**1. ER Diagram Design**

**Database Name: MotorsCertification**

****

**2. Table Design and Data Insertion**

**--(i) Creating orderdetails table**

**CREATE TABLE orderdetails (**

**orderNumber INT PRIMARY KEY,**

**productCode VARCHAR(50),**

**quantityOrdered INT,**

**priceEach FLOAT,**

**orderLineNumber SMALLINT,**

**FOREIGN KEY (orderNumber) REFERENCES orders(orderNumber),**

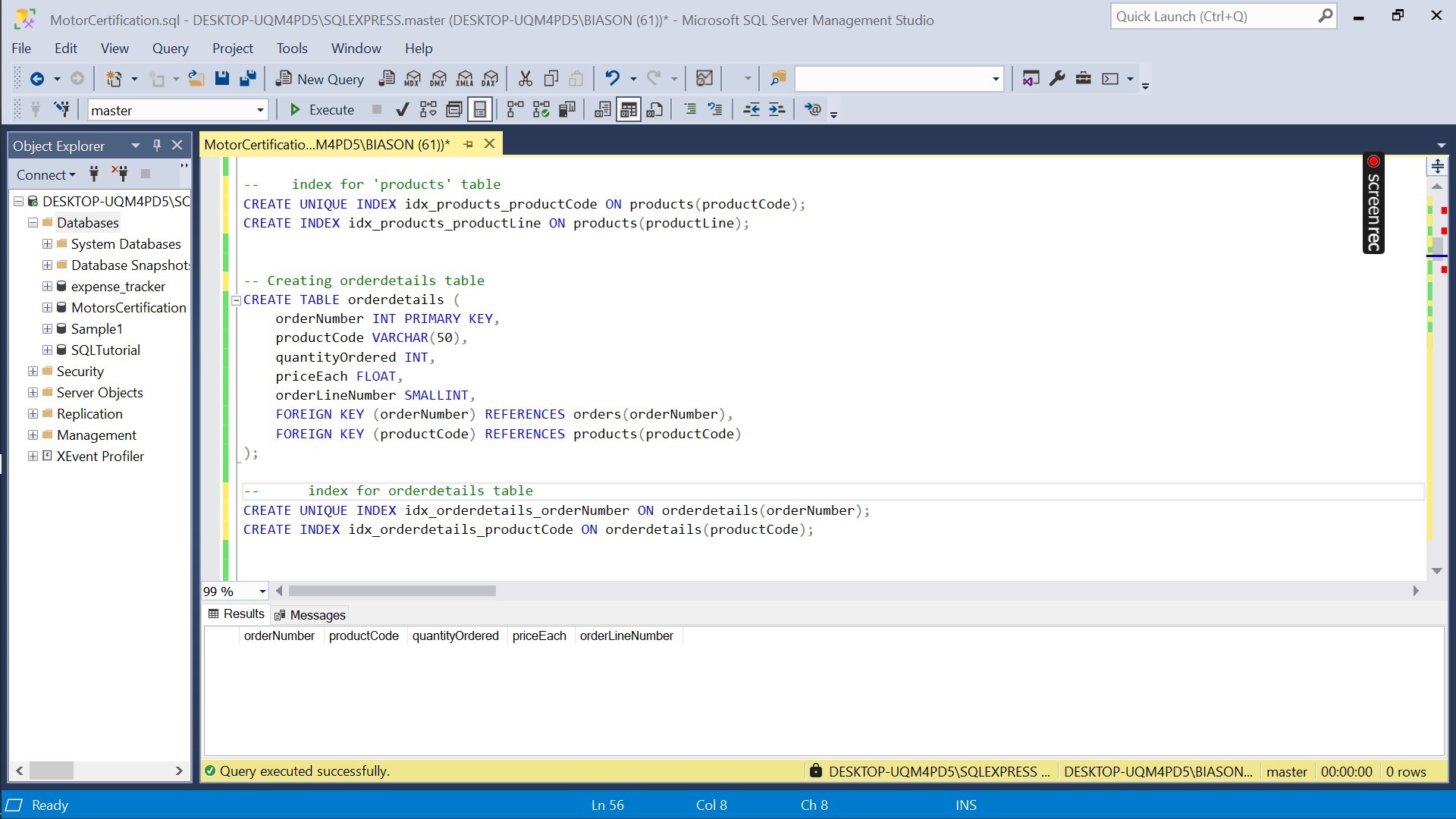
**FOREIGN KEY (productCode) REFERENCES products(productCode)**

