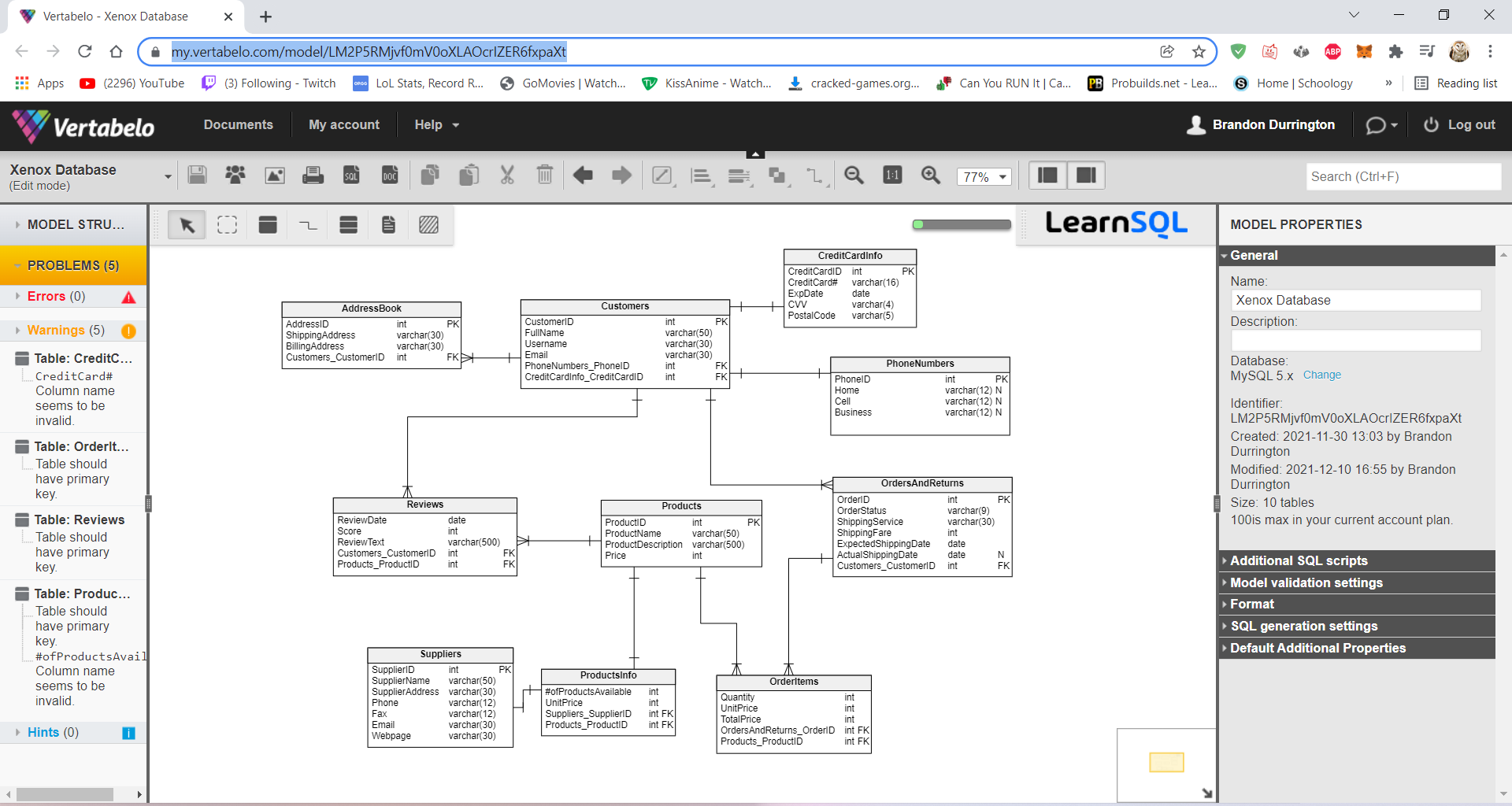
**Xenon**

A futuristic American retail cooperation that sells products of future origin to consumers, it’s a new cooperation investing in the high demand of new and unique items that are used from extraterrestrial life that have been discovered and have a limited supply of.

-Brandon Durrington

Design:

I wanted to make something different that would seem more interesting but still is able to apply to a regular day to day database so I just built on the fact of the items of the store being much more advanced with futuristic-like names. The database is pretty simple but is based on how happy the customers are with these new invented items so I went more into the review/rating part (out of 10) of my database as well as meeting the data requirements.



Implementation:

CREATE TABLE AddressBook (

AddressID int NOT NULL,

ShippingAddress varchar(30) NOT NULL,

BillingAddress varchar(30) NOT NULL,

Customers\_CustomerID int NOT NULL,

CONSTRAINT AddressBook\_pk PRIMARY KEY (AddressID)

);

CREATE TABLE CreditCardInfo (

CreditCardID int NOT NULL,

CreditCard# varchar(16) NOT NULL,

ExpDate date NOT NULL,

CVV varchar(4) NOT NULL,

PostalCode varchar(5) NOT NULL,

CONSTRAINT CreditCardInfo\_pk PRIMARY KEY (CreditCardID)

);

CREATE TABLE Customers (

CustomerID int NOT NULL,

FullName varchar(50) NOT NULL,

Username varchar(30) NOT NULL,

Email varchar(30) NOT NULL,

PhoneNumbers\_PhoneID int NOT NULL,

CreditCardInfo\_CreditCardID int NOT NULL,

CONSTRAINT Customers\_pk PRIMARY KEY (CustomerID)

);

CREATE TABLE OrderItems (

Quantity int NOT NULL,

UnitPrice int NOT NULL,

TotalPrice int NOT NULL,

OrdersAndReturns\_OrderID int NOT NULL,

Products\_ProductID int NOT NULL

);

CREATE TABLE OrdersAndReturns (

OrderID int NOT NULL,

OrderStatus varchar(9) NOT NULL,

ShippingService varchar(30) NOT NULL,

ShippingFare int NOT NULL,

ExpectedShippingDate date NOT NULL,

ActualShippingDate date NULL,

Customers\_CustomerID int NOT NULL,

CONSTRAINT OrdersAndReturns\_pk PRIMARY KEY (OrderID)

);

CREATE TABLE PhoneNumbers (

PhoneID int NOT NULL,

Home varchar(12) NULL,

Cell varchar(12) NULL,

Business varchar(12) NULL,

CONSTRAINT PhoneNumbers\_pk PRIMARY KEY (PhoneID)

);

CREATE TABLE Products (

ProductID int NOT NULL,

ProductName varchar(50) NOT NULL,

ProductDescription varchar(500) NOT NULL,

Price int NOT NULL,

CONSTRAINT Products\_pk PRIMARY KEY (ProductID)

);

CREATE TABLE ProductsInfo (

#ofProductsAvailable int NOT NULL,

UnitPrice int NOT NULL,

Suppliers\_SupplierID int NOT NULL,

Products\_ProductID int NOT NULL

);

CREATE TABLE Reviews (

ReviewDate date NOT NULL,

Score int NOT NULL,

ReviewText varchar(500) NOT NULL,

Customers\_CustomerID int NOT NULL,

Products\_ProductID int NOT NULL

);

CREATE TABLE Suppliers (

SupplierID int NOT NULL,

SupplierName varchar(50) NOT NULL,

SupplierAddress varchar(30) NOT NULL,

Phone varchar(12) NOT NULL,

Fax varchar(12) NOT NULL,

Email varchar(30) NOT NULL,

Webpage varchar(30) NOT NULL,

CONSTRAINT Suppliers\_pk PRIMARY KEY (SupplierID)

);

--ALL RELATIONSHIPS

ALTER TABLE AddressBook ADD CONSTRAINT AddressBook\_Customers FOREIGN KEY (Customers\_CustomerID)

REFERENCES Customers (CustomerID);

ALTER TABLE Customers ADD CONSTRAINT Customers\_CreditCardInfo FOREIGN KEY (CreditCardInfo\_CreditCardID)

REFERENCES CreditCardInfo (CreditCardID);

ALTER TABLE Customers ADD CONSTRAINT Customers\_PhoneNumbers FOREIGN KEY (PhoneNumbers\_PhoneID)

