-

# 

**Lewis University  
CPSC 50900: Database Systems   
Spring 2024 Term Project**

**Student name: Sai Naren Reddy Chittireddy**

**INSTRUCTOR: Raymond Klump**

**project title: E-commerce database**

**Github link: https://github.com/naran88/Milestone1**

Table of Contents

[A. Title Page 1](#_gjdgxs)

[B. Initial Proposal 2](#_30j0zll)

[C. Data Sources 3](#_1fob9te)

[D. Alternative Ways to Store the Data 3](#_2et92p0)

[E. Relational Database Design Process 4](#_tyjcwt)

[F. Relational Database Design 4](#_1t3h5sf)

[G. Data Definition Language (DDL) Scripts 5](#_4d34og8)

[H. Data Manipulation Language Scripts 5](#_2s8eyo1)

[I. Indexes 6](#_17dp8vu)

[J. Views 6](#_3rdcrjn)

[K. Triggers 6](#_26in1rg)

[L. Transactions 7](#_lnxbz9)

[M. Database Security 7](#_35nkun2)

[N. Locking and Concurrent Access 7](#_1ksv4uv)

[O. Backing Up Your Database 7](#_44sinio)

[P. Programming 8](#_2jxsxqh)

[Q. Suggested Future Work 8](#_z337ya)

[R. Activity Log 9](#_3j2qqm3)

# Schedule of Milestones

Here is a schedule that shows when each milestone is due and what sections comprise it.

| Deadline | Sections for which you must demonstrate significant progress |
| --- | --- |
| January 29 at 11:59pm | a. Title page  b. Initial proposal  c. Data sources  d. Alternative ways to store the data  r. Activity Log – at least six entries covering the first two weeks |
| February 12  at 11:59pm | e. Relational database design process  f. Relational database design  g. Data definition language scripts  h. Data manipulation language scripts  r. Activity Log – at least six entries covering the past two weeks |
| February 26 at 11:59pm | i. Indexes  j. Views  l. Transactions  m. Security  n. Locking  o. Backup  r. Activity Log – at least six entries covering the past two weeks |

# B. Initial Proposal

Through our e-commerce database I will store different product data like accessories, bags, beauty, housewares, jewelry, kid’s items, shoes, men’s apparel, and women apparel. This data is important for online shopping, comparison, authorizing efficient product searches, and purchase. Here sources include manufactures, user-generated content, and suppliers. The users include administrators, analysts, and customers aligned from individual shoppers to business exploits in our platform for retail operations. I aim to create a robust e-commerce application for seamless browsing, various product management, serving customer requirements, and purchasing. Our application provides a user friendly interface and ensures to manage the data over multiple product categories.

# C. Data Sources

The dataset discusses NewChic.com's product catalog and the popularity of items from August 2020. It defines what people love and how they rate items. A large dataset is accessible through the Telegram bot for updates and custom queries. This is suitable for researchers to see top products, trends, and understand user preferences. Here data are collected from CSV and JSON files where it compromised product details and popularity metrics. Each of the CSV files includes product attributes like ID, name, category, description, and price. On the other hand JSON files include popularity metrics like purchase, rating, and views.

I collect this data straight from NewChic.com, setting product details and popularity as of early August 2020. The data covers user interactions and ratings, facilitating research on product trends, user tastes, and segmentation. At the same time it provides business opportunities such as optimizing stock and identifying top-selling categories. Perfect for searching what people like and making understanding business decisions.

i collect data from [*https://data.world/jfreex/products-catalog-from-newchiccom/*](https://data.world/jfreex/products-catalog-from-newchiccom/)

# D. Alternative Ways to Store the Data

NoSQL Database with JSON Encoding: Here data is stored using a NoSQL Database like a document-oriented model MongoDB. Each of the product categories is documented and organized with JSON. It allows to accommodate diverse data structures, scalable storage, and flexibility. There are some advantages to using this data storage like easy to grow, support for semi-structured data, and quick development cycles. However, it becomes complex when there is limited support for complex transactions and schema flexibility.

Graph Database for Product Relationships: A graph database Neo4j store data by representing products as nodes and their relationships. This model efficiently represents complex interconnections in product popularity and user tastes. Advantages include fast query performance for relationship-based queries. However, a learning curve and potential inefficiency for simple, tabular data storage compared to traditional databases.

# E. Relational Database Design Process

| **Relation** | **Determinant** | **Dependent** | **Explanation** |
| --- | --- | --- | --- |
| 1 | Accessories |  | Some items like makeup bags and mirrors are categorized under the classification of beauty accessories. |
| 2 | Bags | Accessories | Bags are categorized as accessories. |
| 3 | Beauty | Accessories | Some beauty products like hair accessories, makeup brushes, etc are categorized as accessories. |
| 4 | House | Accessories | Some household products like tableware, decorative items, etc are classified as accessories. |
| 5 | Jewelry | Accessories | Jewelry products are considered accessories. |
| 6 | Kids | Accessories | Some accessories are designed for kids like bags, hair accessories, etc. |
| 7 | Men | Accessories | Some accessories are designed for men like ties, belts, etc. |
| 8 | Shoes | Accessories | Shoes are considered accessories. |
| 9 | Women | Accessories | Some accessories are designed for women like handbags, scarves, etc. |

Based on the identified functional dependencies, the following entity sets or tables are:

| **Name** | **Attributes** |
| --- | --- |
| Accessory **Table** | Accessories\_id, Name\_of\_accessori, total\_price, qty |
| Bag | bags\_id, Name, brand, price, color, Accessories\_Accessories |
| BeautyProduct | beauty\_id, Name, brand, price, Accessories\_Accessories |
| Houseware | house\_id, product\_name, price, qty, Accessories\_Accessories |
| Jewelryltem | jewelry\_id, jewelry\_name, price, brand, qty, Accessories\_Accessories |
| KidAccessory | kids\_id, product\_name, brand, qty, price, Accessories\_Accessories |
| MenAccessory | men\_id, product\_name, qty, price, Accessories\_Accessories |
| Shoe | shoes\_id, name, brand, qty, price, Accessories\_Accessories |
| Women Accessory | women\_id, product\_name, qty, price, Accessories\_Accessories |

**One-to-one relationship:**

**Accessories to bags:** One accessory belongs to many bags which means that the bags are a subcategory of accessories.

**Accessories to Beauty Products:** One accessory is classified into multiple beauty products to define the versatility of accessories within various beauty items.

**Accessories to Houseware:** One accessory is classified into multiple houseware items and provides various advantages of accessories in the household.

**Accessories to JewelryItems:** One accessory is classified into multiple jewelryItems that identify ornamental aspects of accessories.

**Accessories to KidAccessories:**  One accessory is classified into multiple kid accessories to acknowledge specific design and functionality for children.

**Accessories to MenAccessories:** One accessory is classified into multiple men's accessories and involves a wide range of accessories customized for men.

**Accessories to Shoe:** One accessory belongs to many shoes to reflect the association between footwear and accessories.

**Many-to-Many Relationships:**

**Example: BeautyProducts and Accessories:** The many-to-many relationship between accessories and beauty products represents that the beauty products belong to multiple accessory categories and vice versa. It defines the flexibility of a comprehensive representation of the product catalog.

**Multi-valued Attribute:**

If any multi-valued attributes are identified then it need to be normalized. For example, in the product table attributes like “Sizes” have multiple values and it is broken down into individual tables with ProductID and Size as columns.

F. Relational Database Design

**Normalization Analysis:**

**Accessory Table- Attributes:** Accessories\_id, Name\_of\_accessori, total\_price, qty

total\_price is potentially calculated from qty and price of each accessory. This means it should removed to avoid redundancy.

**Bags- Attributes:** bags\_id (Primary Key), Name, brand, price, color, Accessories\_Accessories.

The normalized color into the individual table when multiple colors are associated with a single bag.

**BeautyProduct: Attributes:** beauty\_id (Primary Key), name, brand, price, Accessories\_Accessories.

Create a separate table for BeautyProduct Accessories. Remove transitive dependencies.

**Houseware- Attributes:** house\_id (Primary Key), product\_name, price, qty, Accessories\_Accessories.

Normalized the product\_name into individual tables when multiple products are associated with a single houseware item.

**JewelryItem- Attributes:** jewelry\_id (Primary Key), jewelry\_name, price, brand, qty, Accessories\_Accessories.

Create a separate table for JewelryItem accessories. Remove transitive dependencies.

**KidAccessory- Attributes:** kids\_id (Primary Key), product\_name, brand, qty, price, Accessories\_Accessories.

Create a separate table for KidAccessory accessories. Remove transitive dependencies.

**MenAccessory- Attributes:** men\_id (Primary Key), product\_name, qty, price, Accessories\_Accessories.

Create a separate table for MenAccessory accessories. Remove transitive dependencies.

**Shoe- Attributes:** shoes\_id (Primary Key), name, brand, qty, price, Accessories\_Accessories.

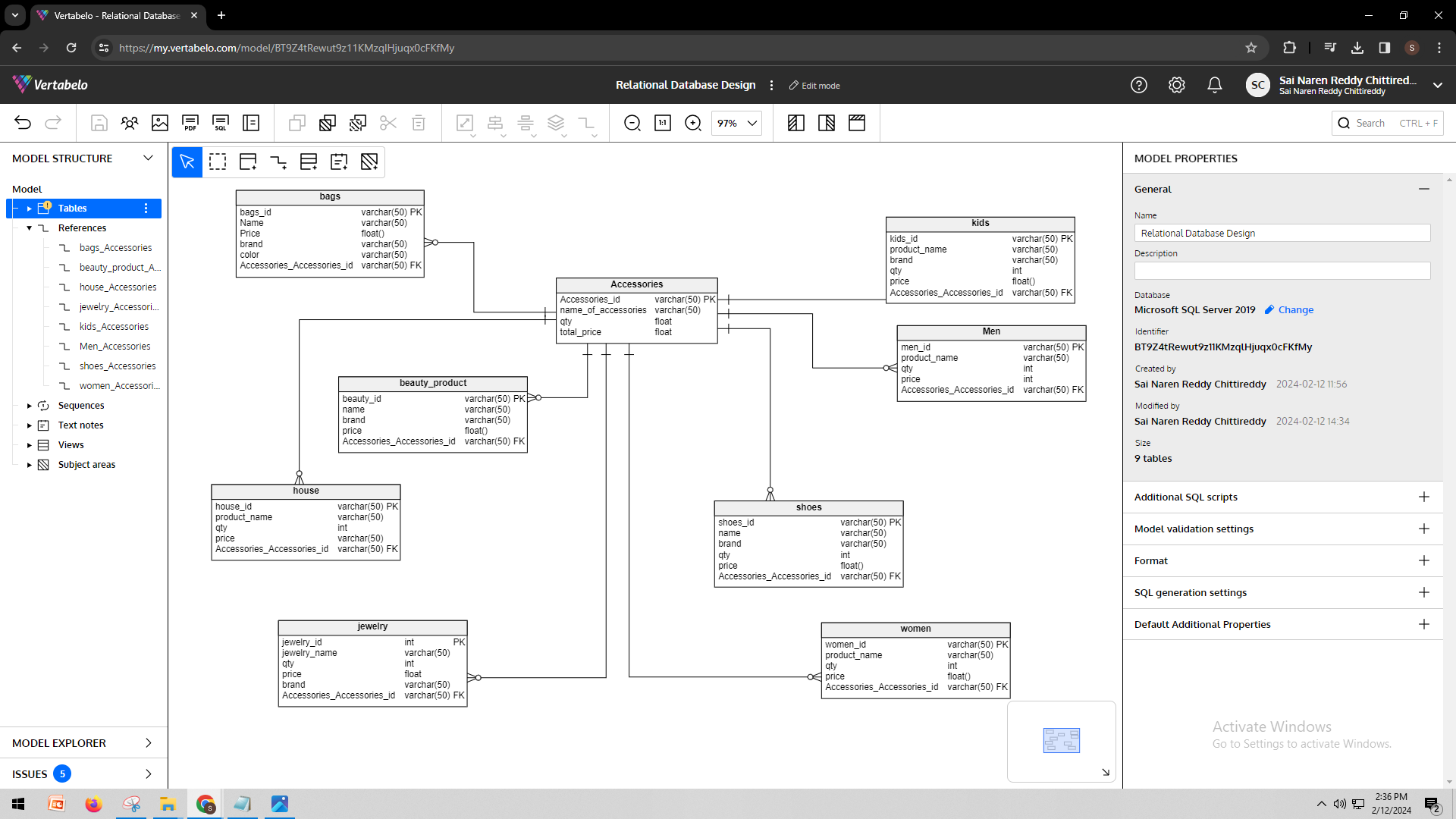
Create a separate table for shoe accessories. Remove transitive dependencies.

**WomenAccessory- Attributes:** women\_id (Primary Key), product\_name, qty, price, Accessories\_Accessories.

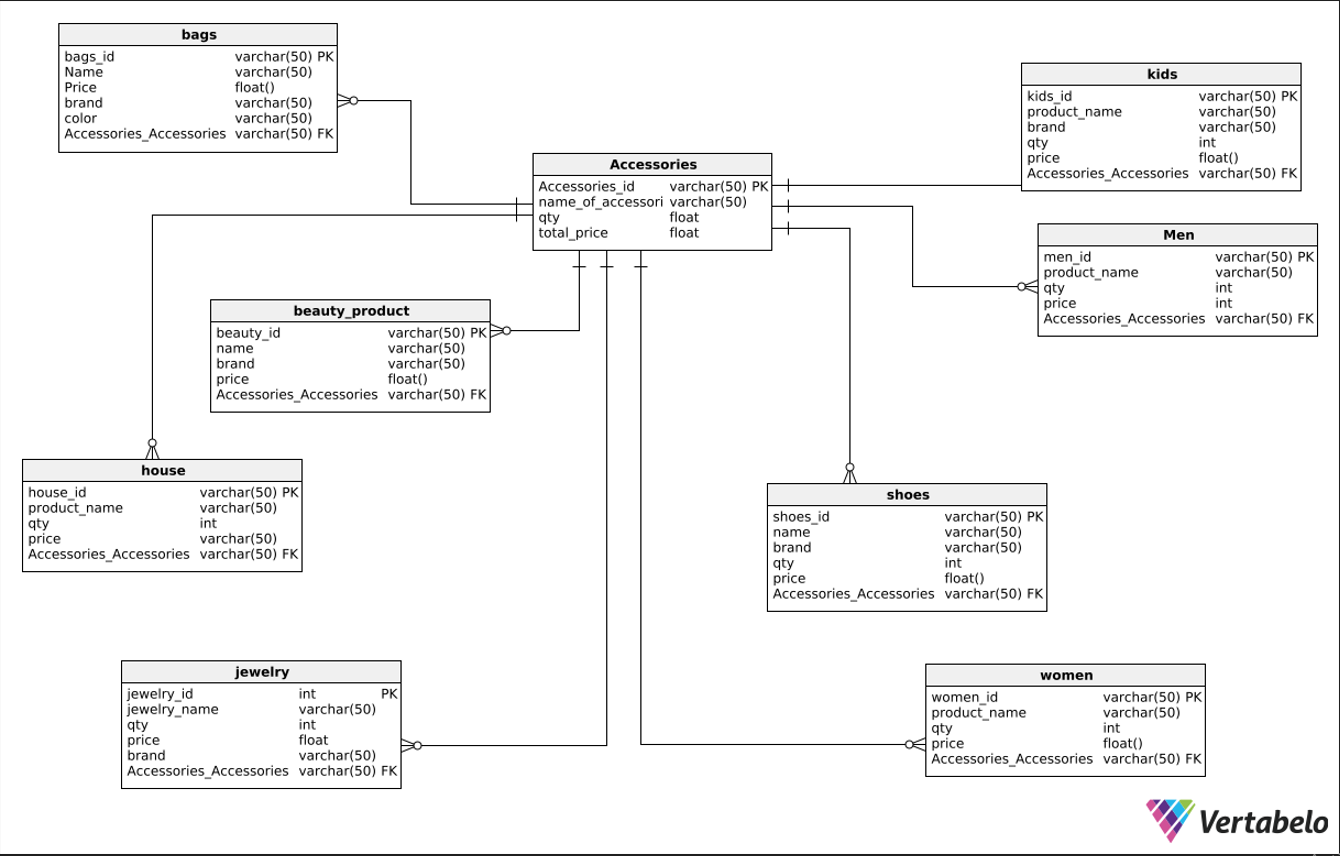
Create a separate table for WomenAccessory accessories. Remove transitive dependencies.

In this normalization analysis, there are no many-to-many relationships identified. So the introduction of bridge entity sets is not required here.