**);**

**-- index for orderdetails table**

**CREATE UNIQUE INDEX idx\_orderdetails\_orderNumber ON orderdetails(orderNumber);**

**CREATE INDEX idx\_orderdetails\_productCode ON orderdetails(productCode);**



**--(ii) Creating customers table**

**CREATE TABLE customers (**

**customerNumber INT PRIMARY KEY,**

**customerName VARCHAR(50),**

**contactLastName VARCHAR(50),**

**contactFirstName VARCHAR(50),**

**phone VARCHAR(50),**

**addressLine1 VARCHAR(50),**

**addressLine2 VARCHAR(50),**

**city VARCHAR(50),**

**state VARCHAR(50),**

**postalCode VARCHAR(15),**

**country VARCHAR(50),**

**salesRepEmployeeNumber INT,**

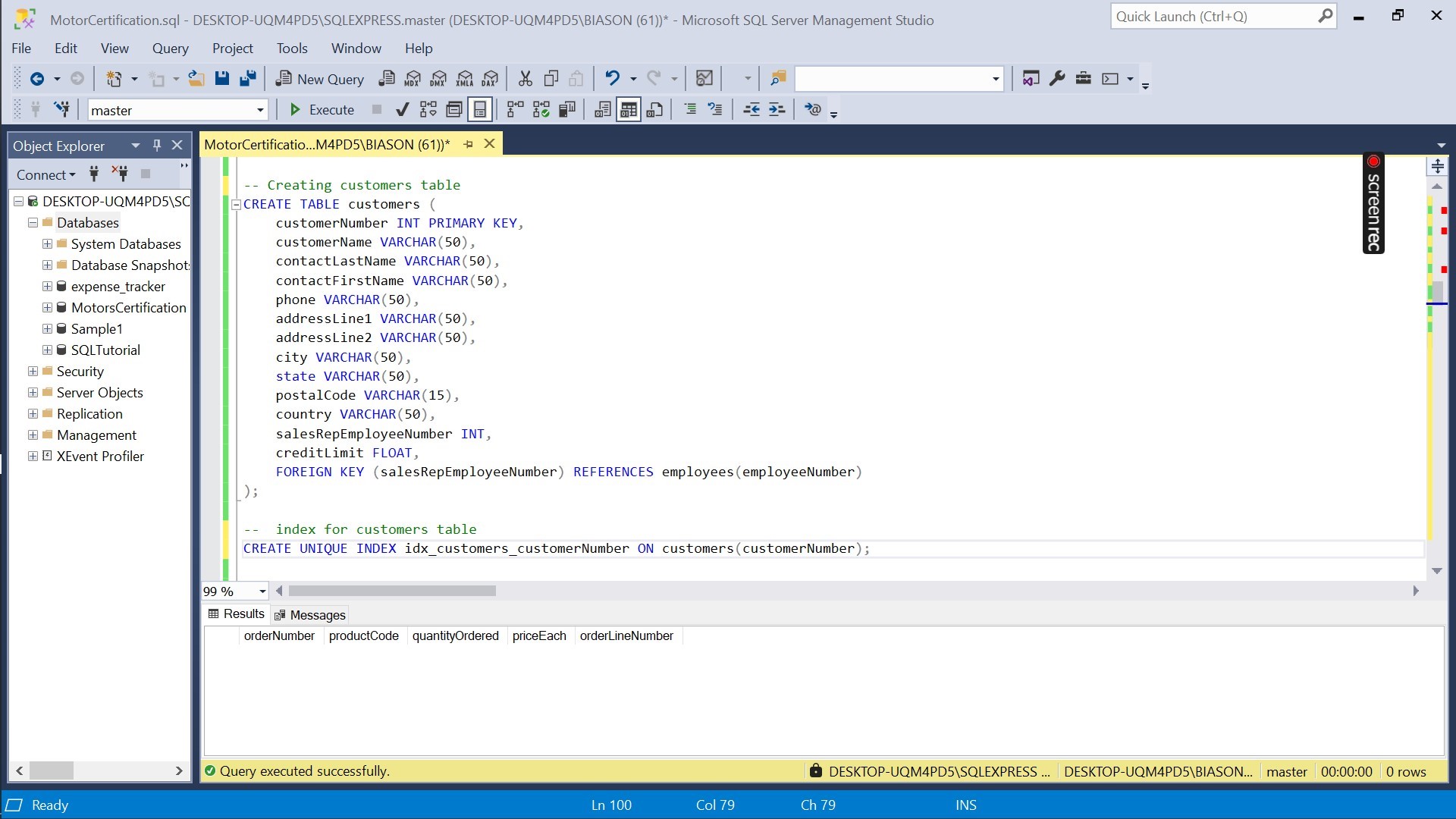
**creditLimit FLOAT,**

**FOREIGN KEY (salesRepEmployeeNumber) REFERENCES employees(employeeNumber)**

**);**

**-- index for customers table**

**CREATE UNIQUE INDEX idx\_customers\_customerNumber ON customers(customerNumber);**

****

**-- (iii) Creating payments table**

**CREATE TABLE payments (**

**customerNumber INT,**

**checkNumber VARCHAR(50),**

**paymentDate DATE,**

**amount FLOAT,**

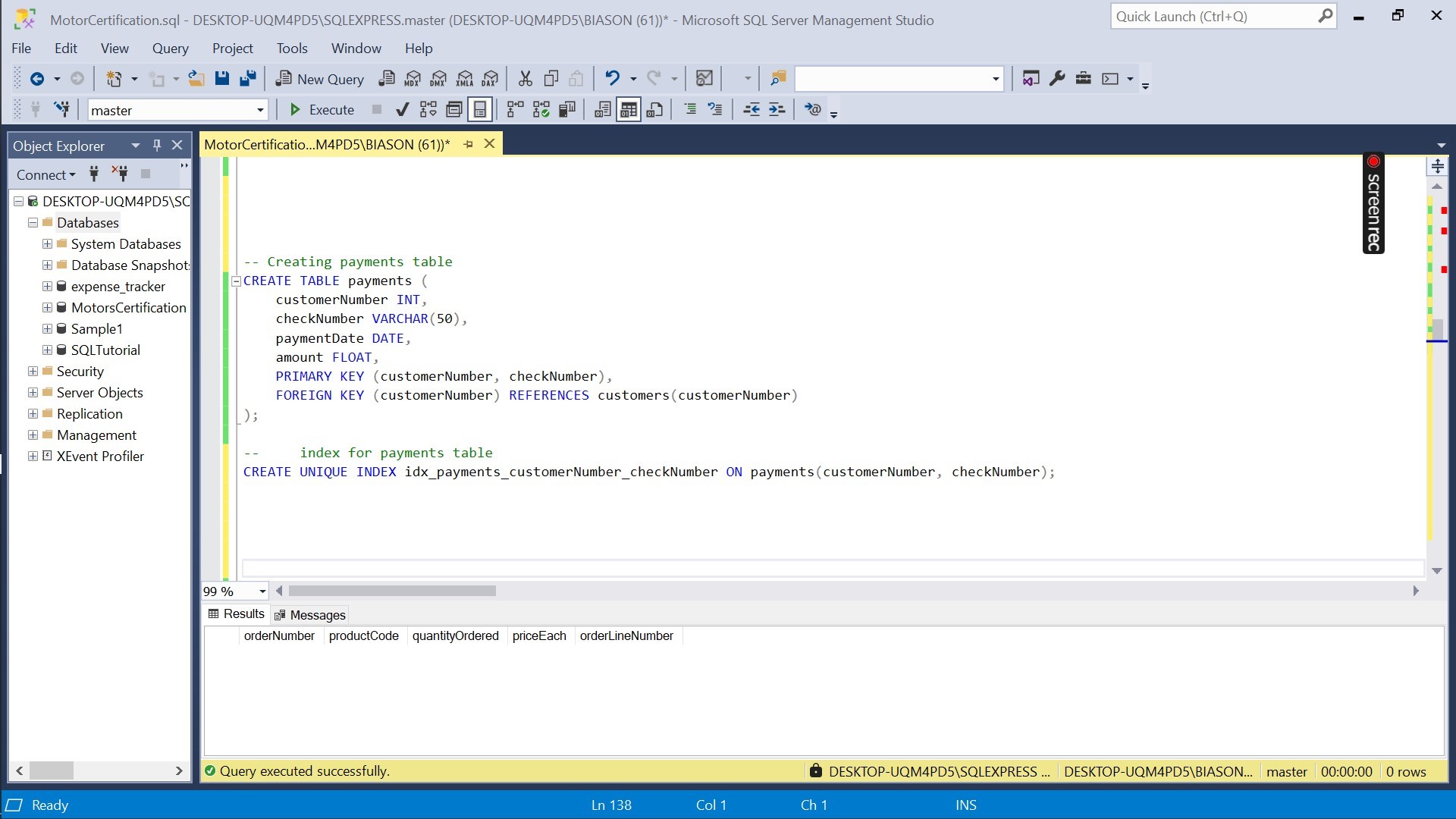
**PRIMARY KEY (customerNumber, checkNumber),**

**FOREIGN KEY (customerNumber) REFERENCES customers(customerNumber)**

**);**

**-- index for payments table**

**CREATE UNIQUE INDEX idx\_payments\_customerNumber\_checkNumber ON payments(customerNumber, checkNumber);**

****

**--(iv) Creating products table**

**CREATE TABLE products (**

**productCode VARCHAR(50) PRIMARY KEY,**

**productName VARCHAR(100),**

**productLine VARCHAR(50),**

**productScale VARCHAR(50),**

**productVendor VARCHAR(50),**

**productDescription TEXT,**

**quantityInStock SMALLINT,**

**buyPrice FLOAT,**

**MSRP FLOAT,**

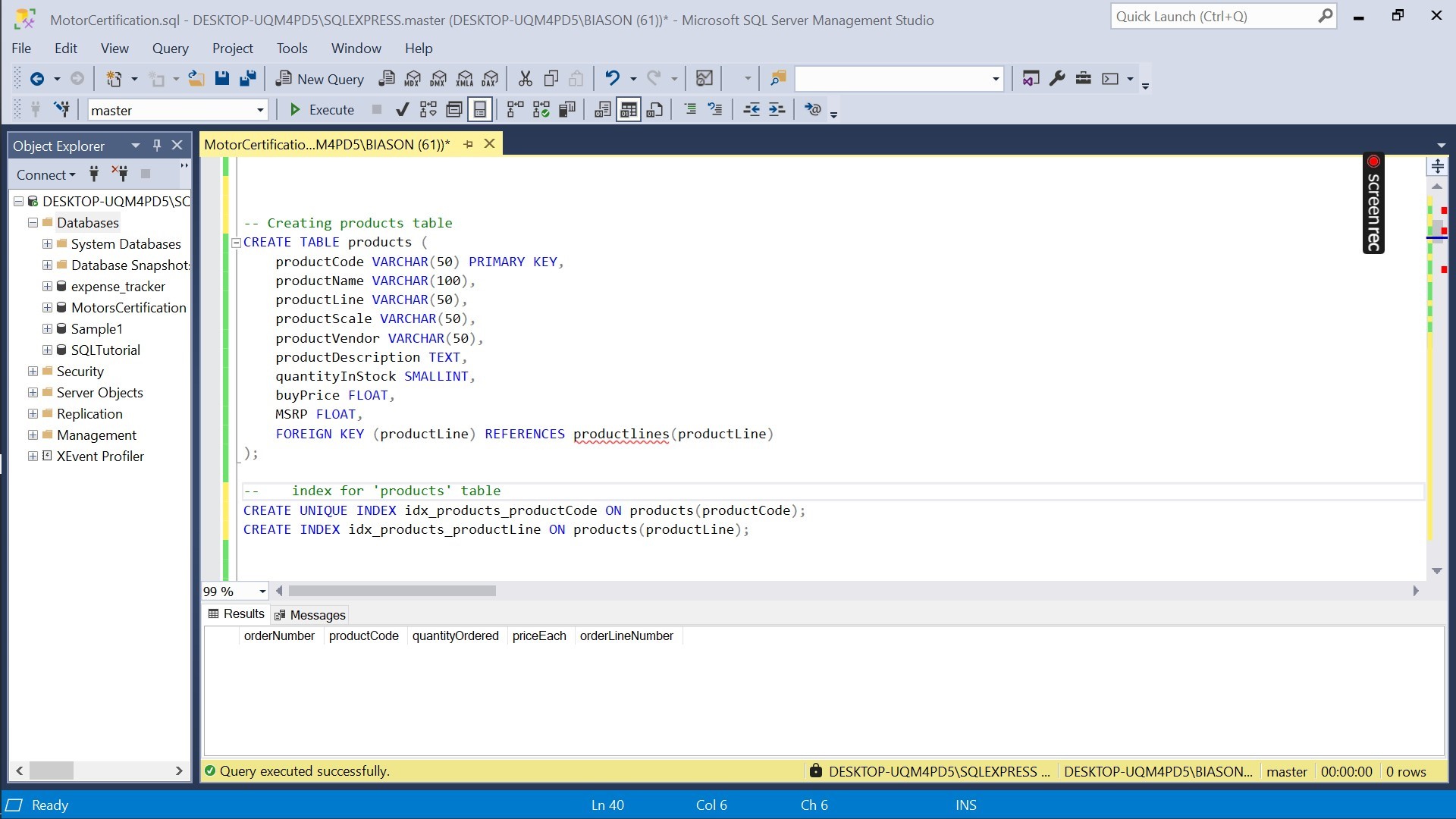
**FOREIGN KEY (productLine) REFERENCES productlines(productLine)**