REFERENCES PhoneNumbers (PhoneID);

ALTER TABLE OrderItems ADD CONSTRAINT OrderItems\_OrdersAndReturns FOREIGN KEY (OrdersAndReturns\_OrderID)

REFERENCES OrdersAndReturns (OrderID);

ALTER TABLE OrderItems ADD CONSTRAINT OrderItems\_Products FOREIGN KEY (Products\_ProductID)

REFERENCES Products (ProductID);

ALTER TABLE OrdersAndReturns ADD CONSTRAINT OrdersAndReturns\_Customers FOREIGN KEY (Customers\_CustomerID)

REFERENCES Customers (CustomerID);

ALTER TABLE ProductsInfo ADD CONSTRAINT ProductsInfo\_Products FOREIGN KEY (Products\_ProductID)

REFERENCES Products (ProductID);

ALTER TABLE ProductsInfo ADD CONSTRAINT ProductsInfo\_Suppliers FOREIGN KEY (Suppliers\_SupplierID)

REFERENCES Suppliers (SupplierID);

ALTER TABLE Reviews ADD CONSTRAINT Reviews\_Customers FOREIGN KEY (Customers\_CustomerID)

REFERENCES Customers (CustomerID);

ALTER TABLE Reviews ADD CONSTRAINT Reviews\_Products FOREIGN KEY (Products\_ProductID)

REFERENCES Products (ProductID);

^^^^^: This is when I created all of my tables and added the relationships between each of the Primary Keys and Foreign Keys of my database

INSERT INTO CreditCardInfo

(CreditCardID, CreditCard#, ExpDate, CVV, PostalCode)

VALUES(1, '4975442746192019', '2025-12-12', 6732, 19904);

INSERT INTO CreditCardInfo

(CreditCardID, CreditCard#, ExpDate, CVV, PostalCode)

VALUES(2, '6011682298379255', '2025-06-20', 9373, 19904);

INSERT INTO CreditCardInfo

(CreditCardID, CreditCard#, ExpDate, CVV, PostalCode)

VALUES(3, '373262568722685', '2023-04-01', 0124, 13210);

INSERT INTO CreditCardInfo

(CreditCardID, CreditCard#, ExpDate, CVV, PostalCode)

VALUES(4, '6011305592942665', '2026-09-15', 8773, 13210);

^^^^^: This is me inserting my CreditCardInfo users

INSERT INTO PhoneNumbers

(PhoneID, Home, Cell, Business)

VALUES(1, '323-758-4094', '940-403-4332', '913-254-2788');

INSERT INTO PhoneNumbers

(PhoneID, Home, Cell)

VALUES(2, '818-365-3573', '401-265-6997');

INSERT INTO PhoneNumbers

(PhoneID, Home, Cell, Business)

VALUES(3, '870-280-6030', '630-422-1386', '312-423-7918');

INSERT INTO PhoneNumbers

(PhoneID, Cell, Business)

VALUES(4, '330-497-9827', '641-705-4463');

^^^^^: This is me adding the phone numbers

INSERT INTO Customers

(CustomerID, FullName, Username, Email, PhoneNumbers\_PhoneID, CreditCardInfo\_CreditCardID)

VALUES(1, 'Richard Gibson', 'Gibbinator', 'Gibbinator@gmail.com', 1, 1);

INSERT INTO Customers

(CustomerID, FullName, Username, Email, PhoneNumbers\_PhoneID, CreditCardInfo\_CreditCardID)

VALUES(2, 'Billy Scheider', 'sillybilly', 'sillybilly@gmail.com', 2, 2);

INSERT INTO Customers

(CustomerID, FullName, Username, Email, PhoneNumbers\_PhoneID, CreditCardInfo\_CreditCardID)

VALUES(3, 'Jaun Garcia', 'geezyman', 'Garcia221@gmail.com', 3, 3);

INSERT INTO Customers

(CustomerID, FullName, Username, Email, PhoneNumbers\_PhoneID, CreditCardInfo\_CreditCardID)

VALUES(4, 'Gwen Flavor', 'flavormon', 'monmon23@gmail.com', 4, 4);

^^^^^: This is where I added all of the customers I would be using

INSERT INTO AddressBook

(AddressID, ShippingAddress, BillingAddress, Customers\_CustomerID)

VALUES(1, '56 Lopney Ave', '56 Lopney Ave', 1);

INSERT INTO AddressBook

(AddressID, ShippingAddress, BillingAddress, Customers\_CustomerID)

VALUES(2, '473 Rocky Rd', '473 Rocky Rd', 2);

INSERT INTO AddressBook

(AddressID, ShippingAddress, BillingAddress, Customers\_CustomerID)

VALUES(3, '419 Bethany Dr', '789 Hungor Dr', 3);

INSERT INTO AddressBook

(AddressID, ShippingAddress, BillingAddress, Customers\_CustomerID)

VALUES(4, '800 Grungy Dr', '800 Grungy Dr', 4);

^^^^^: This is all of the addresses of the customers

INSERT INTO Products

(ProductID, ProductName, ProductDescription, Price)

VALUES(1, 'Lozno 3.0', 'This is a type of shaver that shaves every

hair on body from head to toe.', 415.99);

INSERT INTO Products

(ProductID, ProductName, ProductDescription, Price)

VALUES(2, 'Zaber Saw', 'This saw will cut virtually anything! From paper

to stain steel metal!', 217.99);

INSERT INTO Products

(ProductID, ProductName, ProductDescription, Price)

VALUES(3, 'Guben Coat', 'Most effective jacket at blocking every particle

of wind coming your way and will block all moisture so you will never

go cold again!', 323.99);

INSERT INTO Products

(ProductID, ProductName, ProductDescription, Price)

VALUES(4, 'Zeon AAA Battery', 'New AAA battery made from Zeon material

that will never die and will always be able to power your needs!', 109.99);

INSERT INTO Products

(ProductID, ProductName, ProductDescription, Price)

VALUES(5, 'Copium Moisturizer', 'This new moisturizer will soften your skin

to the absolute softest level there is to being soft.', 99.99);

INSERT INTO Products

(ProductID, ProductName, ProductDescription, Price)

VALUES(6, 'Mimio Fingil Boomerang', 'A boomerang hardwired by alien-like technology

that will come back to you no matter where you are in the universe where ever its been thrown.', 217.99);

^^^^^: All of the new products that Xenon now sells

INSERT INTO Reviews

(ReviewDate, Score, ReviewText, Customers\_CustomerID, Products\_ProductID)

VALUES('2020-06-12', '10', 'Its both attractive and highly adaptable. I would

also like to say thank you to all your staff. Best. Product. Ever!', 1, 6);

INSERT INTO Reviews

(ReviewDate, Score, ReviewText, Customers\_CustomerID, Products\_ProductID)

VALUES('2021-02-11', '9', 'Man, this thing is getting better and better as I

learn more about it. Its really wonderful. I made back the purchase price in just 48 hours!', 1, 3);

INSERT INTO Reviews

(ReviewDate, Score, ReviewText, Customers\_CustomerID, Products\_ProductID)

VALUES('2020-10-01', '1', 'Absolute garbage. Did not shave everything like it promised. Never coming back.', 4, 1);

INSERT INTO Reviews

(ReviewDate, Score, ReviewText, Customers\_CustomerID, Products\_ProductID)

VALUES('2020-07-20', '6', 'The Copium Moisturizer definitely works and softens my skin! But...

