## **Assignment 1**

Student Name 1: Heja Bibani

Student ID 1: 16301173

Student Name 2: Marcus Belcastro

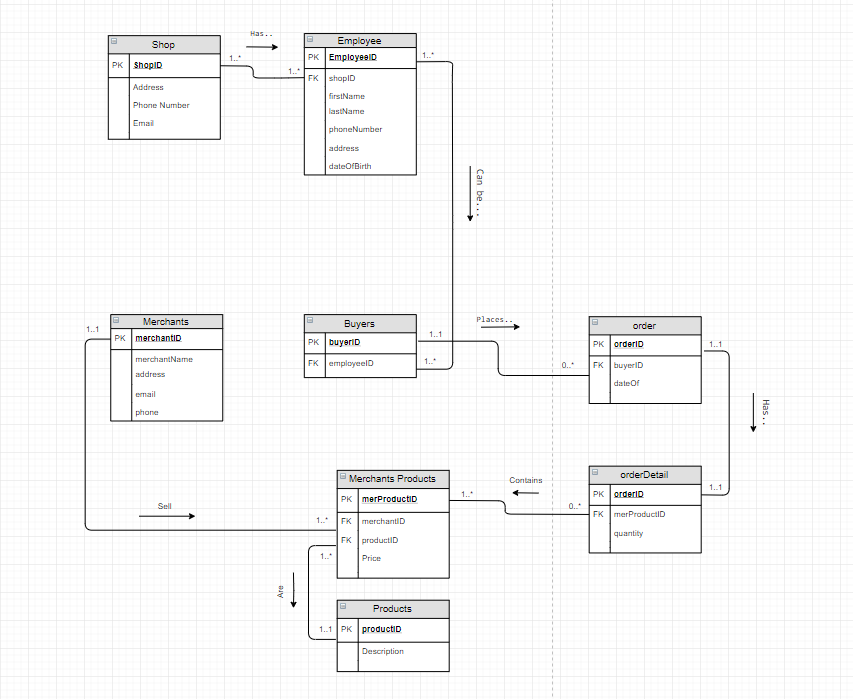
Student Id 2: 19185398

**Declaration:** This statement here declares that this submission is fully the original works of Marcus Belcastro and Heja Bibani.

## **Question 1**

**Logical(diagram):**

We have read through the process about how to complete a GRD and the logical diagram was necessary to produce it. We have followed this same process, and in this case, the logical diagram played a big role during the production of the GRD, and therefore we found that it was absolutely necessary that we included the logical diagram even though it wasn’t explicitly stated. You should see this through question (i) and (ii).

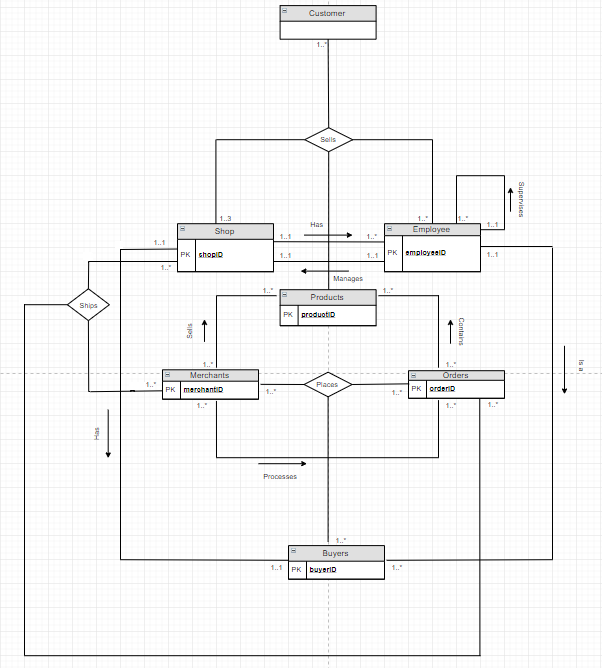
****

1. **ER diagram [two diagrams]**

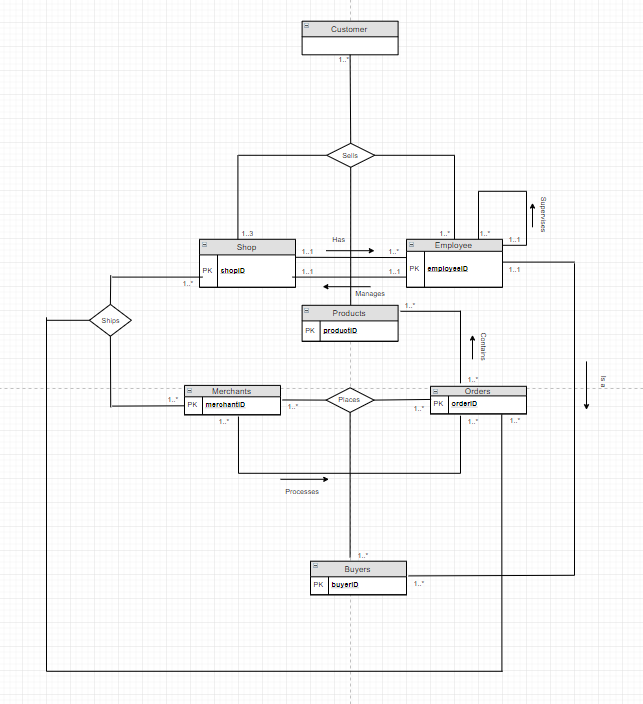
**Topic Redundancy Check:**

We are aware that for the logical diagram; and the logical design of the database, redundancy cannot be avoided, and must be acted upon; because this can cause some actual concerns when dealing with the database (which can cause significant problems). However, within the context of a conceptual diagram, there would be no technical issues with regards to adding “relationships” which are considered redundant, since this would not have a direct impact on the database itself. Then in this case, we are aware that leaving redundancy between relationships is not technically “harmful”. We know that removing redundancy (for conceptual diagrams) is a recommended practice in the industry and have seen this step recommended in lecture 7. Even though that this may be good practice; together, we have addressed the issue that certain information will be missing in the diagram. Then in this case, there are some disadvantages when practicing redundancy on the relationships in conceptual diagrams. Removing redundancy, assumes that the person looking at the diagram is going to make the correct assumptions. In one of our diagrams, we removed the Merchant “sells” products, because this is implied through the participation of ‘processing orders’; and orders ‘contains’ products. When we remove redundancy, the verbs are being replaced with assumptions, and in this case, the diagrams will have less semantic volume (as it’s being translated); since there is less words (verbs). I understand that it is possible, that the owner, or the representatives of the company who are requesting these types of services, may decide that the redundancy check is unnecessary (for conceptual diagrams), and from my perspective, I thought it was very relevant to keep such information. In this case, we have left the diagrams with and without redundancy in separate folders; as a way to demonstrate that we understand the idea.

**With Redundancy:**



**Without Redundancy:**



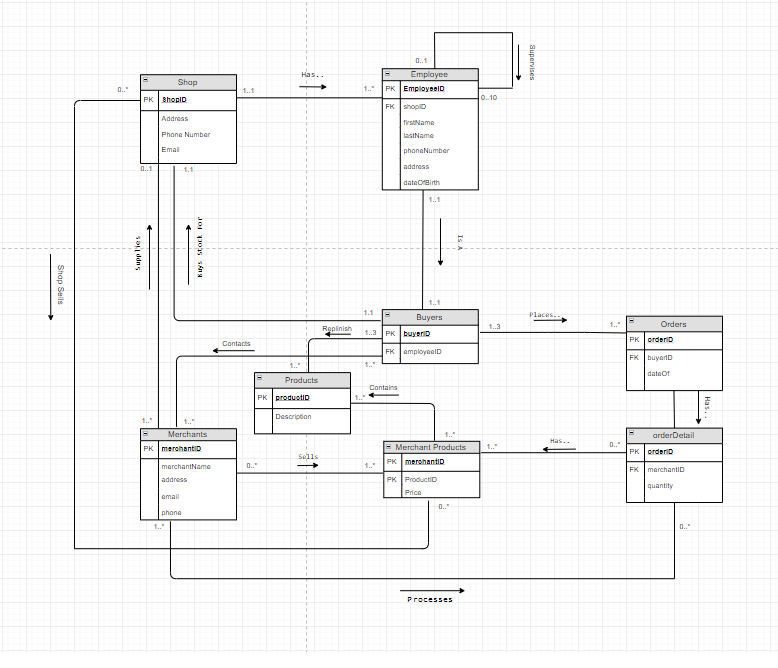
1. **GRD [Refer Logical diagram above]**

We have completed two GRD’s; with the intention of giving an option to the owner; if he has a preference (and for practice). In our case, Marcus felt that option one of the diagrams was better than the other, but I felt that they were around the same; and we could imagine that this type of issue could be produced when we are actually interacting with a business.

**Logical(diagram):**We have read through the process about how to complete a GRD and the logical diagram was necessary to produce it. We have followed this same process, and in this case, the logical diagram played a big role during the production of the GRD, and therefore we found that it was absolutely necessary that we included the logical diagram even though it wasn’t explicitly stated. Please refer to the start of the question for the logical diagram that we have completed.  
  
Option 1:

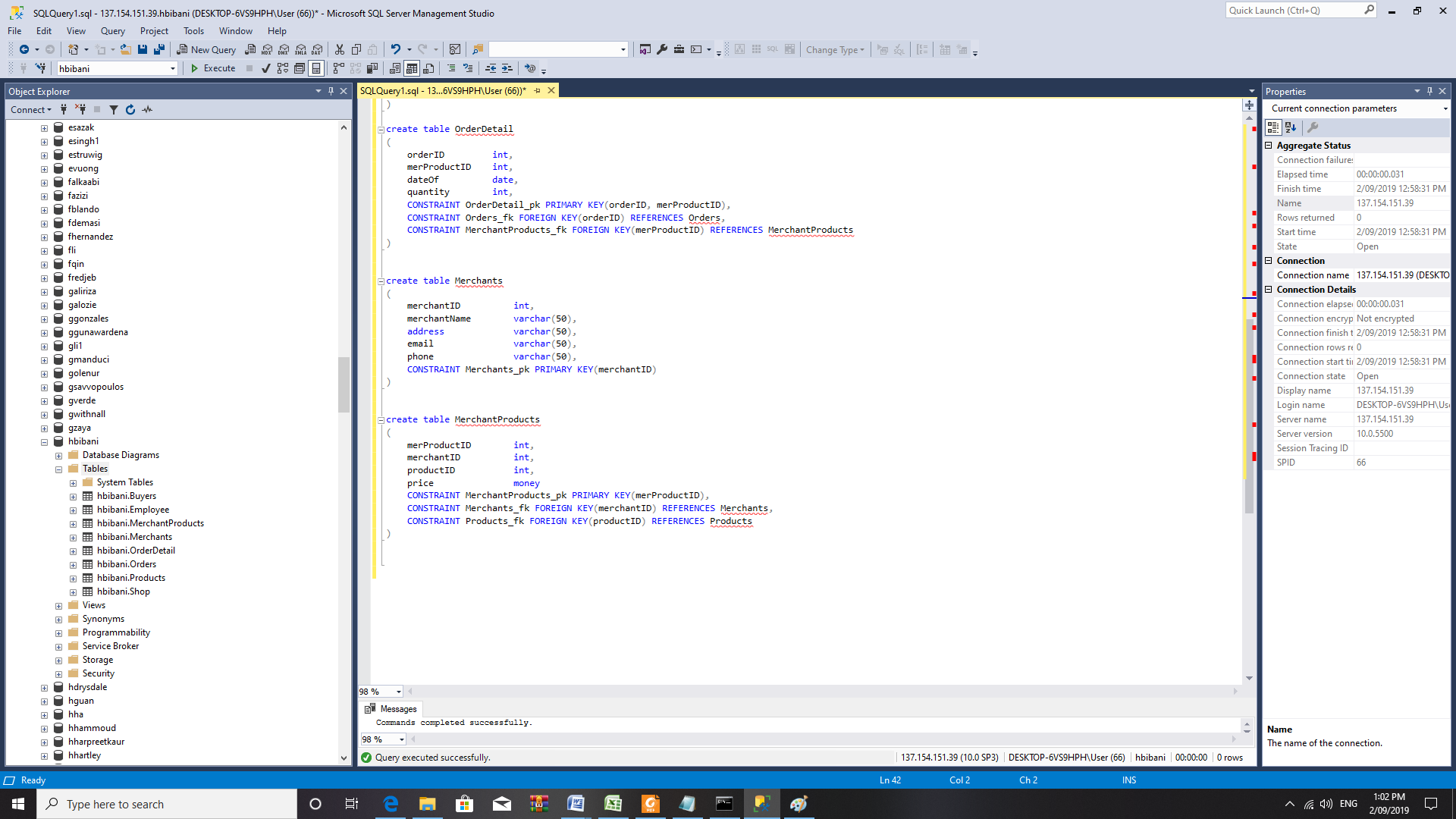
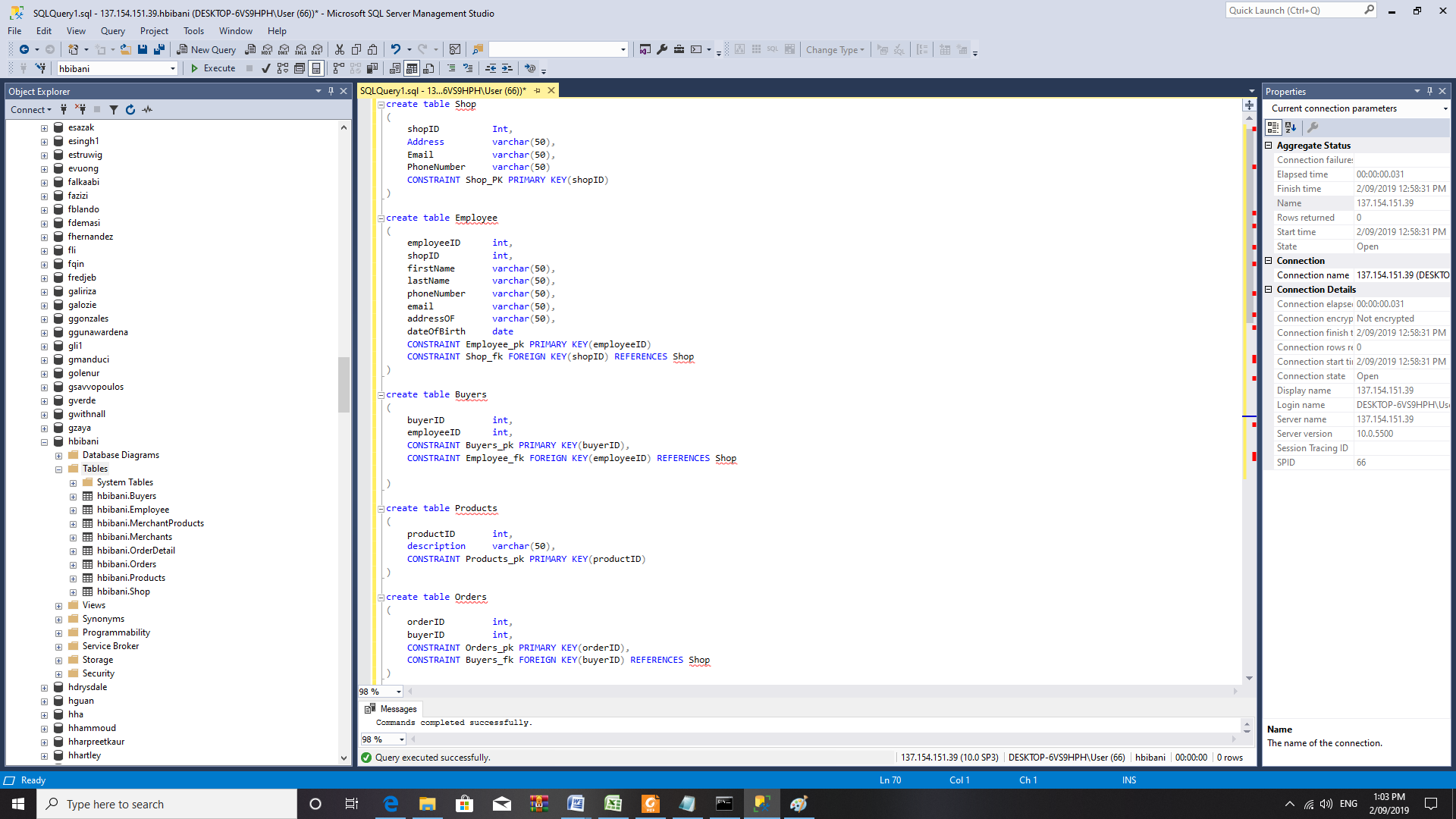


Option 2:



1. **Create Table[two pictures] and Create Statements[SQL]:**

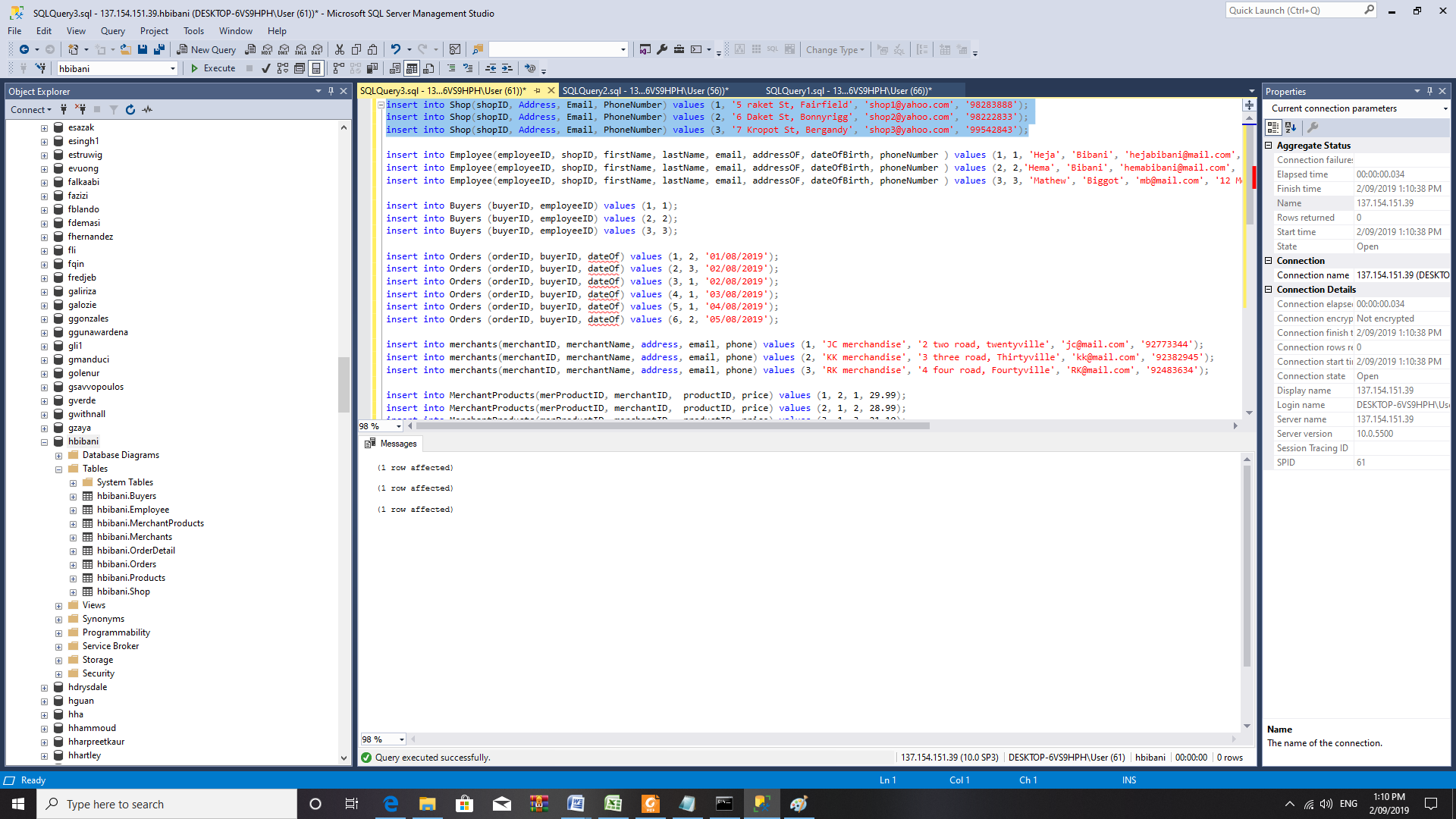


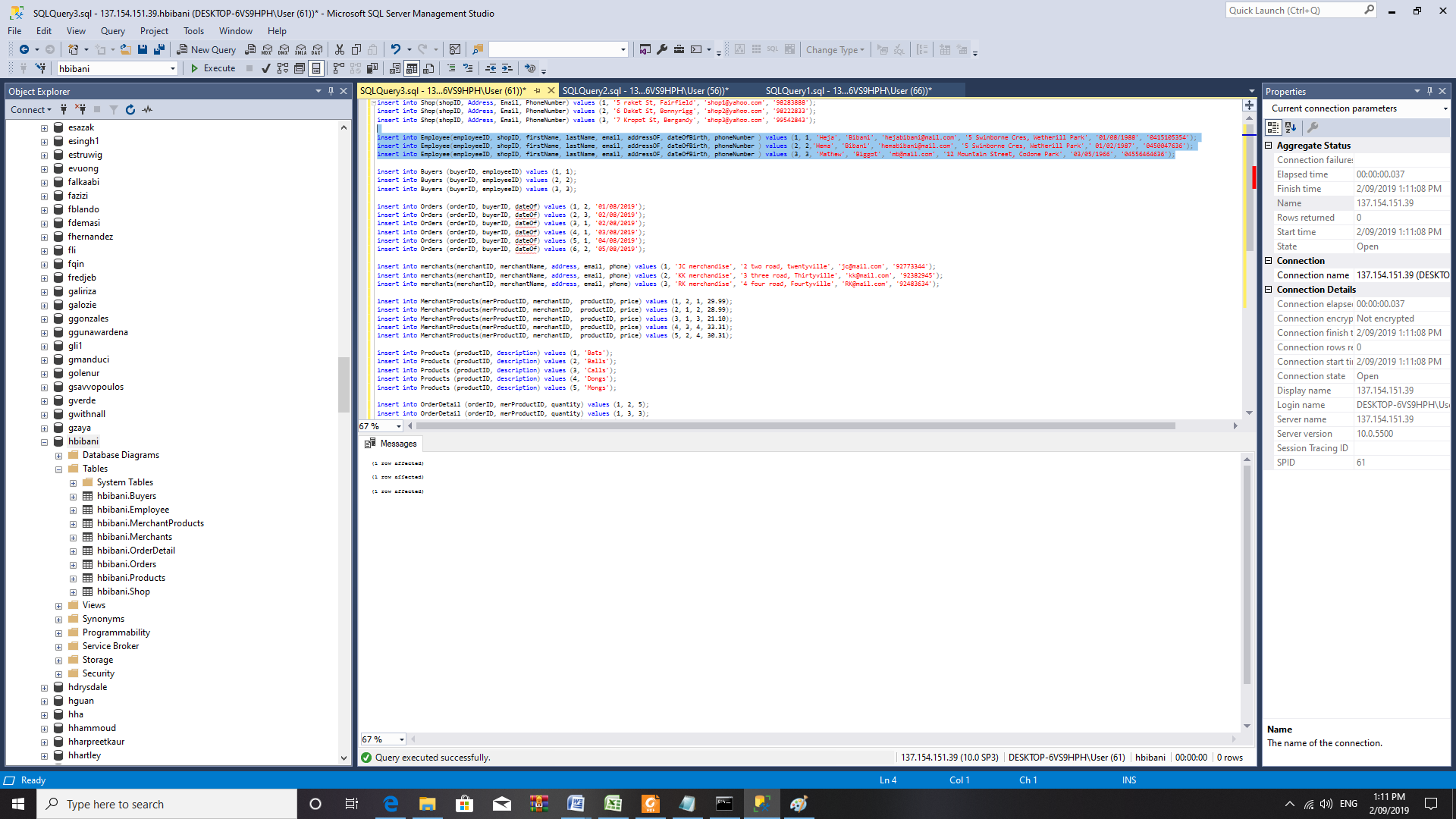


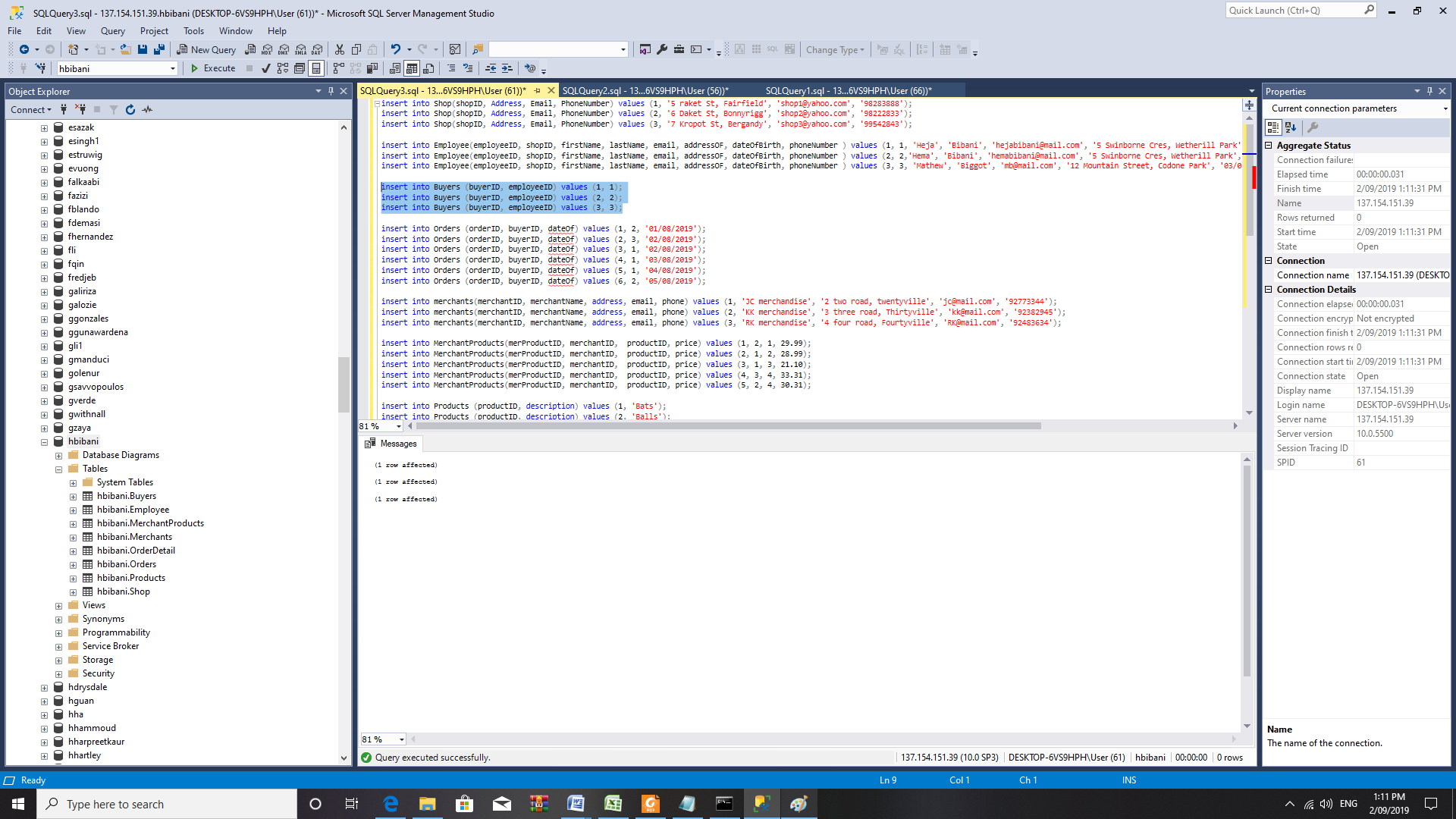
1. **Insert Statement [screenshots and SQL statement] and SELECT statement**

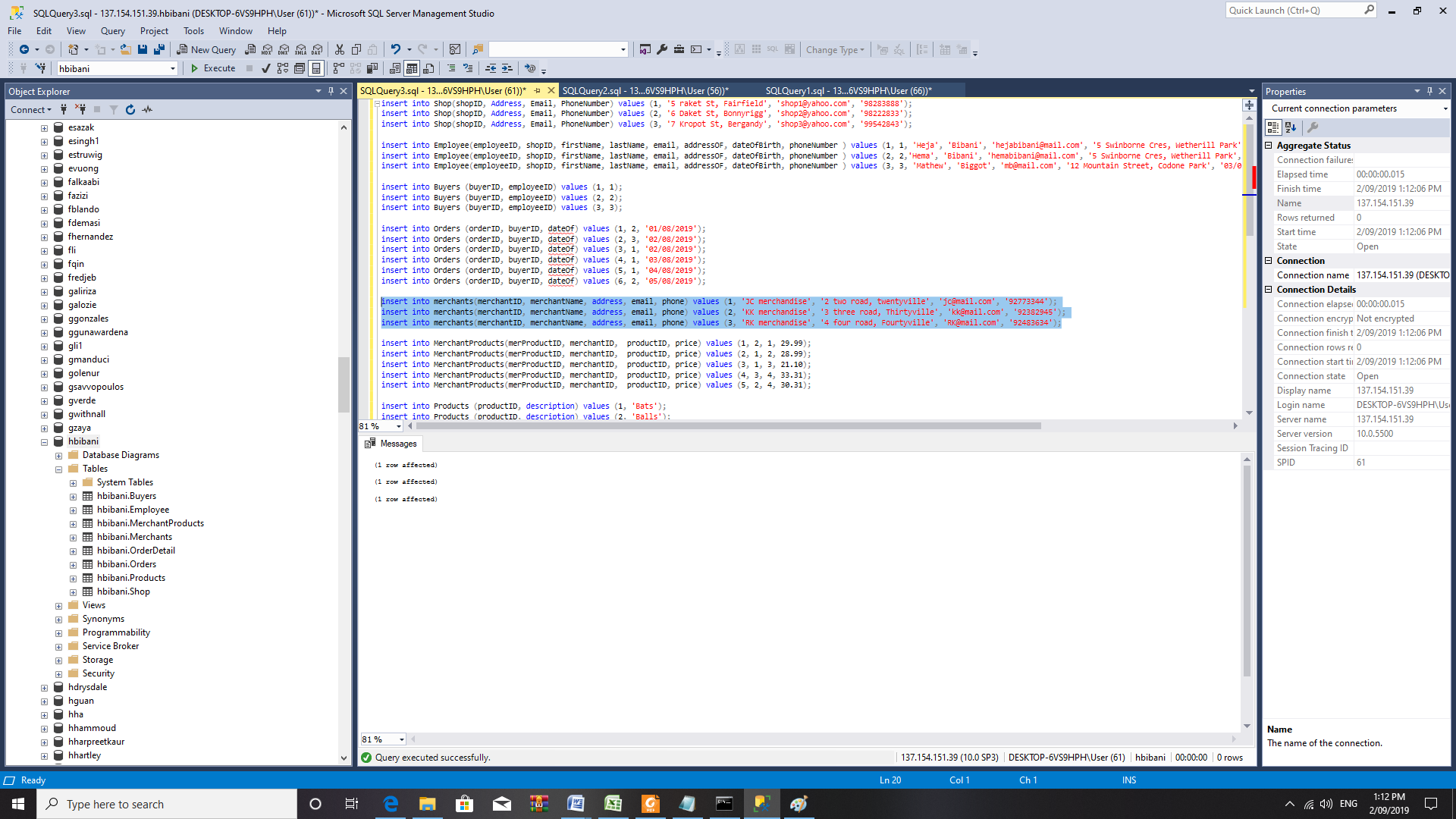
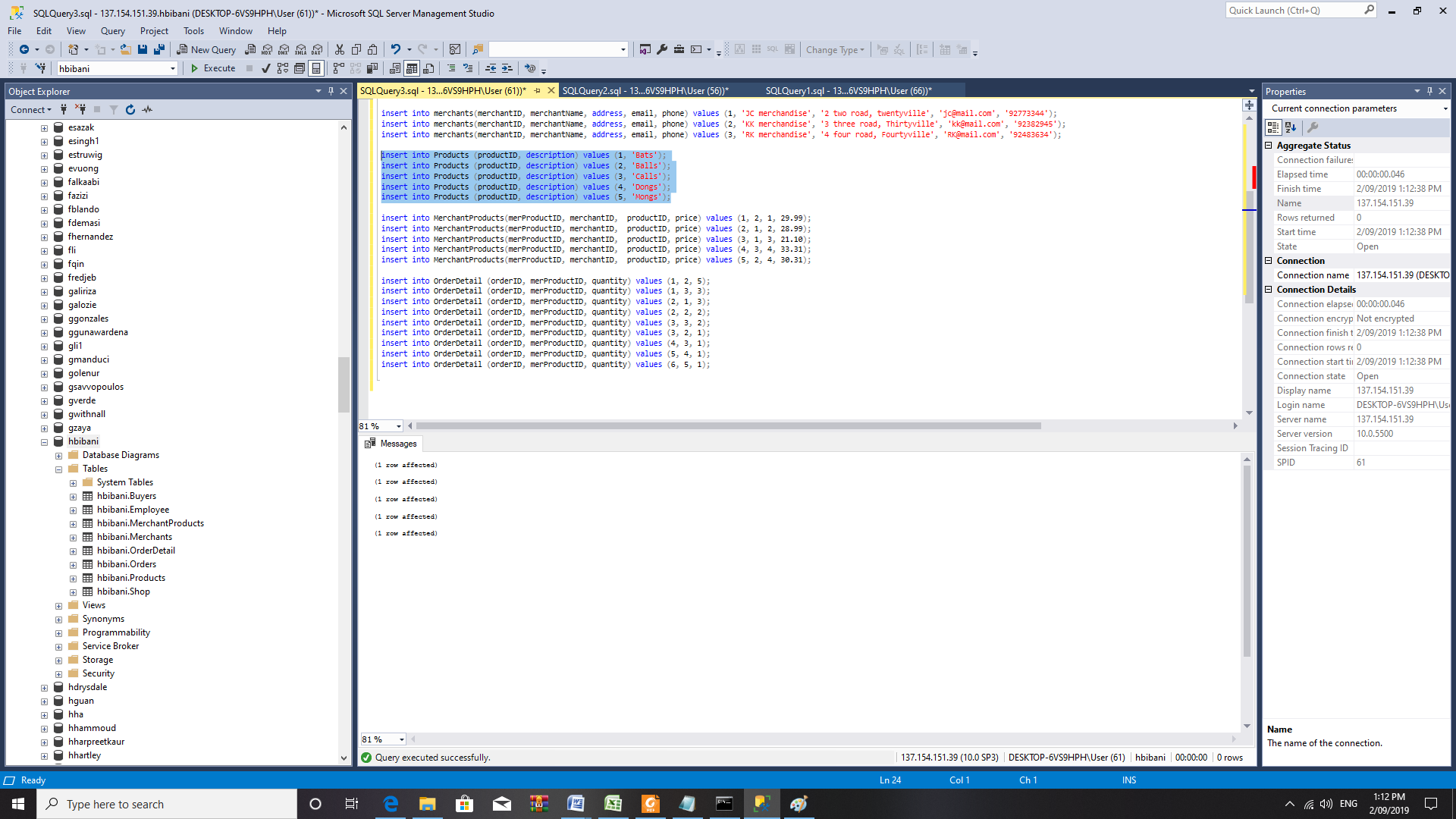