**);**

**-- index for 'products' table**

**CREATE UNIQUE INDEX idx\_products\_productCode ON products(productCode);**

**CREATE INDEX idx\_products\_productLine ON products(productLine);**

****

**-- (v) Creating employees table**

**CREATE TABLE employees (**

**employeeNumber INT PRIMARY KEY,**

**lastName VARCHAR(50),**

**firstName VARCHAR(50),**

**extension VARCHAR(10),**

**email VARCHAR(100),**

**officeCode VARCHAR(10),**

**reportsTo INT,**

**jobTitle VARCHAR(50),**

**FOREIGN KEY (reportsTo) REFERENCES employees(employeeNumber),**

**FOREIGN KEY (officeCode) REFERENCES offices(officeCode)**

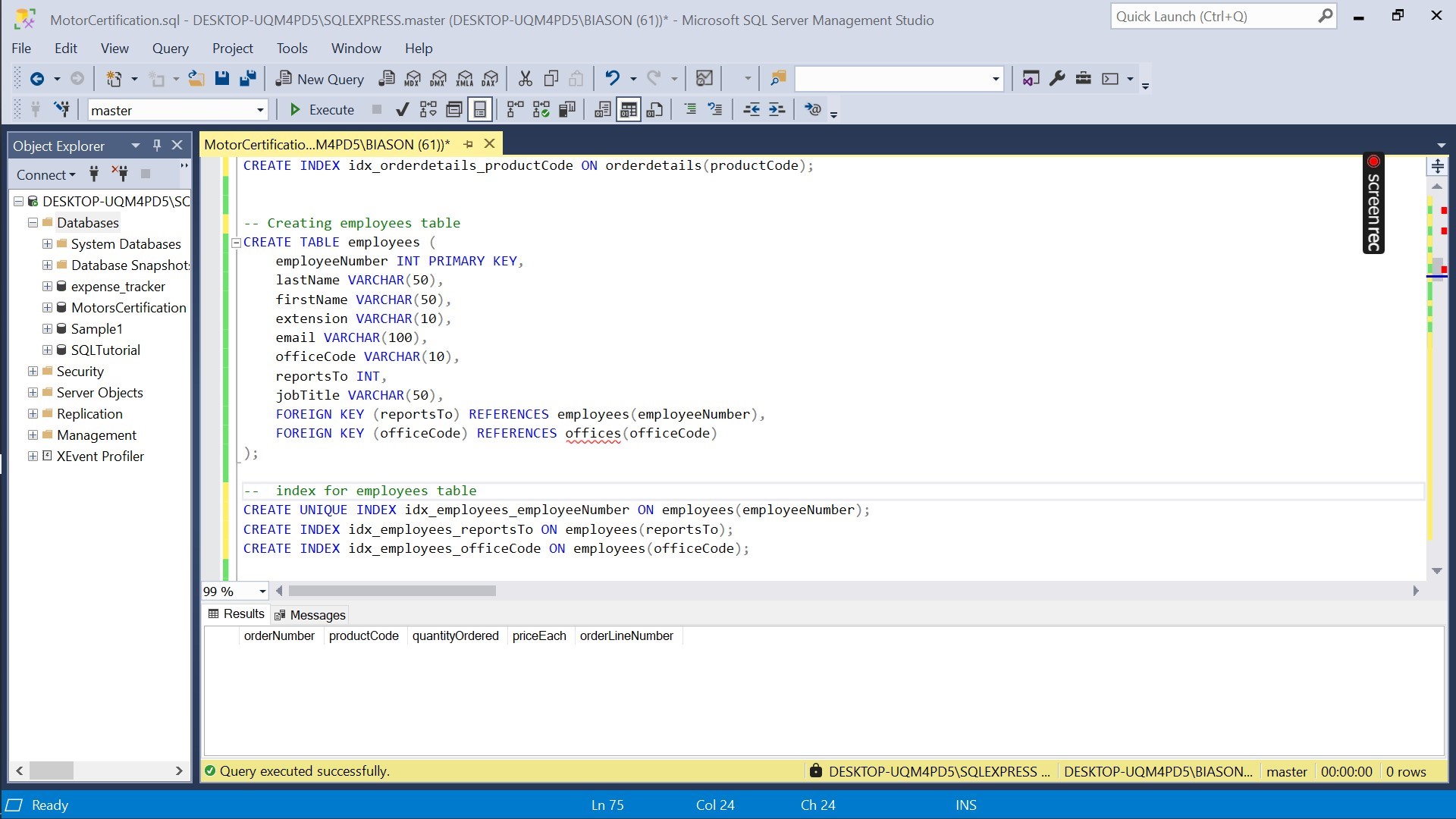
**);**

**-- index for employees table**

**CREATE UNIQUE INDEX idx\_employees\_employeeNumber ON employees(employeeNumber);**

**CREATE INDEX idx\_employees\_reportsTo ON employees(reportsTo);**

**CREATE INDEX idx\_employees\_officeCode ON employees(officeCode);**

****

**--(vi) Creating offices table**

**CREATE TABLE offices (**

**officeCode VARCHAR(10) PRIMARY KEY,**

**city VARCHAR(50),**