I feel like it could be softer.', 3, 5);

^^^^^: All of the reviews, some of my customers made multiple reviews and some made none

INSERT INTO OrdersAndReturns

(OrderID, OrderStatus, ShippingService, ShippingFare, ExpectedShippingDate, ActualShippingDate, Customers\_CustomerID)

VALUES(1, 'Ready', 'USPS', 7, '2022-01-07', '2022-01-10', 2);

INSERT INTO OrdersAndReturns

(OrderID, OrderStatus, ShippingService, ShippingFare, ExpectedShippingDate, Customers\_CustomerID)

VALUES(2, 'Shipped', 'FedEx', 9, '2022-02-07', 3);

INSERT INTO OrdersAndReturns

(OrderID, OrderStatus, ShippingService, ShippingFare, ExpectedShippingDate, Customers\_CustomerID)

VALUES(3, 'Delivered', 'FedEx', 9, '2022-03-07', 1);

^^^^^: Made the list of OrdersAndReturns and since it’s a new place only a small of people are buying items

INSERT INTO OrderItems

(Quantity, UnitPrice, TotalPrice, OrdersAndReturns\_OrderID, Products\_ProductID)

VALUES(1, 415.99, 415.99, 1, 1);

INSERT INTO OrderItems

(Quantity, UnitPrice, TotalPrice, OrdersAndReturns\_OrderID, Products\_ProductID)

VALUES(3, 109.99, 329.97, 2, 4);

INSERT INTO OrderItems

(Quantity, UnitPrice, TotalPrice, OrdersAndReturns\_OrderID, Products\_ProductID)

VALUES(2, 99.99, 199.98, 3, 5);

^^^^^: This is where I denote the exact contents of the order and specifically what was ordered

INSERT INTO Suppliers

(SupplierID, SupplierName, SupplierAddress, Phone, Fax, Email, Webpage)

VALUES(1, 'The Zone', '8275 Pennington St', '740-309-3131', '614-863-5083', 'TheZone@gmail.com', 'TheZone.com');

INSERT INTO Suppliers

(SupplierID, SupplierName, SupplierAddress, Phone, Fax, Email, Webpage)

VALUES(2, 'Bungda', '992 Joy Ridge St', '217-522-8108', '206-200-6203', 'Bungda@hotmal.com', 'Bungda.org');

INSERT INTO Suppliers

(SupplierID, SupplierName, SupplierAddress, Phone, Fax, Email, Webpage)

VALUES(3, 'Hush', '735 Brickyard Lane', '914-450-2155', '574-527-3878', 'Hush@gmail.com', 'Hush.org');

INSERT INTO Suppliers

(SupplierID, SupplierName, SupplierAddress, Phone, Fax, Email, Webpage)

VALUES(4, 'Lockton Inc.', '7853 N. Illinois Rd', '703-966-9518', '717-840-9603', 'Lockton@yahoo.com', 'Lockton.com');

^^^^^: All of my suppliers that give us our products!

INSERT INTO ProductsInfo

(#ofProductsAvailable, UnitPrice, Suppliers\_SupplierID, Products\_ProductID)

VALUES(10, 400, 1, 1);

INSERT INTO ProductsInfo

(#ofProductsAvailable, UnitPrice, Suppliers\_SupplierID, Products\_ProductID)

VALUES(30, 200, 1, 2);

INSERT INTO ProductsInfo

(#ofProductsAvailable, UnitPrice, Suppliers\_SupplierID, Products\_ProductID)

VALUES(22, 300, 2, 3);

INSERT INTO ProductsInfo

(#ofProductsAvailable, UnitPrice, Suppliers\_SupplierID, Products\_ProductID)

VALUES(57, 100, 4, 4);

INSERT INTO ProductsInfo

(#ofProductsAvailable, UnitPrice, Suppliers\_SupplierID, Products\_ProductID)

VALUES(34, 75, 3, 5);

INSERT INTO ProductsInfo

(#ofProductsAvailable, UnitPrice, Suppliers\_SupplierID, Products\_ProductID)

VALUES(18, 200, 3, 6);

^^^^^: Here is all the info for my projects relative to my suppliers

CREATE VIEW CustomersReviews AS

SELECT Cust.CustomerID, Cust.FullName, Prod.ProductName,

Rev.ReviewDate, Rev.Score, Rev.ReviewText

FROM Customers Cust, Products Prod, Reviews Rev

WHERE Cust.CustomerID = Prod.ProductID;

^^^^^: Here is my first view showing the customers reviews

CREATE VIEW CustomersAddress AS

SELECT Cust.CustomerID, Cust.FullName, Addr.ShippingAddress,

Addr.BillingAddress

FROM Customers Cust, AddressBook Addr

WHERE Cust.CustomerID = Addr.AddressID;

^^^^^: Here is the second view showing who lives where and their billing address

CREATE VIEW SuppliersProducts AS

SELECT Supp.SupplierID, Supp.SupplierName, ProdIn.#ofProductsAvailable

FROM Suppliers Supp, ProductsInfo ProdIn

WHERE Supp.SupplierID = ProdIn.Suppliers\_SupplierID;

^^^^^: Third view showing the suppliers products left

CREATE VIEW CustomersOrderStatus AS

SELECT Cust.CustomerID, Cust.FullName, OrdRet.OrderStatus

FROM Customers Cust, OrdersAndReturns OrdRet

WHERE Cust.CustomerID = OrdRet.Customers\_CustomerID;

^^^^^: Fourth view showing the satus of customers orders

CREATE PROC spDelawareCustomers

AS

SELECT FullName, PostalCode

FROM Customers JOIN CreditCardInfo

ON CreditCardInfo.CreditCardID =

Customers.CreditCardInfo\_CreditCardID

WHERE PostalCode = 19904

ORDER BY Fullname;

^^^^^: first procedure was able to get all Delaware customers using the Delaware zip code 19904

CREATE PROC spCustomerCardExpiration

AS

SELECT FullName, ExpDate

FROM Customers JOIN CreditCardInfo

ON CreditCardInfo.CreditCardID =

Customers.CreditCardInfo\_CreditCardID

WHERE (4 > DATEDIFF(year, GETDATE(), ExpDate))

ORDER BY Fullname;

^^^^^: Second procedure and was able to show whose care expires soon meaning in the next 4 years

CREATE PROC spKnownShippingDate

AS

SELECT FullName, OrderStatus, ExpectedShippingDate, ActualShippingDate

FROM Customers JOIN OrdersAndReturns

ON Customers.CustomerID =

OrdersAndReturns.Customers\_CustomerID

WHERE ActualShippingDate IS NOT NULL

ORDER BY Fullname;

^^^^^: third procedure and was able to show which order a customer order has its shipping date set

CREATE PROC spBadReviews

AS

SELECT ProductName, ReviewDate, Score, Reviewtext

FROM Products JOIN Reviews

ON Products.ProductID =

Reviews.Products\_ProductID

WHERE Score <= 6

ORDER BY ProductName;

^^^^^: fourth procedure showing which product got a bad review which means 6 or less

CREATE FUNCTION fnNameSearcher

(@CustomerName varchar(50))

RETURNS int

BEGIN

RETURN (SELECT CustomerID FROM Customers

WHERE FullName = @CustomerName);

END;

^^^^^: first function that you can use to name search in the database

CREATE FUNCTION fnProductsAvailablePrice()

RETURNS int

BEGIN

RETURN (SELECT SUM(#ofProductsAvailable) FROM ProductsInfo);

END;

^^^^^: second function where you can et all of the prodicts available

CREATE TRIGGER Customers\_INSERT\_UPDATE

ON Customers

AFTER INSERT, UPDATE

AS

UPDATE Customers

SET FullName = UPPER(FullName)

WHERE CustomerID IN (SELECT CustomerID FROM Inserted);

GO

CREATE TRIGGER ProductsInfo\_INSERT\_UPDATE

ON ProductsInfo

AFTER INSERT, UPDATE

AS

UPDATE ProductsInfo

SET UnitPrice = ROUND(UnitPrice, -2)

WHERE

);

GO

CREATE TRIGGER Reviews\_INSERT\_UPDATE

ON Reviews

AFTER INSERT, UPDATE

AS

UPDATE Reviews

SET Score = ROUND(Score, 1));

GO

BEGIN TRANSACTION

UPDATE Products

SET Price = Price + 50

COMMIT

GO

BEGIN TRANSACTION

UPDATE ProductsInfo

SET #ofProductsAvailable = #ofProductsAvailable + 10

COMMIT

GO

BEGIN TRANSACTION

UPDATE ProductsInfo

SET UnitPrice = UnitPrice + 100

COMMIT

GO

BEGIN TRANSACTION

UPDATE ProductsInfo

SET UnitPrice = UnitPrice - 100

COMMIT

CREATE ROLE Xenon\_Relations;

GRANT UPDATE

ON Customers to Xenon\_Relations;

GRANT INSERT

ON CreditCardInfo TO Xenon\_Relations;

ALTER ROLE Xenon\_Relations ADD MEMBER JuliaSusans;

CREATE ROLE Xenon\_CustomerService;