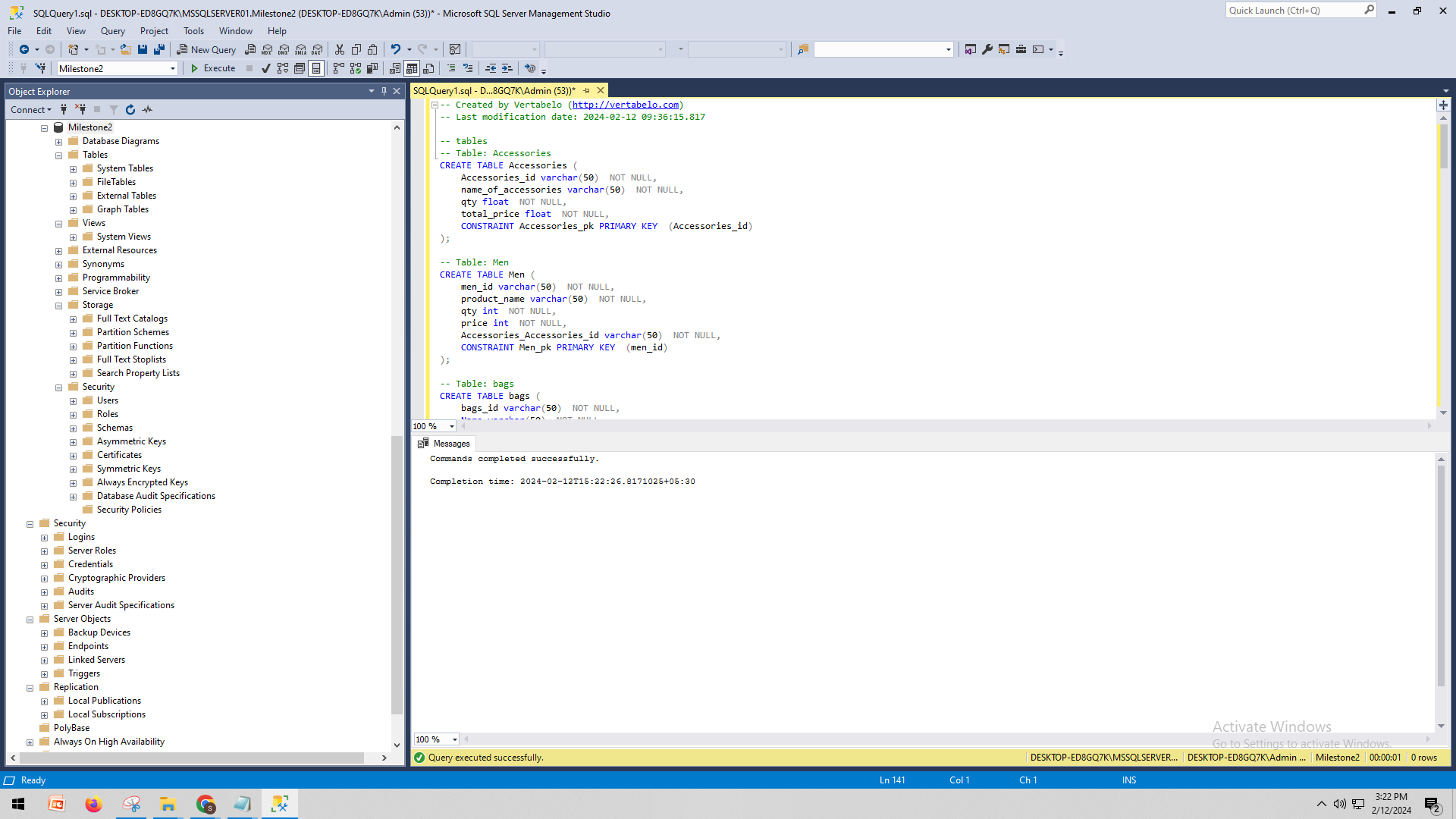
Select a foreign key to establish the relationship between tables and ensure data integrity. To provide more efficiency, introduce the substitute key as a primary key. It helps to enhance performance and provide a stable identifier while natural key change. This approach provides a robust relational database design to support relationships and improve performance.

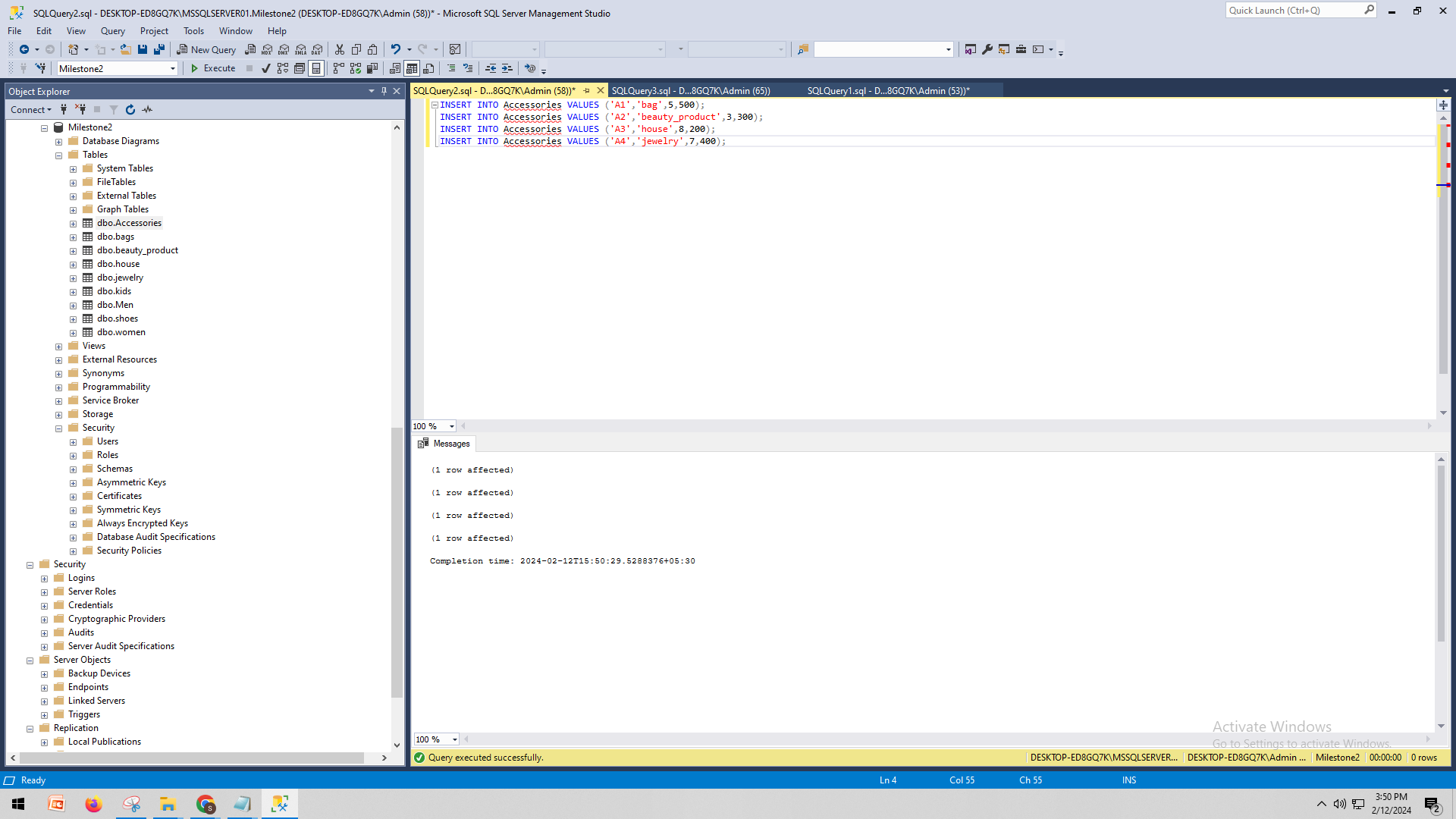
**

*After export from vertebelo:*

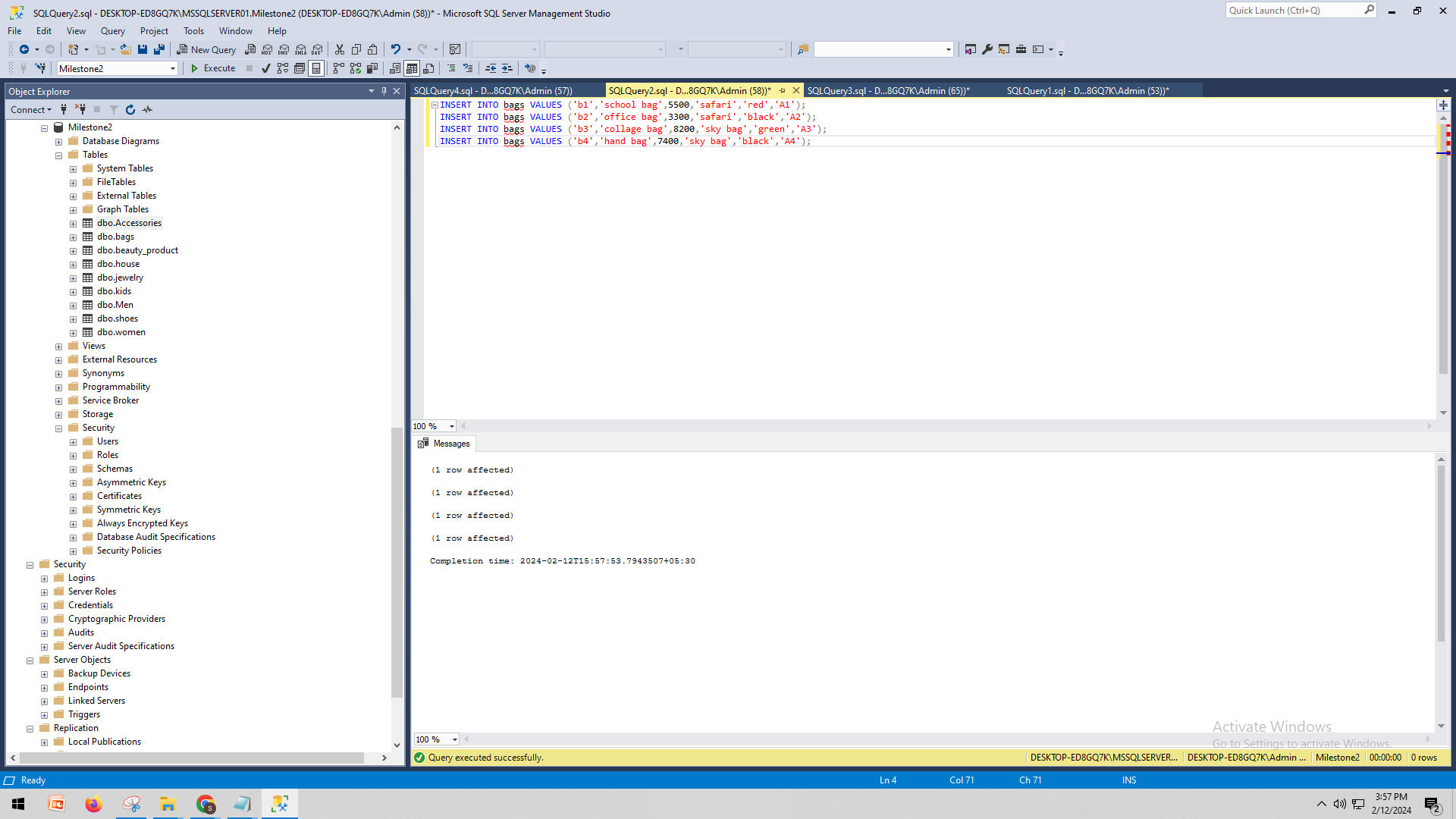
**

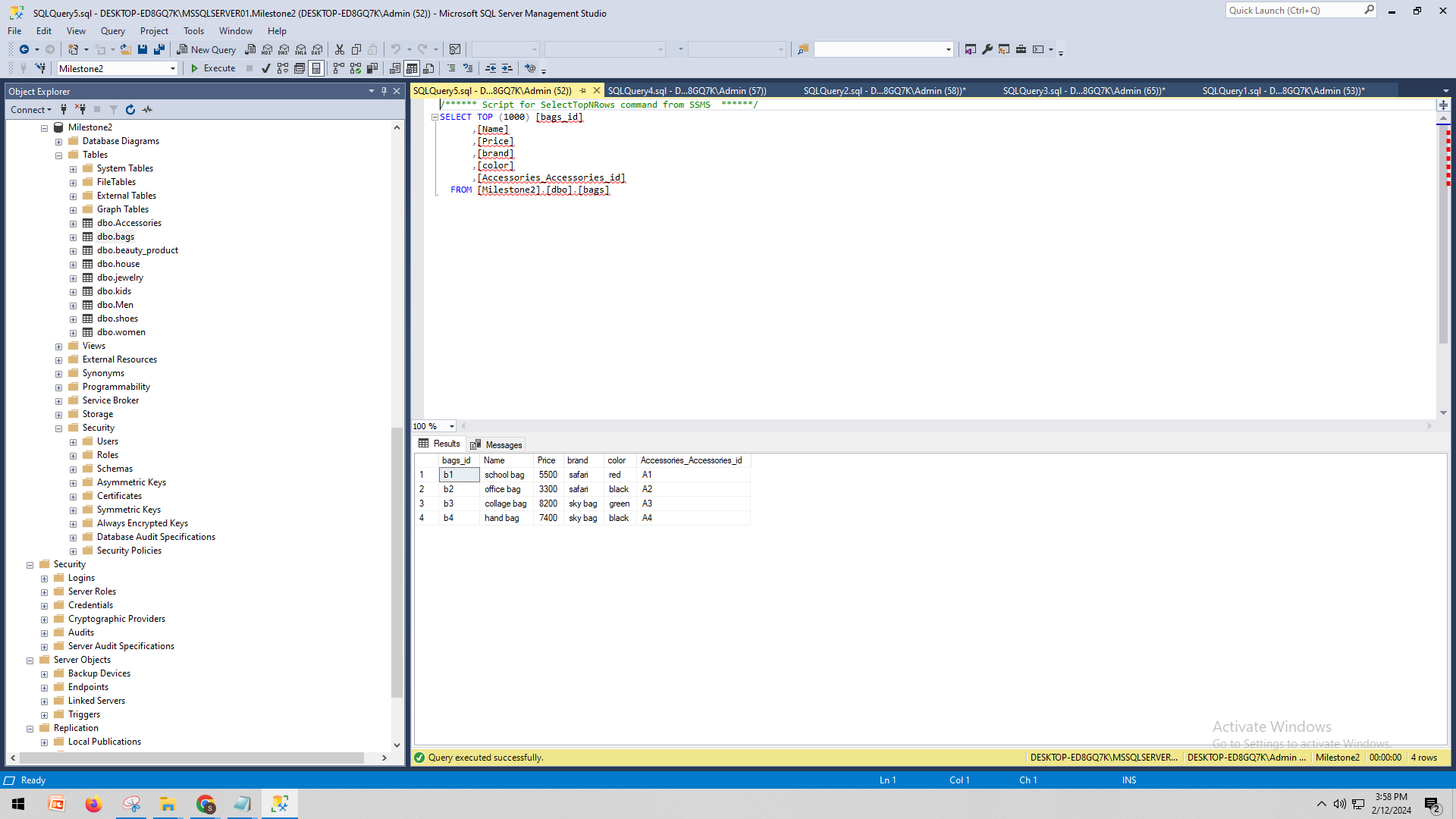
# G. Data Definition Language (DDL) Scripts