**phone VARCHAR(50),**

**addressLine1 VARCHAR(100),**

**addressLine2 VARCHAR(100),**

**state VARCHAR(50),**

**country VARCHAR(50),**

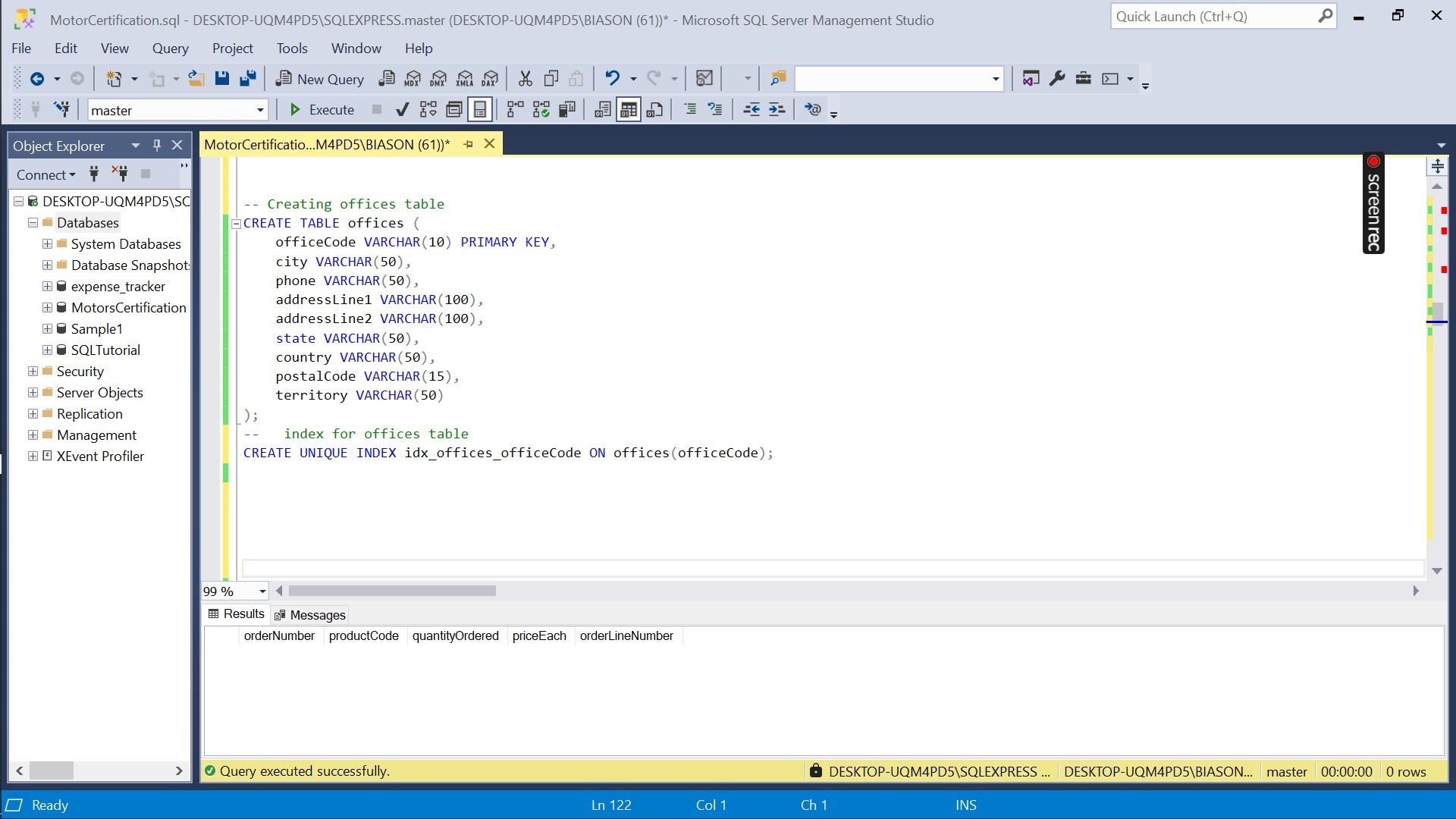
**postalCode VARCHAR(15),**

**territory VARCHAR(50)**

**);**

**-- index for offices table**

**CREATE UNIQUE INDEX idx\_offices\_officeCode ON offices(officeCode);**

****

**--(vii) Creating orders table**

**CREATE TABLE orders (**

**orderNumber INT PRIMARY KEY,**

**orderDate DATE,**

**requiredDate DATE,**

**shippedDate DATE,**

**status VARCHAR(50),**

**comments TEXT,**

**customerNumber INT,**

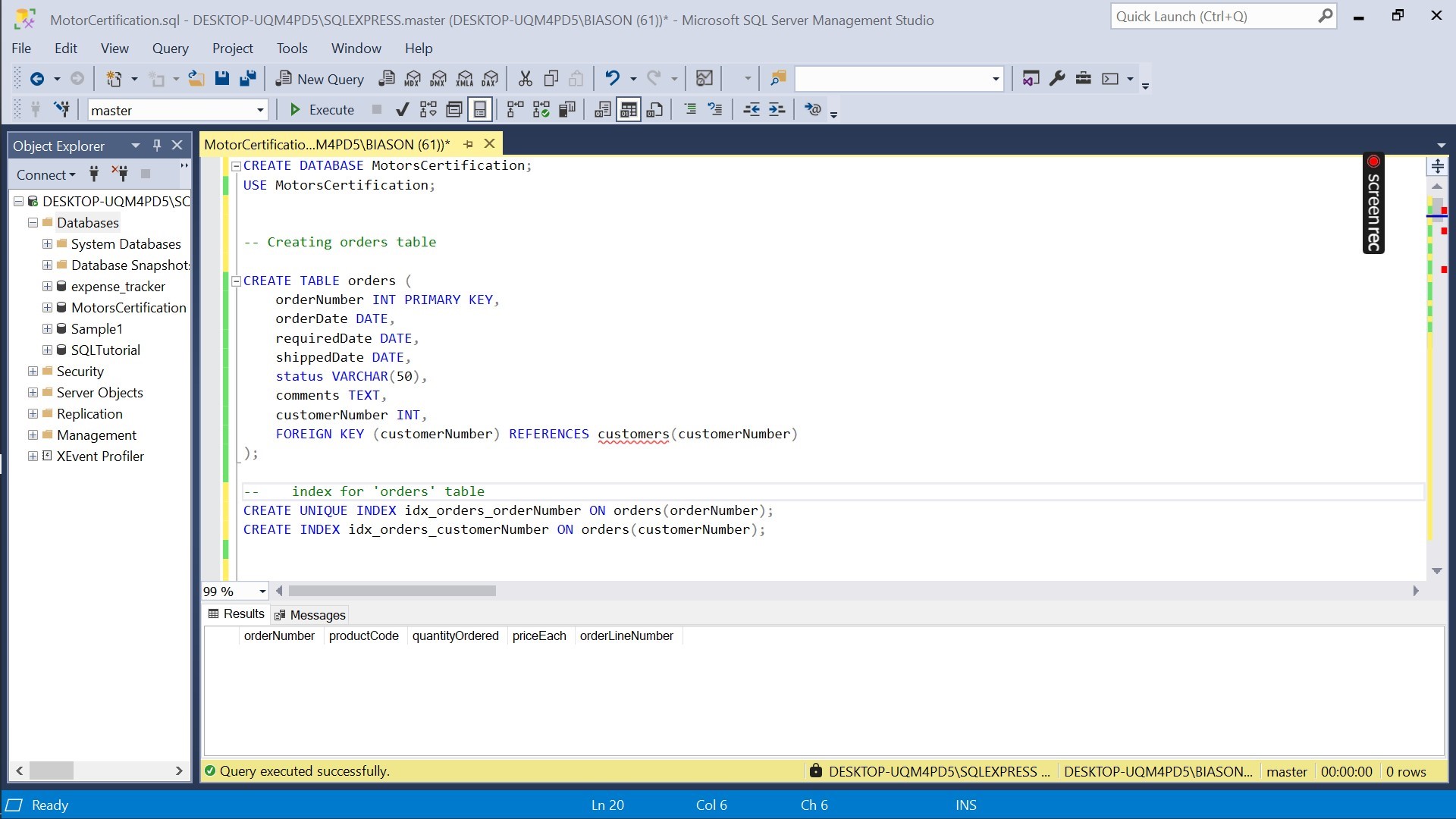
**FOREIGN KEY (customerNumber) REFERENCES customers(customerNumber)**

**);**

**-- index for 'orders' table**

**CREATE UNIQUE INDEX idx\_orders\_orderNumber ON orders(orderNumber);**

**CREATE INDEX idx\_orders\_customerNumber ON orders(customerNumber);**

****

**-- (viii) Creating productlines table**

**CREATE TABLE productlines (**

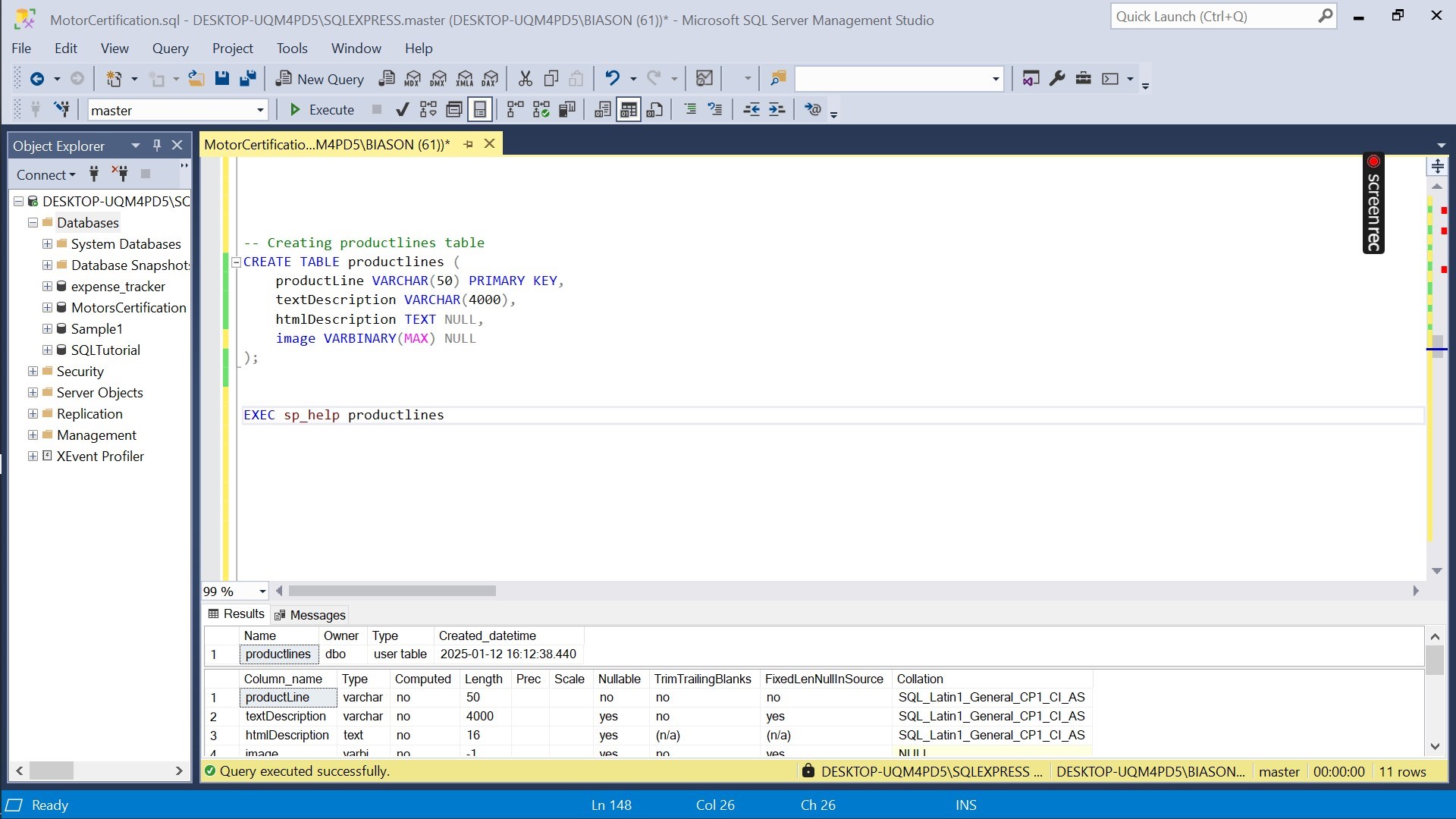
**productLine VARCHAR(50) PRIMARY KEY,**