GRANT INSERT, UPDATE

ON Customers to Xenon\_CustomerService;

GRANT INSERT, UPDATE

ON CreditCardInfo TO Xenon\_CustomerService;

GRANT INSERT, UPDATE

ON PhoneNumbers TO Xenon\_CustomerService;

GRANT INSERT, UPDATE

ON AddressBook TO Xenon\_CustomerService;

ALTER ROLE Xenon\_Relations ADD MEMBER GonnyHuger;

ALTER ROLE Xenon\_Relations ADD MEMBER NannySonson;

CREATE ROLE Xenon\_Manager;

GRANT INSERT, UPDATE

ON Customers to Xenon\_Manager;

GRANT INSERT, UPDATE

ON CreditCardInfo TO Xenon\_Manager;

GRANT INSERT, UPDATE

ON PhoneNumbers TO Xenon\_Manager;

GRANT INSERT, UPDATE

ON AddressBook TO Xenon\_Manager;

GRANT INSERT, UPDATE

ON OrdersAndReturns TO Xenon\_Manager;

GRANT INSERT, UPDATE

ON Products TO Xenon\_Manager;

GRANT INSERT, UPDATE

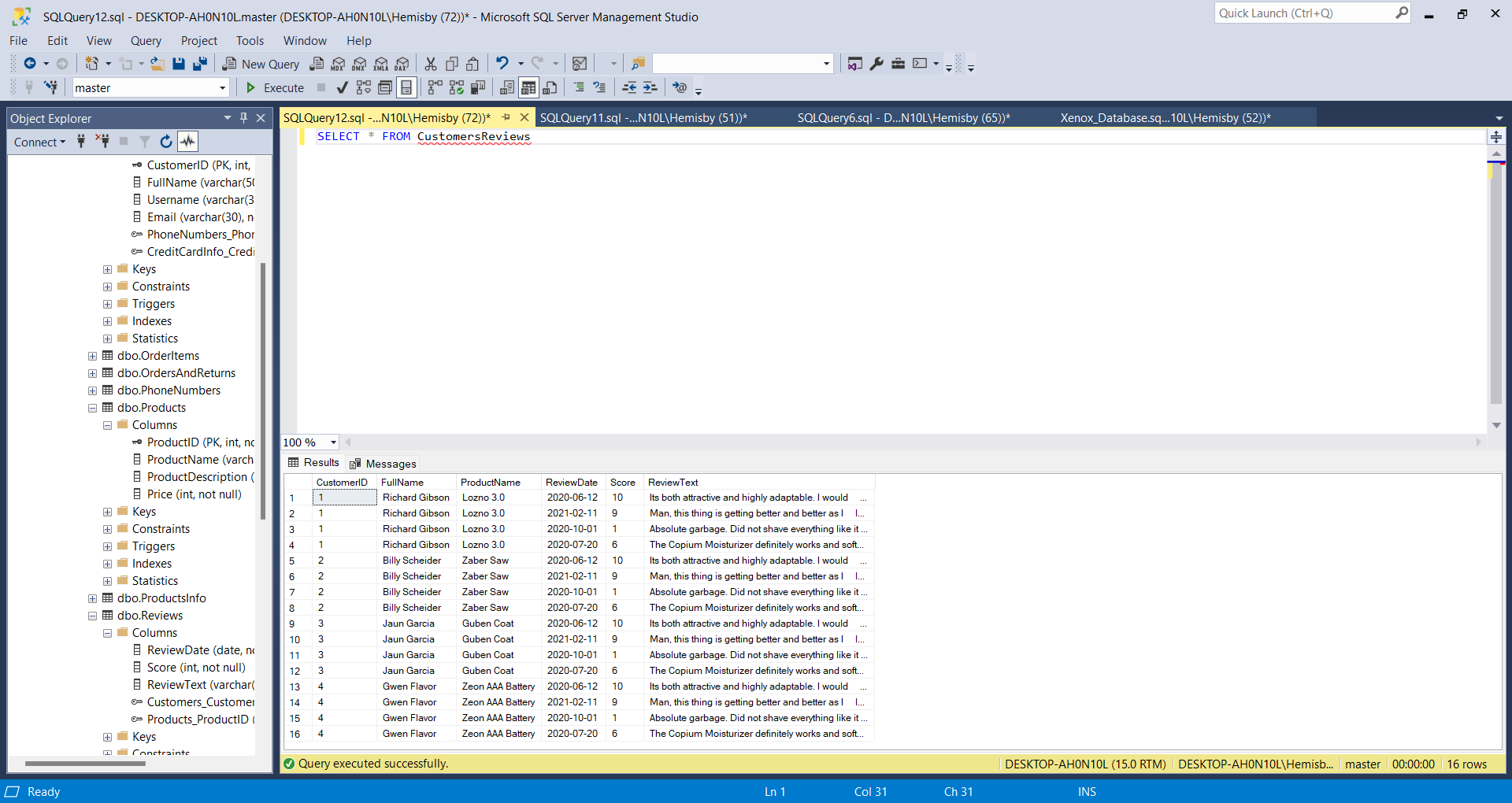
ON ProductsInfo TO Xenon\_Manager;

