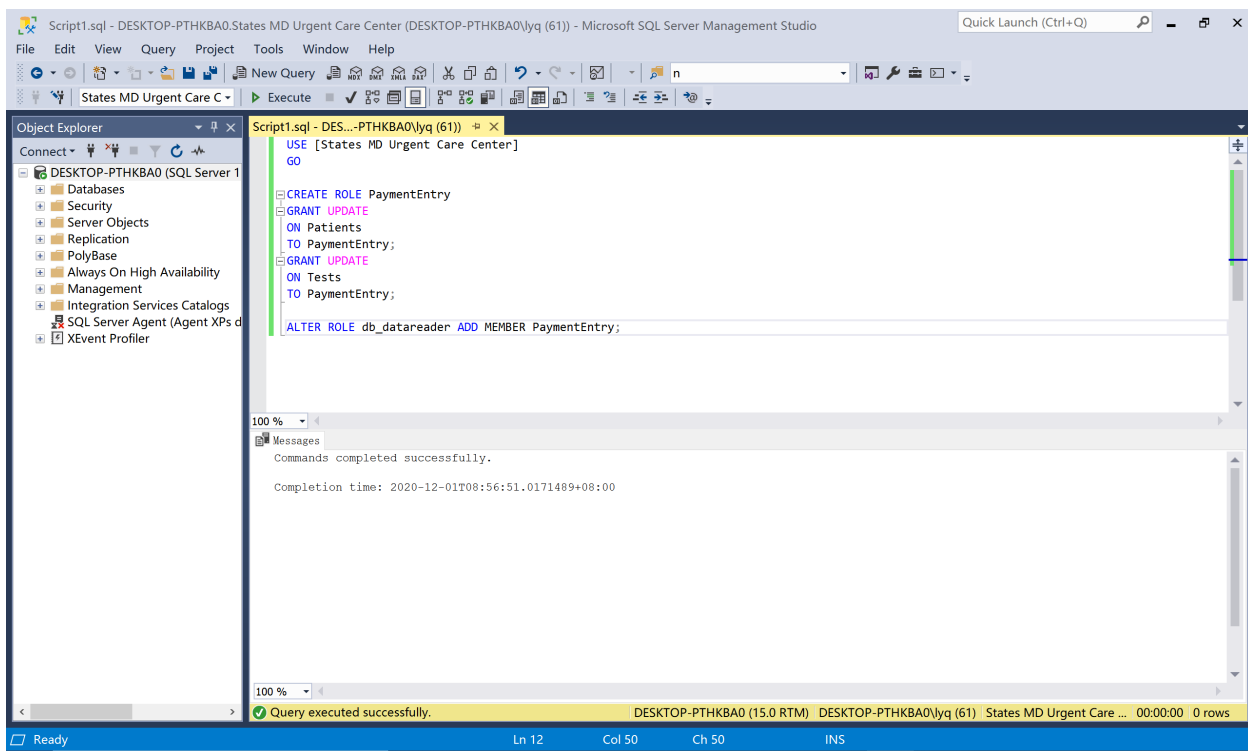
The 'Results' pane at the bottom shows the output of the query, which is a single row from the 'Insurance' table:

PatientID	Company	IssueDate	ExpiryDate	Coverage
1008	OMT	2014-03-22 00:00:00	2024-03-22 00:00:00	90%

The status bar at the bottom indicates 'Query executed successfully.' and 'DESKTOP-PTHKBA0 (15.0 RTM) DESKTOP-PTHKBA0\lyq (65) States MD Urgent Care... 00:00:00 1 rows'.