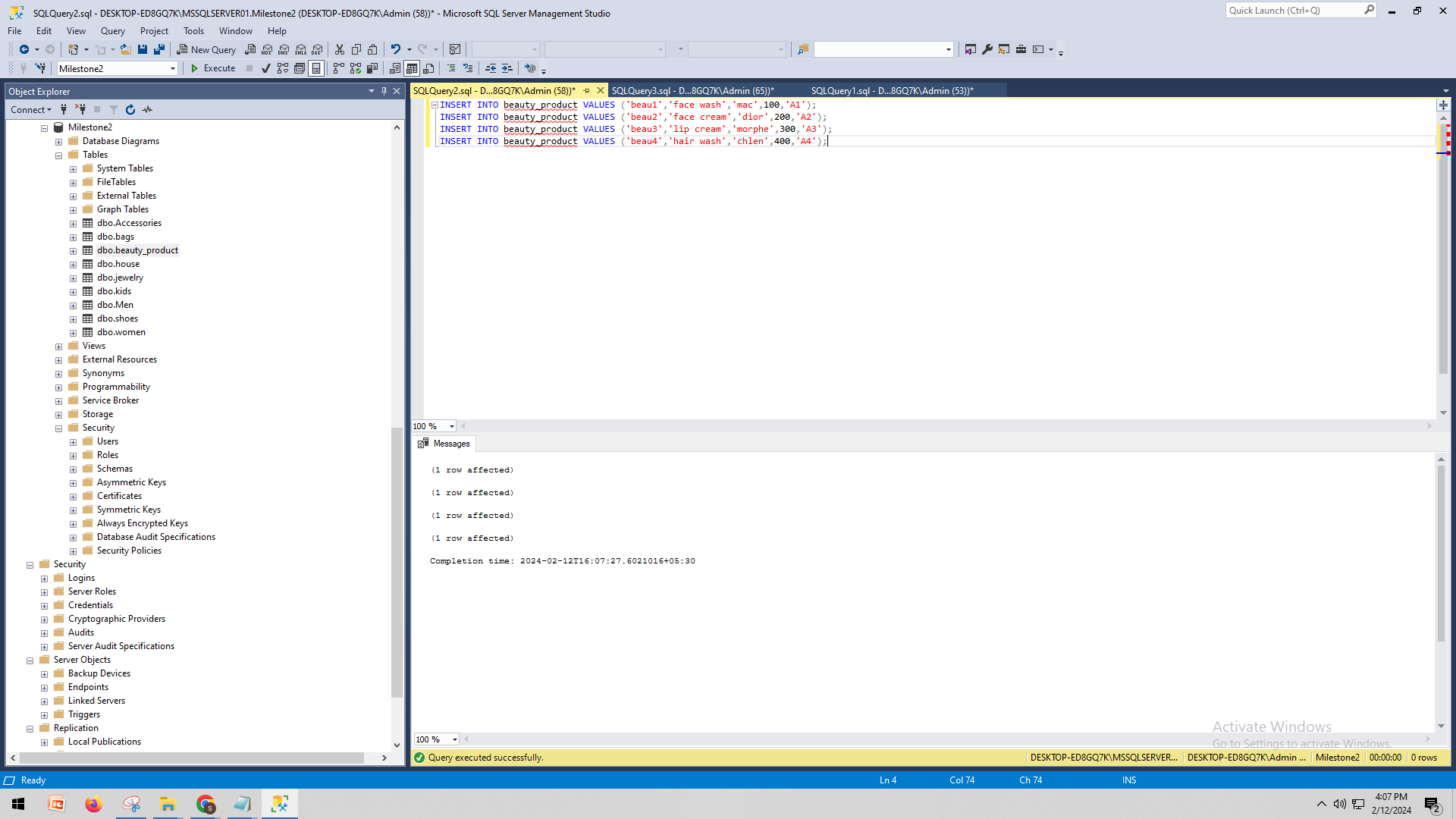


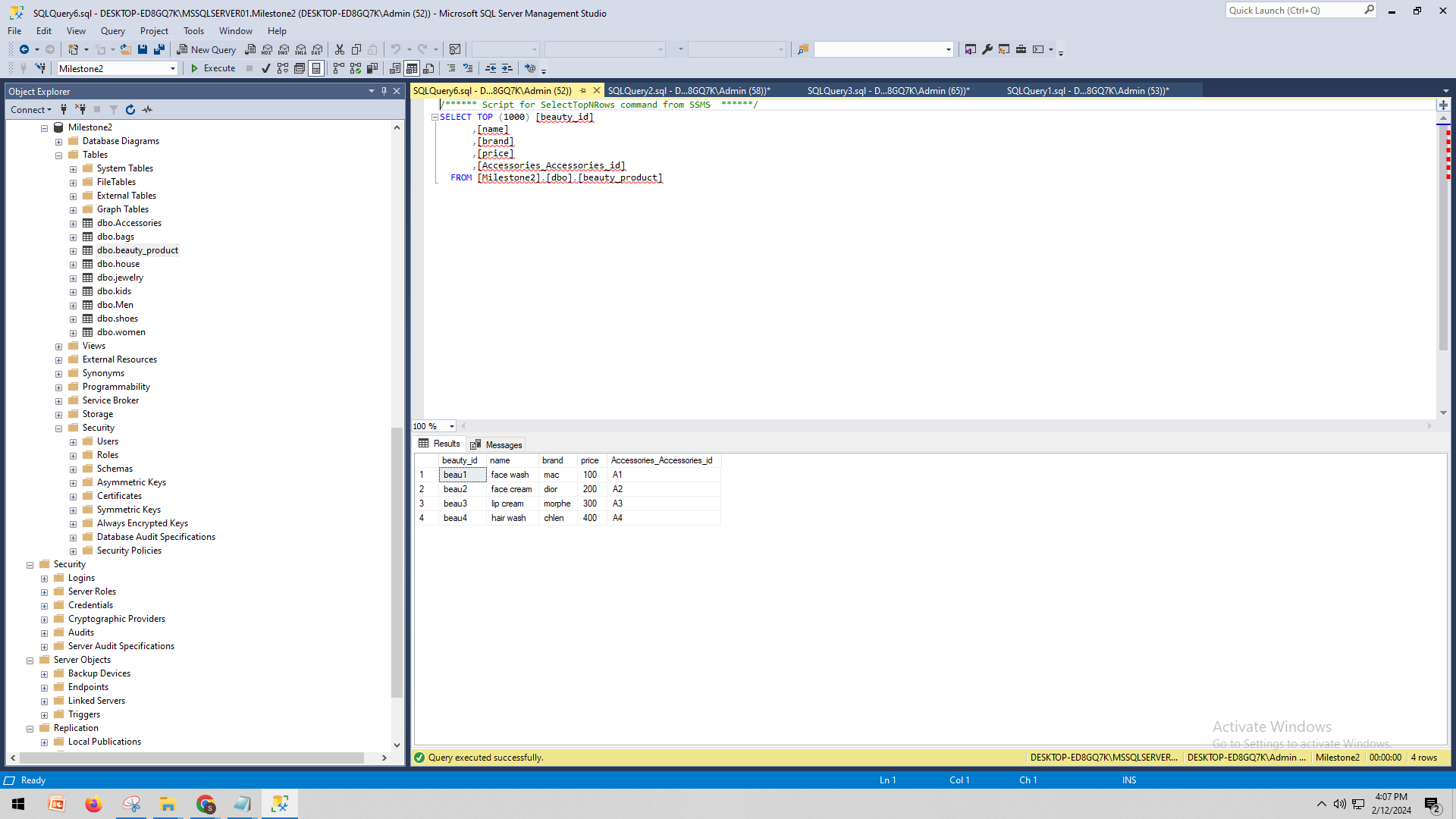


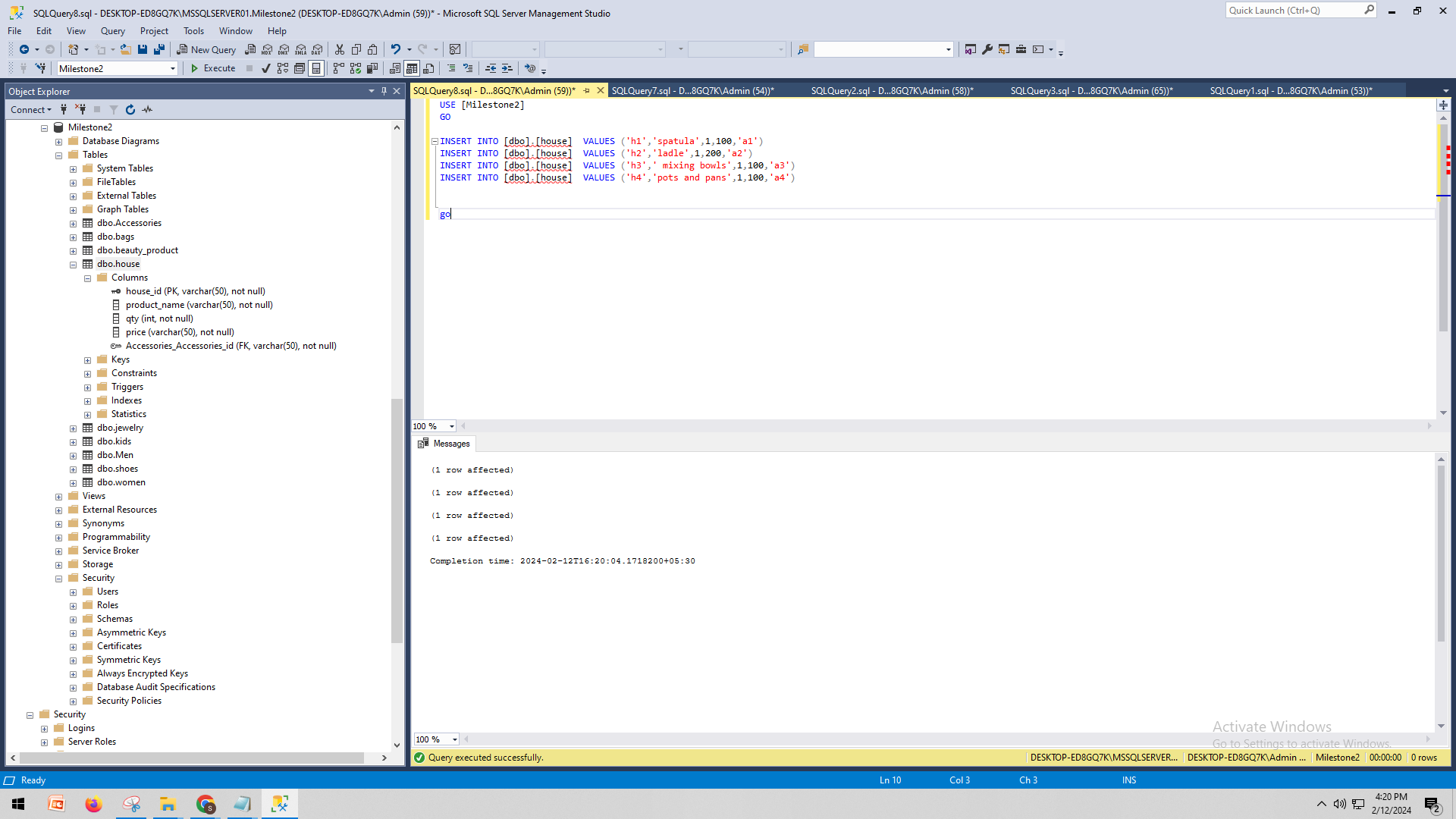


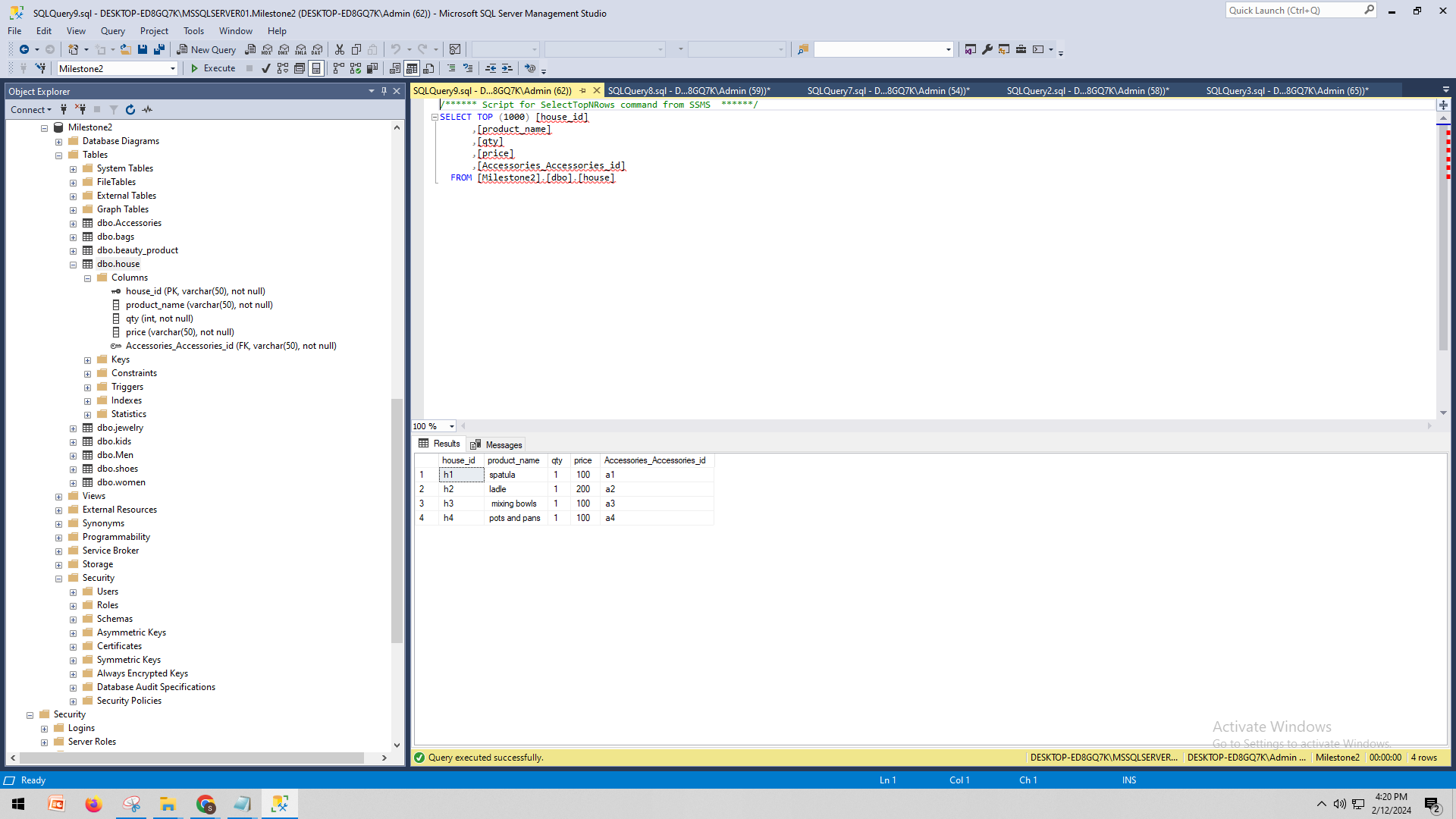


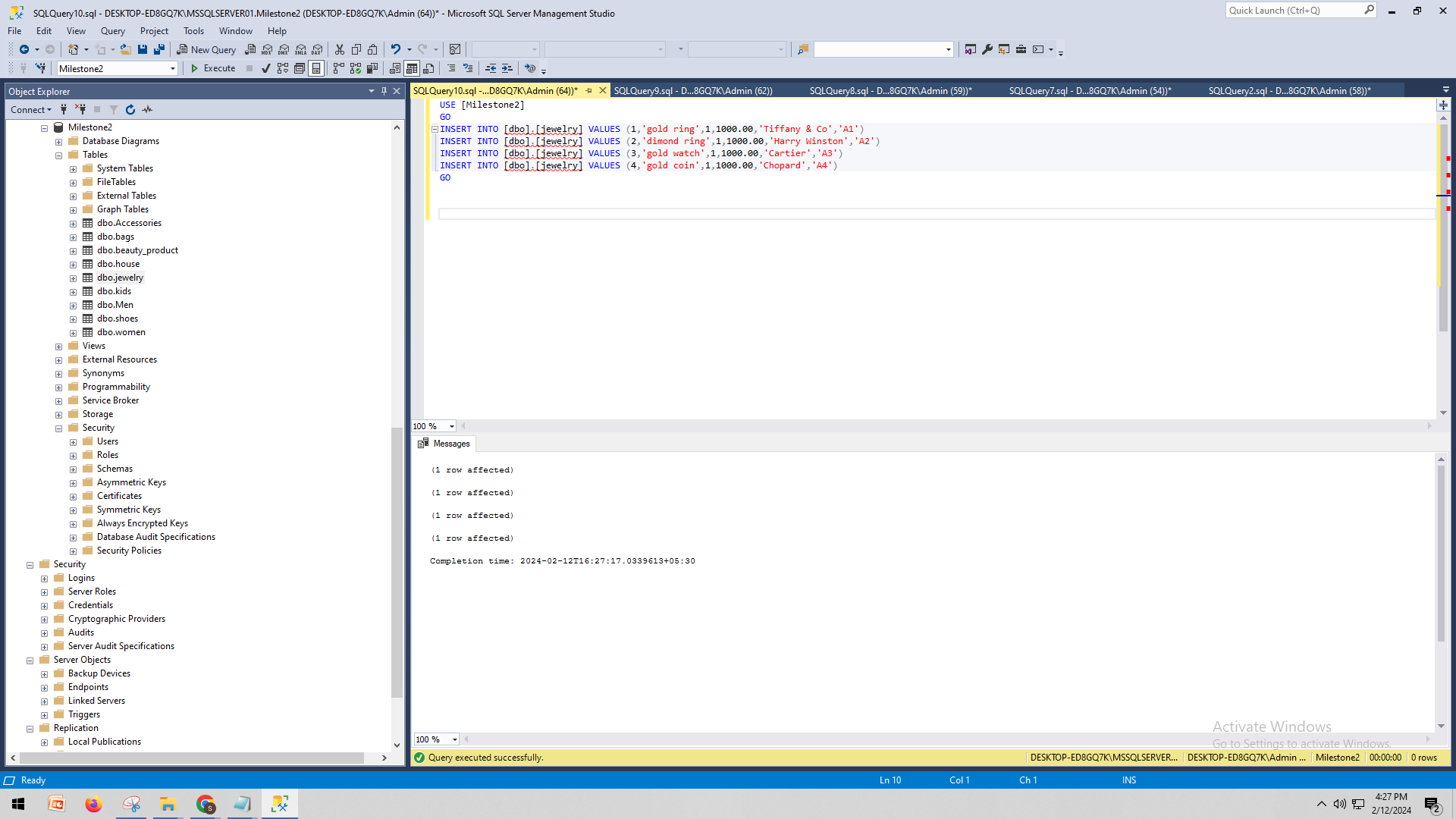


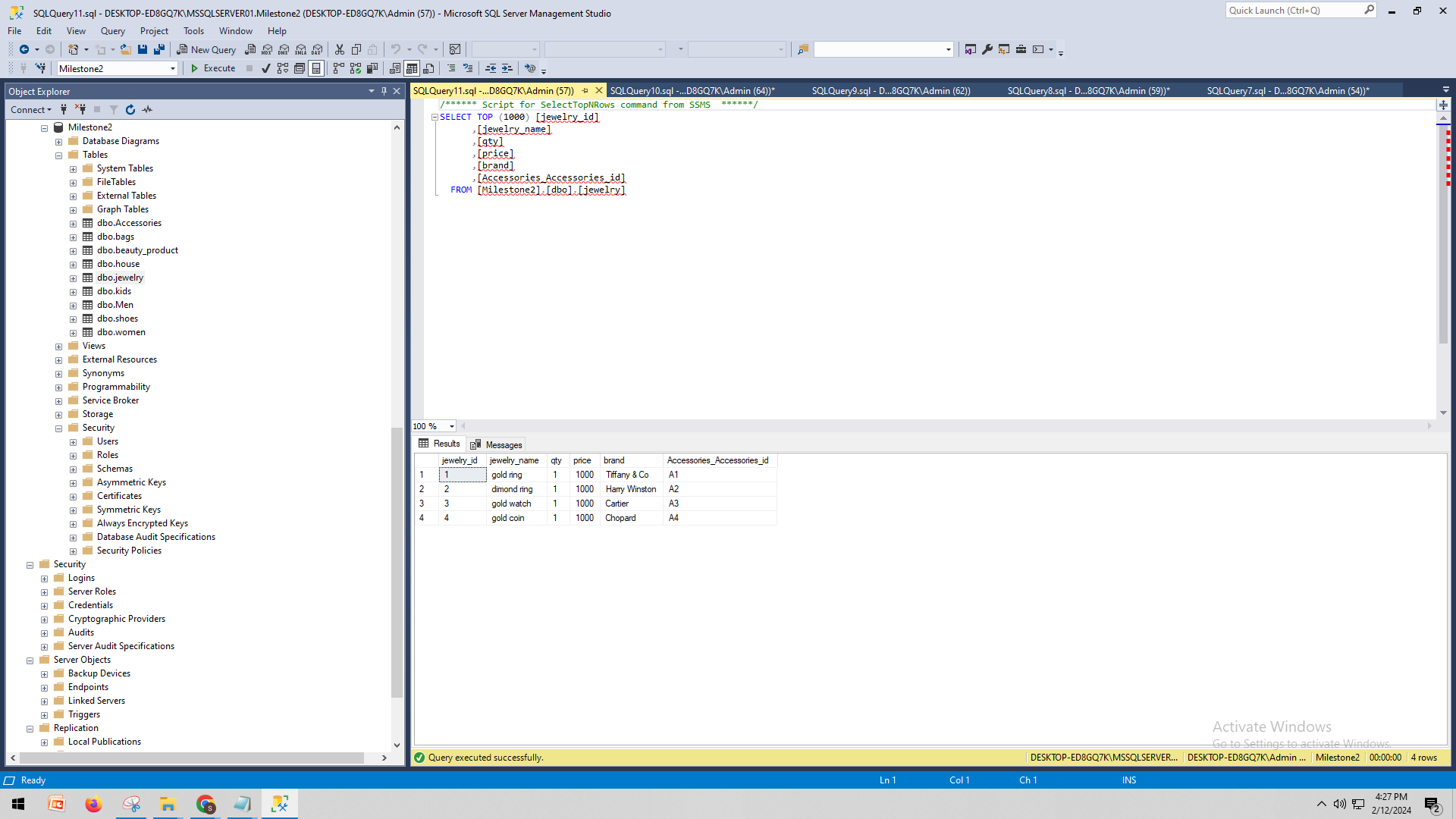


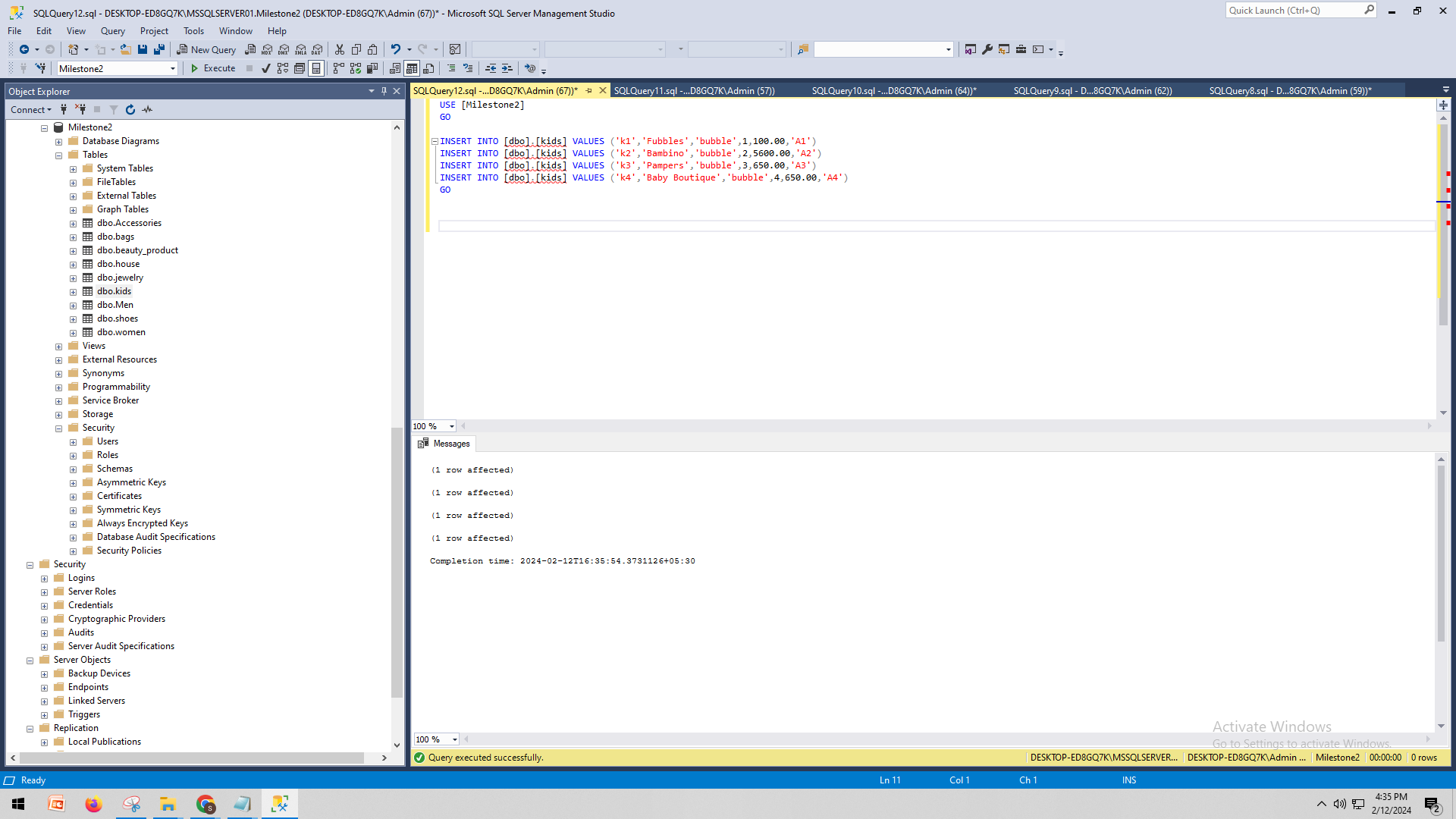


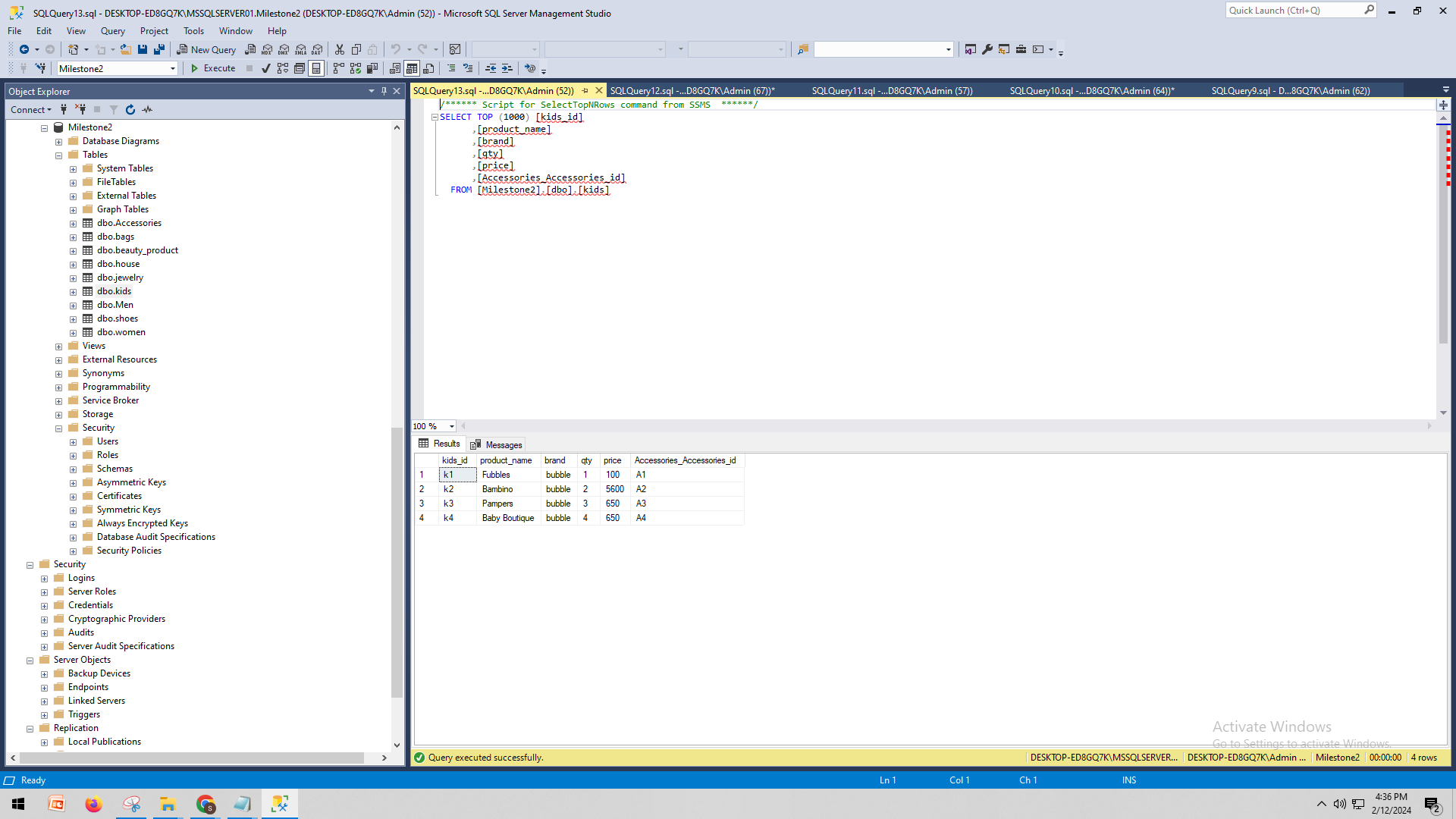


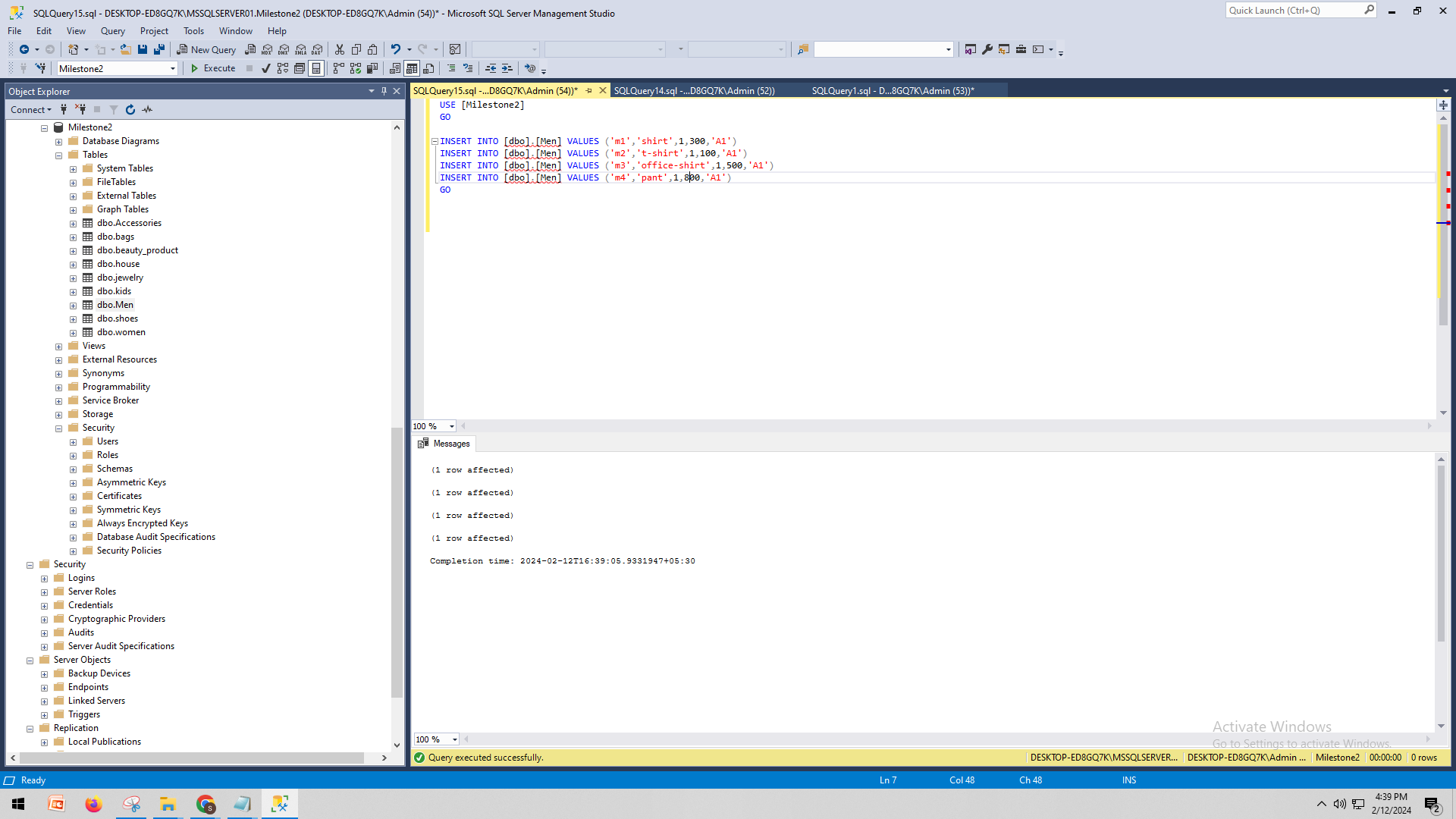


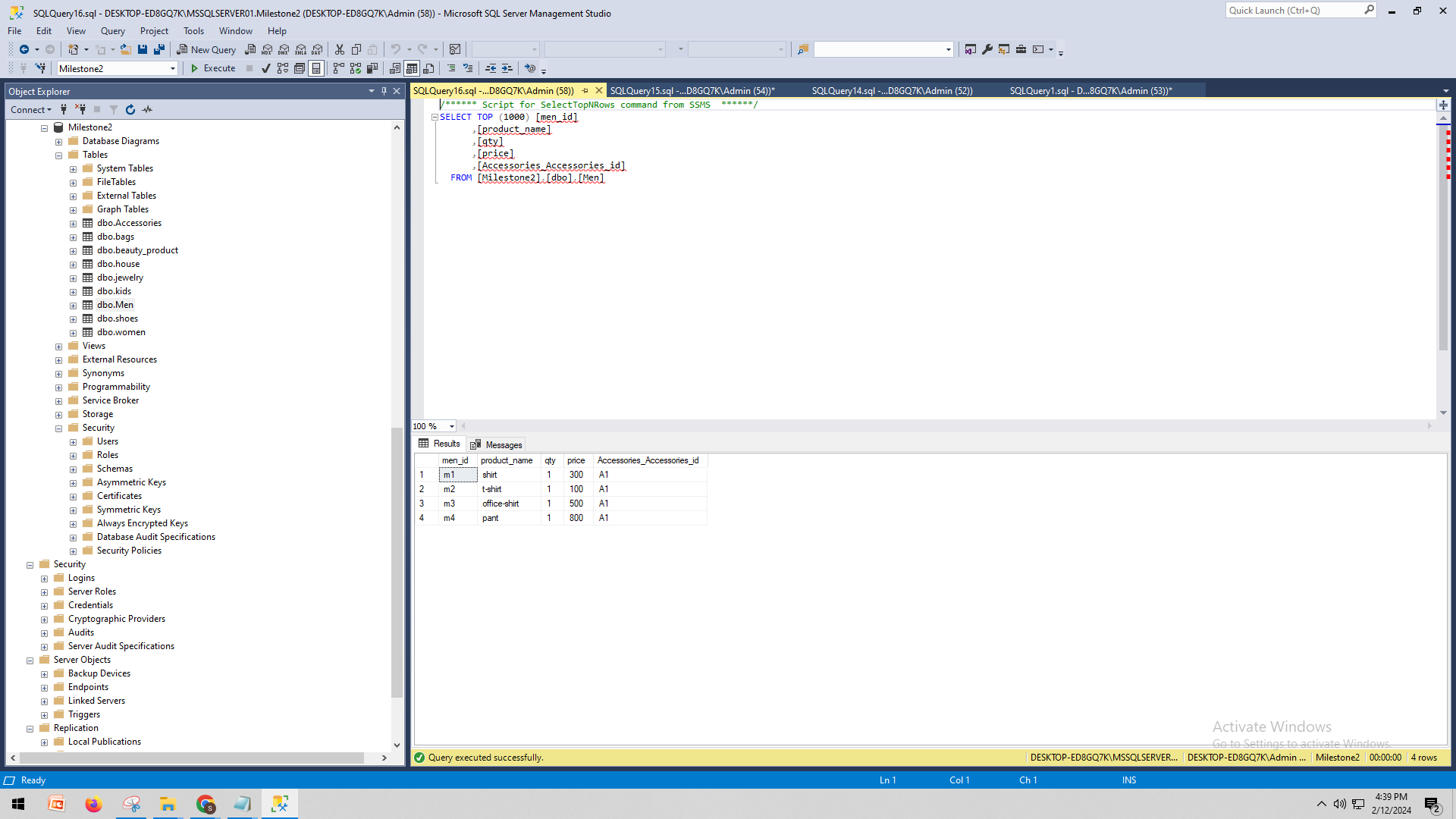


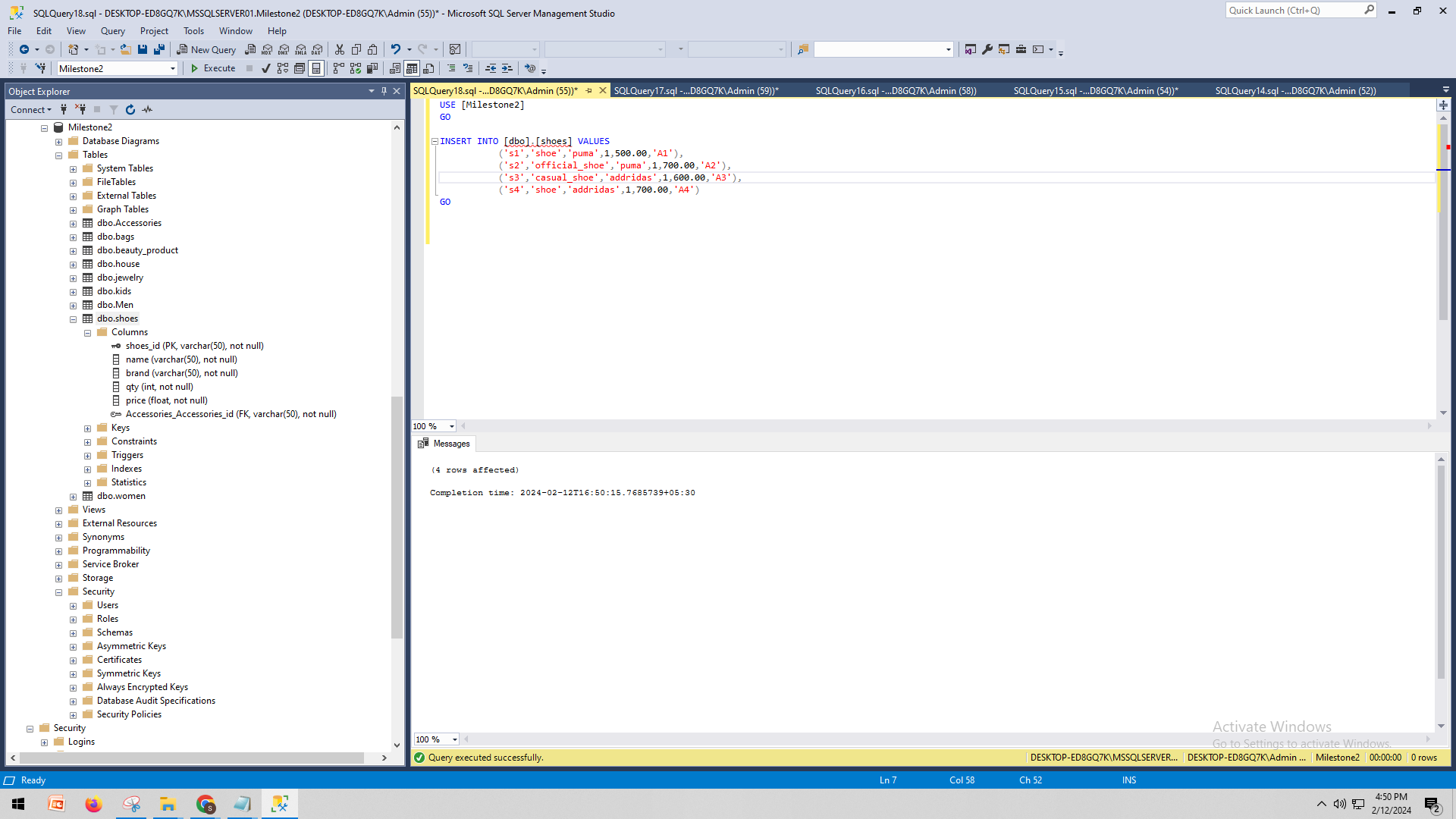