**textDescription VARCHAR(4000),**

**htmlDescription TEXT NULL,**

**image VARBINARY(MAX) NULL**

**);**

****

**4. Delete the columns in productlines which are useless that do not infer anything.**

**SELECT \* FROM productlines; -- viewing the productlines table first**

**-- In my case, columns 'htmlDescription' and 'image' are not important since they have no values/items.**

**-- Deleting 'htmlDescription' and 'image'**

**ALTER TABLE productlines**

**DROP COLUMN htmlDescription, image;**

**-- verifying if the dropped columns are removed**

**SELECT \* FROM productlines;**

**5. Use a select statement to verify all insertions as well as updates.**

**-- Verifying data in 'orderdetails' table**

**SELECT \* FROM orderdetails;**

**-- Verifying data in 'employees' table**

**SELECT \* FROM employees;**

**-- Verifying data in 'payments' table**

**SELECT \* FROM payments;**

**-- Verifying data in 'products' table**

**SELECT \* FROM products;**

**-- Verifying data in 'customers' table**

**SELECT \* FROM customers;**

**-- Verifying data in 'offices' table**

**SELECT \* FROM offices;**

**-- Verifying data in 'orders' table**

**SELECT \* FROM orders;**

**6. Find out the highest and the lowest amount.**

**-- Find the highest amount in the 'payments' table**

**SELECT MAX(amount) AS HighestAmount FROM payments;**

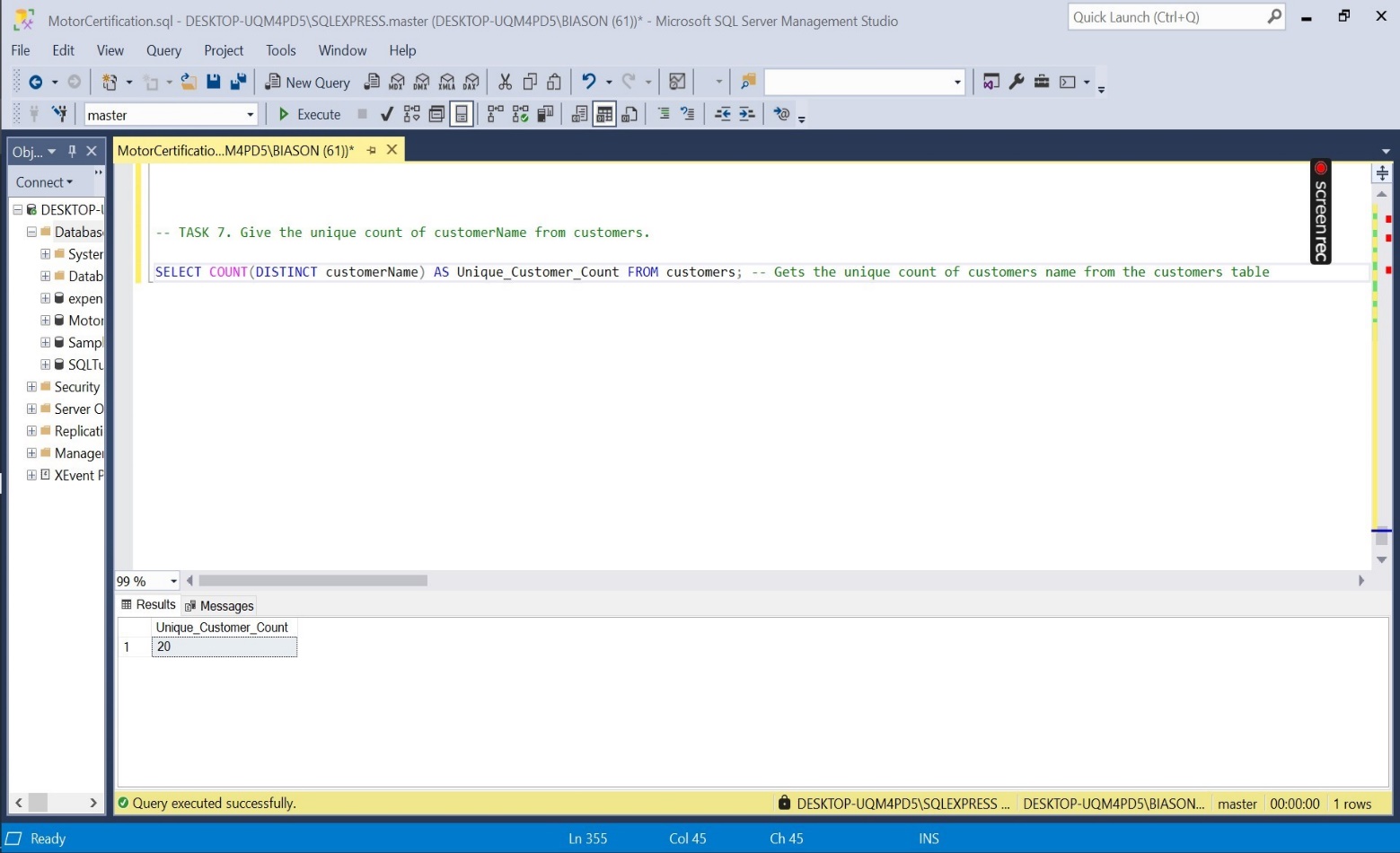
**-- Find the lowest amount in the 'payments' table**

**SELECT MIN(amount) AS LowestAmount FROM payments;**

**7. Give the unique count of customerName from customers.**

**-- TASK 7. Give the unique count of customerName from customers.**

**SELECT COUNT(DISTINCT customerName) AS Unique\_Customer\_Count FROM customers; -- Gets the unique count of customers name from the customers table**

****

**8. Create a view from customers and payments named cust\_payment and select customerName, amount, contactLastName, contactFirstName who have paid. Truncate and Drop the view after operation.**

**-- Create a view named 'cust\_payment'**

**CREATE VIEW cust\_payment AS**

**SELECT cst.customerName, py.amount, cst.contactLastName, cst.contactFirstName**

**FROM customers cst**

**JOIN payments py ON cst.customerNumber = py.customerNumber;**

**-- Select from the view to verify**

**SELECT \* FROM cust\_payment;**

**-- Truncate and drop the view after operation**

**DROP VIEW cust\_payment;**

**9. Create a stored procedure on products which displays productLine for Classic Cars.**

**-- TASK 9. Creating a Stored Procedure for products**

**-- Create a stored procedure to display productLine for Classic Cars**

**CREATE PROCEDURE GetClassicCars**

**AS**

**BEGIN**

**SELECT productLine**

**FROM products**

**WHERE productLine = 'Classic Cars';**

**END;**

**-- Executing the stored procedure**

**EXEC GetClassicCars;**

****

**10.Create a function to get the creditLimit of customers less than 96800**

**-- Creating a function to get creditLimit of customers less than 96800**

**CREATE FUNCTION FindCreditLimitsBelow96800()**

**RETURNS TABLE**

**AS**

**RETURN**

**(**

**SELECT customerNumber, customerName, creditLimit**

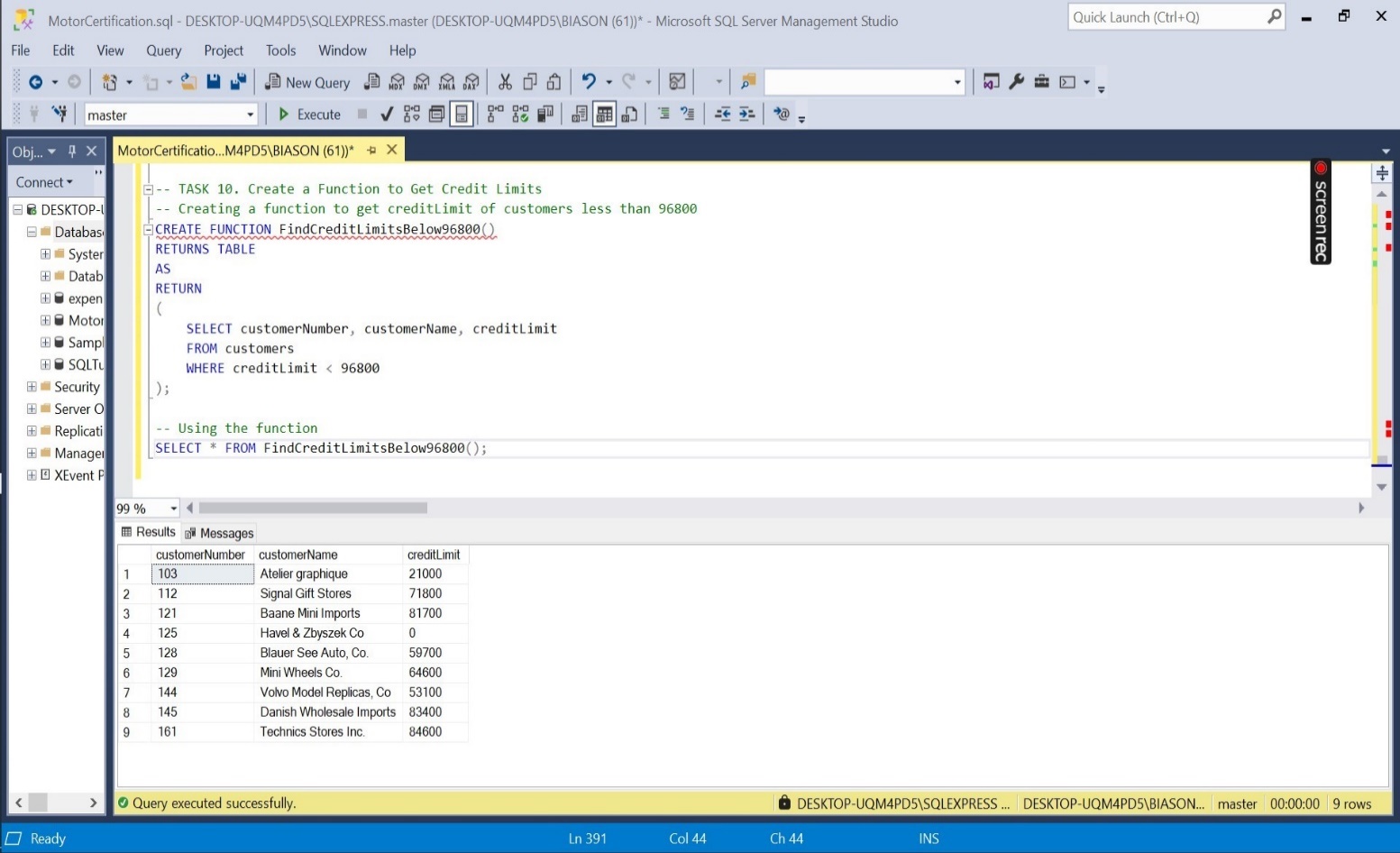
**FROM customers**

**WHERE creditLimit < 96800**

**);**

**-- Using the function**

**SELECT \* FROM FindCreditLimitsBelow96800();**

****

**-- TASK 11. Create Trigger to store transaction record for employee table which displays employeeNumber, lastName, FirstName**