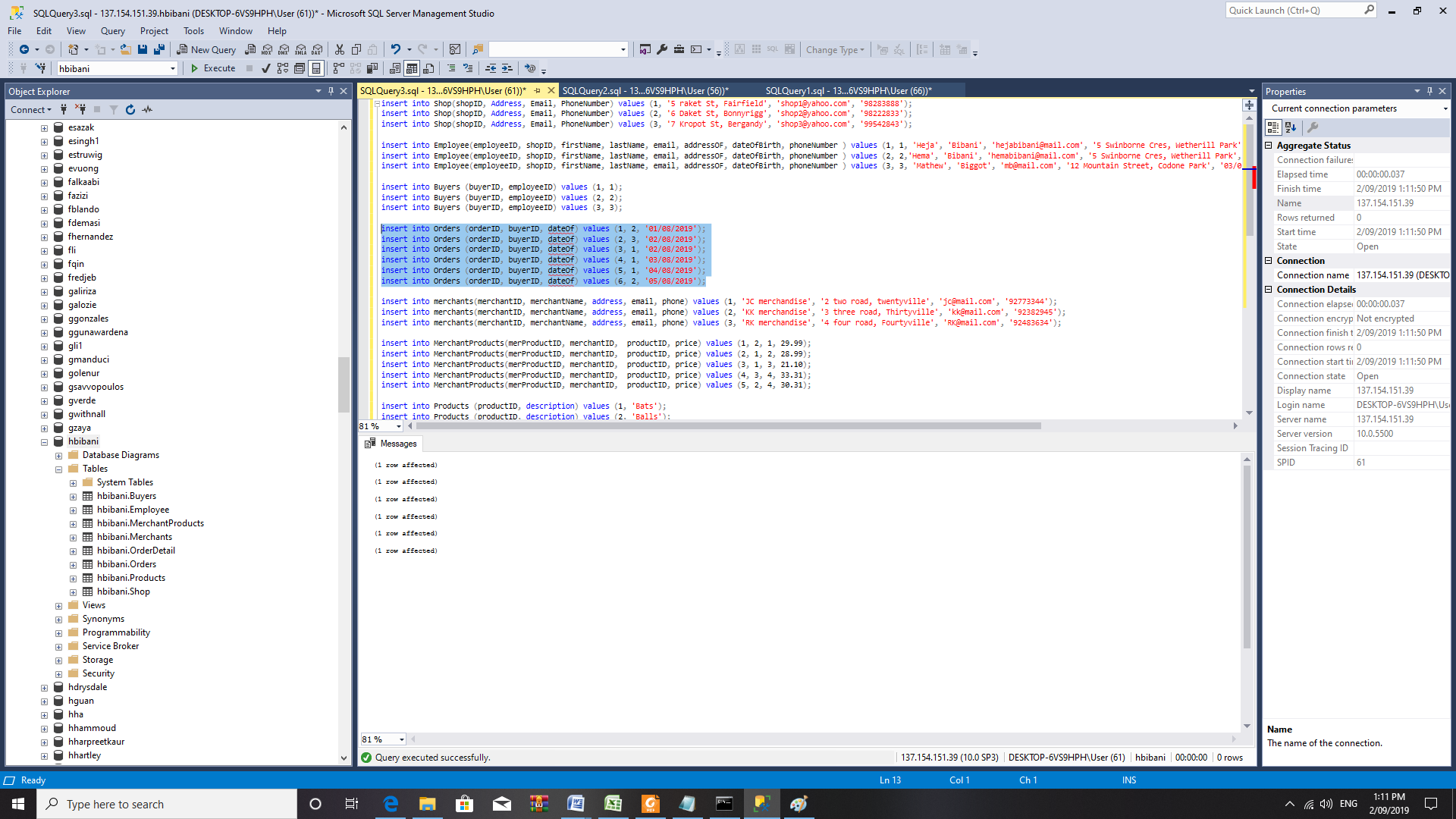
1. Screenshots

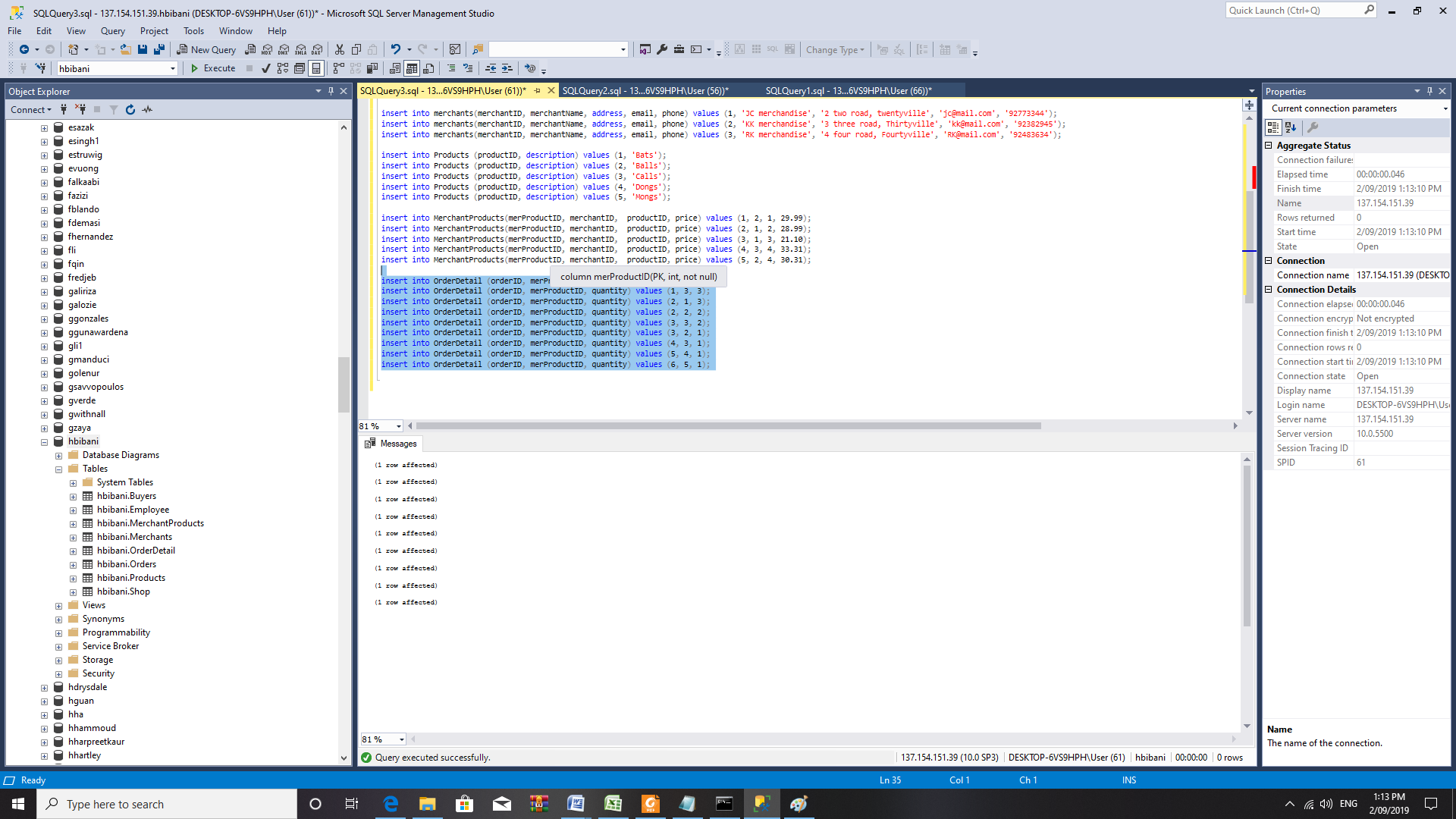


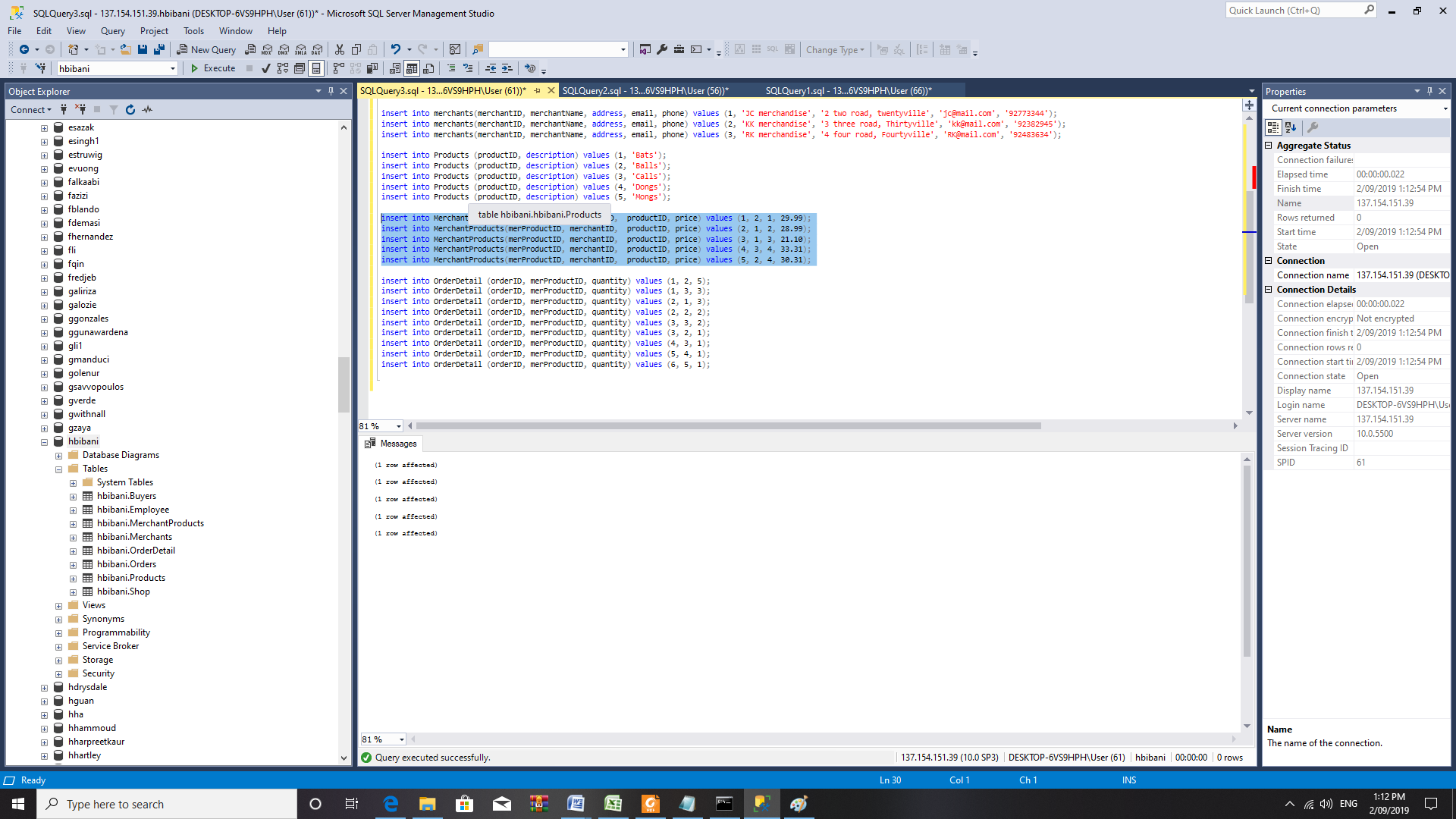




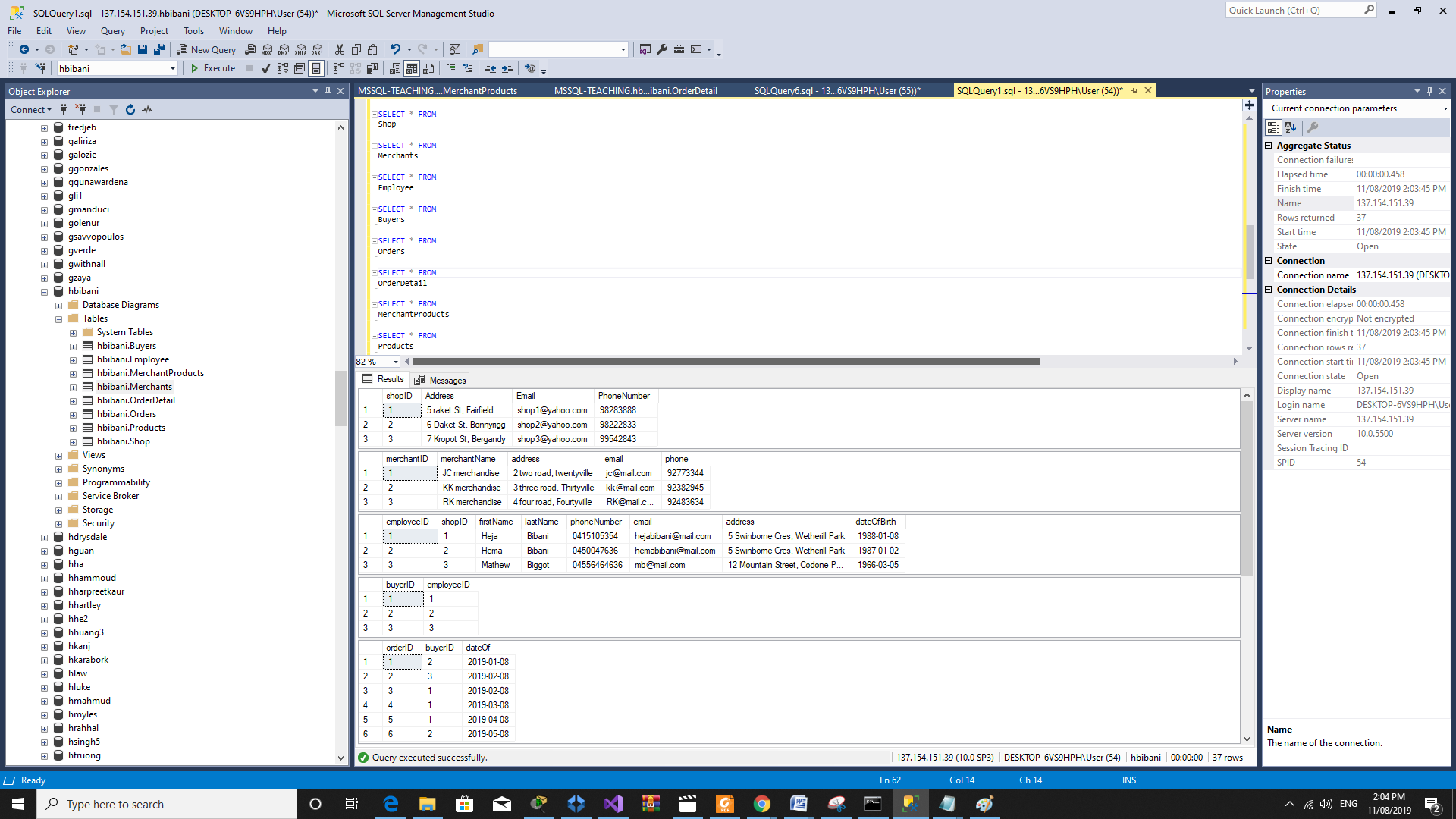
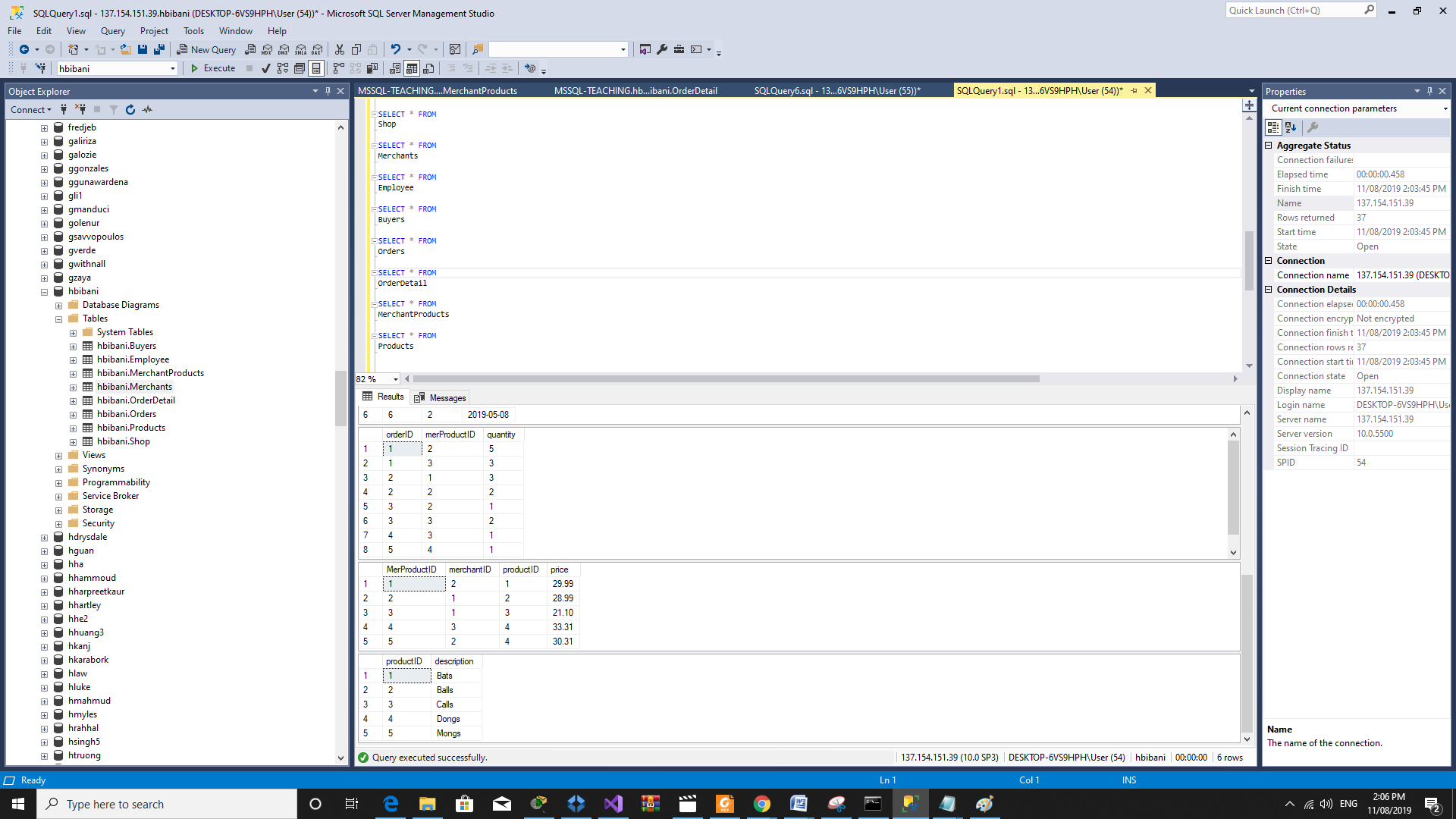
  
  
  
  