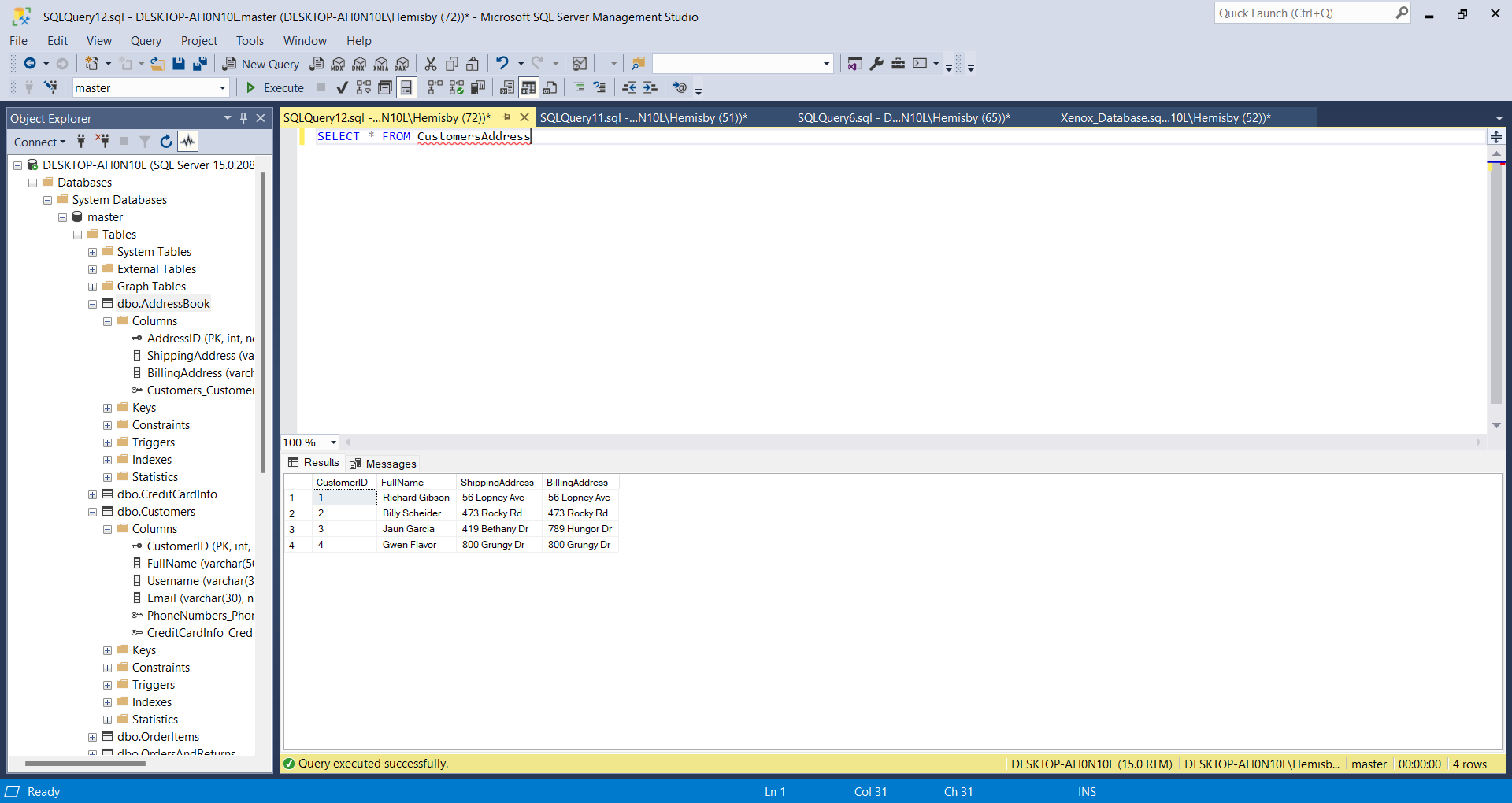
GRANT INSERT, UPDATE

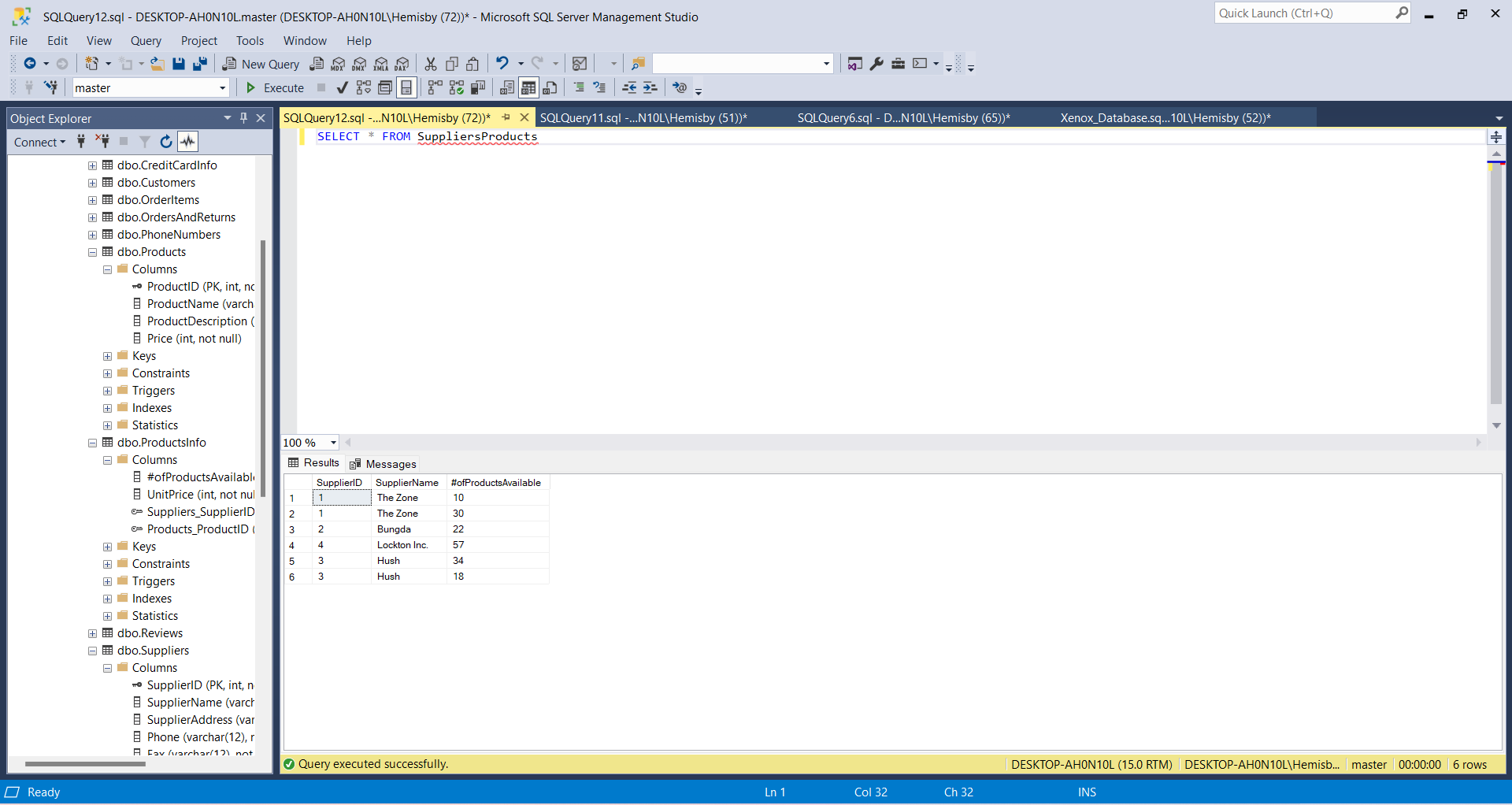
ON OrderItems TO Xenon\_Manager;

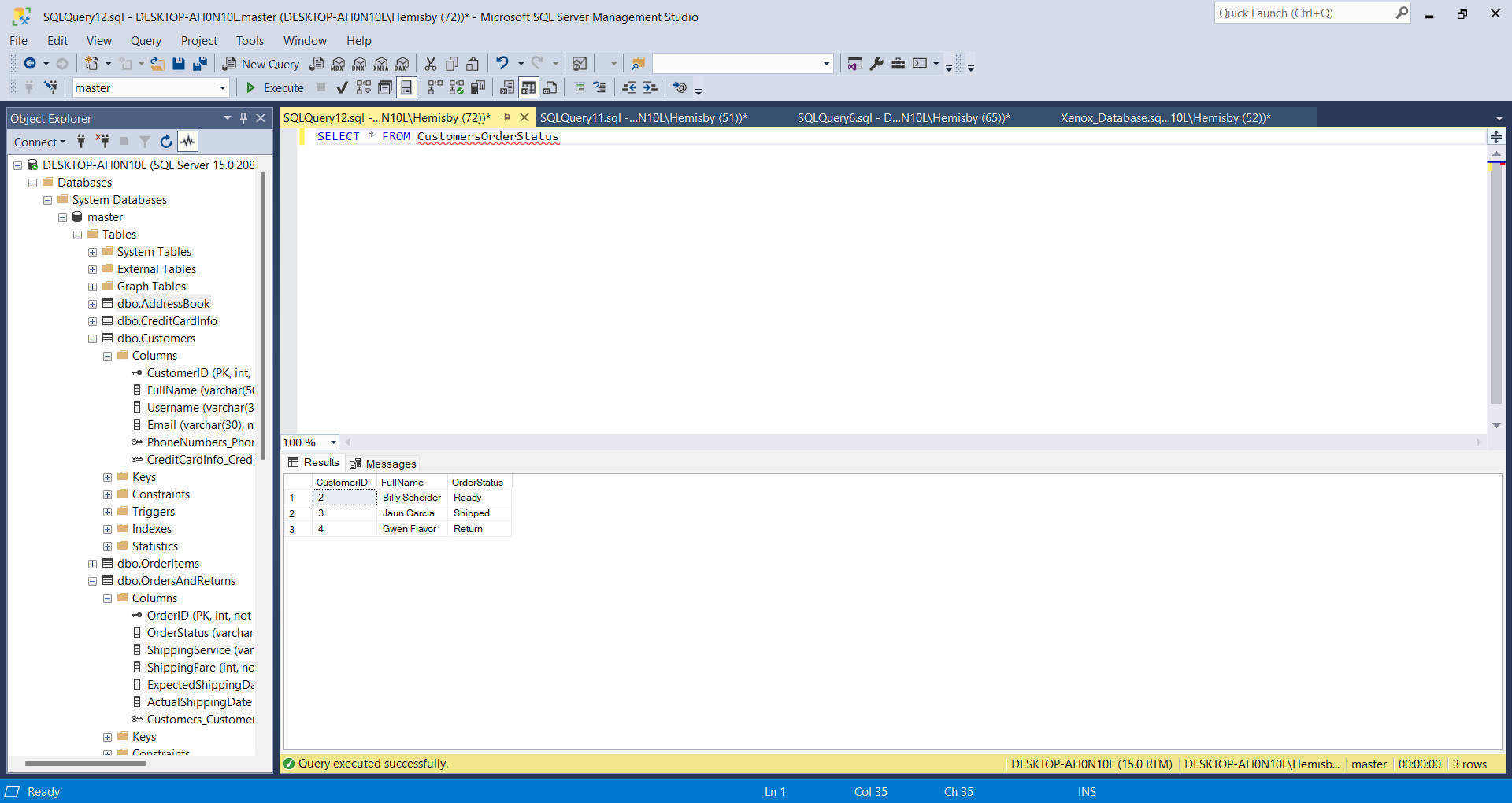
ALTER ROLE Xenon\_Manager ADD MEMBER BoboJoe;