**-- and office code upon insertion**

**-- Creating a trigger for insertion on the employees table**

**CREATE TRIGGER trig\_InsertEmployee**

**AFTER INSERT**

**ON employees**

**FOR EACH Row**

**BEGIN**

**-- Assuming there is a table named 'employeeTransactions' to store employee transaction records**

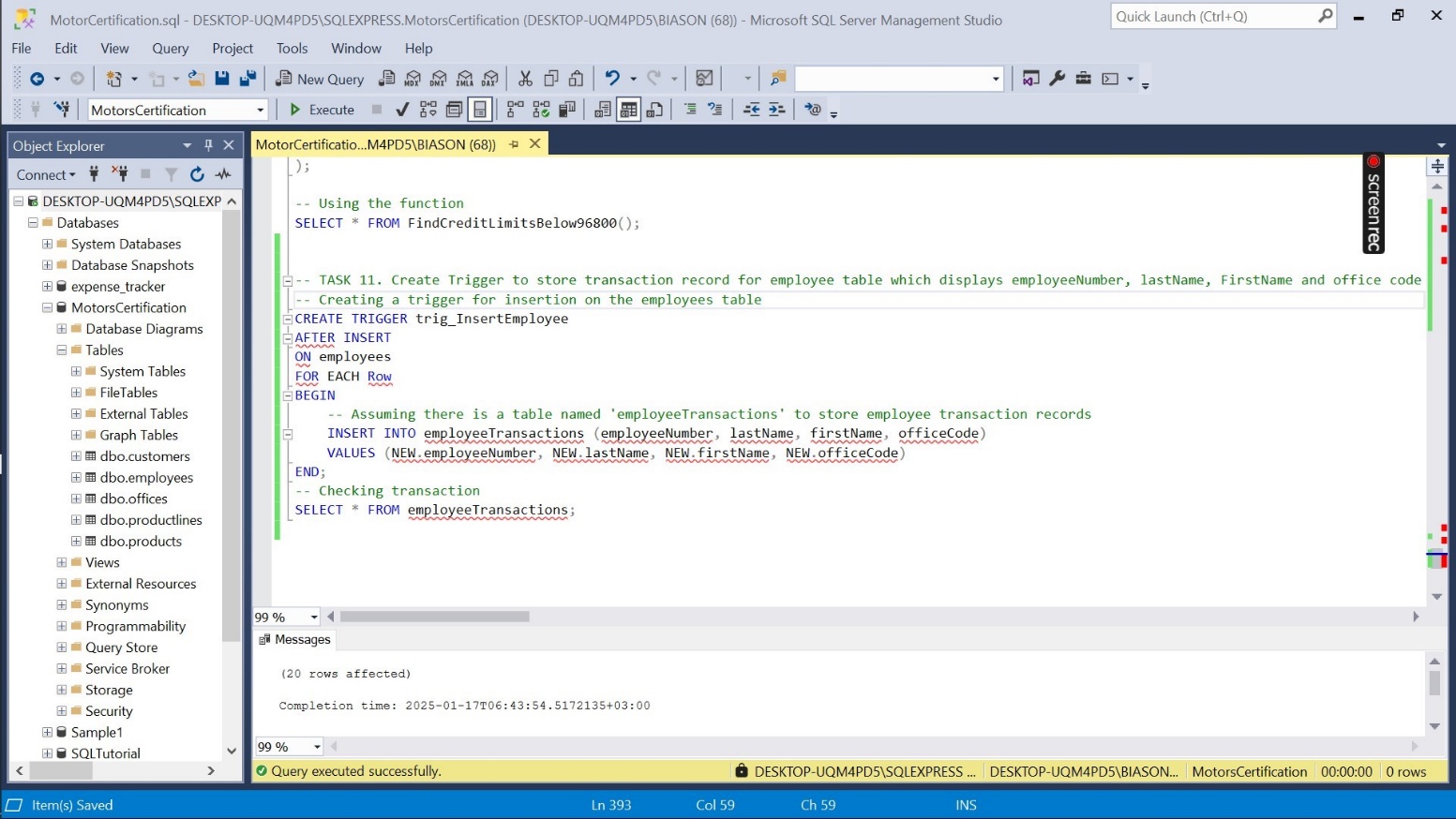
**INSERT INTO employeeTransactions (employeeNumber, lastName, firstName, officeCode)**

**VALUES (NEW.employeeNumber, NEW.lastName, NEW.firstName, NEW.officeCode)**

**END;**

**-- Checking transaction**

**SELECT \* FROM employeeTransactions;**

****

**-- TASK 12. Create a Trigger to display customer number if the amount is greater than 10,000**

**CREATE TRIGGER log\_high\_transaction**

**AFTER INSERT ON trsansactions**

**FOR EACH ROW**

**-- Assuming there is a table 'high\_vallu\_transactions' that stores customer transaction greater than 10000**

**BEGIN**

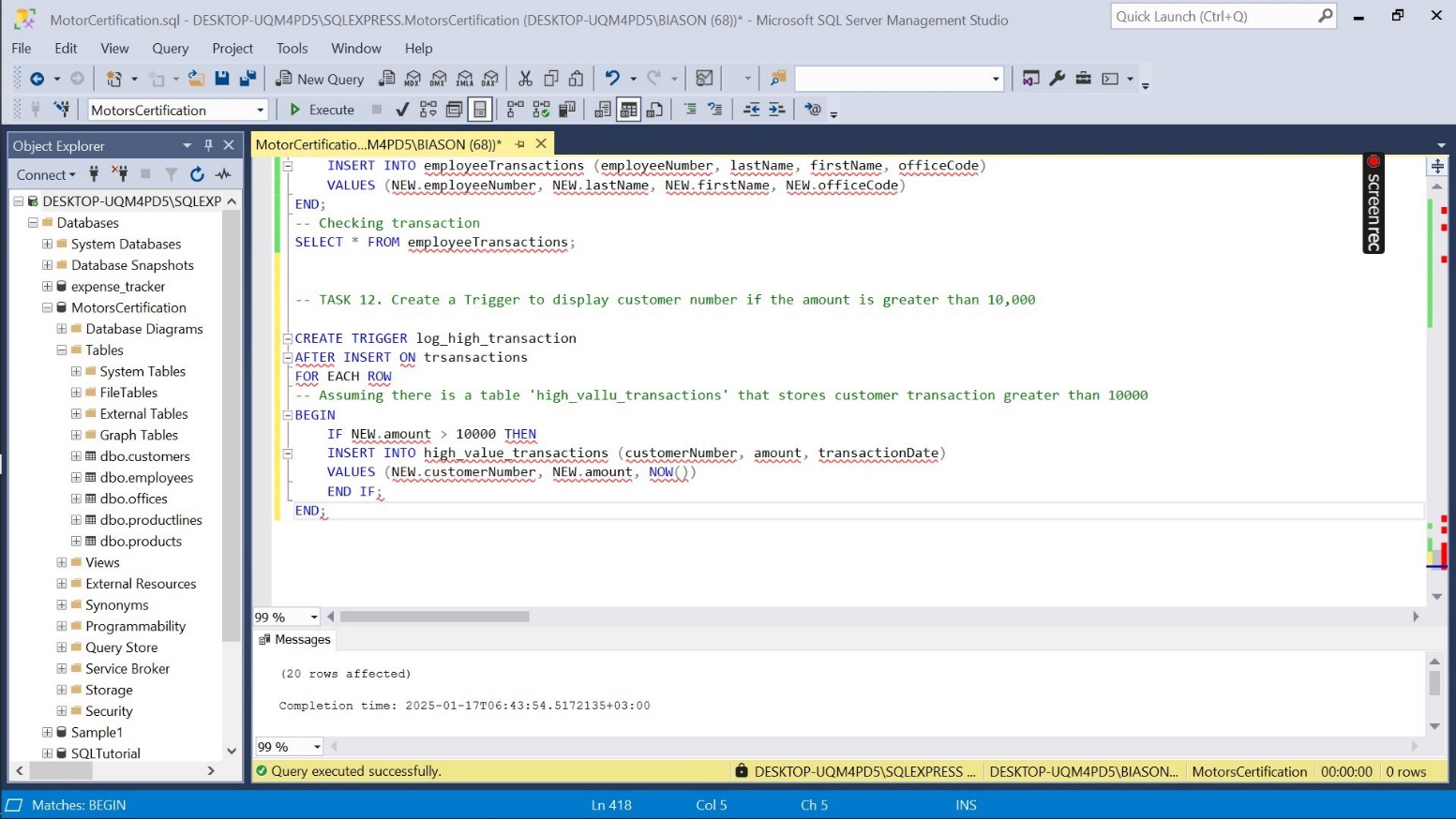
**IF NEW.amount > 10000 THEN**

**INSERT INTO high\_value\_transactions (customerNumber, amount, transactionDate)**

**VALUES (NEW.customerNumber, NEW.amount, NOW())**

**END IF;**

**END;**

****

**TASK 13. Create Users, Roles and Logins according to 3 Roles: Admin, HR, and Employee. Admin can view full database and has full access, HR can view and access only employee and offices table. Employee can view all tables only.**