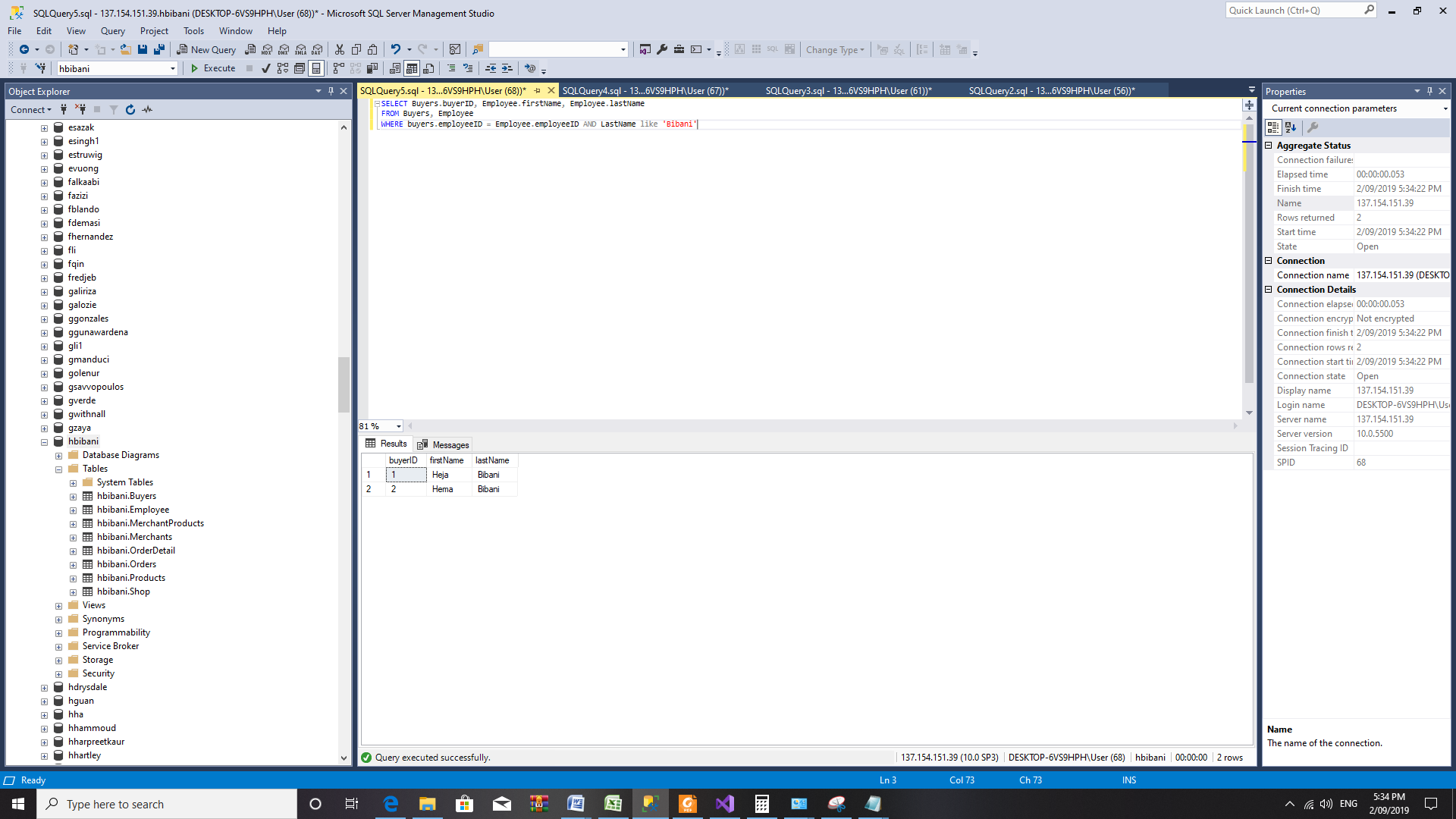
1. SELECT Statements:



1. **Buyer with ‘BIBANI’ last name [two Pictures] and Code**



We have decided to use two statements with the intention to demonstrate our ability to use different SQL statements to produce the same output.

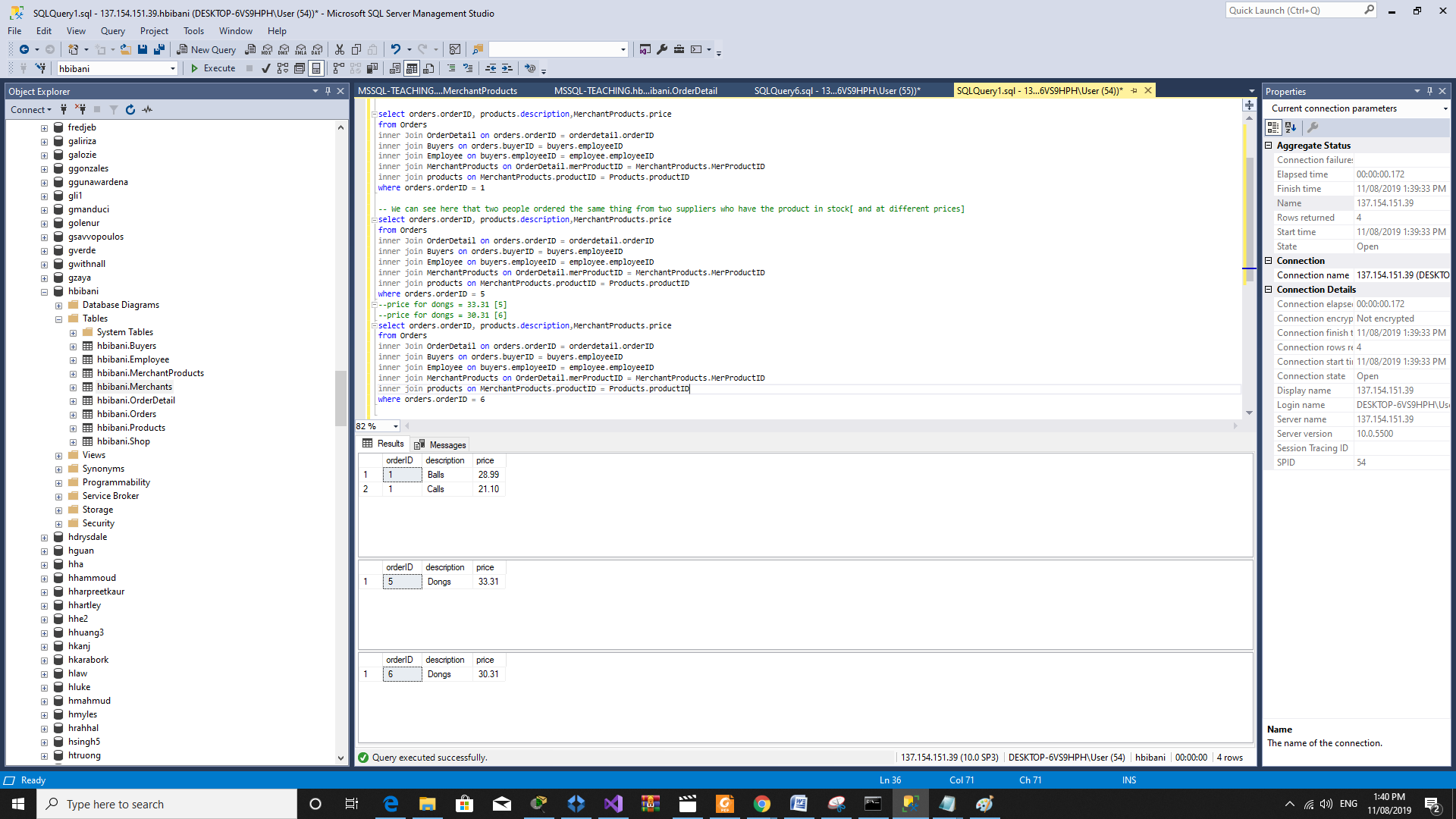


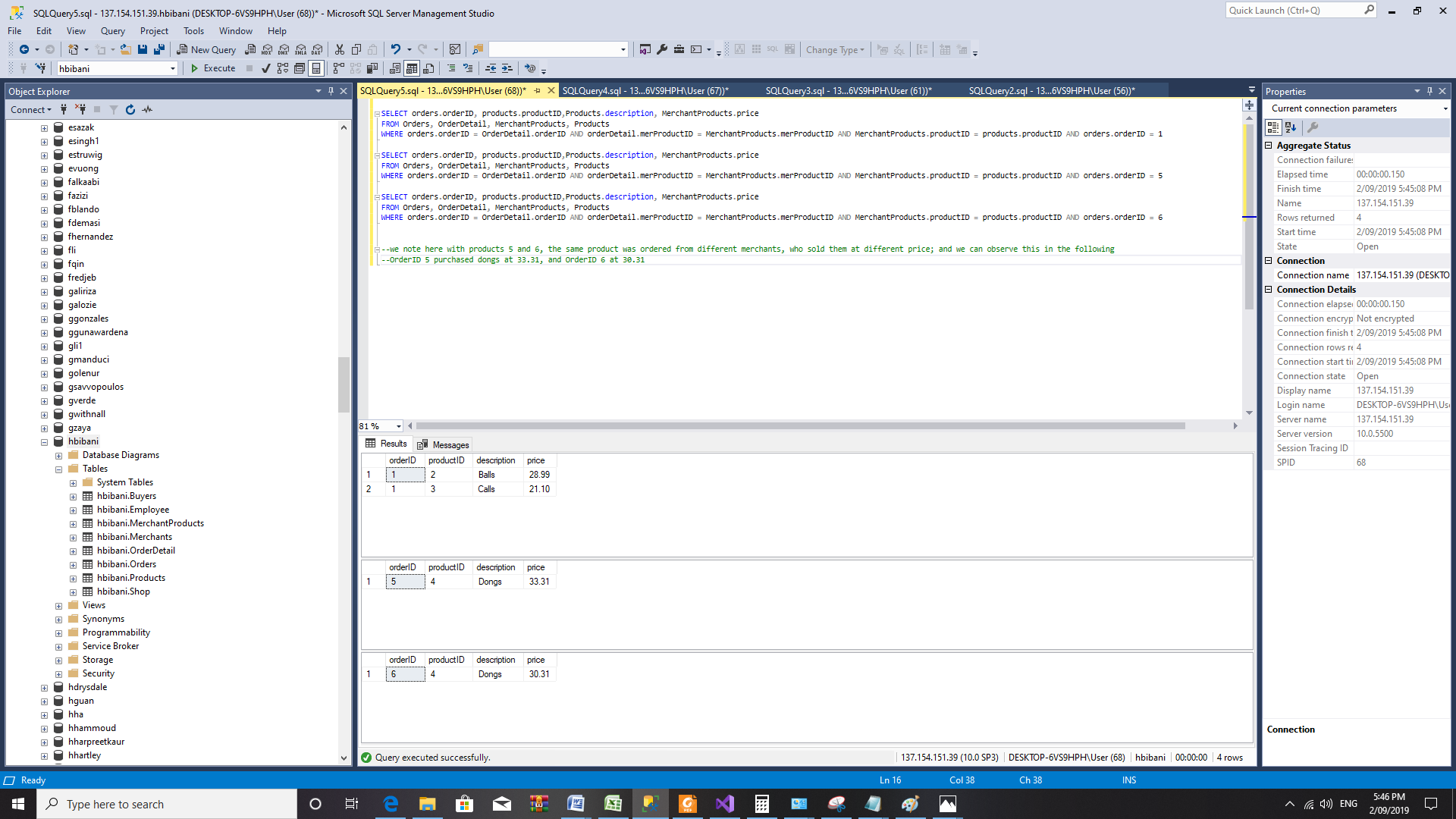
1. **Order with Price [two screen shots; two options]**



We note here with products 5 and 6, the same product was ordered from different merchants, who sold them at different prices; and we can observe this in the following case.

For instance; OrderID 5 purchased dongs at 33.31, and OrderID 6 purchased the same product at 30.31. I’ve had experience as a buyer for a company and thought that it was imperative that this feature was available; since this is a knowable constraint whilst working in this particular field. In some cases, it is reasonable to purchase a product with a higher cost; if for instance, that product is out of stock from the main supplier.





1. **Listing with Order Name [one screenshot]**

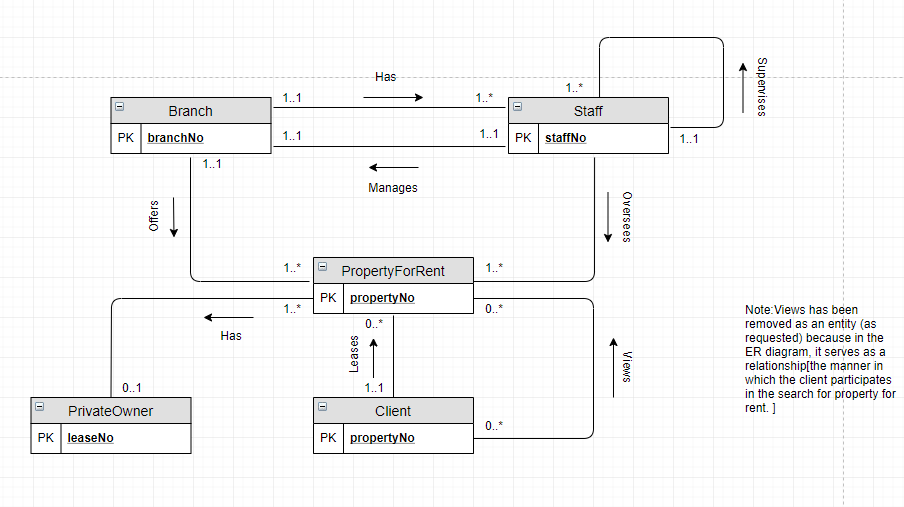


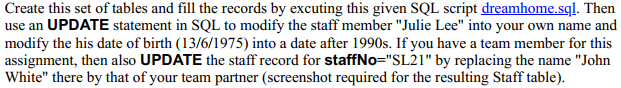


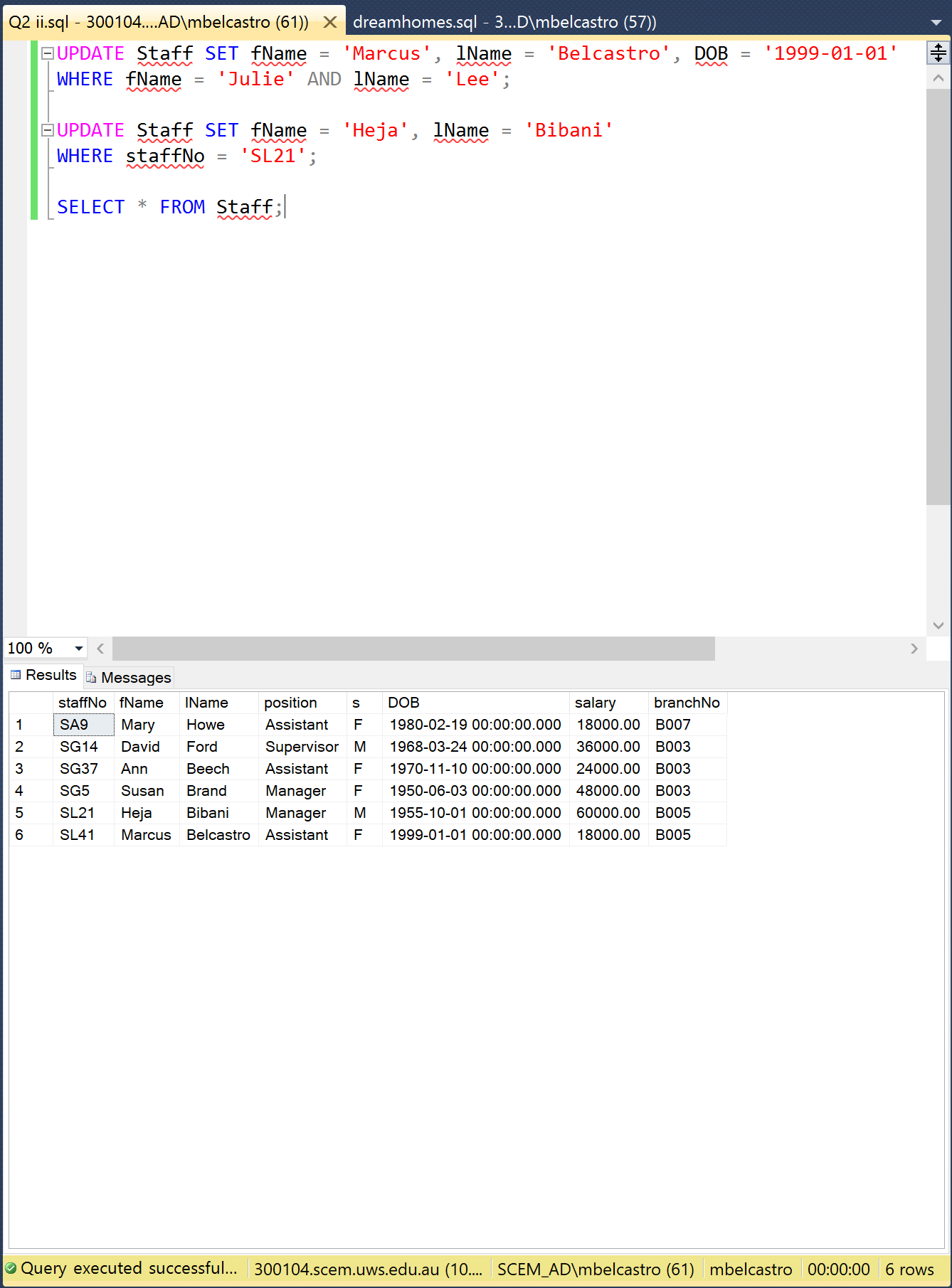
## **Question 2**

1. **ER diagram [One diagram]**

We have removed the entity ‘views’; this is a weak entity; that has its own attributes in the logical diagram, and therefore only a requirement for the actual database itself (logical). This can be the case for any verb, since it has the capacity to possess attributes (for instance, the date in which the action (viewing) was completed). In the ER diagram; views is represented as a relationship. Technically, the ER diagram should had been created first, and we would expect, that the relationship ‘views’ would have been identified as an entity during the process of producing the ***logical*** design.

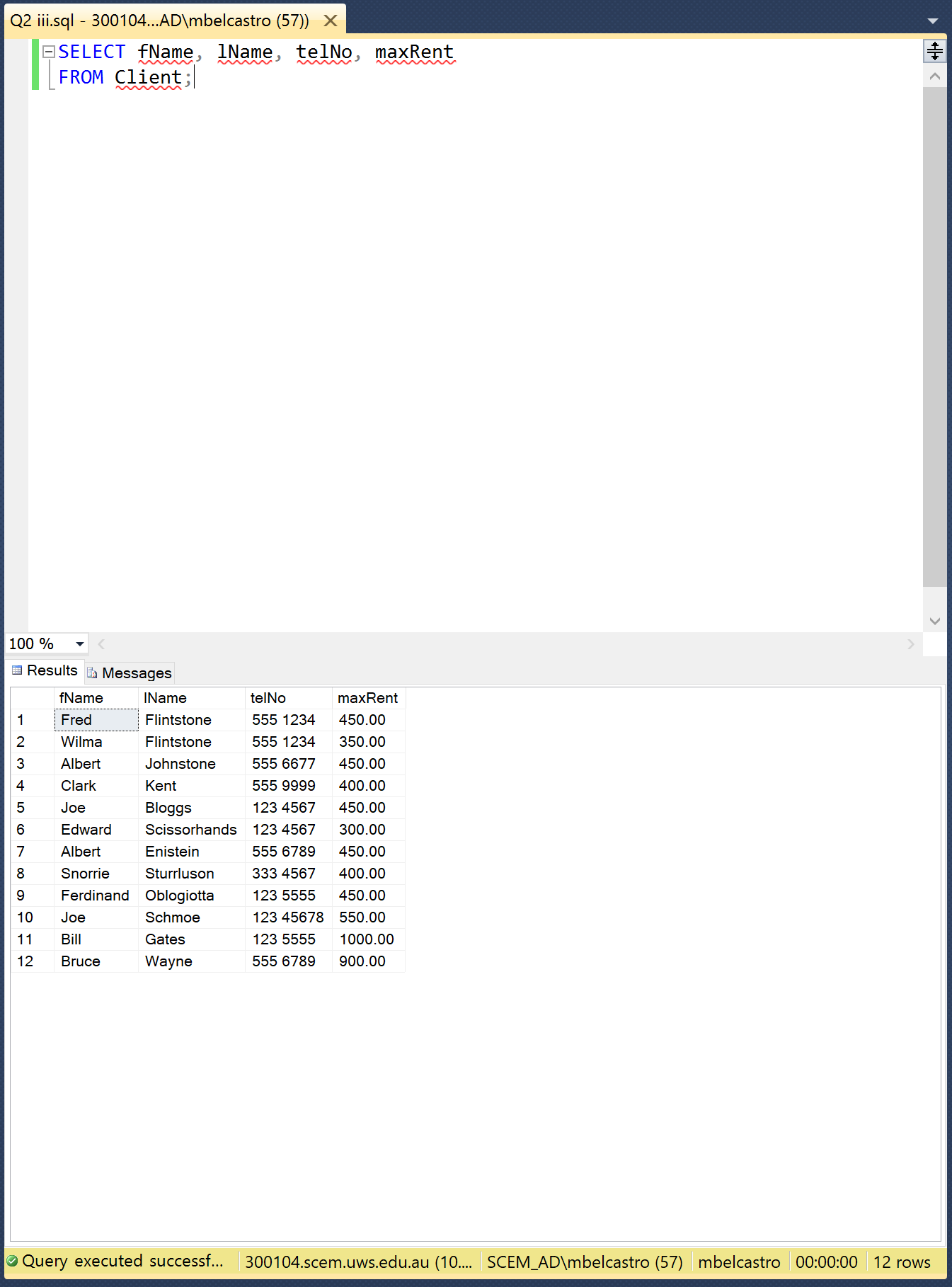


1. **SQL Update Statements[screenshots]**



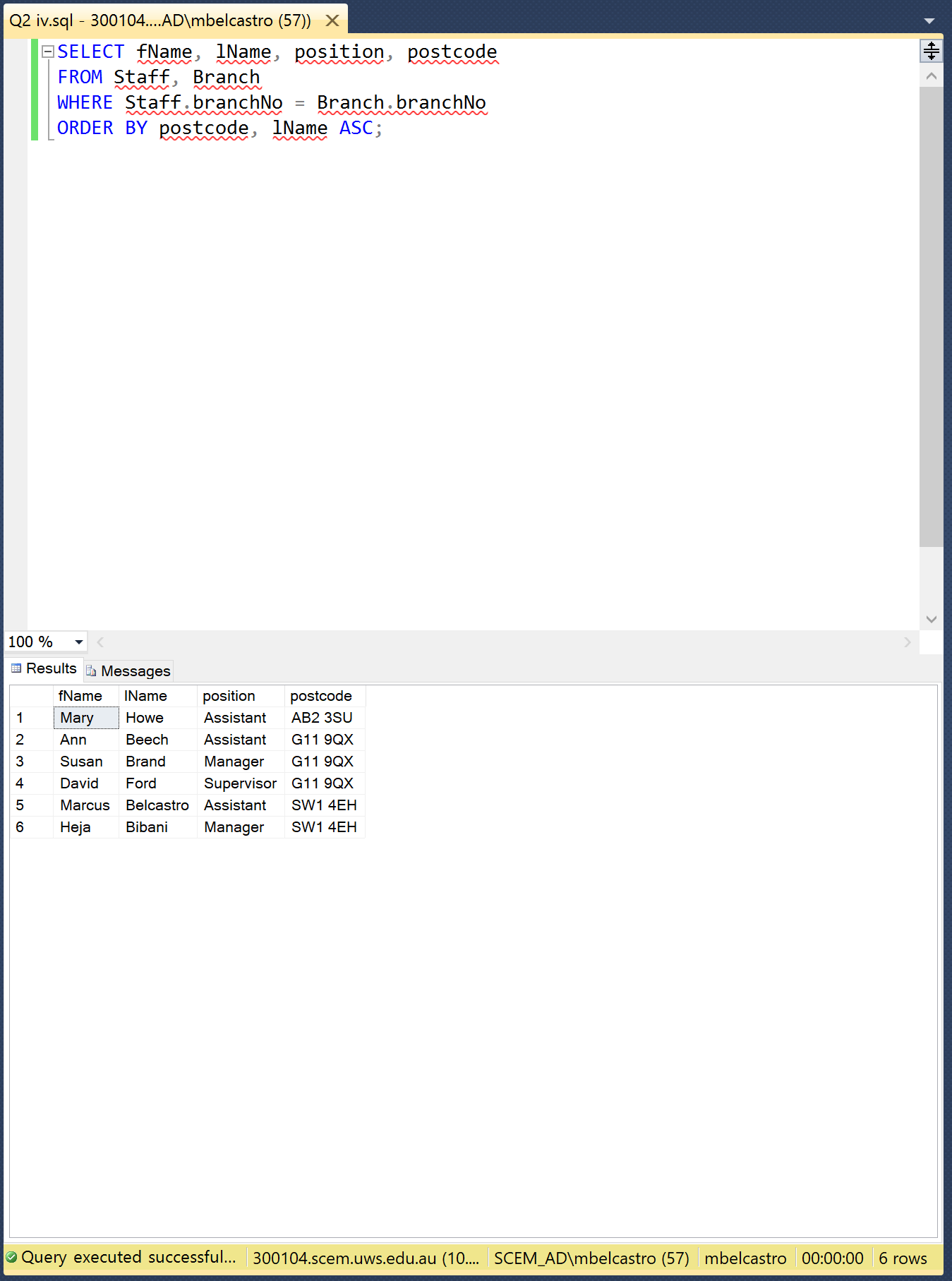
1. **SELECT Statement [FROM CLIENT]**



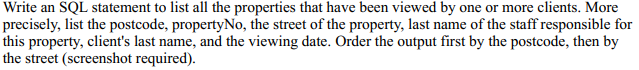


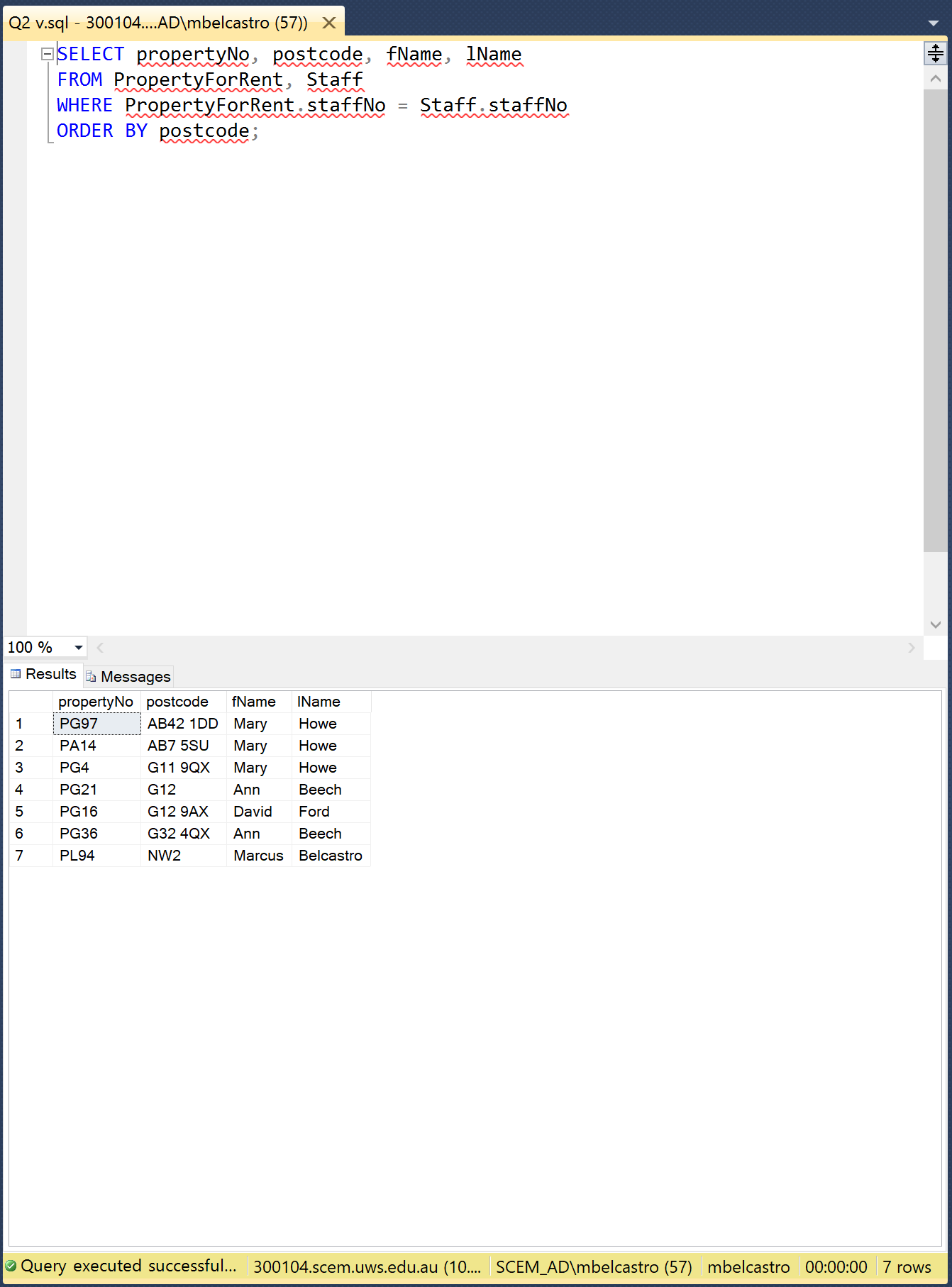
1. **SELECT, ORDER BY postcode , last name Ascending**





1. **SELECT property**

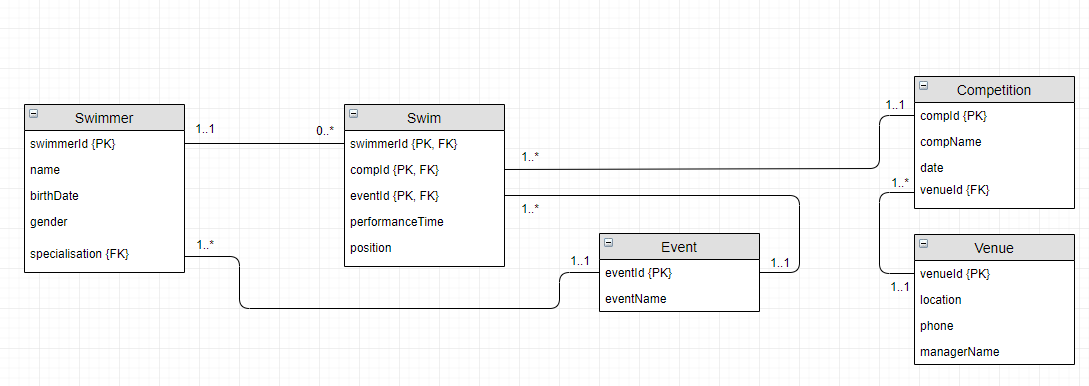
****



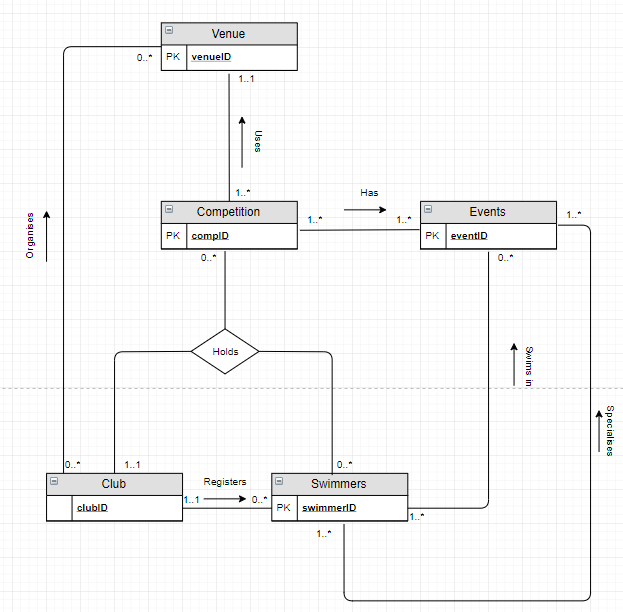
## **Question 3**

We have included the logical diagram into the question; as we have stated; it is very important to include the logical diagram; because it is imperative in the construction of the GRD.