# Scripts-1

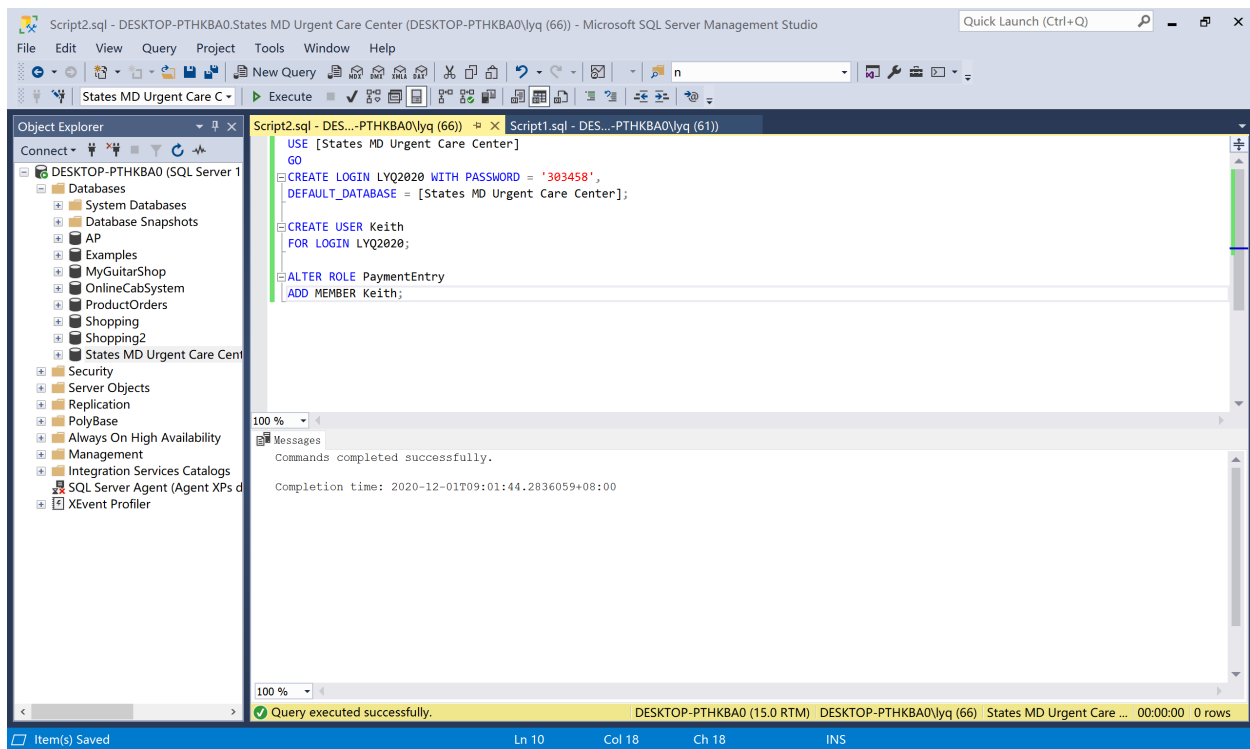
In this script, I create a user-defined database role named Paymententry in the State MD Urgent Care Center database and give permissions of UPDATE to both the table Patients and the table Tests.





## Scripts-2

In this script, I create a login ID named 'LYQ2020' with the password '303458' and assign a user named Keith for login to the PaymentEntry role.



### **III. Conclusions**

#### **Project Analysis:**

During finish this project, This project gave me a better comprehension of database logic, and gave me inspiration of designing a database. Although I meet a lot of problems from database design to implementation, from data insert to data testing. I could be able to solve them in the end. And one of the important thing is to identify data elements, tables and assign them into logic relationship. I think the ER diagram is very useful for database design since it clearly displayed tables relationships and contribute to the construction of database structure. The other thing I think is essential too is to implement database with sql language without errors, It takes long time to analysis the project requirements and the data flow and structure , but it is most important for any project. If you do something wrong in the beginning. It will become so hard to change something in the middle of the project since I wrote wrong syntax and it did took time to correct it. Throughout this time I could write syntax and test them by several view, function, stored procedure or scripts more similarly.

**Remarks:**

After I finish this project, I learned a lot about how to analysis statistics and implement a database. I think establishing the right relationship between tables is as same important as creating right syntax for designing a database.