**-- Creating Roles**

**CREATE ROLE admin\_role;**

**CREATE ROLE hr\_role;**

**CREATE ROLE employee\_role;**

**-- Granting privileges to roles**

**-- admin privileges**

**GRANT SELECT, INSERT, UPDATE, DELETE ON DATABASE::MotorCertification TO admin\_role;**

**-- hr privileges**

**GRANT SELECT, INSERT, UPDATE, DELETE ON employees TO hr\_role;**

**GRANT SELECT, INSERT, UPDATE, DELETE ON offices TO hr\_role;**

**-- employee privileges**

**GRANT SELECT ON customers TO employee\_role;**

**GRANT SELECT ON employees TO employee\_role;**

**GRANT SELECT ON offices TO employee\_role;**

**GRANT SELECT ON productlines TO employee\_role;**

**GRANT SELECT ON products TO employee\_role;**

**GRANT SELECT ON orders TO employee\_role;**

**GRANT SELECT ON payments TO employee\_role;**

**--creating logins**

**CREATE LOGIN admin\_role WITH PASSWORD = 'AdminStrong@123';**

**CREATE LOGIN hr\_role WITH PASSWORD = 'hR@123';**

**CREATE LOGIN employee\_role WITH PASSWORD = 'emPloyee@123';**

**-- creating users for each login in the database**

**CREATE USER admin\_user FOR LOGIN admin\_login;**

**CREATE USER hr\_user FOR LOGIN hr\_login;**

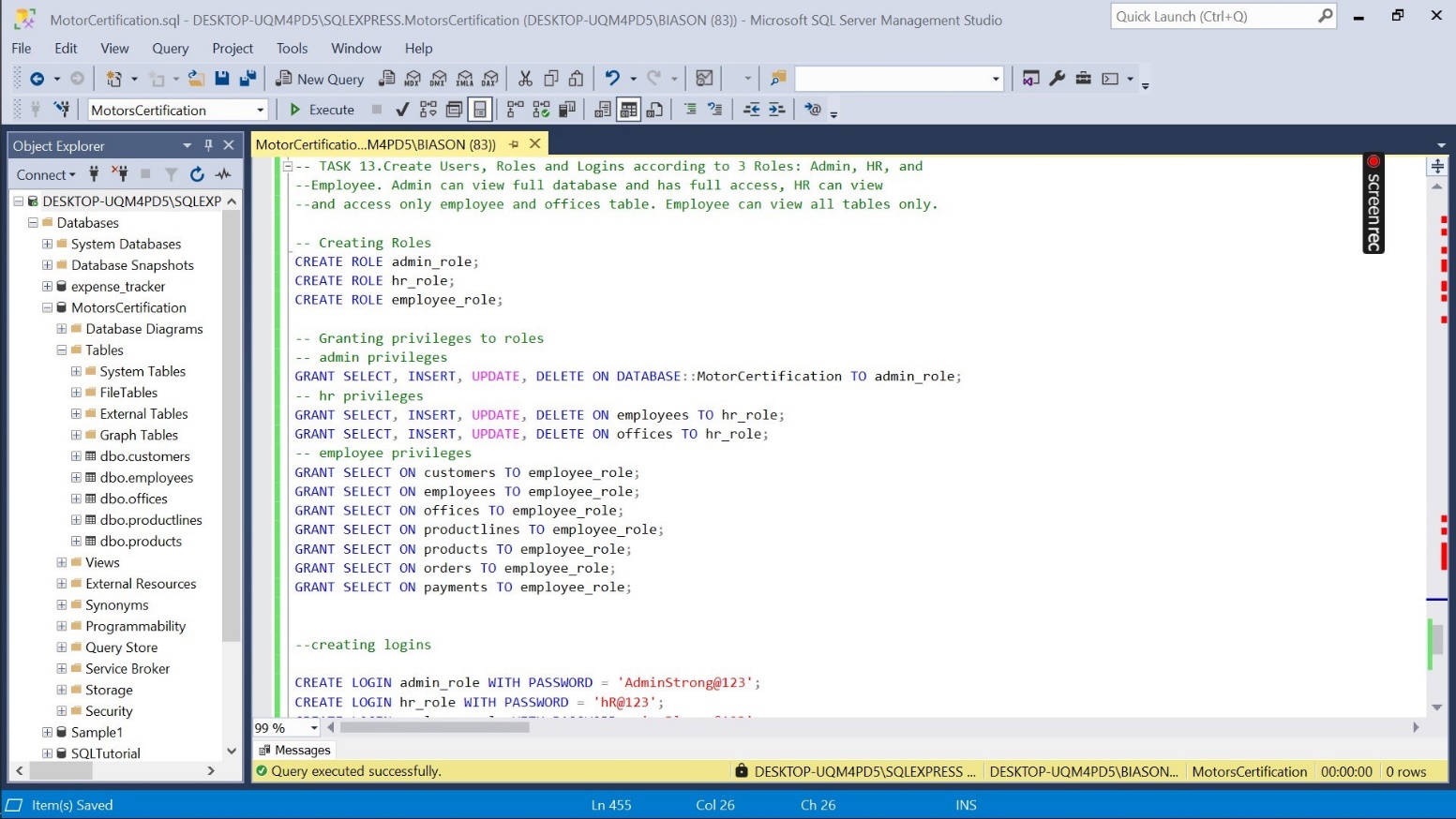
**CREATE USER employee\_user FOR LOGIN employee\_login;**

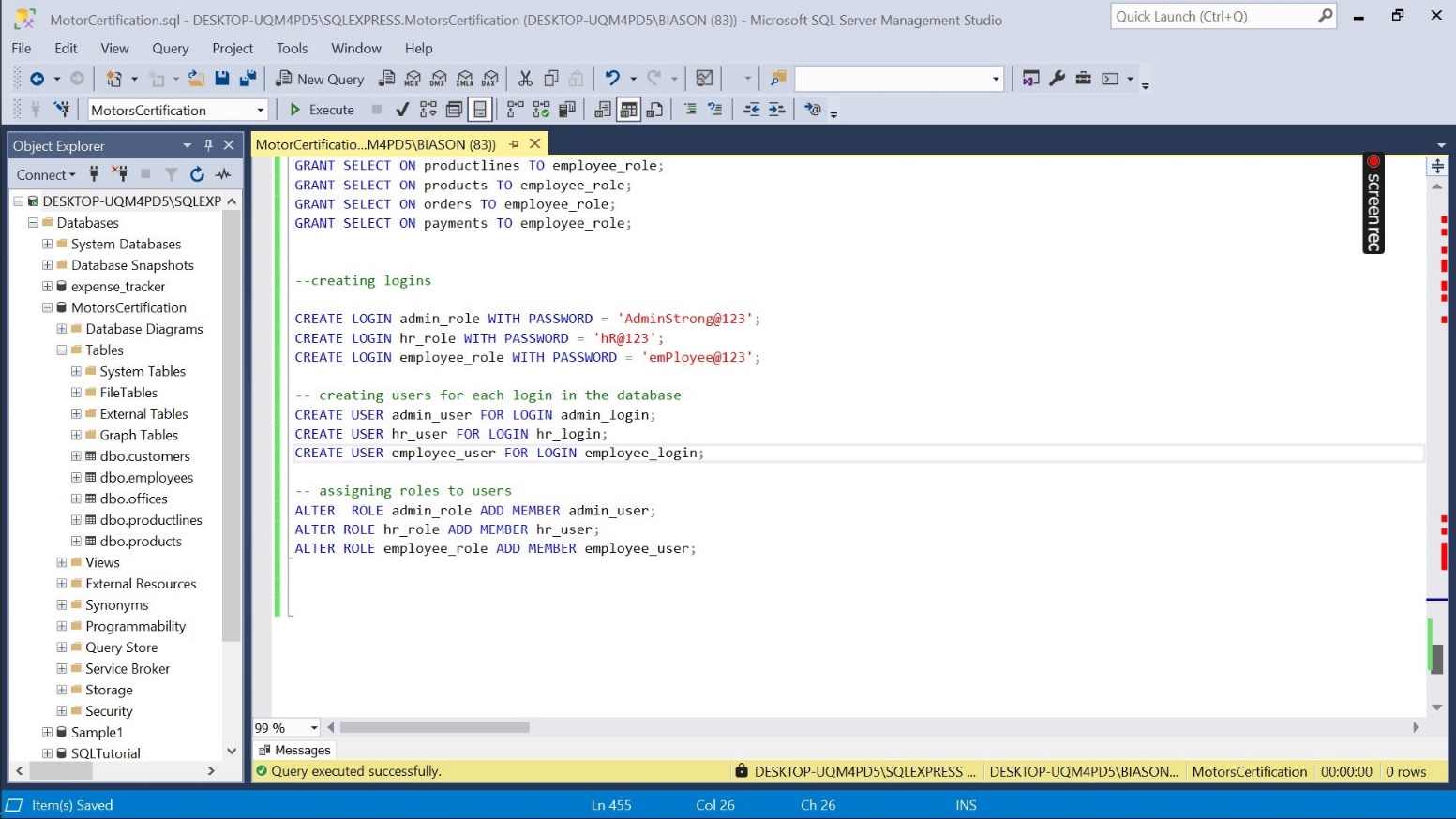
**-- assigning roles to users**

**ALTER ROLE admin\_role ADD MEMBER admin\_user;**

**ALTER ROLE hr\_role ADD MEMBER hr\_user;**

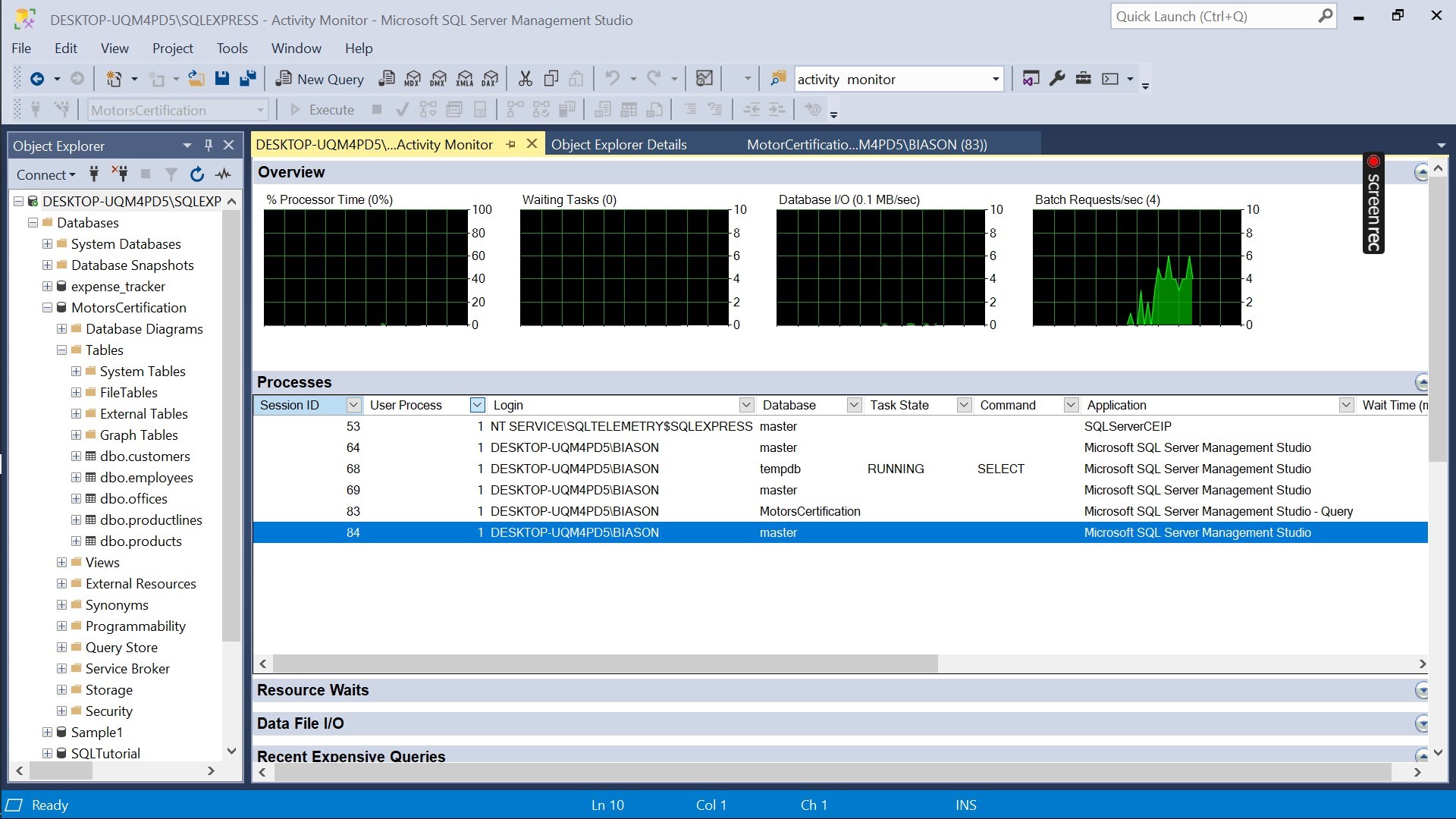
**ALTER ROLE employee\_role ADD MEMBER employee\_user;**