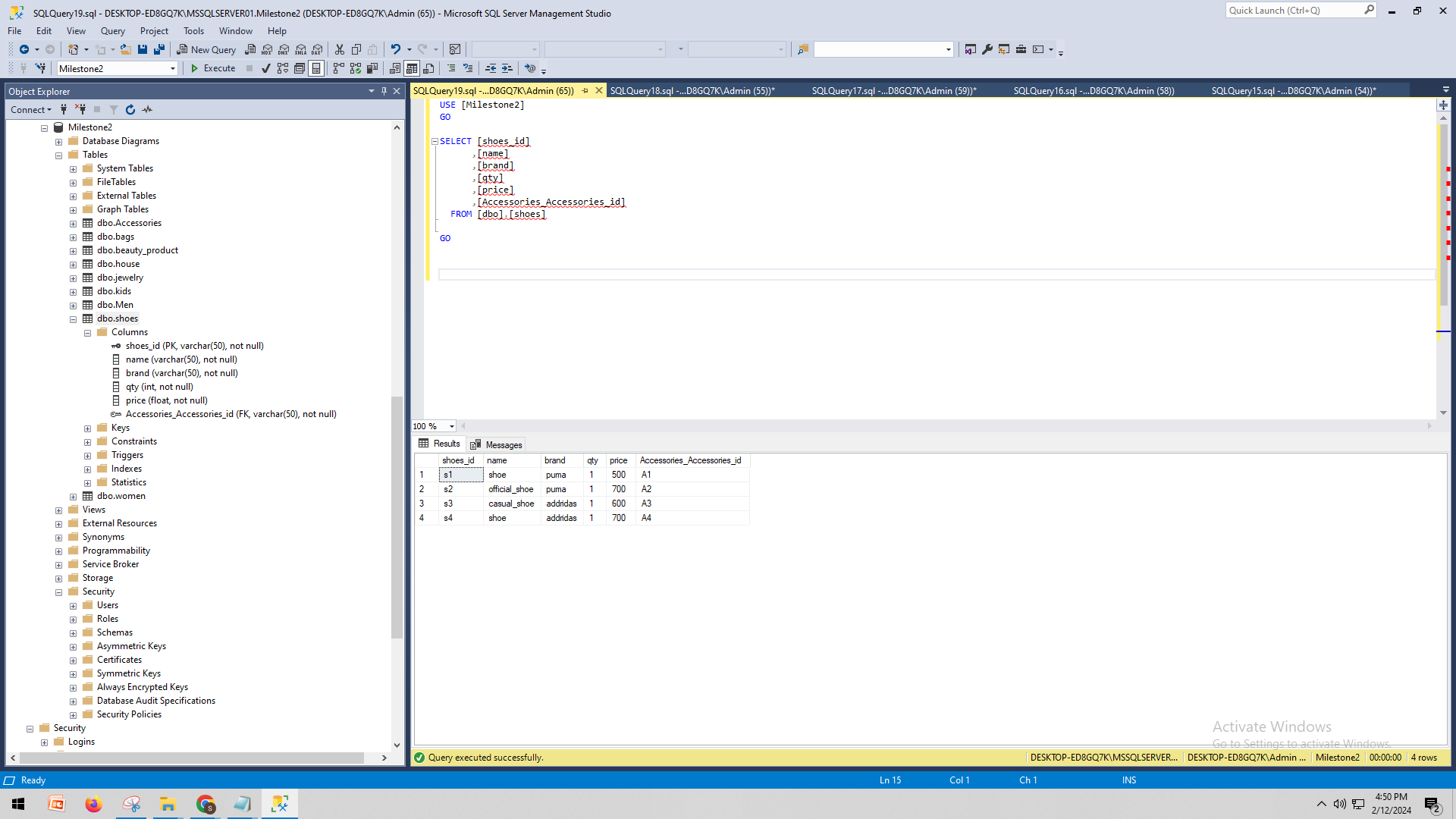


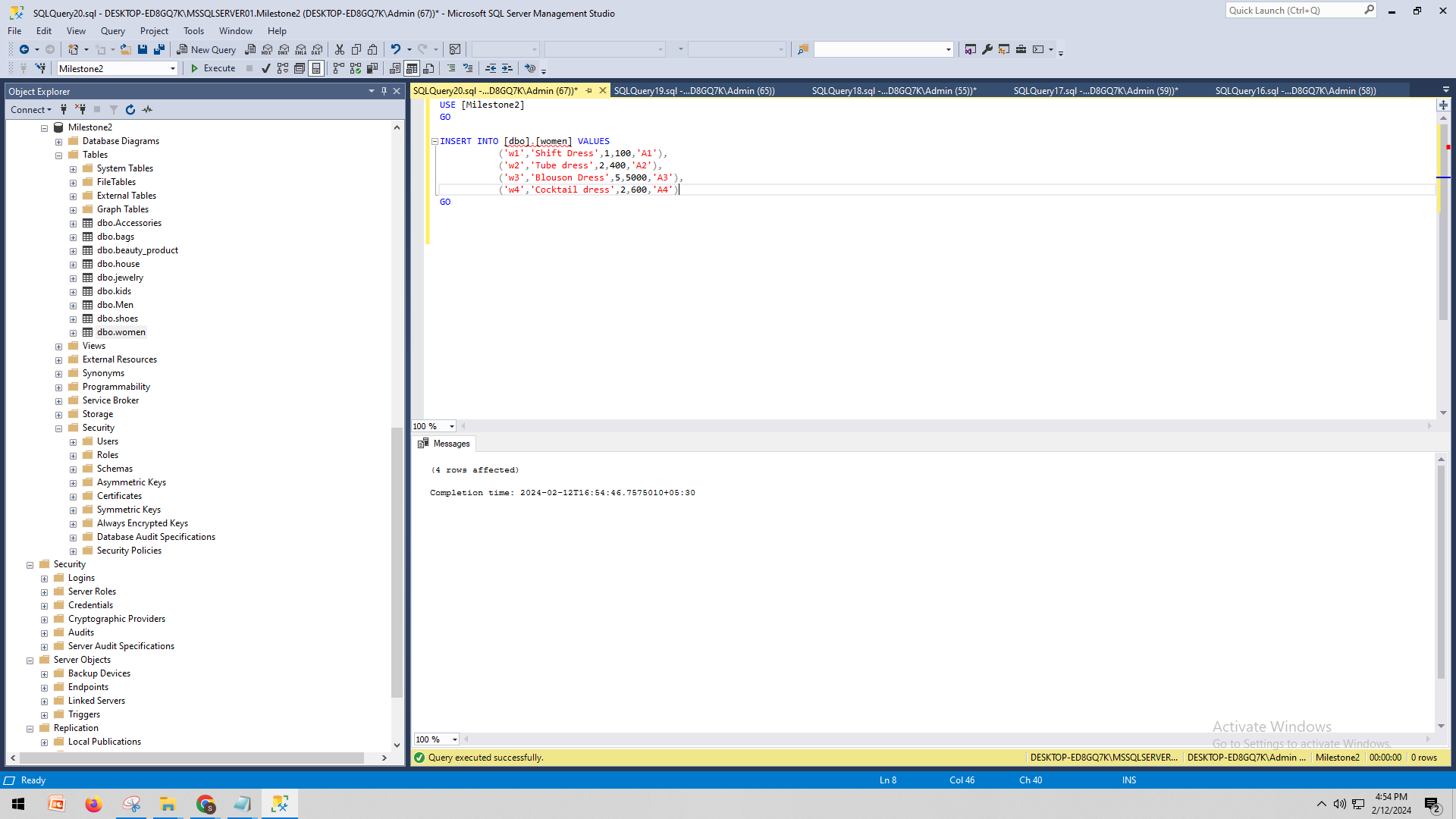


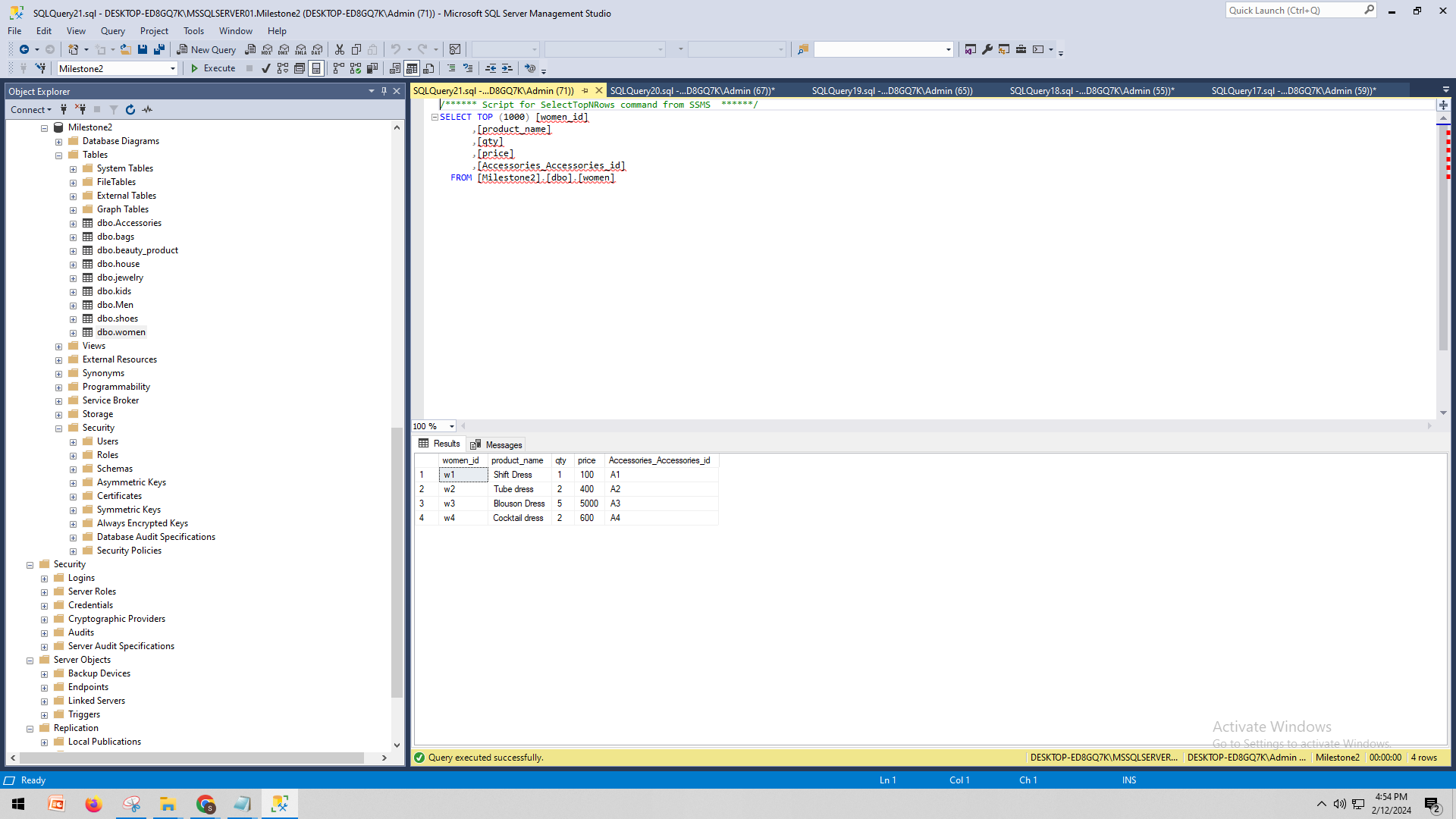












# H. Data Manipulation Language Scripts

* *1 point each for the two insert statements*

USE [Milestone2]

GO

INSERT INTO [dbo].[bags] VALUES

('b5','official\_bag',300.00,'american tourister','red','A1')

GO



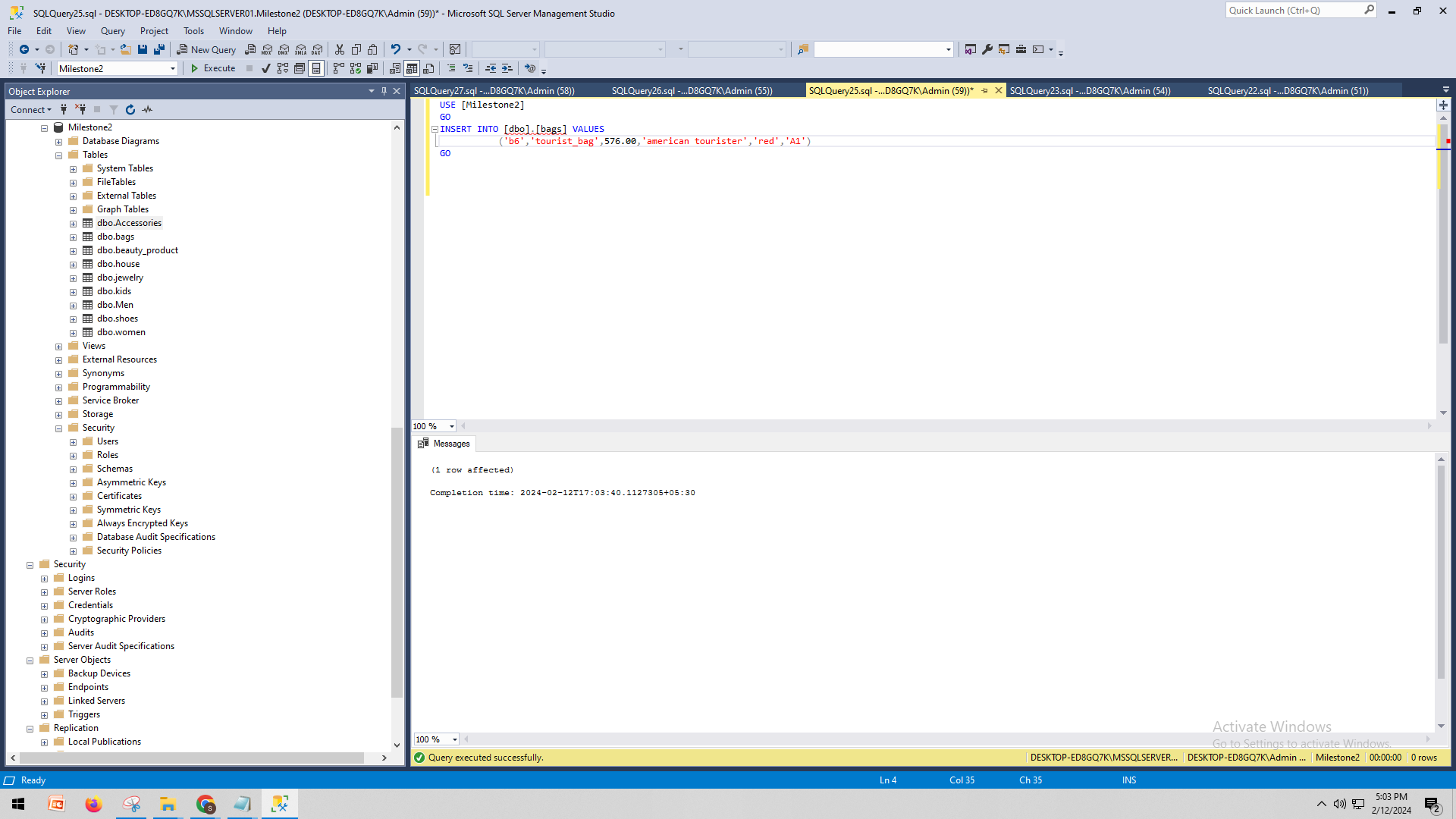
USE [Milestone2]

GO

INSERT INTO [dbo].[bags] VALUES

('b6','tourist\_bag',576.00,'american tourister','red','A1')

GO



* *1 point each for the two update statements*

*USE [Milestone2]*

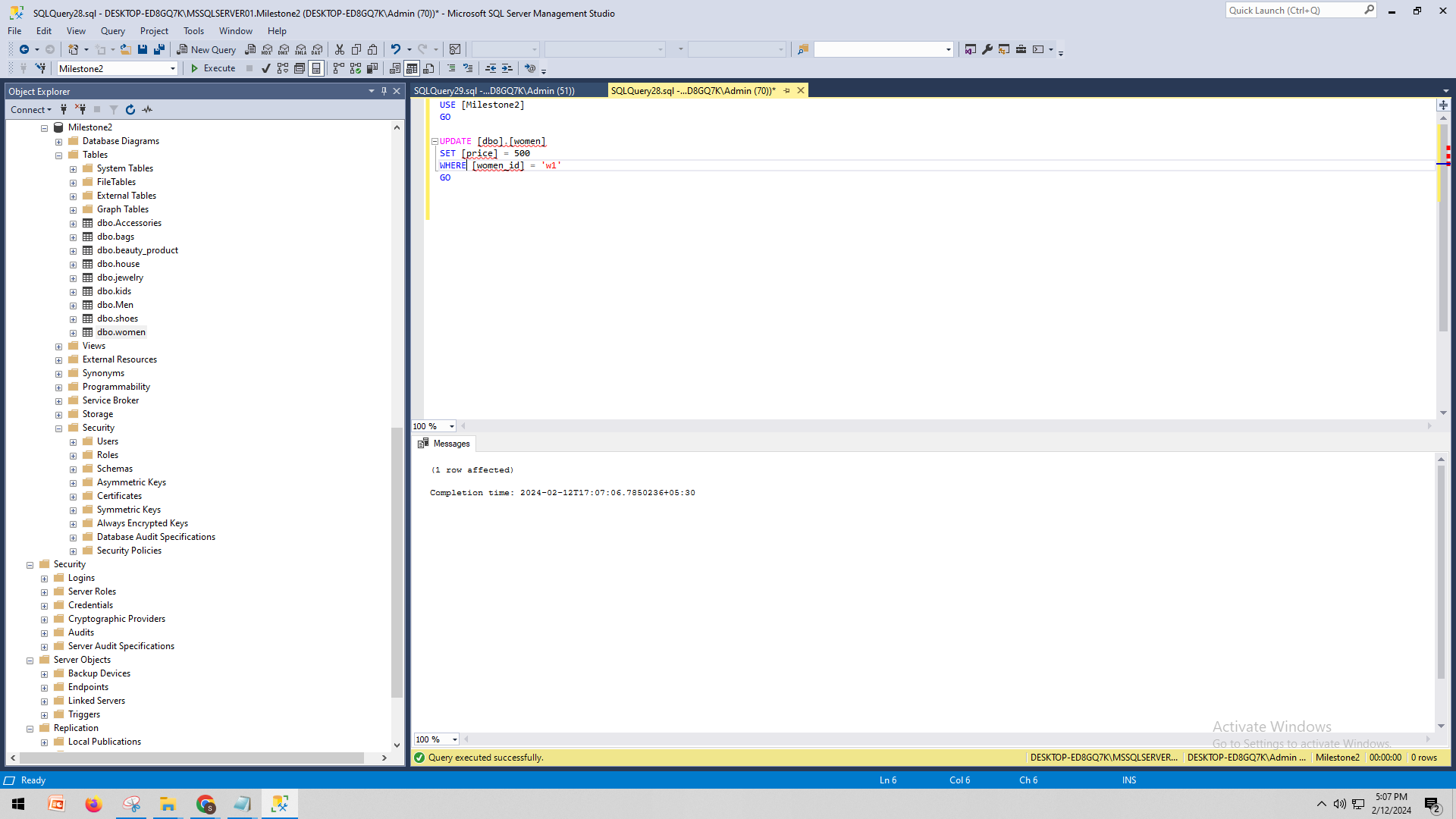
*GO*

*UPDATE [dbo].[women]*

*SET [price] = 500*

*WHERE [women\_id] = 'w1'*

*GO*

**

*USE [Milestone2]*

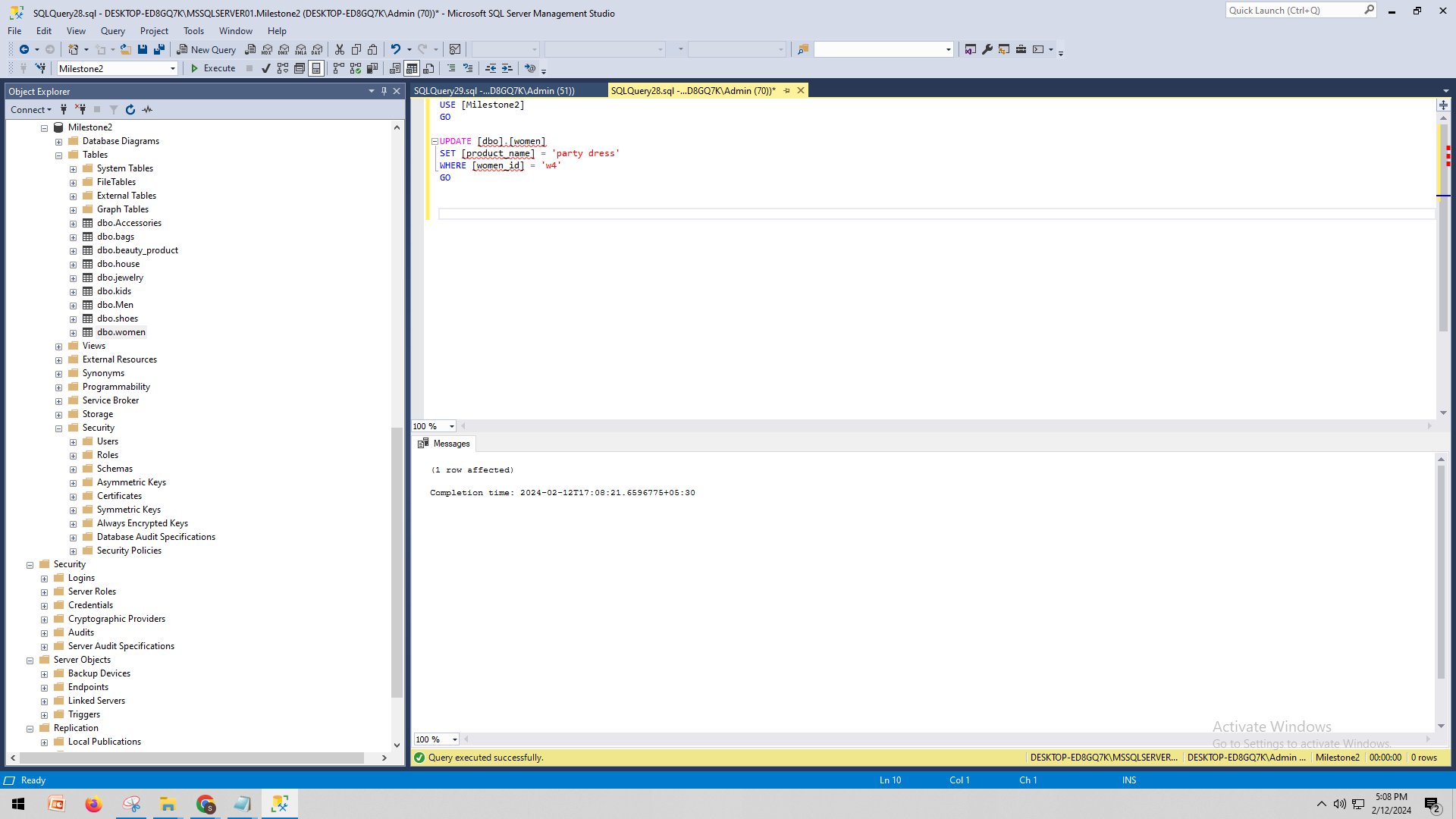
*GO*

*UPDATE [dbo].[women]*

*SET [product\_name] = 'party dress'*

*WHERE [women\_id] = 'w4'*

*GO*

**

* *1 point for the delete statement*

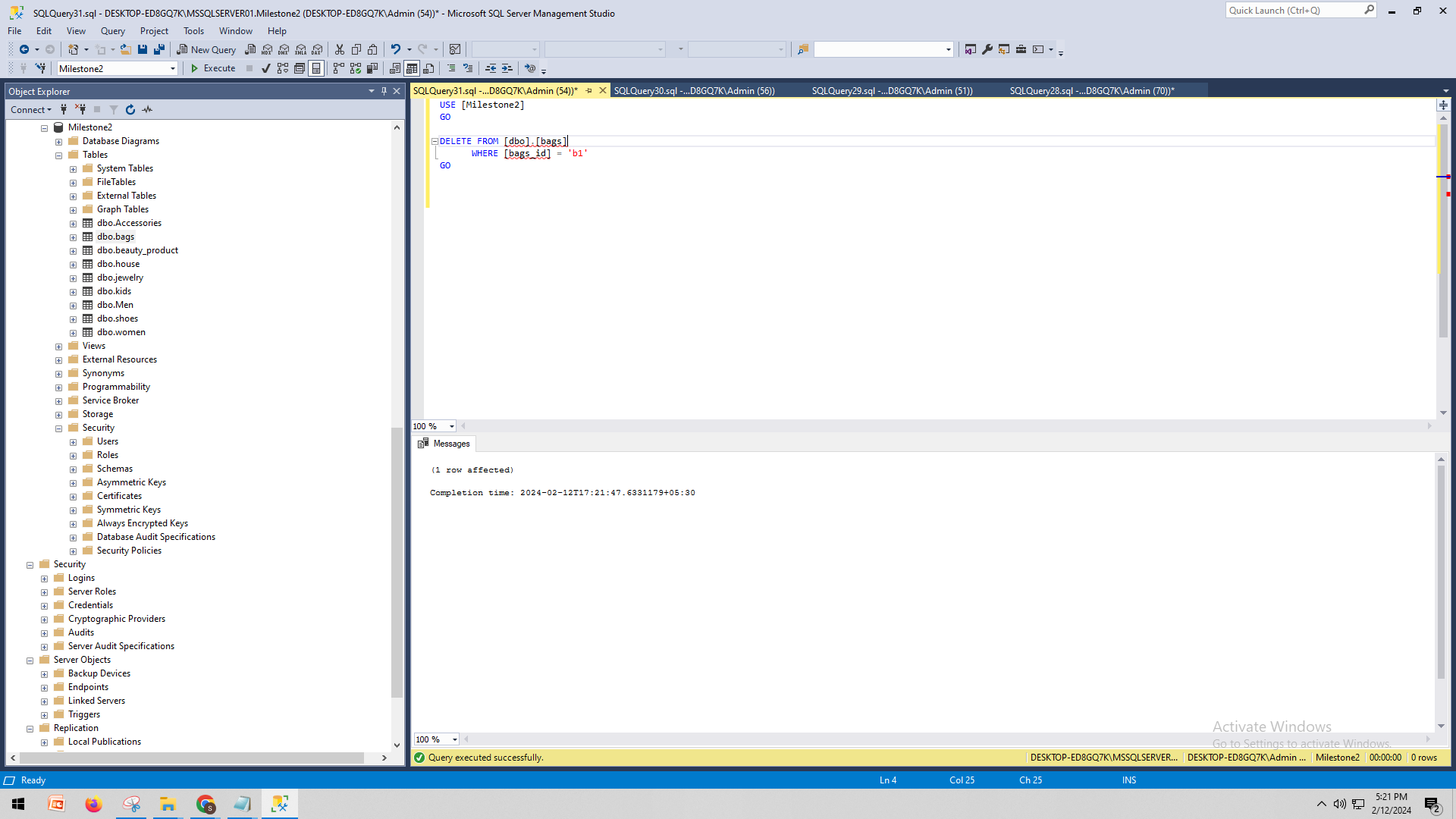
*USE [Milestone2]*

*GO*

*DELETE FROM [dbo].[bags]*

*WHERE [bags\_id] = 'b1'*

*GO*

**

* *1 point for the simple select statement*

SELECT [bags\_id]