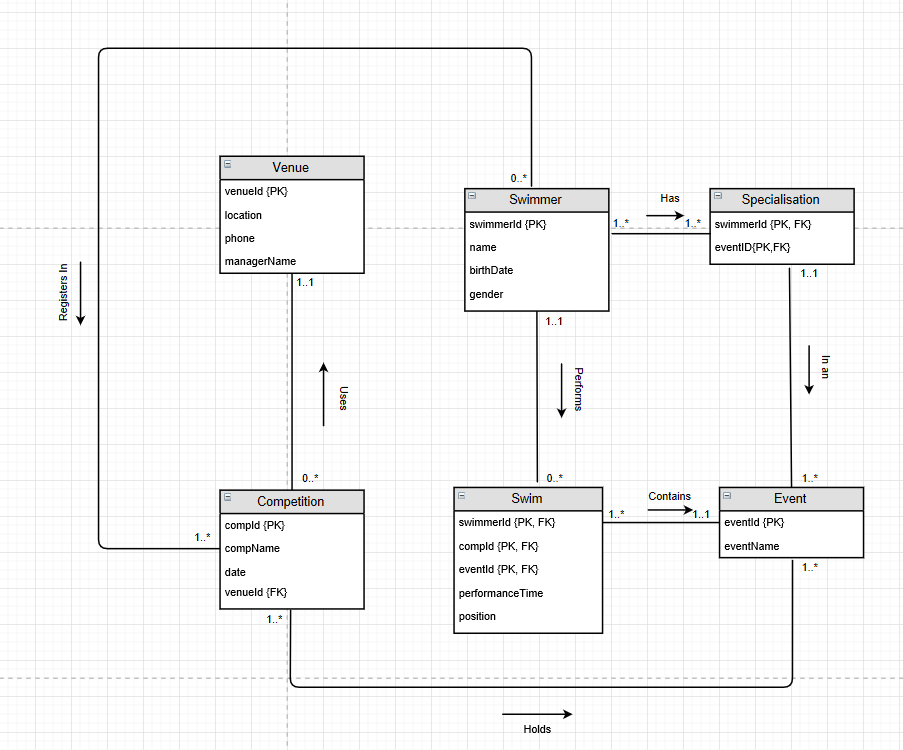
**Relational/Logical Diagram:**



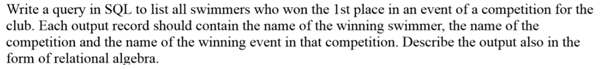
1. **ER diagram [one screenshot]**



1. **GRD [one screenshot]**



1. **SQL query all swimmers who won 1st Place and Relational Algebra**



SELECT Swimmer.name, Swim.position, Competition.compName, Event.eventName

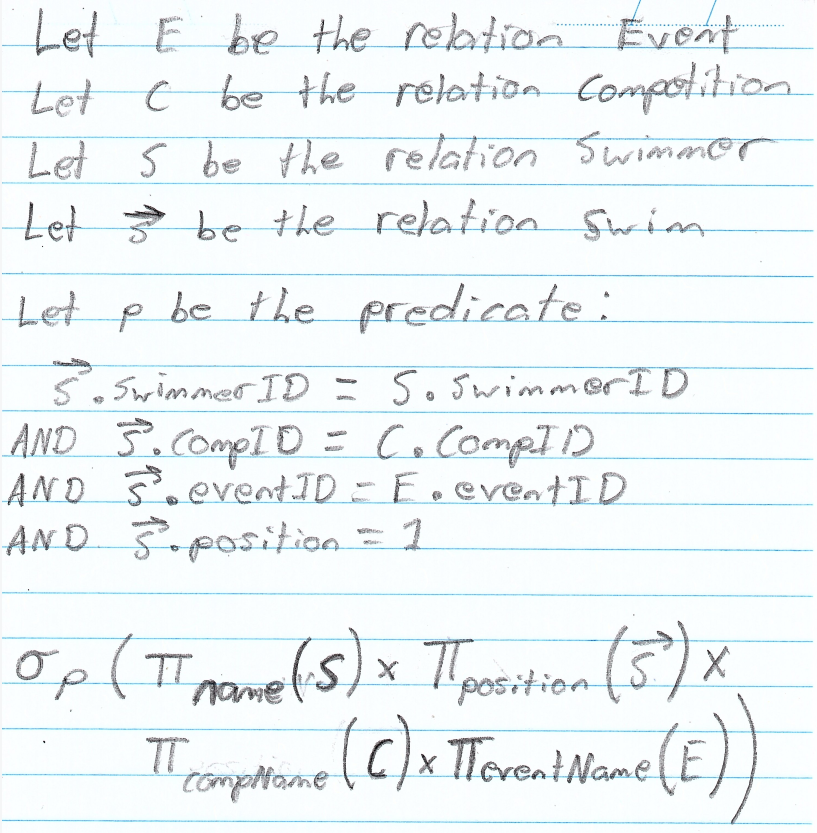
FROM Swimmer, Swim, Competition, Event

WHERE Swim.swimmerId = Swimmer.swimmerId AND

Swim.compId = Competition.compId AND

Swim.eventId = Event.eventId AND

Swim.position = 1;



1. **SQL query**



SELECT Swimmer.name, Swim.position, Competition.compName, Event.eventName

FROM Swimmer, Swim, Competition, Event

WHERE Swim.swimmerId = Swimmer.swimmerId AND

Swim.compId = Competition.compId AND

Swim.eventId = Event.eventId AND

Swim.eventId = Swimmer.specialisation AND

Swim.position = 1

## **Question 4**

1. **Alpha practical 5**

Video store is storing information about users and the videos that they have leased to in the store. A user can only borrow 5 items at a time.

**The three entities which are involved in the relationship are as following**

1. Video

Attributes:

* videoID (primary key)
* videoTitle
* dateEntered
* genre

1. User

Attributes:

* userID
* fName
* lName
* address
* DOB

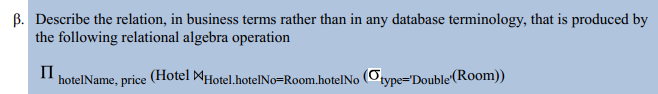
1. BorrowInfo

Attributes:

* BorrowID(primary key)
* userID(Foreign Key)
* videoID(Foreign Key)
* dateBorrowed

Note: We have established that the userID, videoID and dateborrowed are together candidate keys, but felt that this approach would not be ergonomic and user friendly, and have decided to produce borrowID as the primary key(chosen candidate key).

1. **Beta and Gamma practical 6**

**Beta:**

Answer: Show the prices and the name of the hotels that have rooms that are of type double.

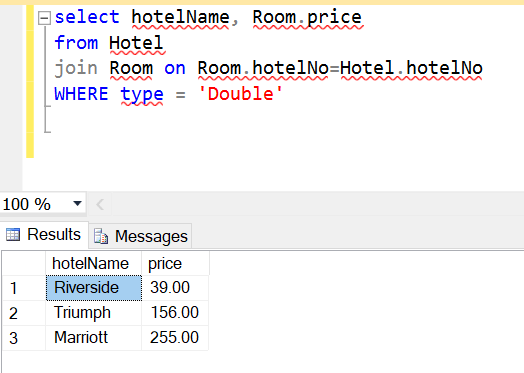
**Extra:**

select hotelName, Room.price

from Hotel

join Room on Room.hotelNo=Hotel.hotelNo

WHERE type = 'Double'



**Gamma:**





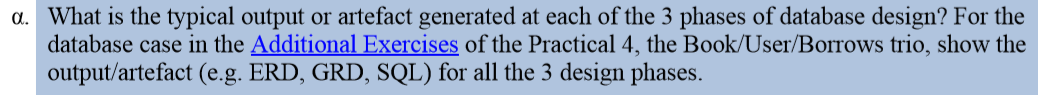
**Extra Statement:**

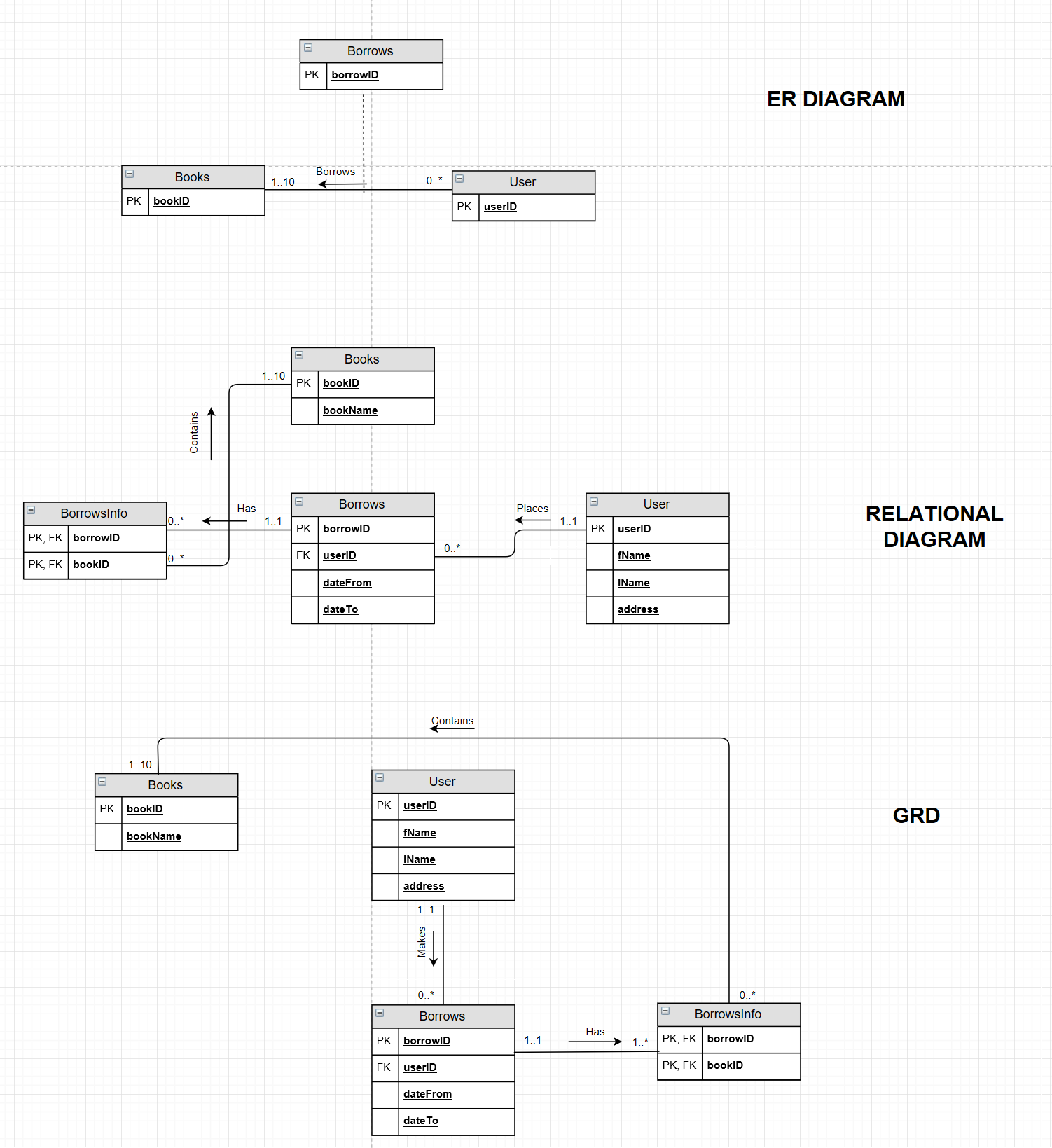
select hotelName

from Hotel

WHERE City = 'Paris'

1. **Practical 7 Alpha**





SQL as output:  
  
create table User

(

userID Int,

firstName varchar(50),

lastName varchar(50),

addressOF varchar(50),

PhoneNumber varchar(50),

CONSTRAINT User\_PK PRIMARY KEY(userID)

)

create table Books

(

bookID Int,

bookName varchar(50),

CONSTRAINT Books\_PK PRIMARY KEY(booksID)

)

create table Borrows

(

borrowID int,

userID int,

firstName varchar(50),

lastName varchar(50),

dateFrom date,

dateTo date,

CONSTRAINT Borrows\_pk PRIMARY KEY(borrowID)

CONSTRAINT User\_fk FOREIGN KEY(userID) REFERENCES User

)

create table BorrowsInfo

(

borrowID int,

bookID int

CONSTRAINT OrderDetail\_pk PRIMARY KEY(borrowID, bookID),

CONSTRAINT Borrows\_fk FOREIGN KEY(borrowID) REFERENCES Borrows,

CONSTRAINT book\_fk FOREIGN KEY(bookID) REFERENCES Books

)