****

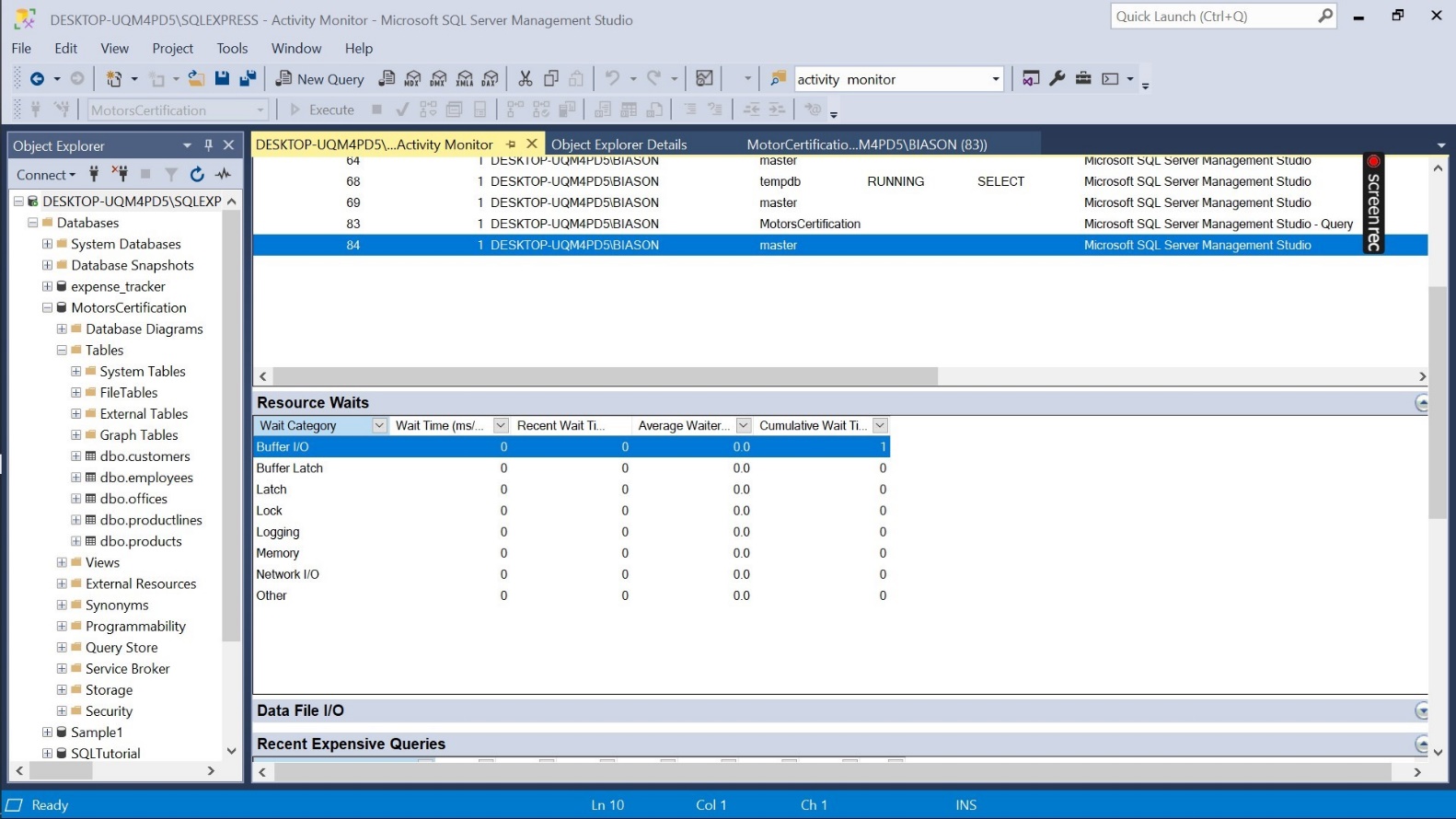
****

**TASK 15. Open Activity Monitor and list down some minor observations including Processes, Resource Waits, and Active Expensive Queries.**

1. **Processes- Displays the active sessions and processes**

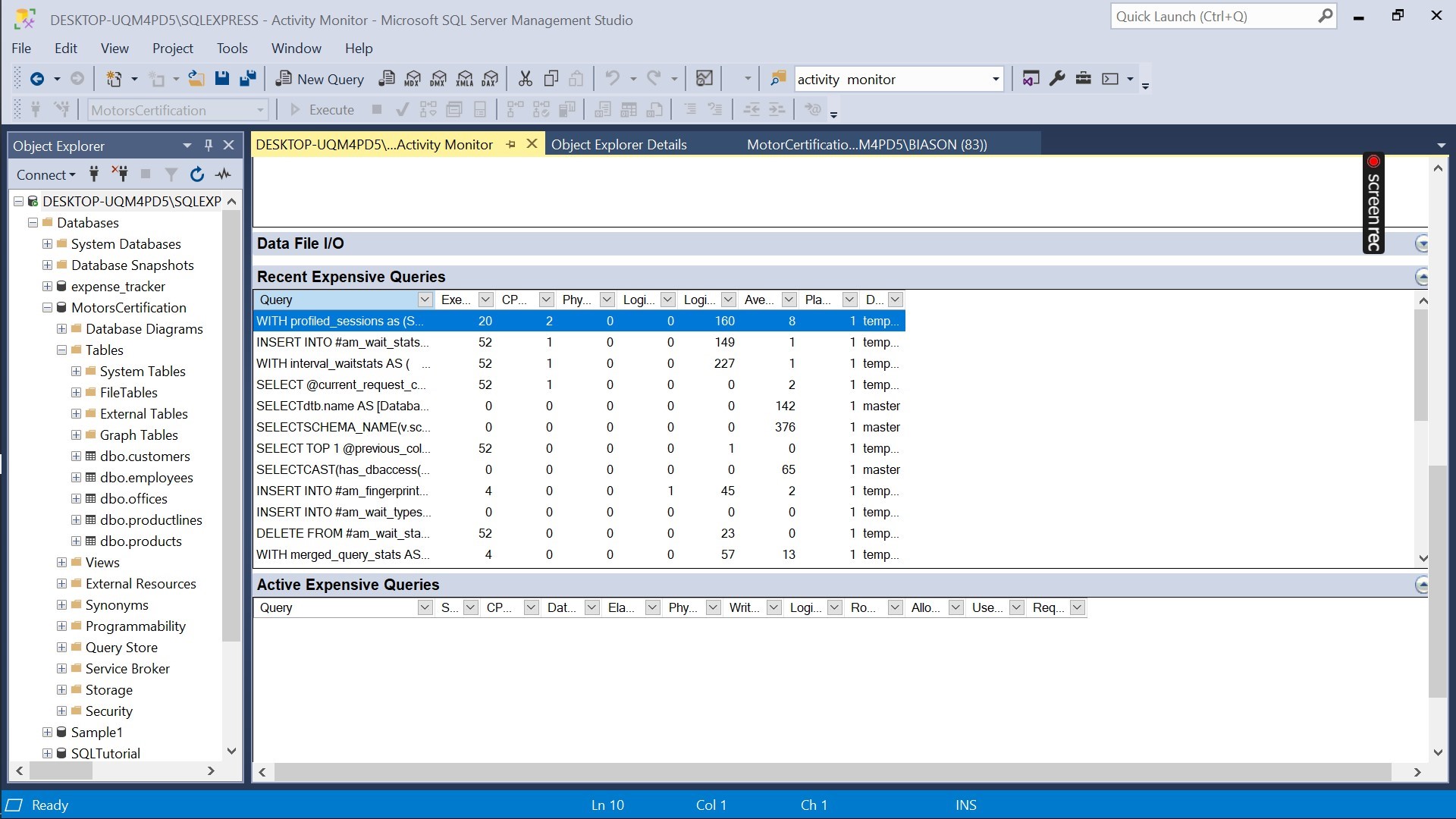
****

1. **Resource waits- shows how long the queries are waiting for resources**

****

1. **Active Expensive Queries- Shows queries that are currently running and are resource intensive.**

* **In my case there are no active expensive queries there are only recent expensive queries**

****