,[Name]

,[Price]

,[brand]

,[color]

,[Accessories\_Accessories\_id]

FROM [Milestone2].[dbo].[bags]



* *2 points each for the 2 join statements*

*SELECT TOP (1000)*

*w.[women\_id],*

*w.[product\_name] AS women\_product\_name,*

*w.[qty] AS women\_qty,*

*w.[price] AS women\_price,*

*m.[men\_id],*

*m.[product\_name] AS men\_product\_name,*

*m.[qty] AS men\_qty,*

*m.[price] AS men\_price,*

*w.[Accessories\_Accessories\_id] AS women\_accessories\_id,*

*m.[Accessories\_Accessories\_id] AS men\_accessories\_id*

*FROM*

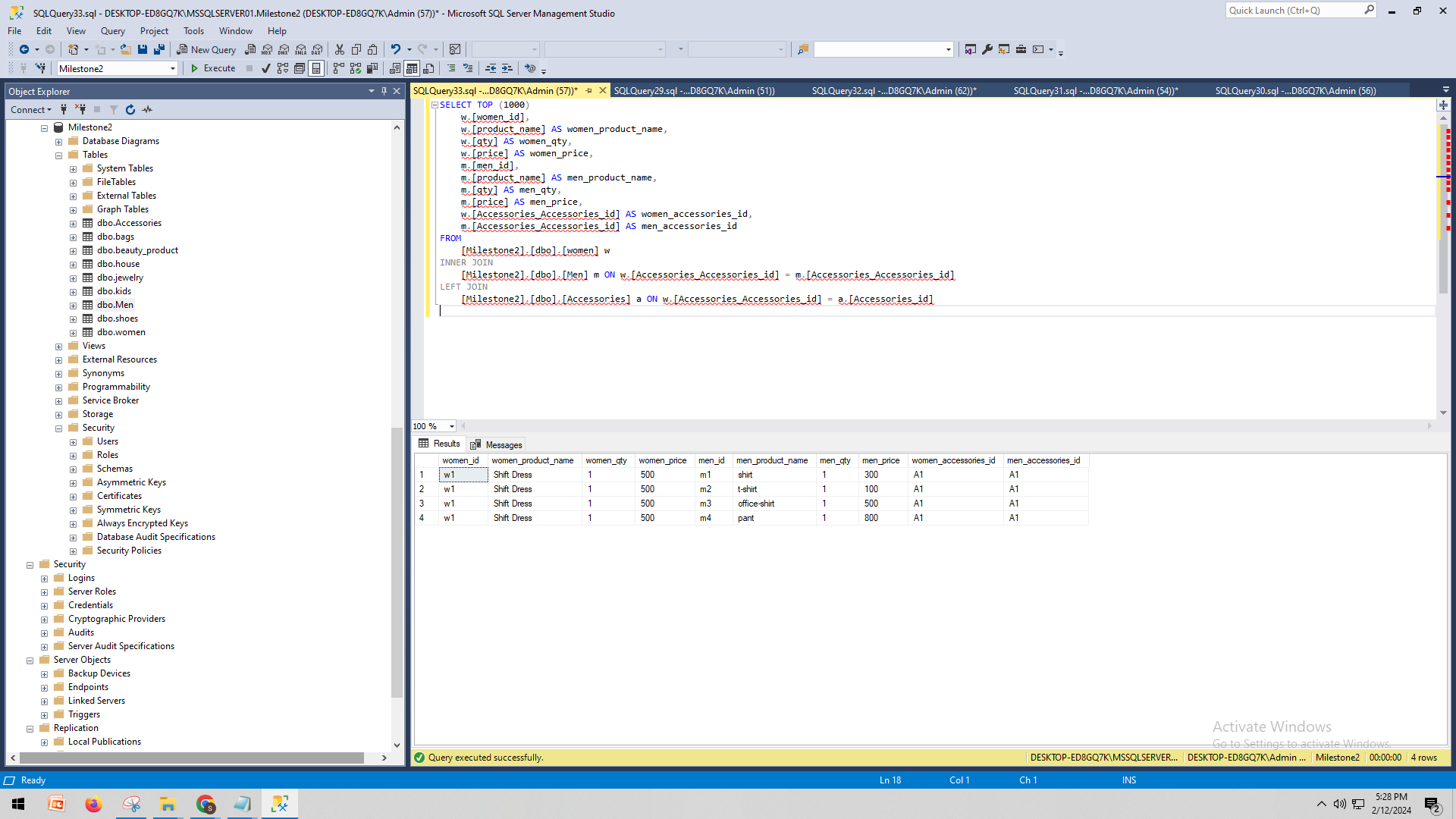
*[Milestone2].[dbo].[women] w*

*INNER JOIN*

*[Milestone2].[dbo].[Men] m ON w.[Accessories\_Accessories\_id] = m.[Accessories\_Accessories\_id]*

*LEFT JOIN*

*[Milestone2].[dbo].[Accessories] a ON w.[Accessories\_Accessories\_id] = a.[Accessories\_id]*

**

* *2 points each for the two that use summary statements*

*SELECT*

*COUNT([shoes\_id]) AS total\_shoes,*

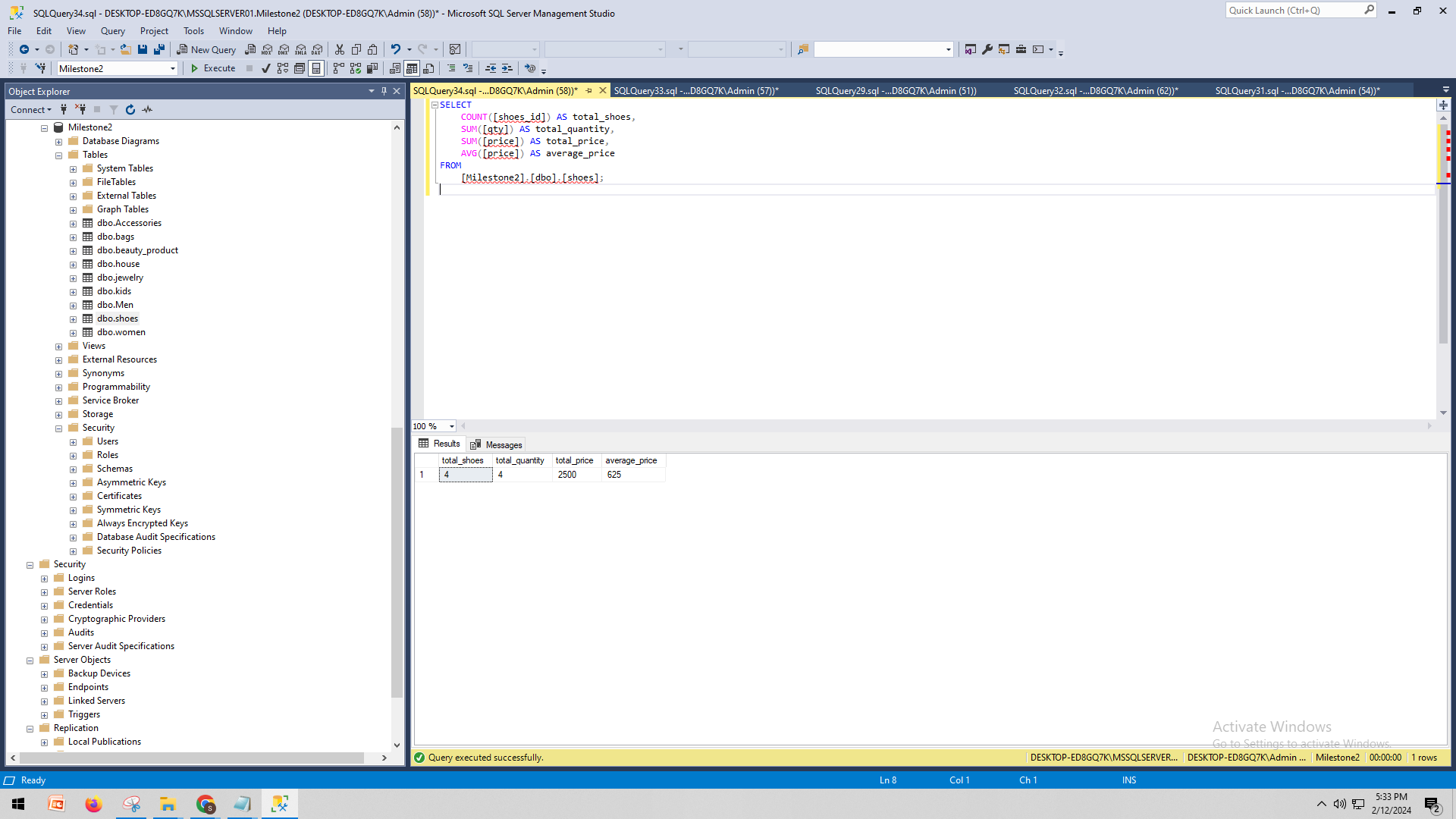
*SUM([qty]) AS total\_quantity,*

*SUM([price]) AS total\_price,*

*AVG([price]) AS average\_price*

*FROM*

*[Milestone2].[dbo].[shoes];*

**

*SELECT*

*COUNT([jewelry\_id]) AS total\_jewelry\_items,*

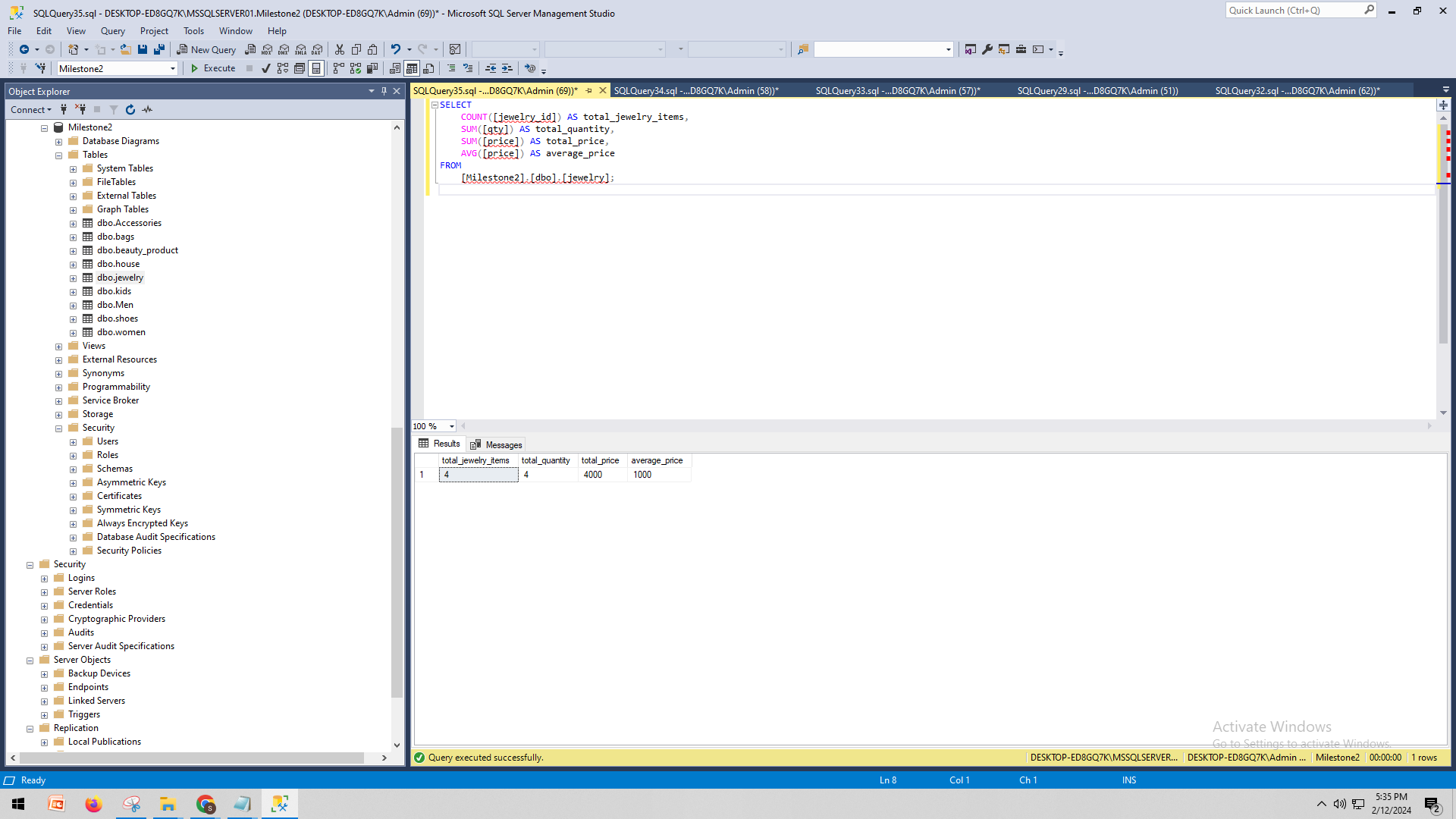
*SUM([qty]) AS total\_quantity,*

*SUM([price]) AS total\_price,*

*AVG([price]) AS average\_price*

*FROM*

*[Milestone2].[dbo].[jewelry];*

**

* *2 points for the multi-table query*

*SELECT*

*m.[men\_id] AS men\_id,*

*m.[product\_name] AS men\_product\_name,*

*m.[qty] AS men\_qty,*

*m.[price] AS men\_price,*

*w.[women\_id] AS women\_id,*

*w.[product\_name] AS women\_product\_name,*

*w.[qty] AS women\_qty,*

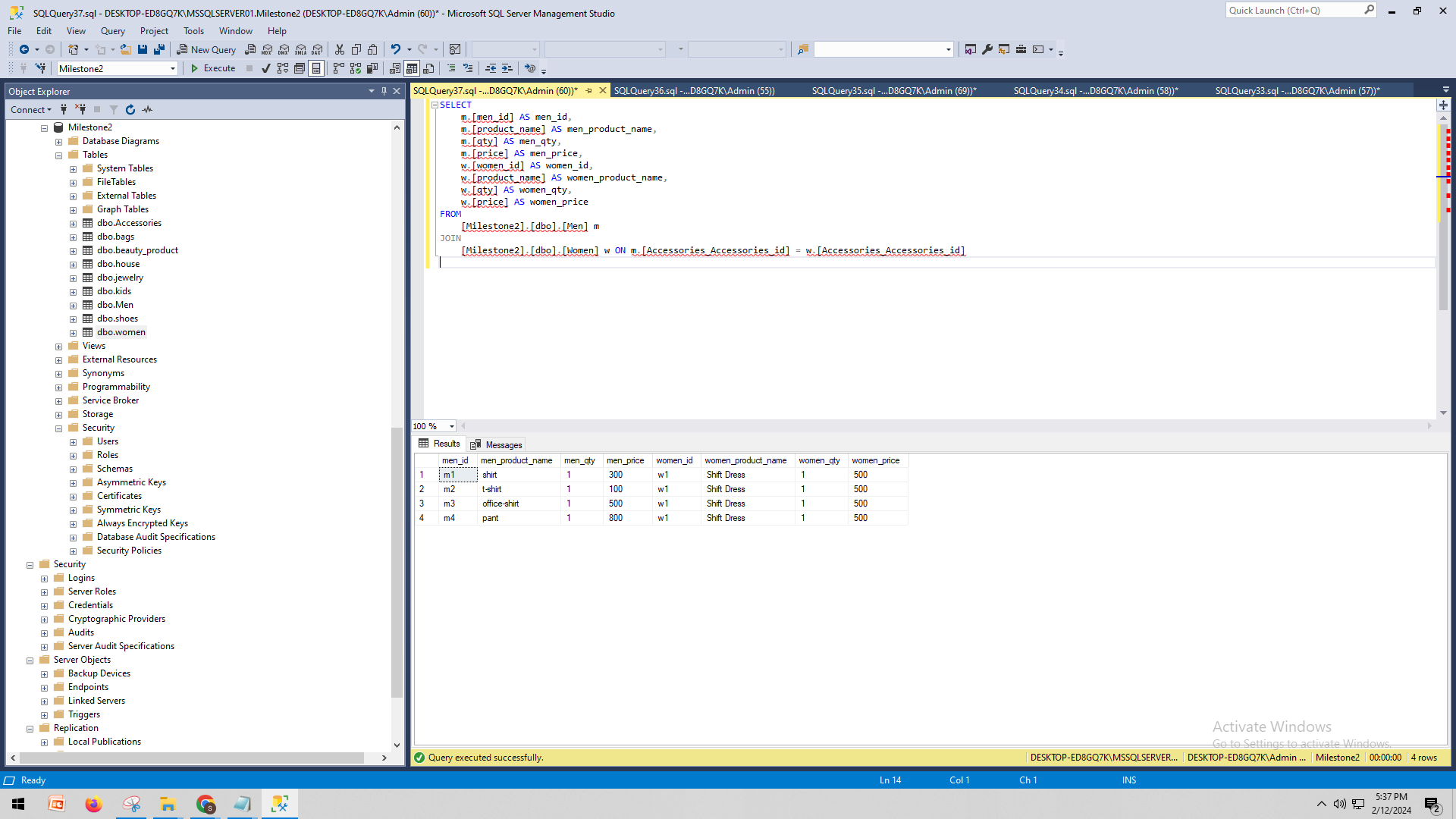
*w.[price] AS women\_price*

*FROM*

*[Milestone2].[dbo].[Men] m*

*JOIN*

*[Milestone2].[dbo].[Women] w ON m.[Accessories\_Accessories\_id] = w.[Accessories\_Accessories\_id]*

**

* *2 points for the query of your choice.*

*SELECT TOP (1000) [beauty\_id]*

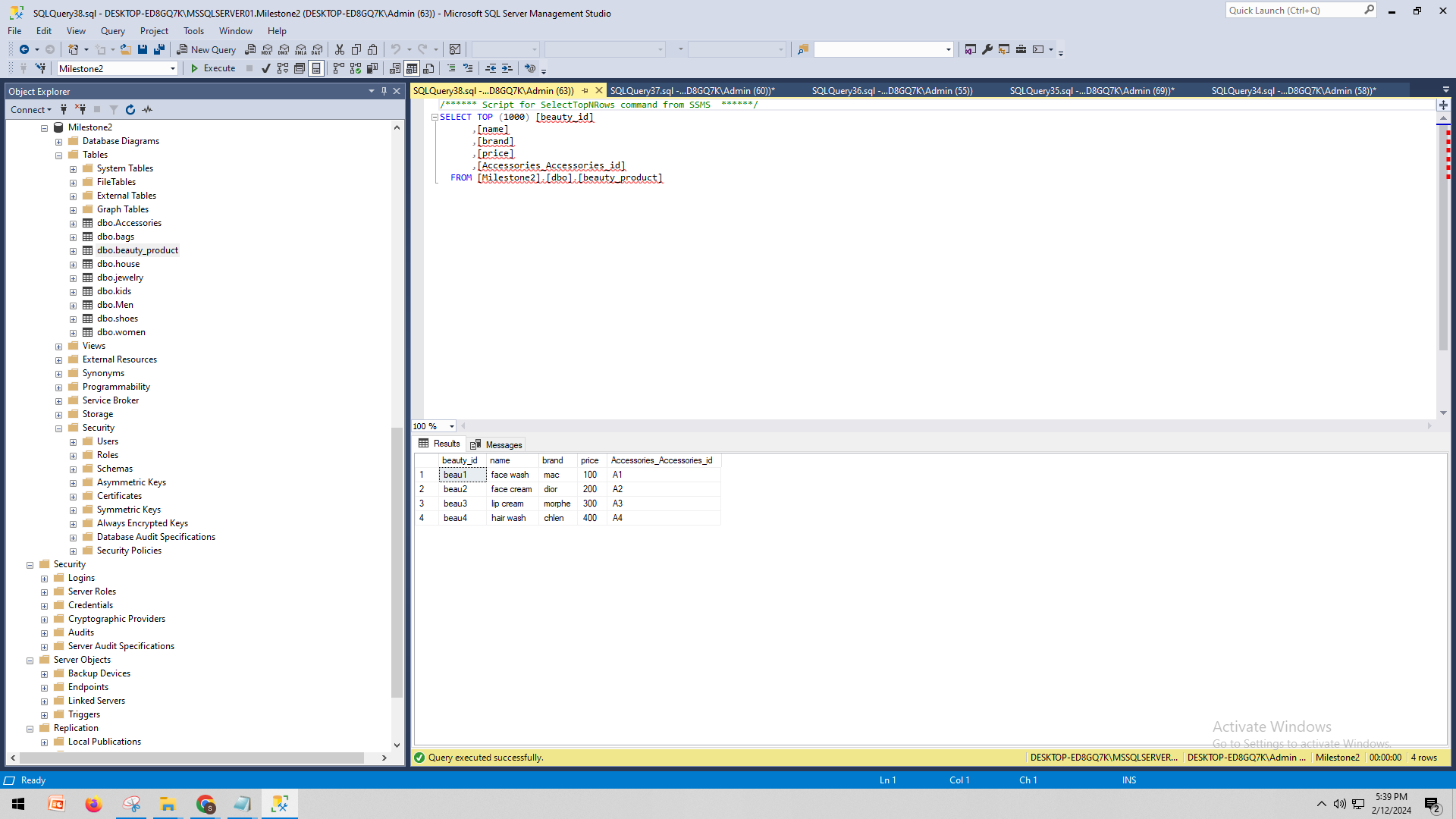
*,[name]*

*,[brand]*

*,[price]*

*,[Accessories\_Accessories\_id]*

*FROM [Milestone2].[dbo].[beauty\_product]*

**

*SELECT TOP (1000) [house\_id]*

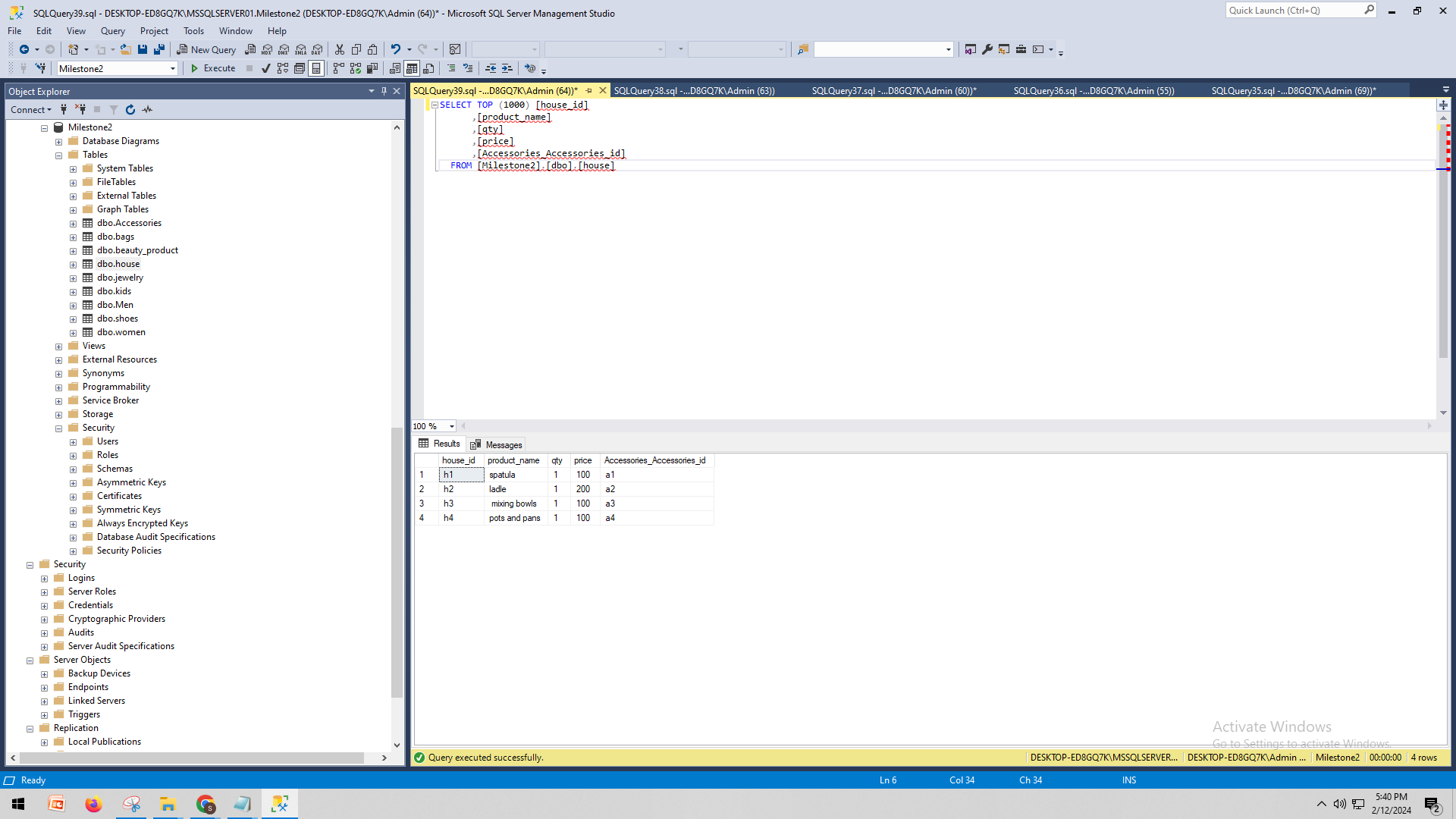
*,[product\_name]*

*,[qty]*

*,[price]*

*,[Accessories\_Accessories\_id]*

*FROM [Milestone2].[dbo].[house]*

**

# I. Indexes

*Description: Improve the performance of your design by adding indexes to various tables. Show the SQL needed to add the indexes. Explain why you chose the ones you added. Explain how you would demonstrate the impact the indexes had on the performance of various queries.*

*Rubric: Your work will be graded as follows:*

* *3 points for clearly defining at least three indexes and explaining why you chose them.*
* *3 points for showing the sql needed to generate the indexes*
* *2 points for explaining how you would demonstrate the performance improvement afforded by the indexes.*

*Total points possible: 8*

ENTER YOUR INDEX WORK HERE

# J. Views

*Description: Add two views to your database to provide easy access to combinations of data from multiple tables.*

*Rubric: Your work will be graded as follows:*

* *2 points for including the SQL for generating the two views in your Word document*
* *2 points for including screenshots for the data contained in each view in your Word document*
* *2 points for explaining why each view is a valuable addition to your database*

*Total points possible: 6*

ENTER YOUR WORK WITH VIEWS HERE

# K. Triggers

*Description: Add a trigger to a table so that data will be updated when a certain event occurs*

*Rubric: Your work will be graded as follows:*

* *2 points for including the SQL for the trigger in your Word document*
* *2 points for clearly explaining the purpose of the trigger*
* *2 points for a screenshot and explanation that shows the trigger in action.*

*Total points possible: 6*

ENTER YOUR WORK WITH TRIGGERS HERE

# L. Transactions

*Description: Demonstrate that you know how to define and use a transaction. Why are transactions important for ensuring ACID behavior?*

*Rubric: Your work will be graded as follows:*

* *3 points for clearly explaining the importance of transactions to ensuring ACID behavior*
* *3 points for including a screenshot and accompanying explanation of a MySQL transaction.*

*Total points possible: 6*

ENTER YOUR WORK WITH TRANSACTIONS HERE

# M. Database Security

*Description: Identify the different kinds of users who will use your database. Write GRANT statements to define the privileges for these different kinds of users.*

*Rubric: Your work will be graded as follows:*

* *4 points for clearly identifying and describing the various kinds of users who will use the databases and identifying and justifying what privileges each should have.*
* *4 points for writing GRANT statements that assign privileges to these different kinds of users.*
* *4 points for demonstrating with screenshots that your GRANT statements do distinguish among different kinds of users in regard to what they can do with the database.*

*Total points possible: 12*

ENTER YOUR WORK WITH DATABASE SECURITY HERE

# N. Locking and Concurrent Access

*Description: Explain the purpose of locking tables and show how to do that to prevent inconsistencies that may arise in your data when concurrent transactions take place.*

*Rubric: Your work will be graded as follows:*

* *3 points for clearly explaining an example that shows why you should lock tables to prevent inconsistencies.*
* *2 points for providing a screenshot and accompanying explanation of locking tables.*

*Total points possible: 5*

ENTER YOUR WORK WITH LOCKING AND CONCURRENT ACCESS HERE

# O. Backing Up Your Database

*Description: How you will back up your database. What commands will you issue? How frequently will the commands run? How can they be automated? Where will the backups be stored?*

*Rubric: Your work will be graded as follows:*

* *6 points for clearly explaining and justifying your database backup strategy, including the frequency with which you will back up the database, how you will automate backups, where you will store them, and how you will secure them. You will earn three points for addressing each factor (frequency, location, automation, and security)*
* *2 points for providing a screenshot of the command you would issue to back up the database and for including a portion of the resulting file.*

*Total points possible: 8*

ENTER YOUR WORK ON DATABASE BACKUPS HERE

# P. Programming

*Description: Write a Python, Java, or PHP program that generates a report that contains a subset of the data from your database. Include the code for your Python program in your Word document, and also post the program to your GitHub repository.*

*Rubric: Your work will be graded as follows:*

* *10 points for writing a Python script (and including its code in the Word doc) that will pull data from a database and store it to a text file and present it to the screen. Your code must have comments in it that explain how it works. You will be awarded 3 points for successfully connecting to the database, 3 points for successfully querying it, and 4 points for presenting the data to the screen and to a file. Internal comments count for 2 points.*
* *2 points for posting the code to GitHub*
* *6 points for showing a screenshot of your running the script and showing the results it produces on the screen.*

*Total points possible: 18*

ENTER YOUR PYTHON, PHP, or JAVA DATABASE PROGRAMMING WORK HERE

# Q. Suggested Future Work

*Description: Describe the limitations of your current database and explain how you or someone else could improve the design to address these shortcomings. Also describe how you might take advantage of leverage cloud services to increase the performance and availability of your database. Finally, explain the advantages and disadvantages of storing your data in a NoSQL format instead.*

*Rubric: Your work will be graded as follows:*

* *3 points for clearly describing the limitations of your databases*
* *3 points for explaining how you would address these shortcomings*
* *3 points for explaining how you might migrate the database to the cloud and describing what advantages you might gain from doing that.*
* *3 points for explaining the advantages and disadvantages of storing your data in a document-based NoSQL format instead.*

*Total points possible: 12*

ENTER YOUR SUGGESTED FUTURE WORK IDEAS HERE

# R. Activity Log

*Description: As an appendix, the team will keep a frequently updated diary or log of their activity. What did you or your team study in this class each day? What did you learn? What did you accomplish or build or design? You don't have to enter something every day, but there should be at least three entries each week. Since I have eight weeks, that means you should make 3 posts to the Activity Log each week, for a total of at least 24 posts. Each post will be worth 1 point.*

*If you are working as part of a team, make sure you clearly identify which team member worked on which tasks. The Activity Log should help me figure out how each team member contributed to the project. If I cannot discern who worked on what aspects of the project from the activity log, no points will be awarded for it.*

*Total points possible: 24*

MAKE AT LEAST THREE ENTRIES PER WEEK. CLEARLY IDENTIFY WHAT EACH PERSON ON YOUR TEAM ACCOMPLISHED. YOU MUST SHARE THE RESPONSIBILITY OF COMPLETING THE PROJECT.

**Week 1**

Day 1:

Discussed project requirement and responsibility.

Research e-commerce data structures.

Research potential technology for database implementation.

Day 3:

In meeting, share findings and decide on database schema.

Drafted the initial proposal document.

Explored security considerations for the database.

Day 5

Work on the data fields for each product category.

Research best practices for user authentication.

**Week 2**

Day 1:

Reviewed and finalized the database schema.

Explore the data migration strategies.

Day 3:

Began database design and develop initial tables.

Investigate all potential API integrations.

Day 5:

Discussed progress and addressed roadblocks.

Explore options for hosting databases.

Drafted project title.

**Week 3**

**Day 1:**

Manage detailed review of initial database design.

Investigate normalization techniques to ensure its integrated data.

To find query performance research on index strategies.

**Day 3:**

Based on the feedback collaboratively worked on refining the database scheme.

Investigate options to implement translation and ensure data consistency.

Discussed potential challenges to mitigated strategies.

**Day 5:**

Within the database scheme, implement primary and foreign key constraints.

Manage code review sessions to ensure consistency.

For the database investigate backup and recovery strategies.

**Week 4**

**Day 1:**

To enhance the database security examine data encryption methods.

In the database investigate methods to manage concurrency and locking.

Discuss potential challenges and their solutions.

**Day 3:**

Managing load testing of initial database design to find performance bottlenecks.

Discuss the initial version of user documentation.

Collaborate on API integration plan.

**Day 5:**

Review progress until now and discuss potential improvement.

Concluded decision of the hosting provider for a database.

Explore e-commerce database.