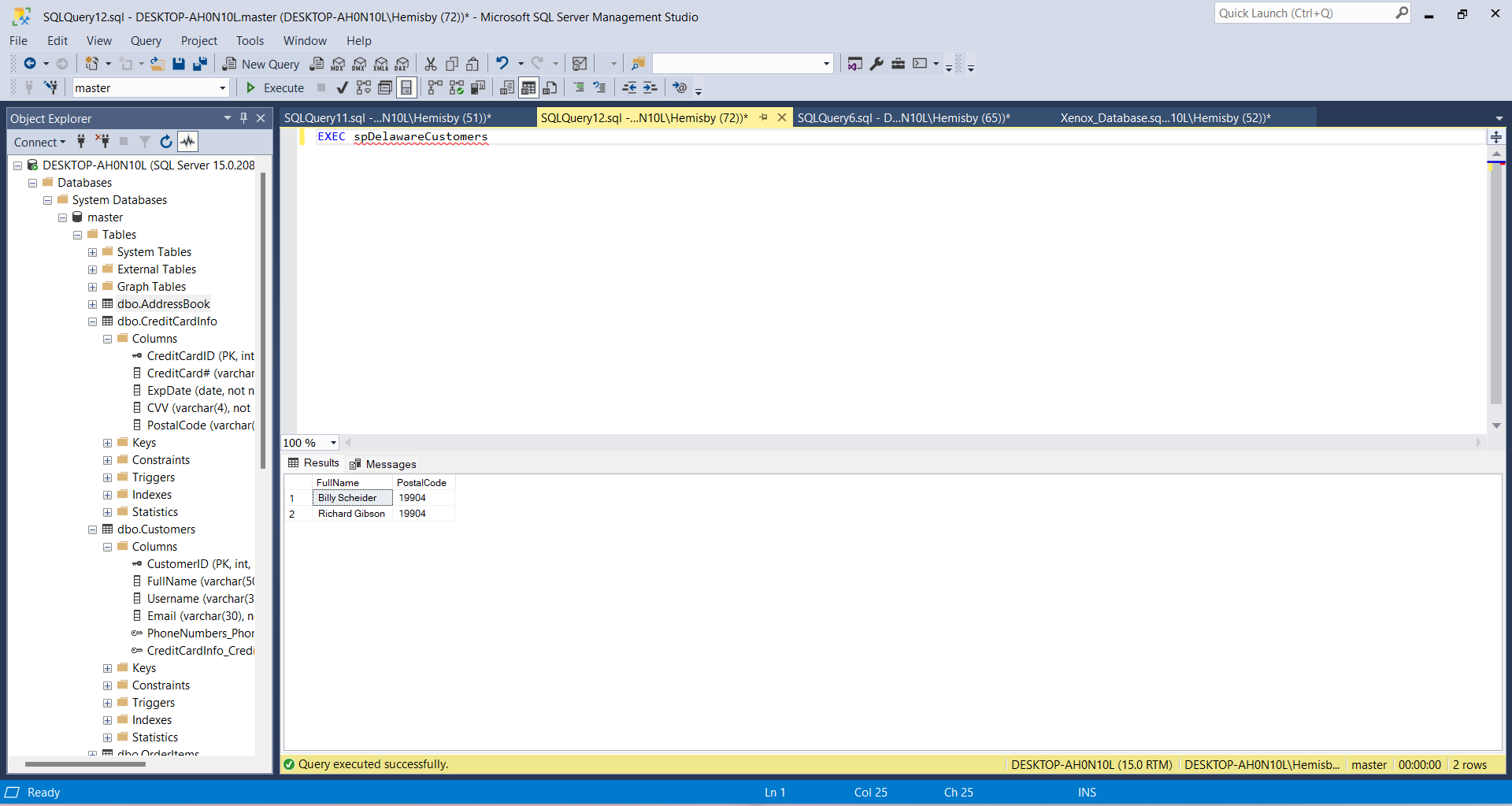
Testing:

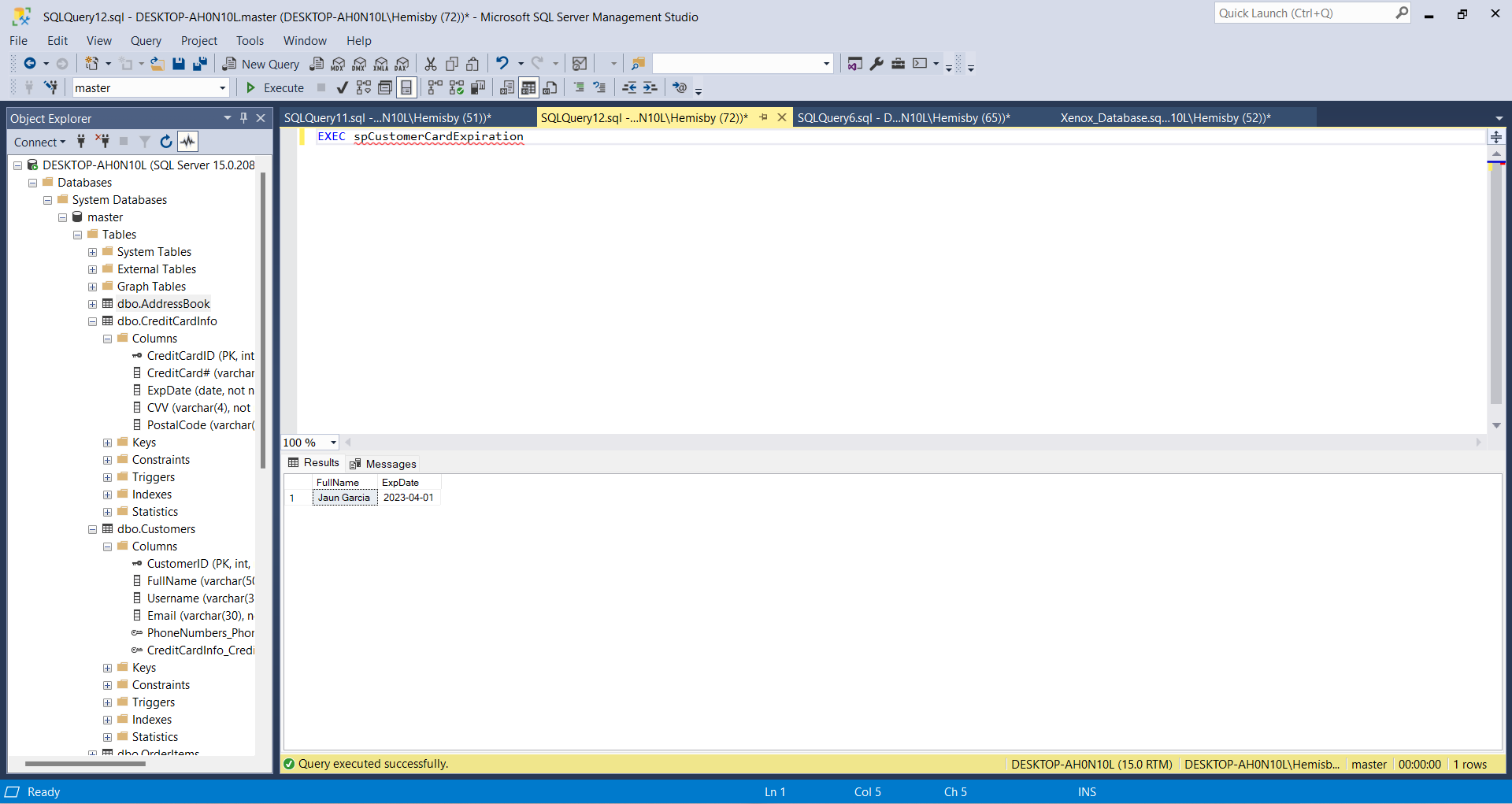


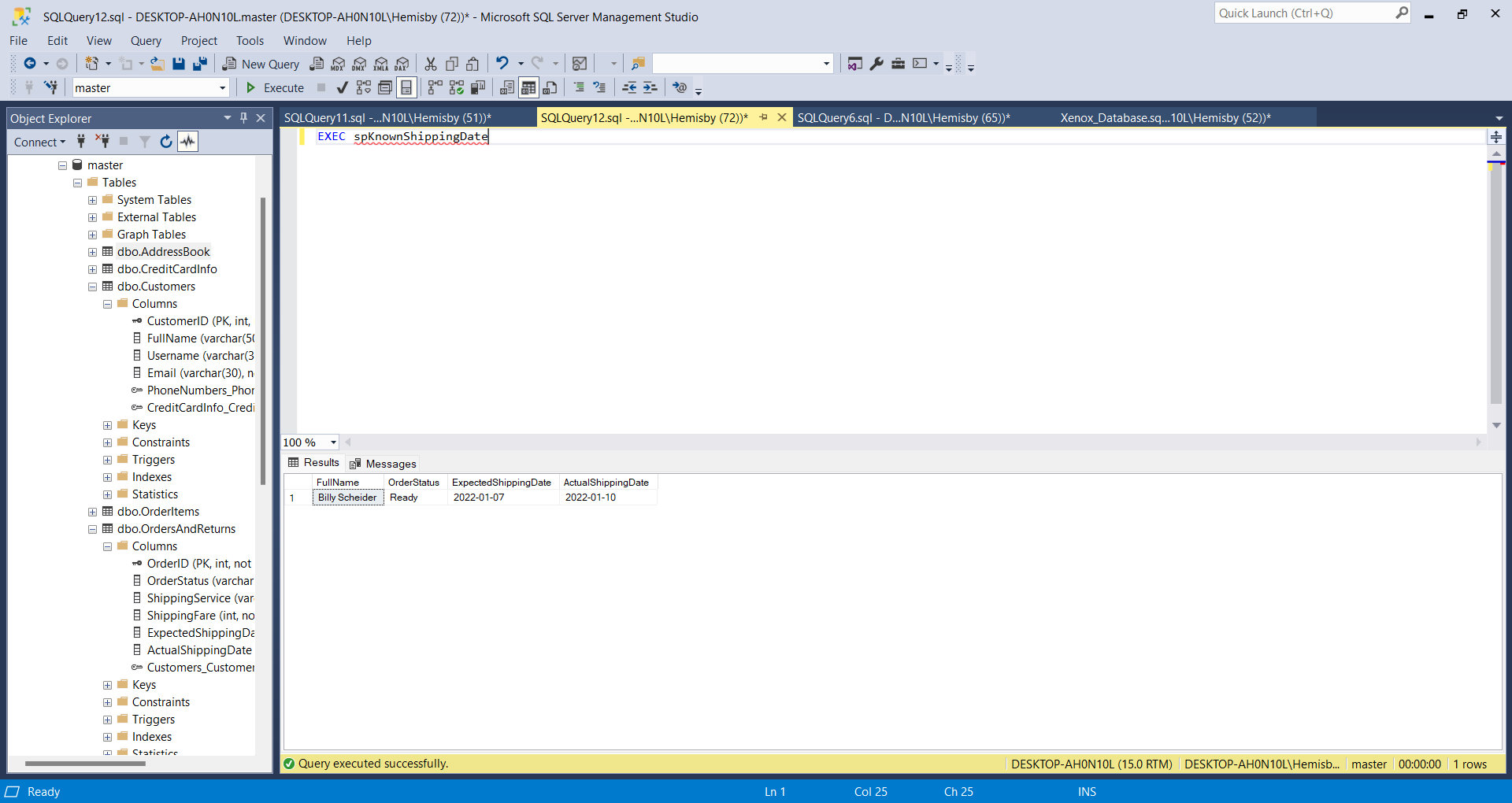


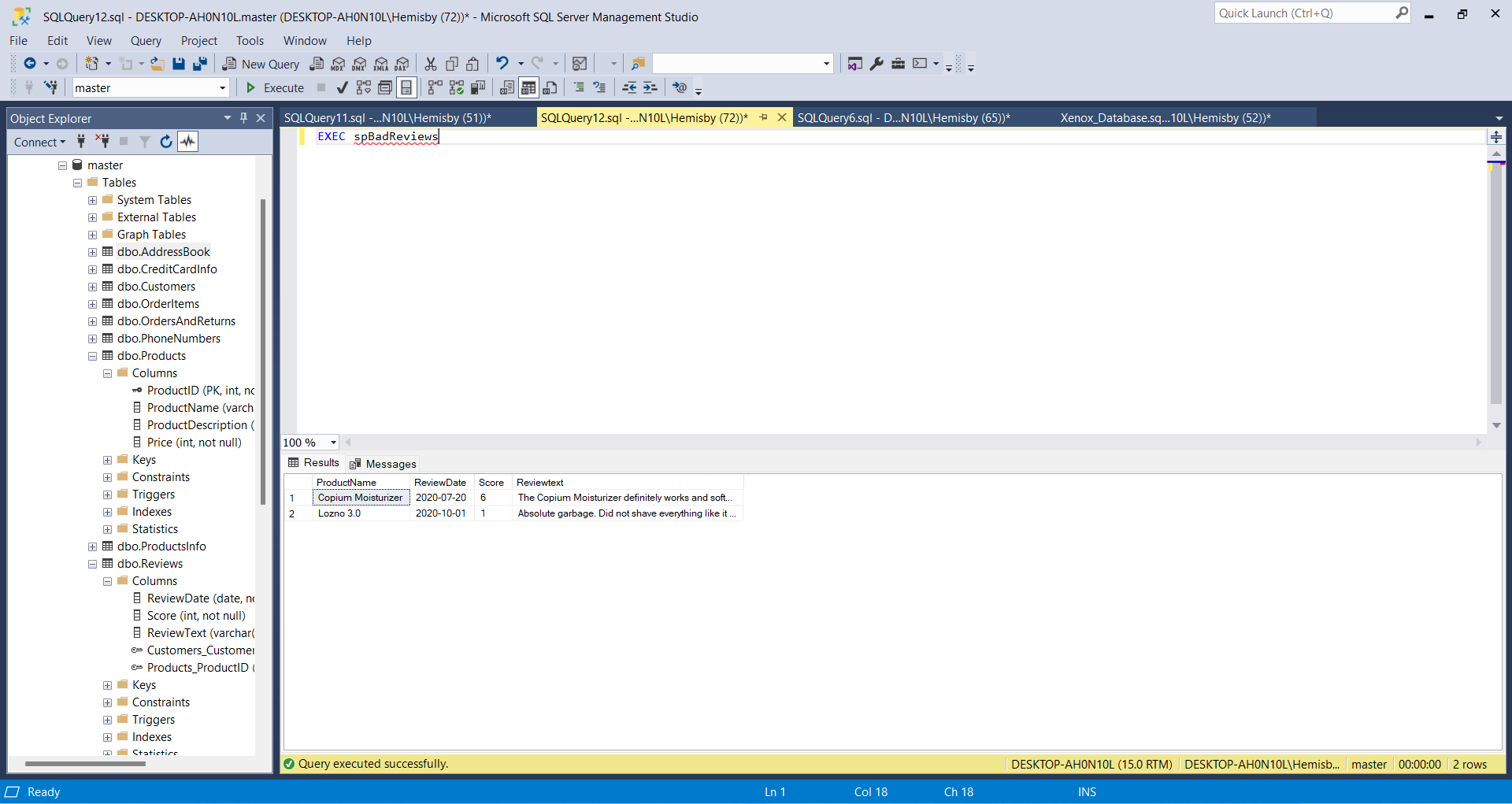


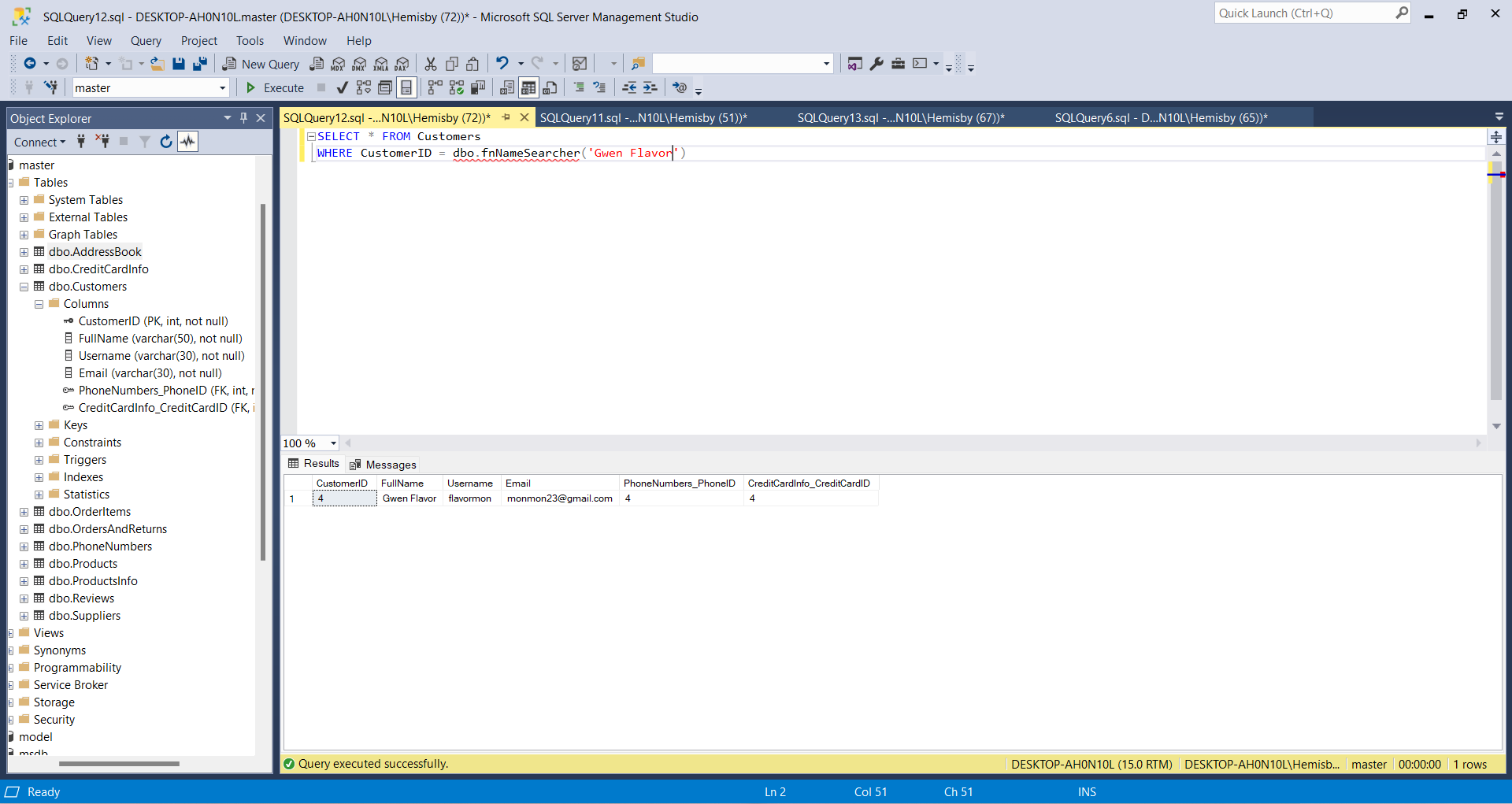


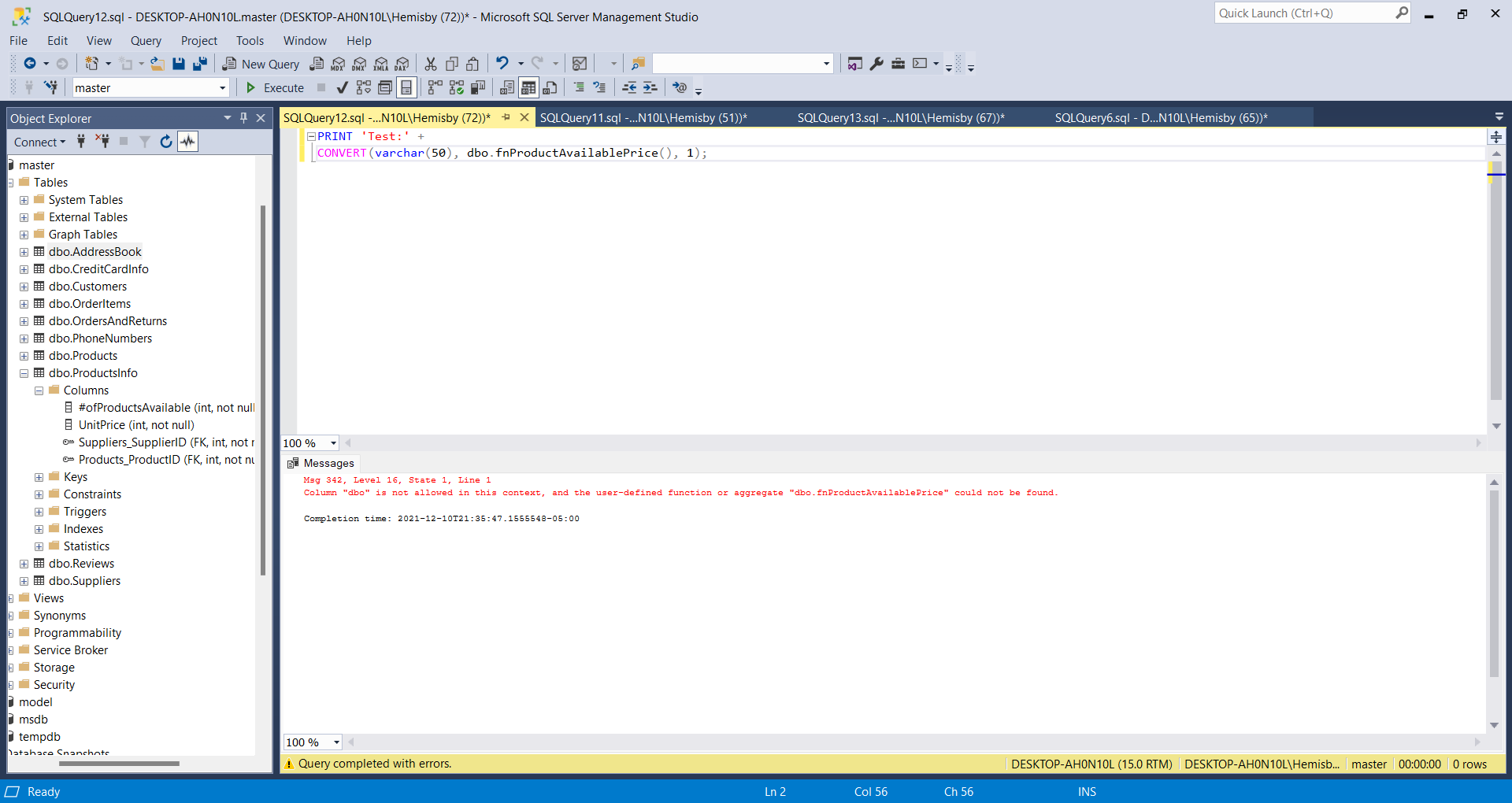












Conclusions:

I really enjoyed this project because it allowed me to learn a lot about how much work actually goes into a database and how satisfying it is when you complete it. Before I had absolutely no idea what I was doing at the beginning of the year now I am creating a whole database which is really exciting and learned a lot. Some stuff I do need to practice more because I am very interested n database and would love to get a job in database but not too sure if I will get a job so I do have to keep my options open. Overall, it took extremely long to figure how to do this the right way and took a lot of more time editing and inserting the data (more than I thought) but overall I enjoyed myself.

In actuality though Xenon is a newer company and I tried to show that with its lack of items and reviews but emphasized the futuristic origin of the products by explaining what they did in the product description. The functions and views I made so it will be easier to d things that will be done often like with the fnNameSearcher for example

Appendix:

