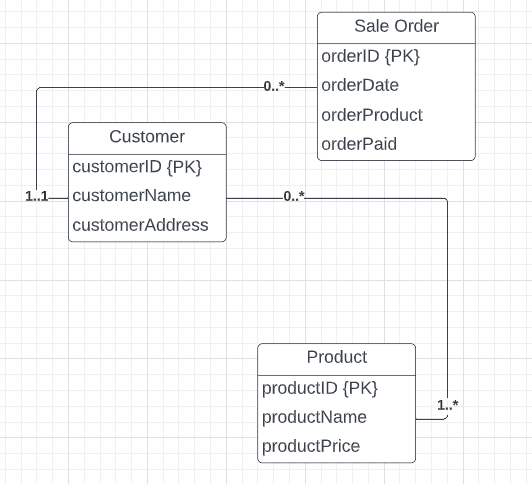
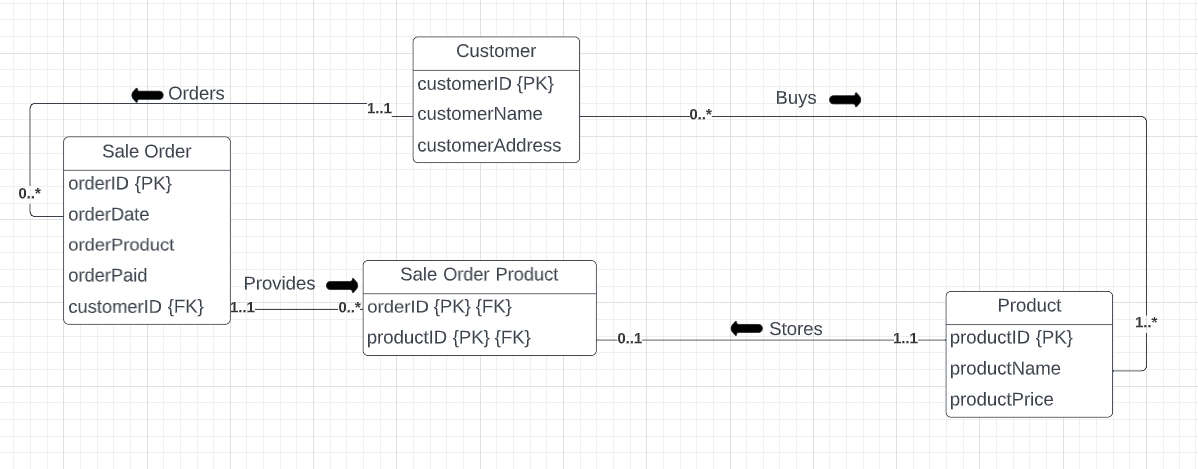
**Database Design and Development Assignment 1:**

Question 1: Basic database design and queries:

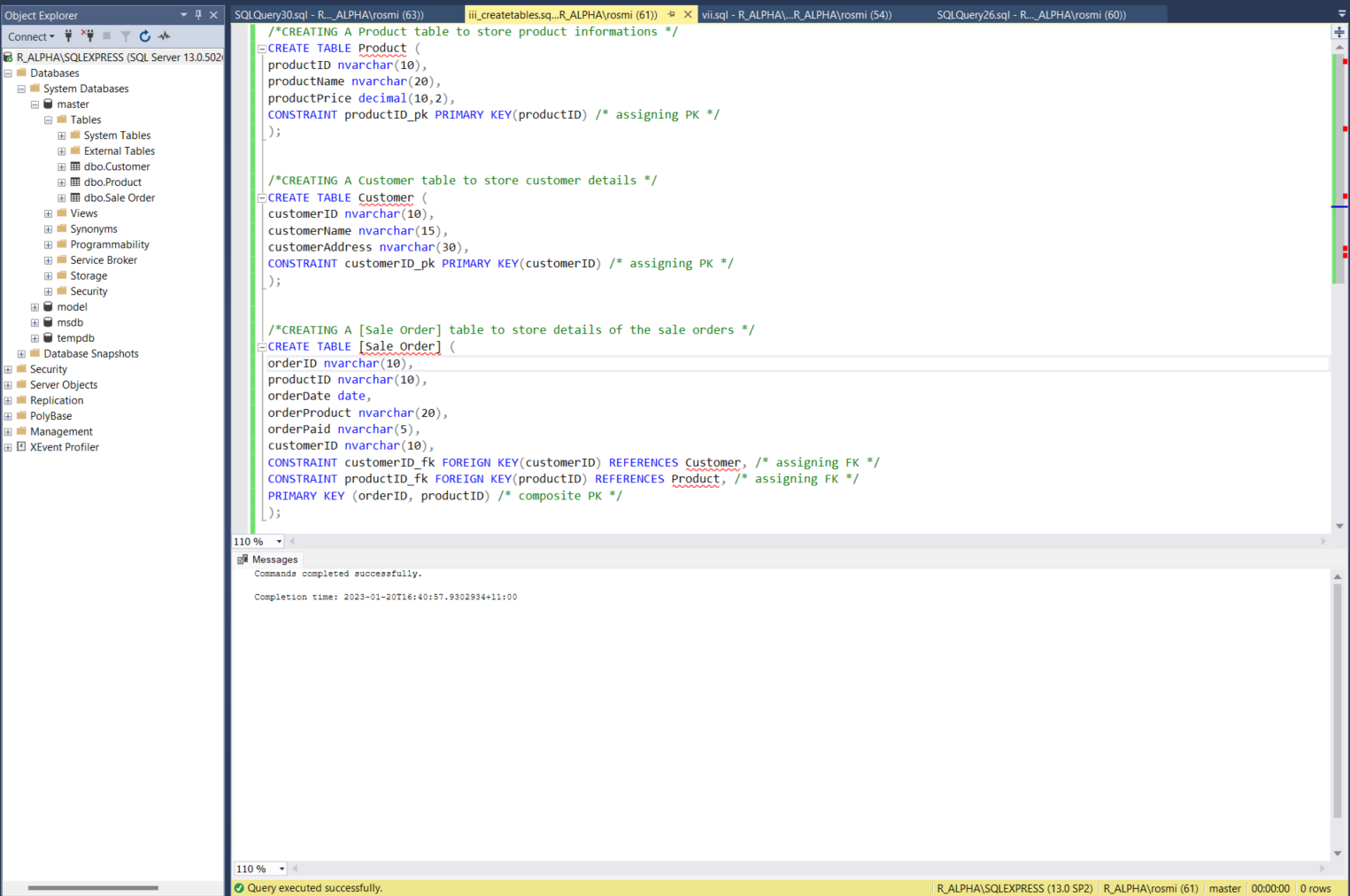
Argos is a small home product store. Argos wants to create a simple database to keep track of its products prices, customers and their sale orders. For simplicity, products will be recorded as a **product** which stores its name and price, each **customer** will be recorded for the name and address, and each **sale order** will be recorded for its date, all the products included in the order and whether the order has been paid yet.

**i.** Design a minimum database (a database of only essential attributes) to fulfil this purpose, and draw the ER diagram for your design. Indicate on the ER diagram the primary keys and the relationship multiplicities. By minimum we mean that you don't need to add anything that is not explicitly stated in the requirements, unless it's one of your artificially generated keys. For instance, you don't need to include a telephone number or email address for the entity corresponding to a customer.

**ii.** Draw the Global Relation Diagram (GRD) corresponding to the ER diagram in the above, indicating all the primary keys and foreign keys. The GRD should be in a similar form to Figure 17.9 (page 554 or 516 for edition 5) of the textbook. Note, all the attributes should be kept in this diagram too.

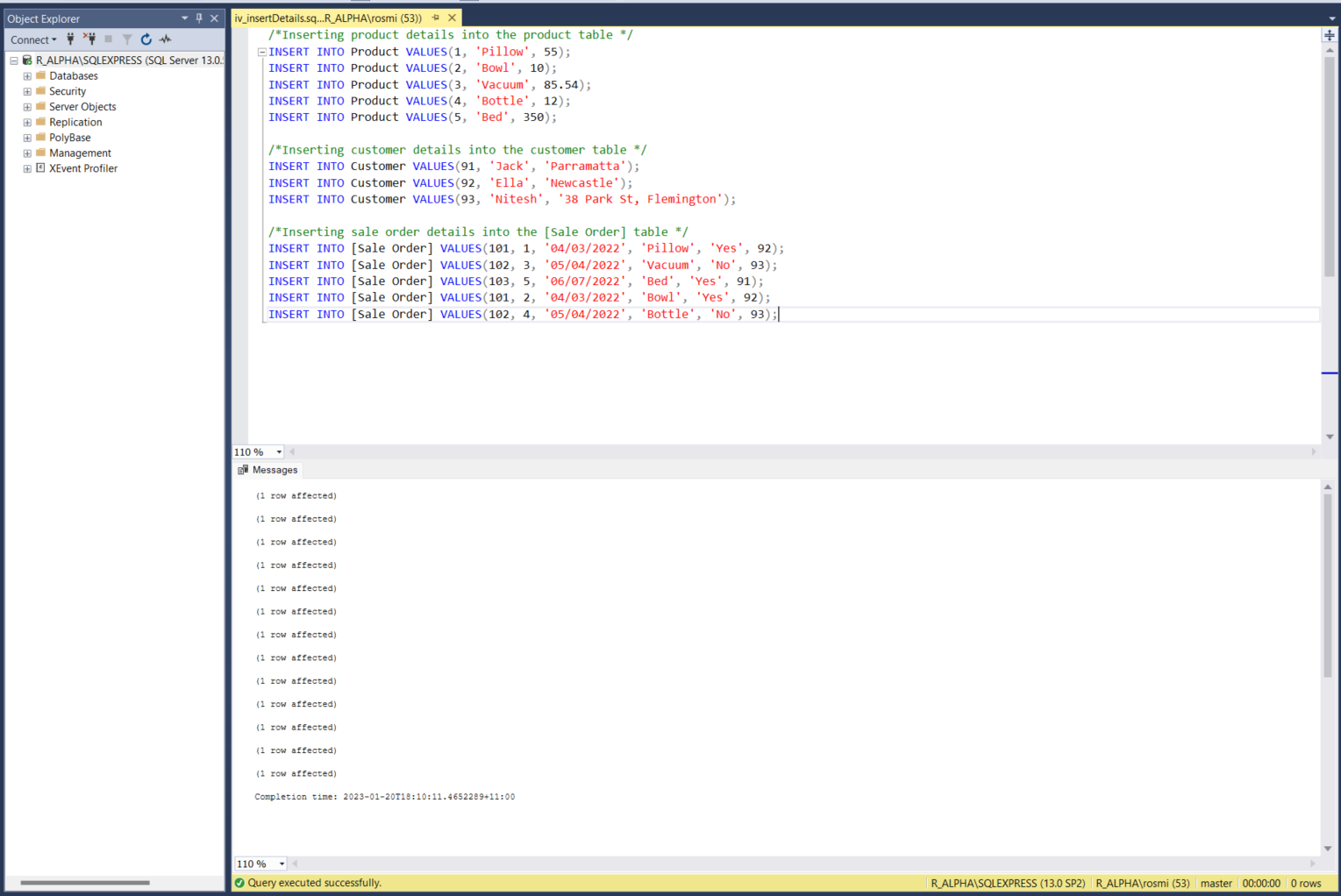
**iii.** Write an SQL script (of statements) that generates all the tables for your designed database from above.

|  |
| --- |
| /\*CREATING A Product table to store product informations \*/  CREATE TABLE Product (  productID nvarchar(10),  productName nvarchar(20),  productPrice decimal(10,2),  CONSTRAINT productID\_pk PRIMARY KEY(productID) /\* assigning PK \*/  );  /\*CREATING A Customer table to store customer details \*/  CREATE TABLE Customer (  customerID nvarchar(10),  customerName nvarchar(15),  customerAddress nvarchar(30),  CONSTRAINT customerID\_pk PRIMARY KEY(customerID) /\* assigning PK \*/  );  /\*CREATING A [Sale Order] table to store details of the sale orders \*/  CREATE TABLE [Sale Order] (  orderID nvarchar(10),  productID nvarchar(10),  orderDate date,  orderProduct nvarchar(20),  orderPaid nvarchar(5),  customerID nvarchar(10),  CONSTRAINT customerID\_fk FOREIGN KEY(customerID) REFERENCES Customer, /\* assigning FK \*/  CONSTRAINT productID\_fk FOREIGN KEY(productID) REFERENCES Product, /\* assigning FK \*/  PRIMARY KEY (orderID, productID) /\* composite PK \*/  ); |

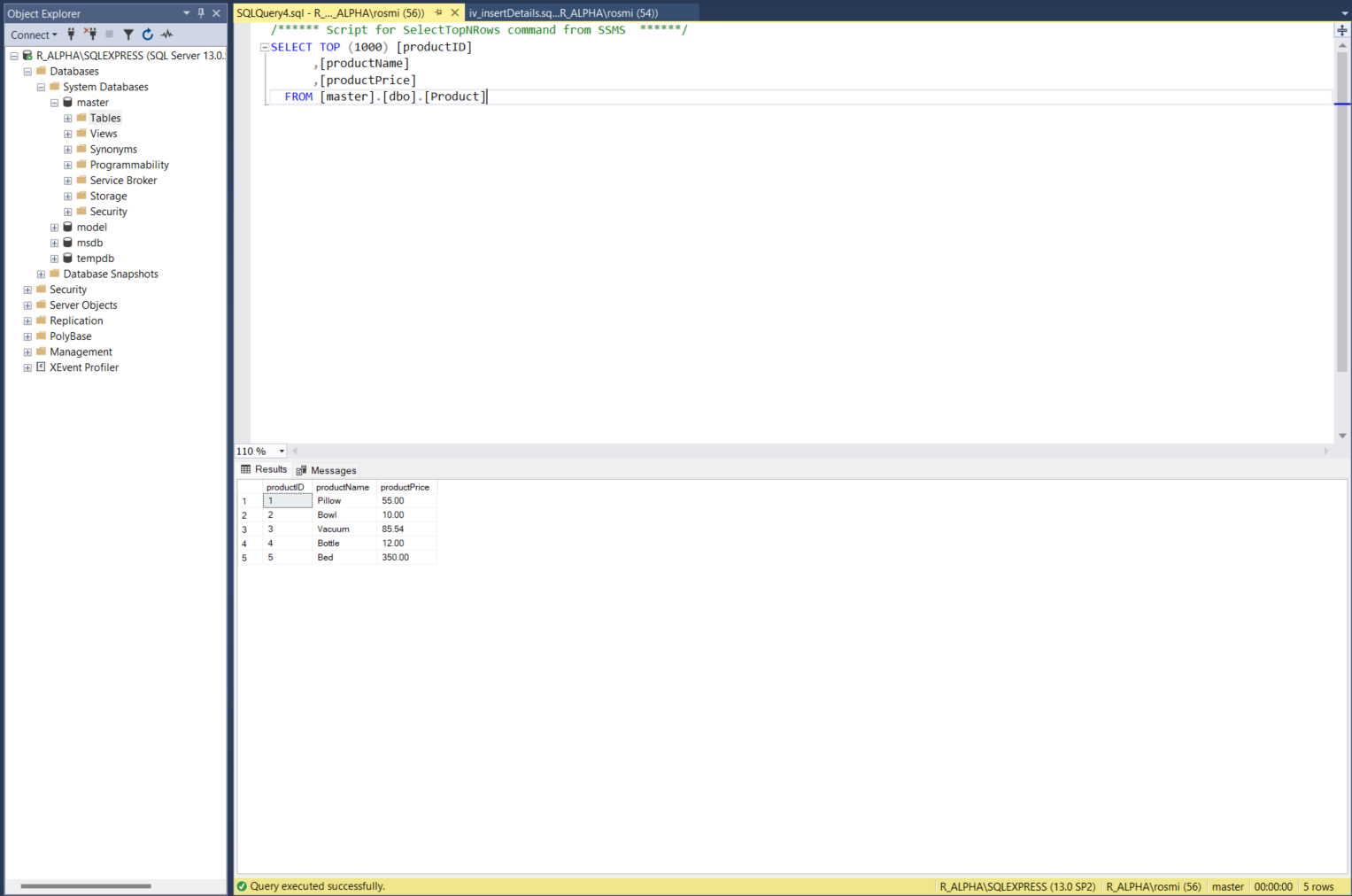


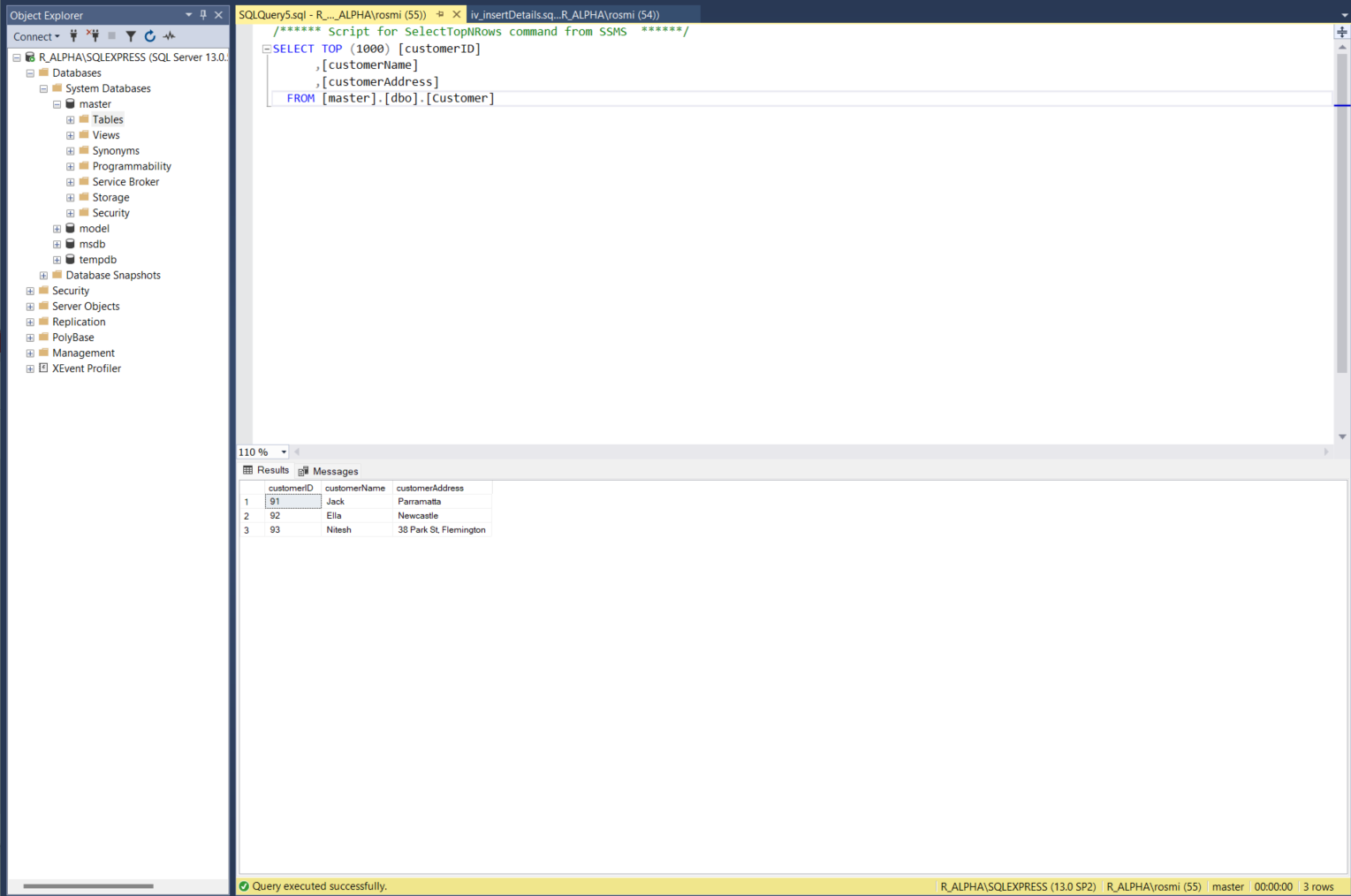
**iv.** Write an SQL script to insert sufficient records into your tables. Each table should contain no less than 3 records. At least 2 sale orders should each contain 2 or more products of the ordered merchandise. Screenshots are required for the records of all the tables.

|  |
| --- |
| /\*Inserting product details into the product table \*/  INSERT INTO Product VALUES(1, 'Pillow', 55);  INSERT INTO Product VALUES(2, 'Bowl', 10);  INSERT INTO Product VALUES(3, 'Vacuum', 85.54);  INSERT INTO Product VALUES(4, 'Bottle', 12);  INSERT INTO Product VALUES(5, 'Bed', 350);  /\*Inserting customer details into the customer table \*/  INSERT INTO Customer VALUES(91, 'Jack', 'Parramatta');  INSERT INTO Customer VALUES(92, 'Ella', 'Newcastle');  INSERT INTO Customer VALUES(93, 'Nitesh', '38 Park St, Flemington');  /\*Inserting sale order details into the [Sale Order] table \*/  INSERT INTO [Sale Order] VALUES(101, 1, '04/03/2022', 'Pillow', 'Yes', 92);  INSERT INTO [Sale Order] VALUES(102, 3, '05/04/2022', 'Vacuum', 'No', 93);  INSERT INTO [Sale Order] VALUES(103, 5, '06/07/2022', 'Bed', 'Yes', 91);  INSERT INTO [Sale Order] VALUES(101, 2, '04/03/2022', 'Bowl', 'Yes', 92);  INSERT INTO [Sale Order] VALUES(102, 4, '05/04/2022', 'Bottle', 'No', 93); |

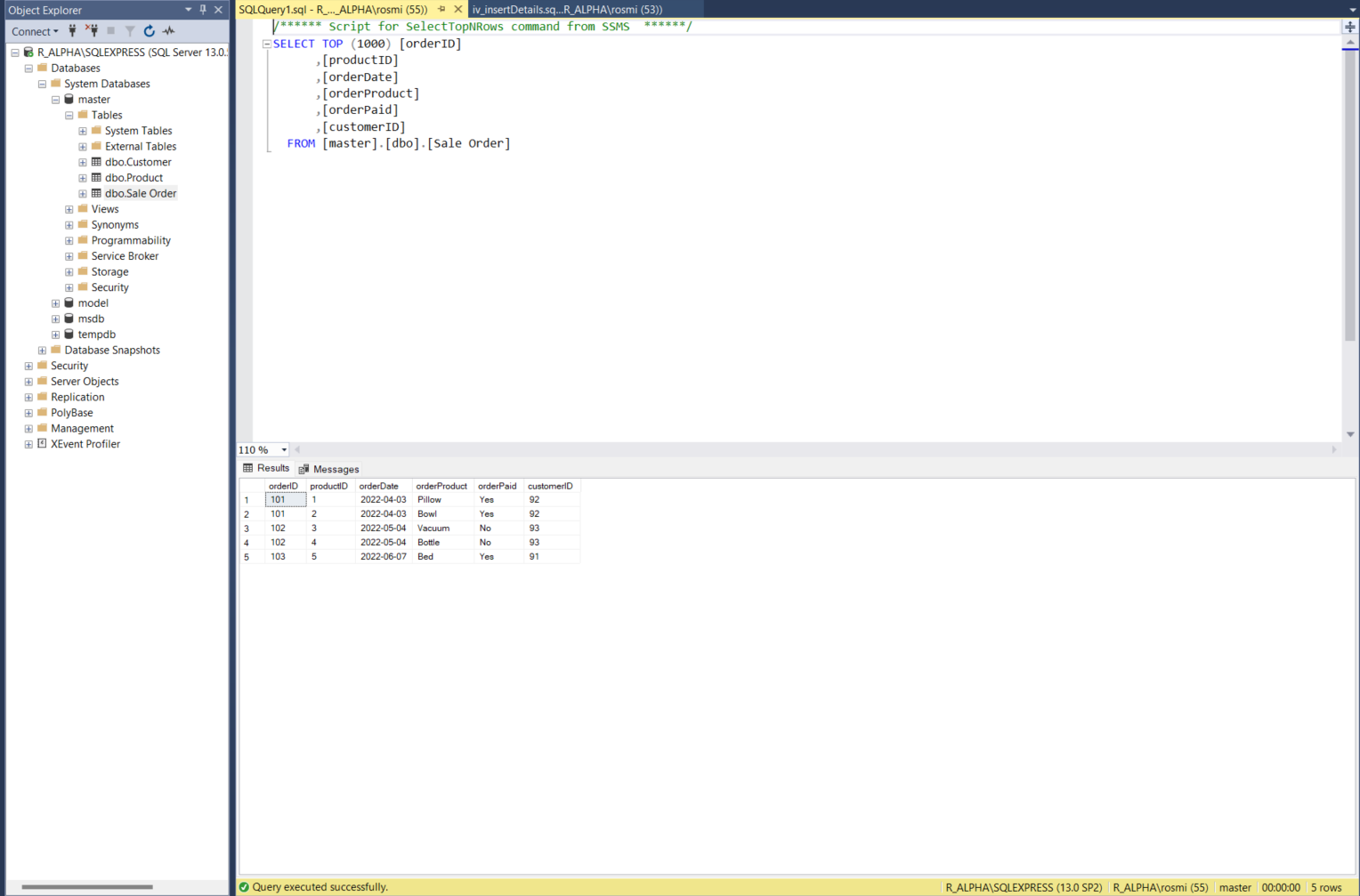


**PRODUCT TABLE:**

****

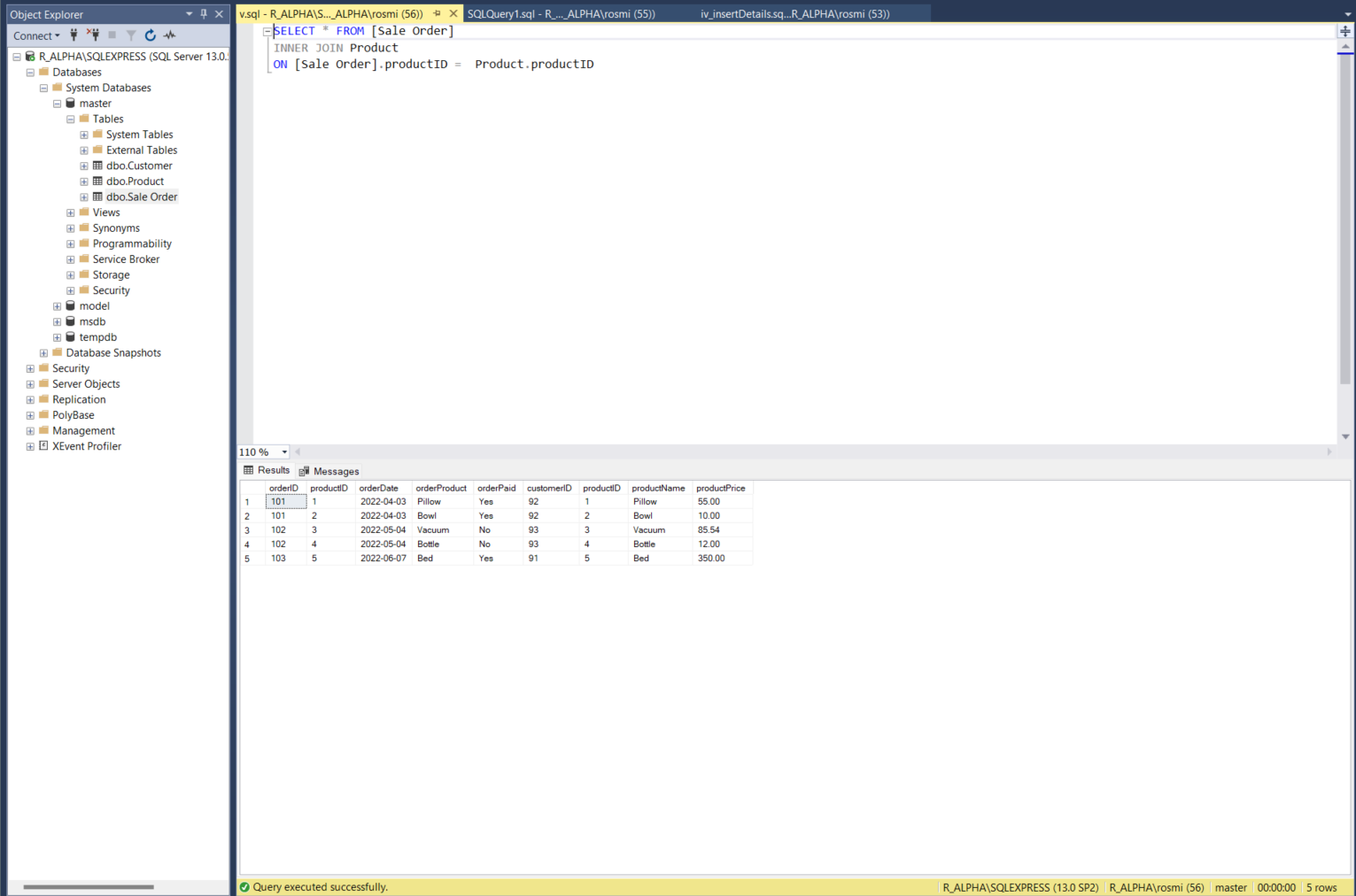
**CUSTOMER TABLE:**

**SALE ORDER TABLE:**

****

**v.** For a given sale order (number), write an SQL statement to list all the product names and their corresponding prices for the sale order (screenshot required).

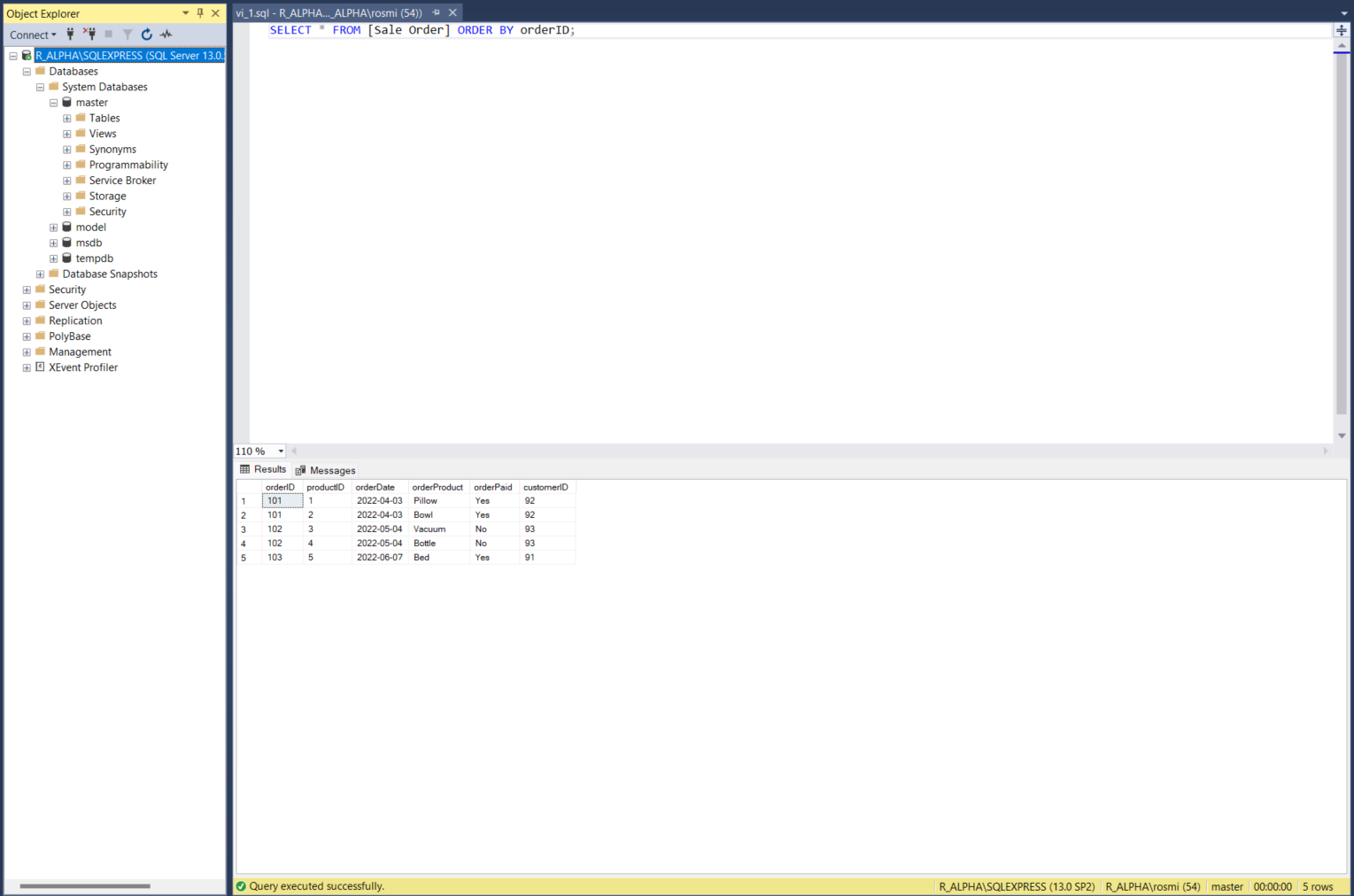
|  |
| --- |
| SELECT \* FROM [Sale Order]  INNER JOIN Product  ON [Sale Order].productID = Product.productID |



**vi.** List all the sale orders by their order number, date, and the name and the address of the customer who places that order (screenshot required).

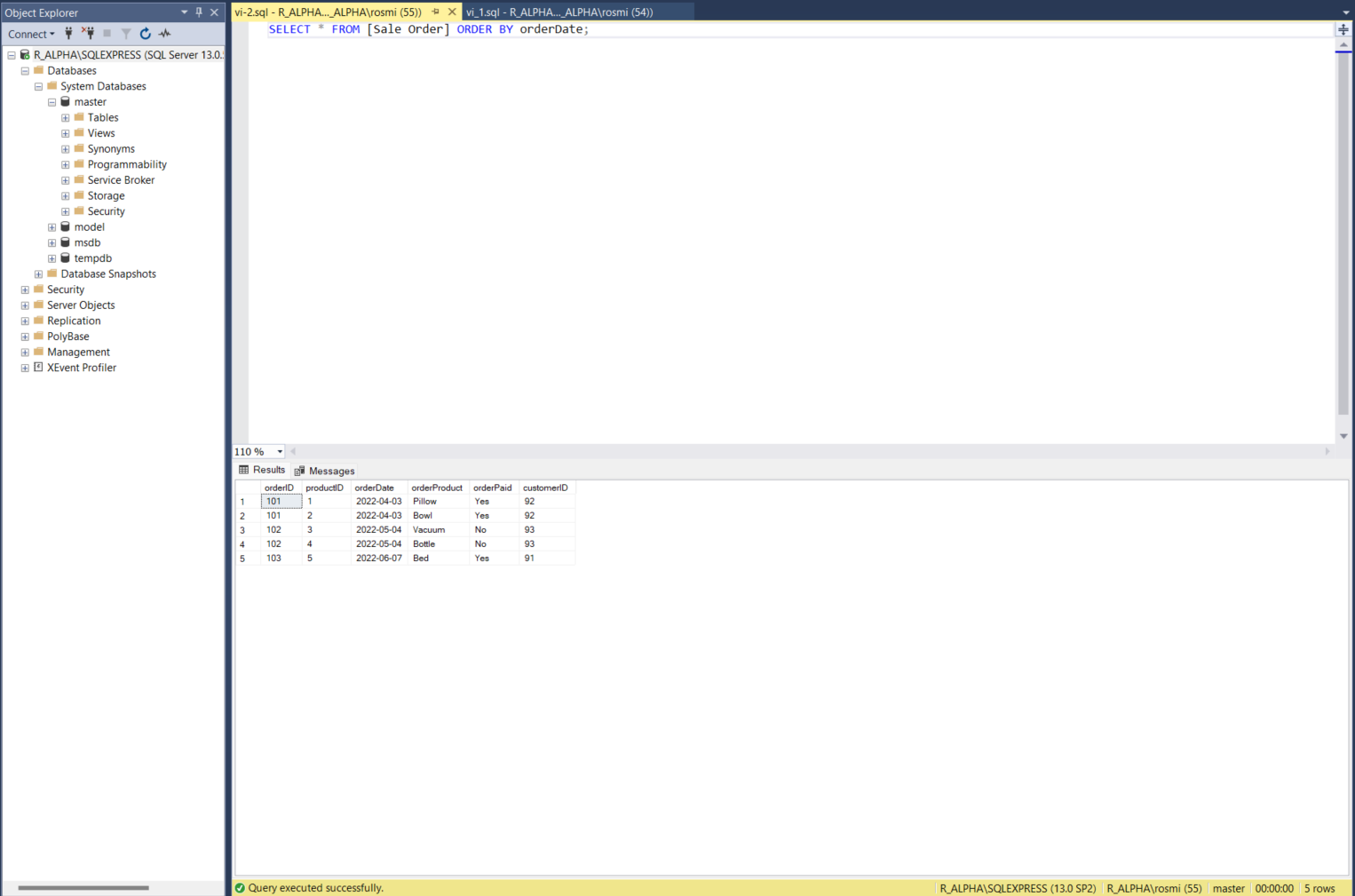
***Sale orders by their order number(orderID)*:**

|  |
| --- |
| SELECT \* FROM [Sale Order] ORDER BY orderID; |



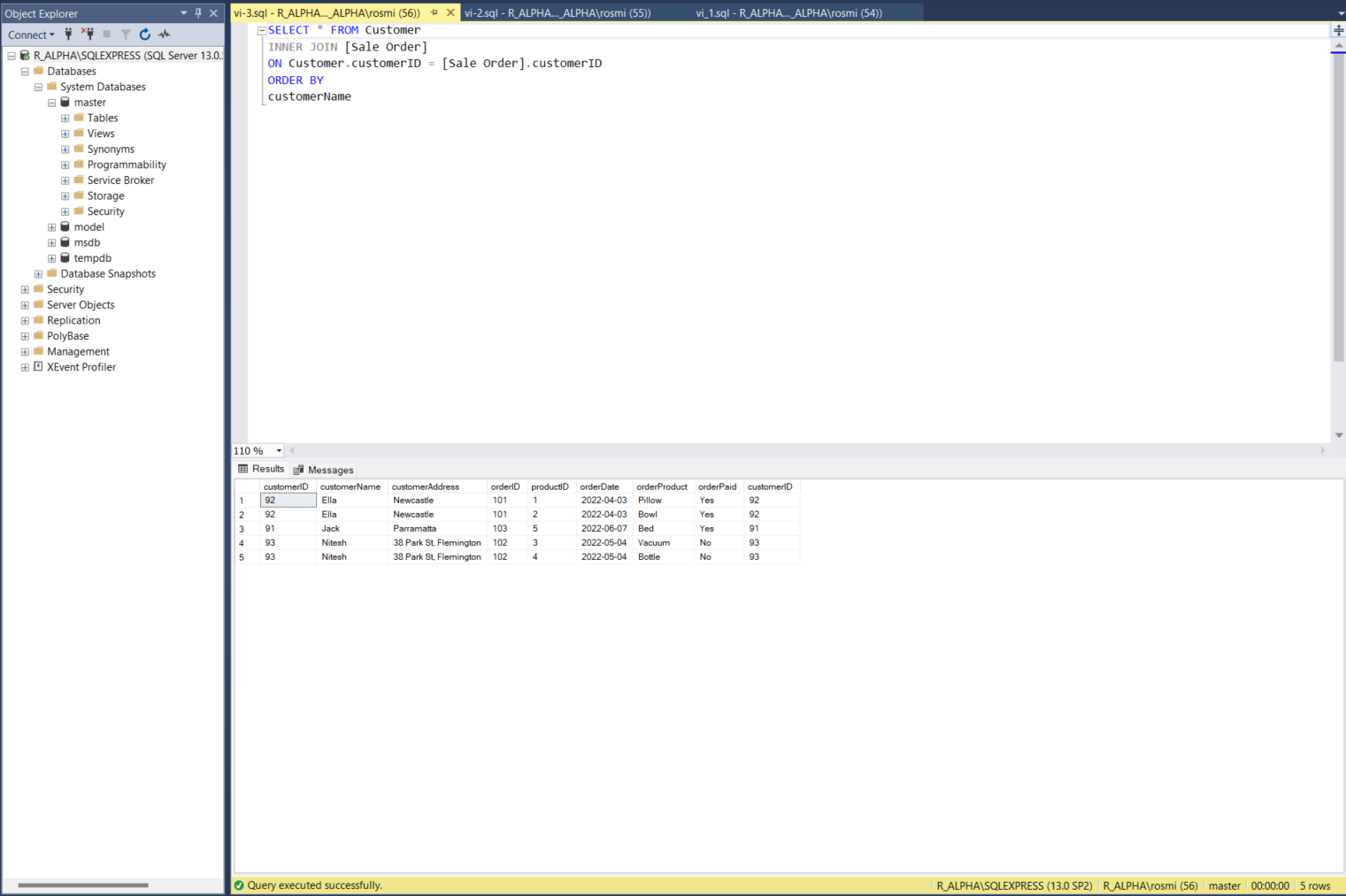
***Sale orders by their order date(orderDate)*:**

|  |
| --- |
| SELECT \* FROM [Sale Order] ORDER BY orderDate; |



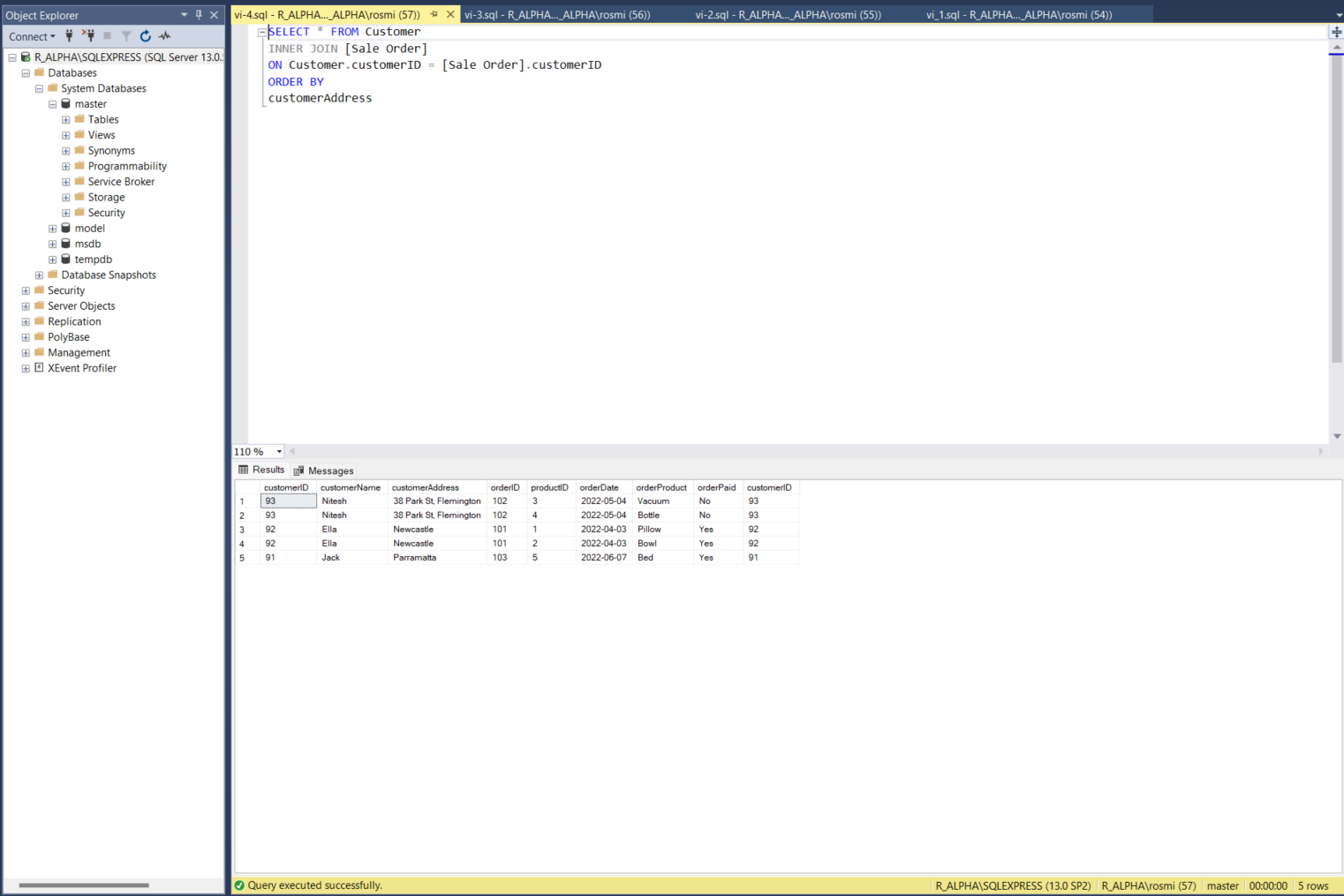
***Sale orders by customer's name*:**

|  |
| --- |
| SELECT \* FROM Customer  INNER JOIN [Sale Order]  ON Customer.customerID = [Sale Order].customerID  ORDER BY  customerName |



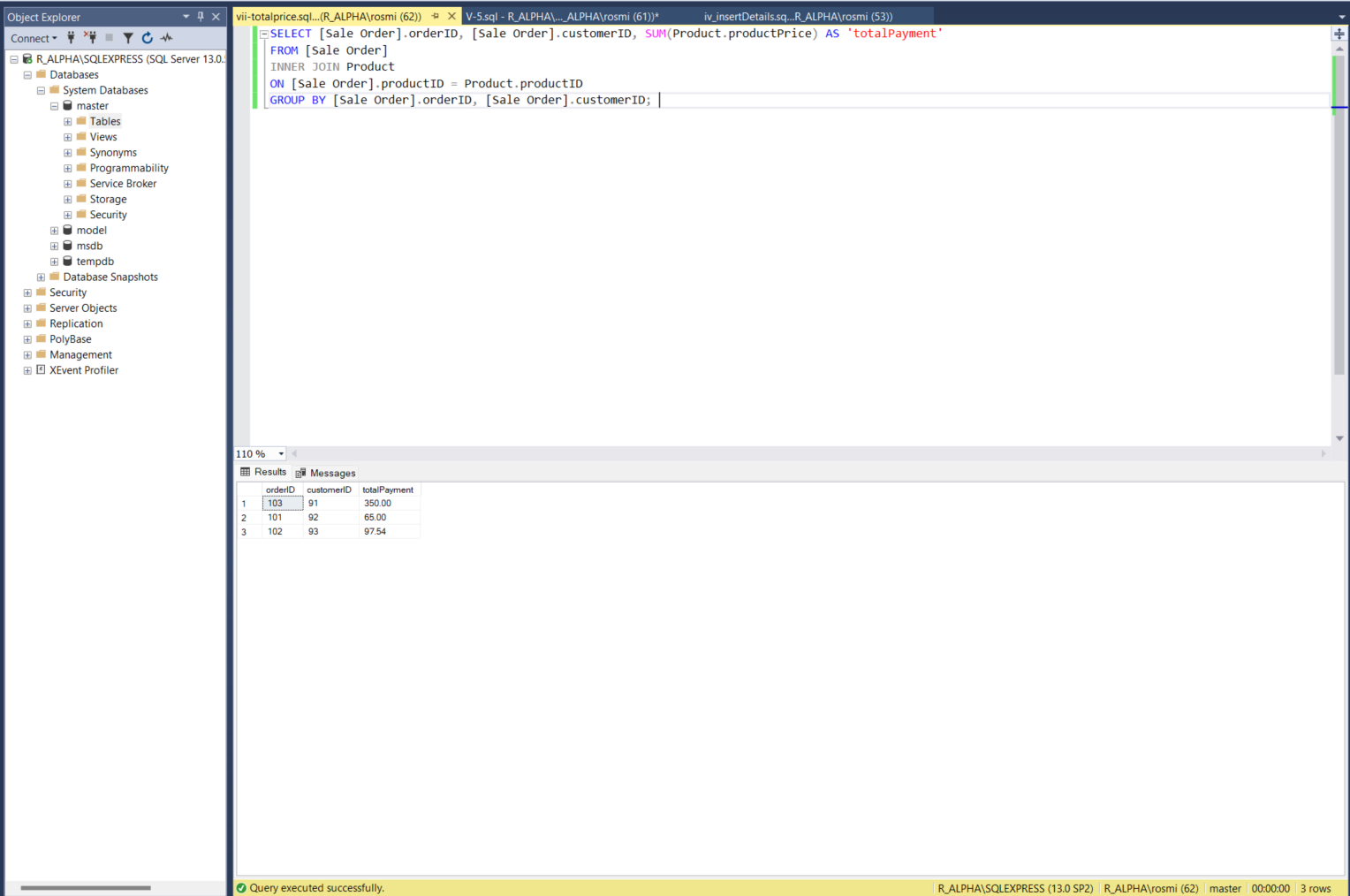
***Sale orders by customer's address*:**

|  |
| --- |
| SELECT \* FROM Customer  INNER JOIN [Sale Order]  ON Customer.customerID = [Sale Order].customerID  ORDER BY  customerAddress |

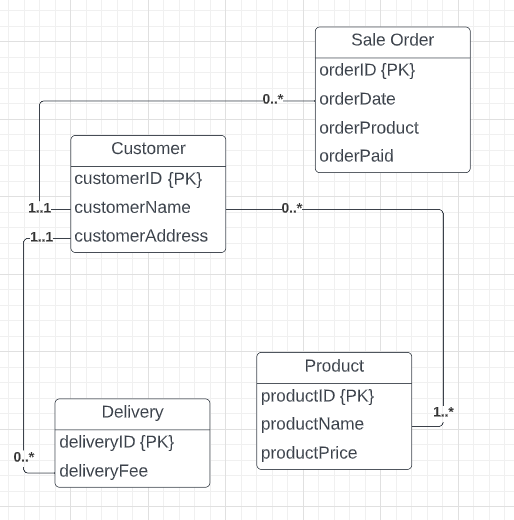


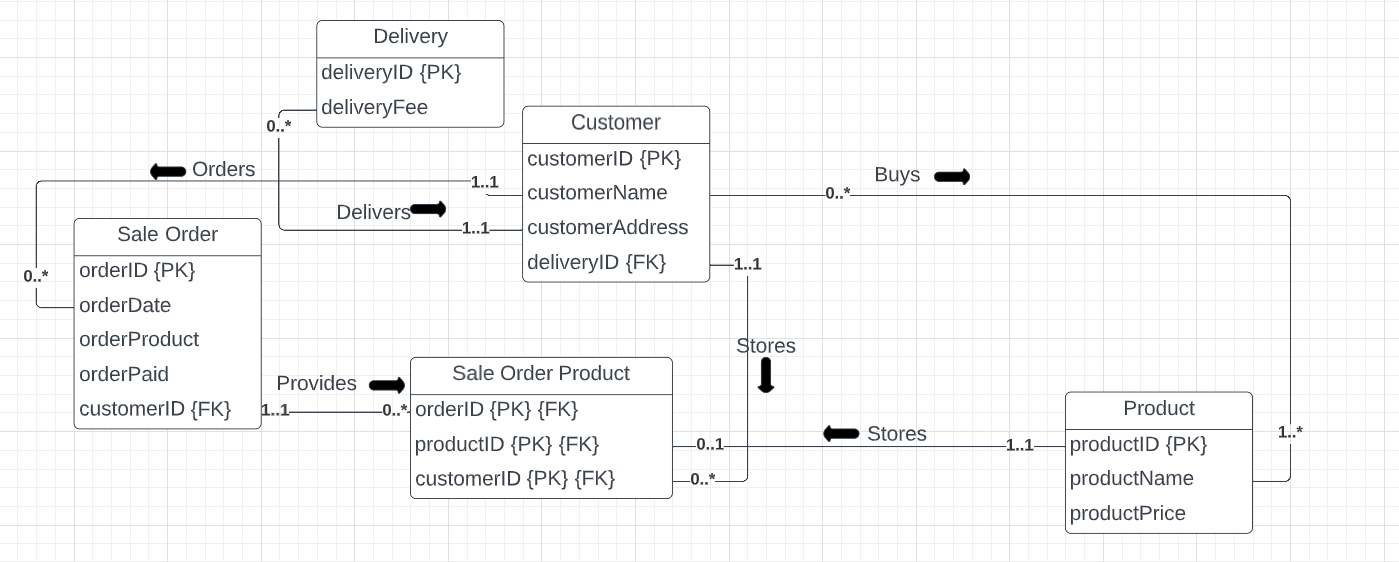
**vii.** Write an SQL statement that lists the sale order (number) and the total price of the corresponding order, for all the orders in the database (screenshot required).

|  |
| --- |
| SELECT [Sale Order].orderID, [Sale Order].customerID, SUM(Product.productPrice) AS 'totalPayment'  SELECT [Sale Order].orderID, [Sale Order].customerID, SUM(Product.productPrice) AS 'totalPayment'  FROM [Sale Order]  INNER JOIN Product  ON [Sale Order].productID = Product.productID  GROUP BY [Sale Order].orderID, [Sale Order].customerID; |

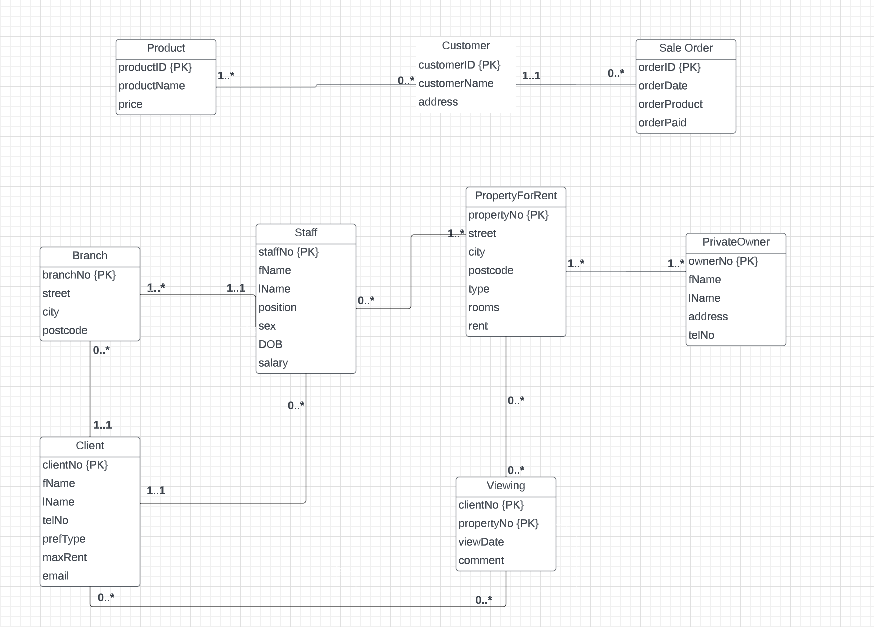


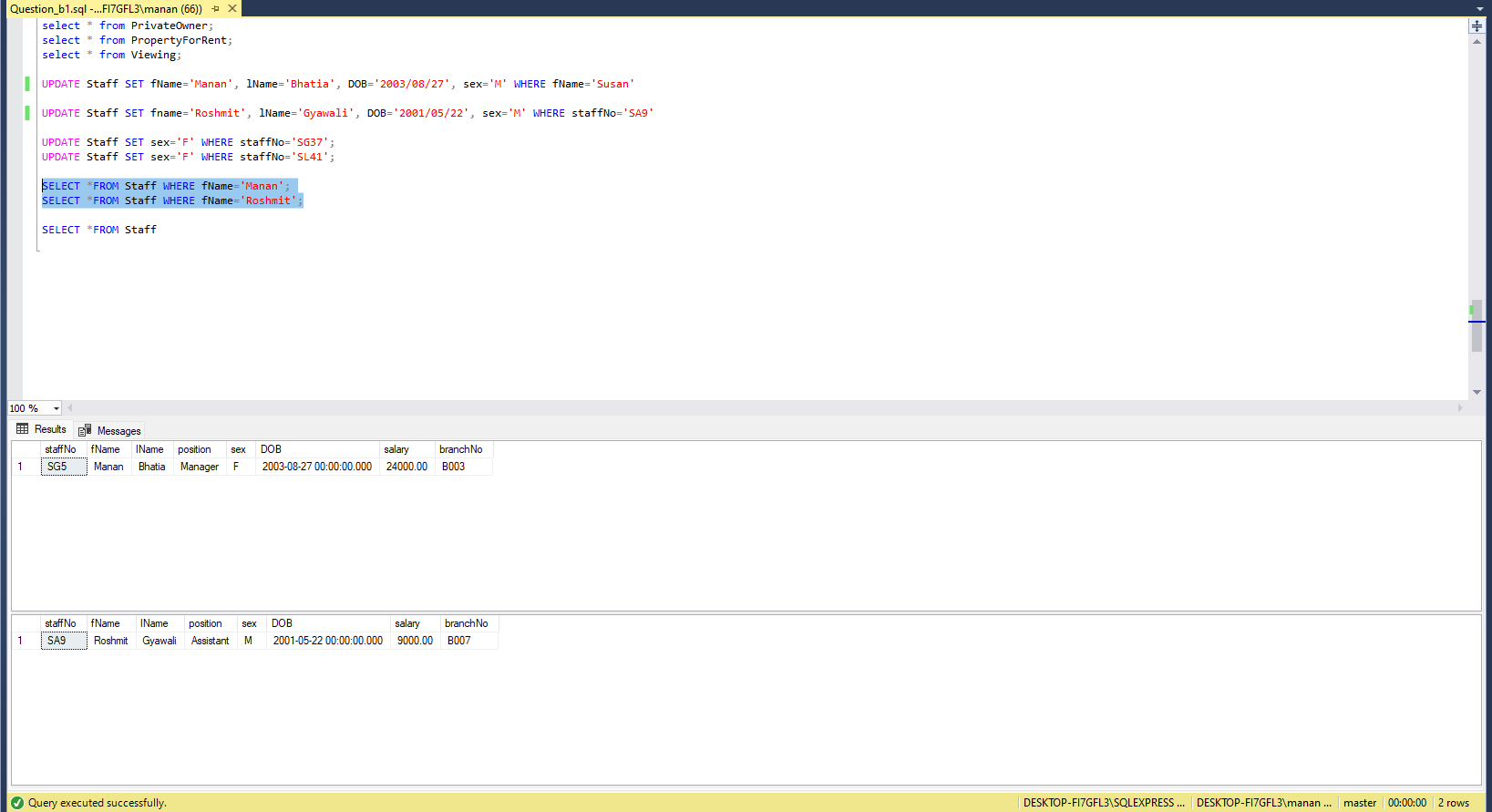
**viii.** Extend the database so that a customer can opt for a home delivery of their ordered product. The home delivery fee will be charged according to the postcode of the customer address. In other words, each delivery to an address of the same postcode will incur the same delivery cost. Draw a separate ERD and a separate GRD for this extended database.

 (ERD)

(GRD)

Q2. More on SQL queries

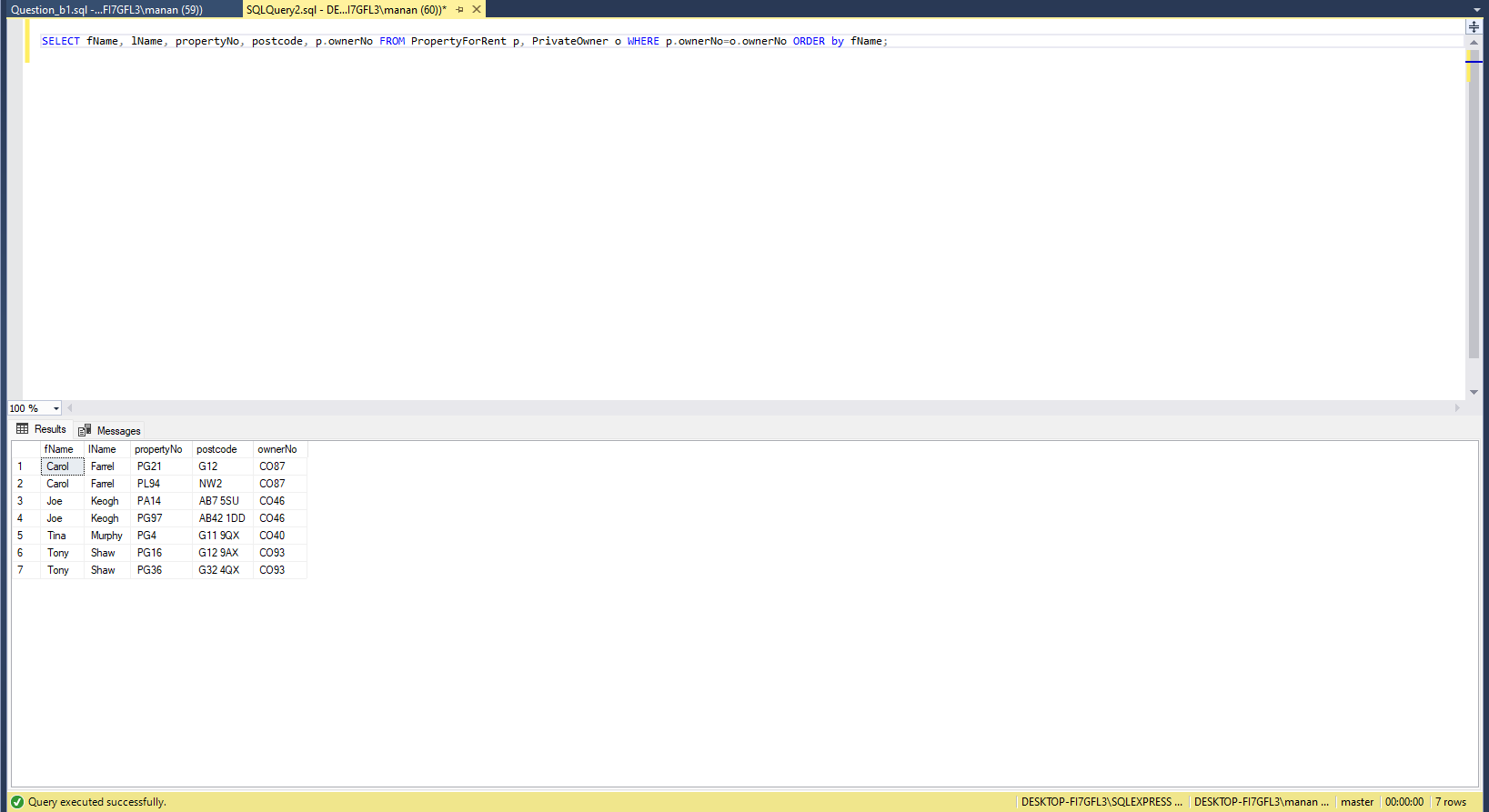
**i.** Draw an ERD to represent the above table-linking diagram (which is essentially what we would call a relation diagram). The ERD should bear fewer entity types than the number of tables in the above displayed diagram. That is, the table or tables that essentially represent relationships should be represented as relationships on the ERD, not as entities. 

**ii.** Create this set of tables and fill the records by executing the given SQL script dreamhome.sql (link added to Assignment1 on vUWS). Then use an UPDATE statement in SQL to modify the staff member "Susan Brand" into your own name and modify her date of birth (03/06/1940) into your date of birth. If you have a team member for this assignment, then also UPDATE the staff record for staffNo = "SA9" by replacing the name "Mary Howe" by that of your team partner.

*Comments:* Overall, the SQL Script of Dreamhouse has been updated from the original names given to the names of Manan and Roshmit, as well as there date of birth.

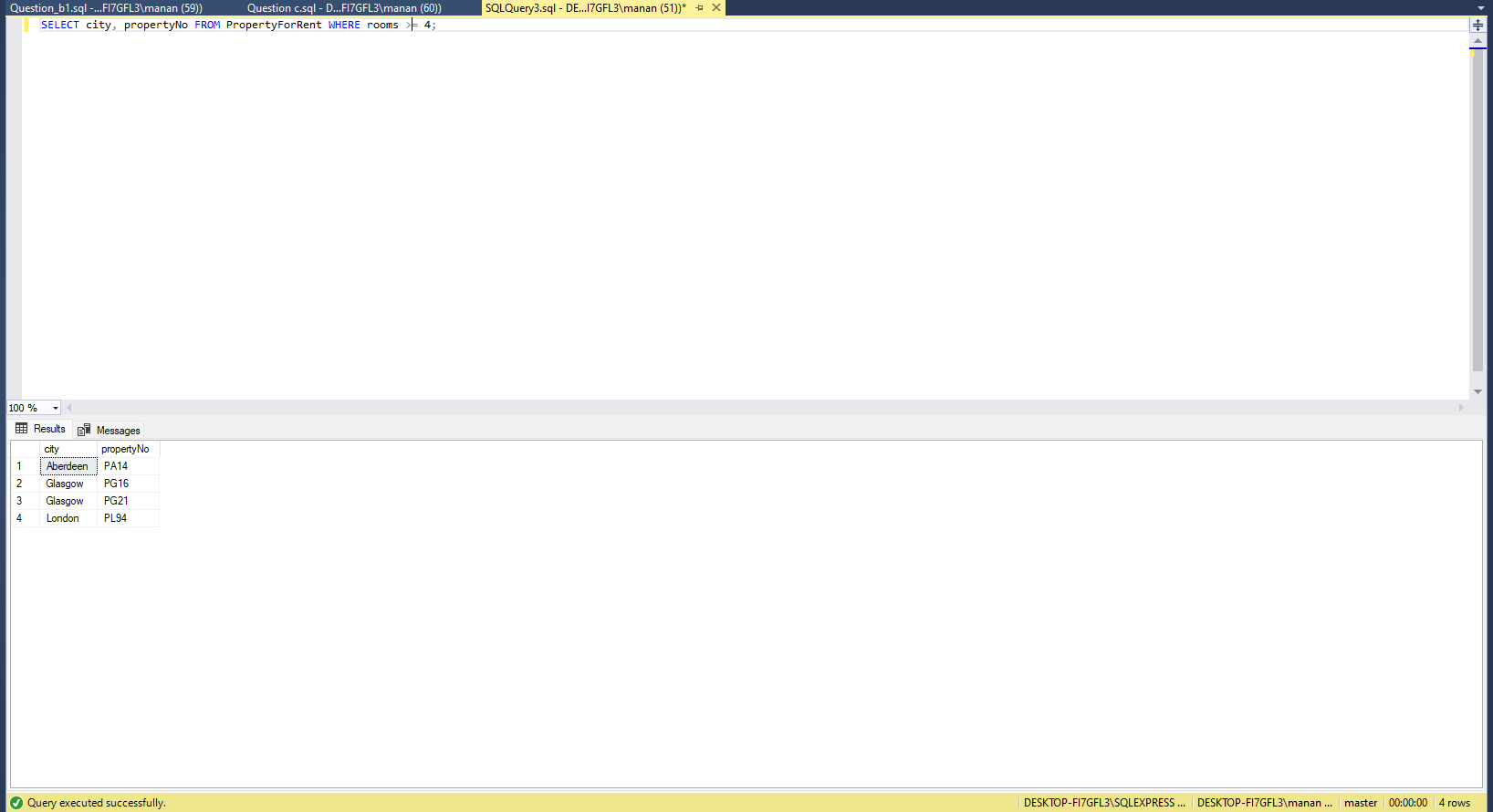
**iii.** Write an SQL statement to list propertyNo, postcode, and the name of the owner of the property. Sort the output according to the owner name alphabetically (screenshot required).

|  |
| --- |
| SELECT fName, lName, propertyNo, postcode, p.ownerNo  FROM PropertyForRent p, PrivateOwner o  WHERE p.ownerNo=o.ownerNo ORDER by fName; |



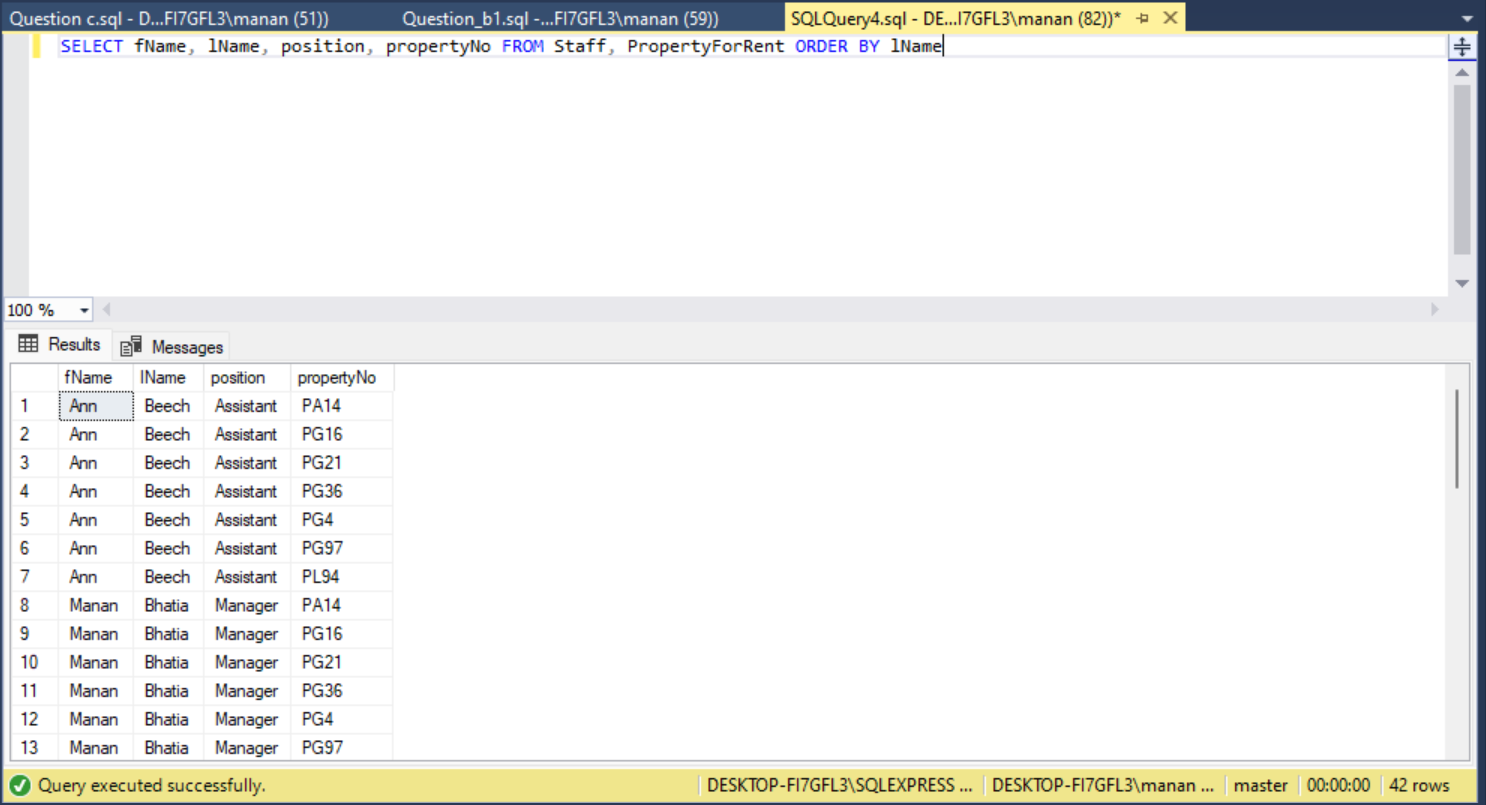
**iv.** Write an SQL statement to list by the city the total number of properties that have 4 or more rooms for rent (screenshot required).

|  |
| --- |
| SELECT city, propertyNo  FROM PropertyForRent WHERE rooms >= 4; |



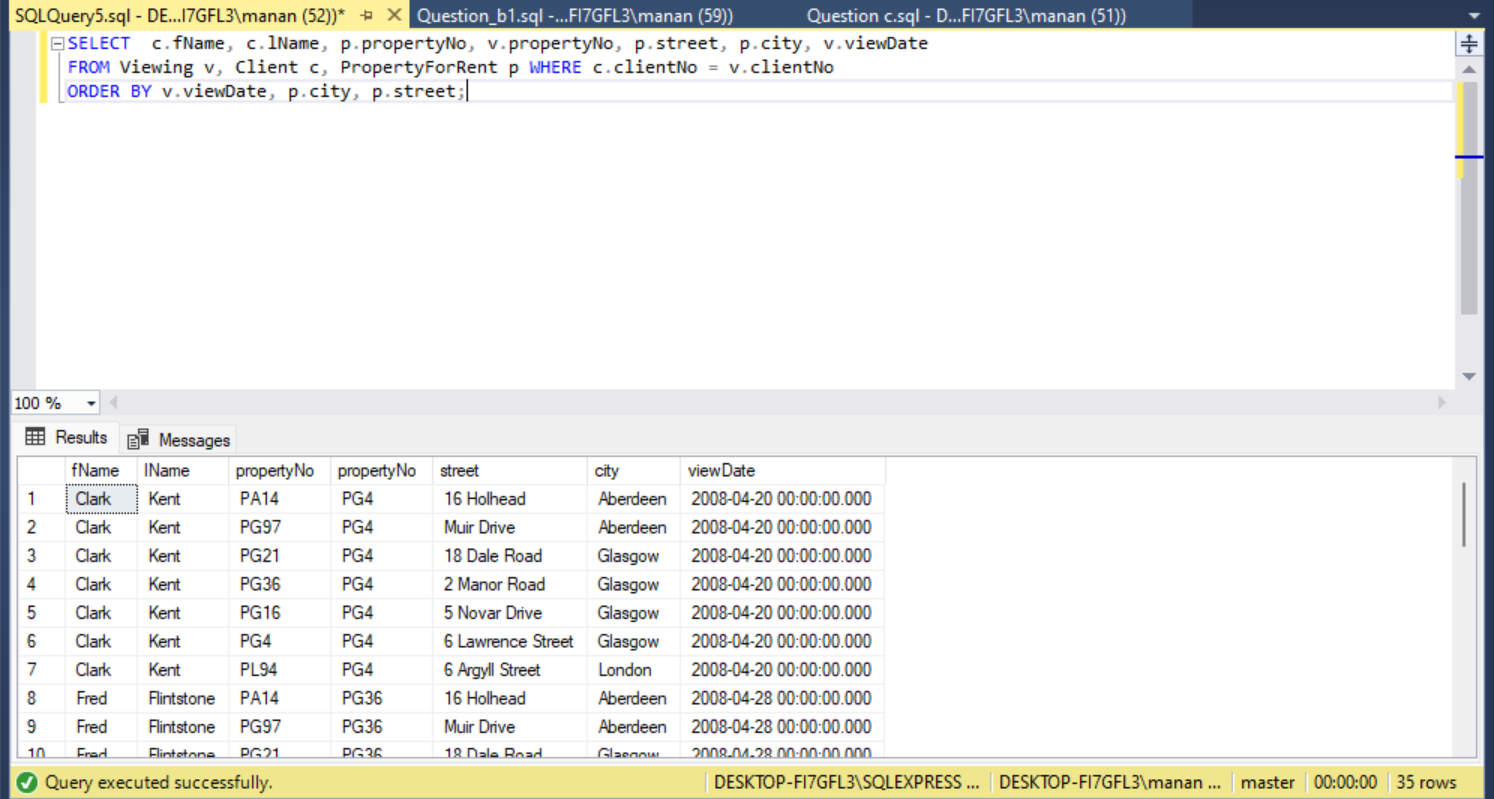
**v.** Write an SQL statement to list staff name, position, and the number of properties managed by the staff member, ordered alphabetically according to the last name (screenshot required).

|  |
| --- |
| SELECT fName, lName, position, propertyNo  FROM Staff, PropertyForRent  ORDER BY lName |



**vi.** Write an SQL statement to list, for all the properties that have been viewed by one or more clients, the client name, the propertyNo of the property the client viewed, the street and the city of the property, and the viewing date. Order the output first by the client name, then by the viewing date (latest first), then by the city and finally by the street name (screenshot required).

|  |
| --- |
| SELECT c.fName, c.lName, p.propertyNo, v.propertyNo, p.street, p.city, v.viewDate  FROM Viewing v, Client c, PropertyForRent p  WHERE c.clientNo = v.clientNo  ORDER BY v.viewDate, p.city, p.street; |



Comments: From part iii to vi, listing all the known attributes from its entities has given unique results.

Q3. Database modelling - case study

In this part, you are asked to design a database to support a university system that provides a platform for students to enrol in a single programme of study and complete a number of fixed core courses specific to that programme as well as a number of electives taken from other programmes. This design will be the basis for part of the forthcoming Assignment 2. The major business requirements are summarised below in the Mini Case: University of Western Sydney. You are asked to develop a detailed Entity-Relationship model for this mini case. Your ER model should consist of a detailed ER diagram integrated with itemised discussions on the features of the entities and relationships and all the assumptions you made. The ER diagram and the accompanying document should identify keys, constraints, entity types, relationship types, specialisation/generalisation, etc. The ER diagram should in general have more than 5 entity types and no more than 15. This means that students will have to selectively identify those most important entity types and relationship types for their work and for the fulfilment of the key business functionalities. More concretely, your deliverables should include:

**i.** A list of supplementary business rules or assumptions that underpin your particular database design. When pertinent, link them to the part of your database design with short comments.

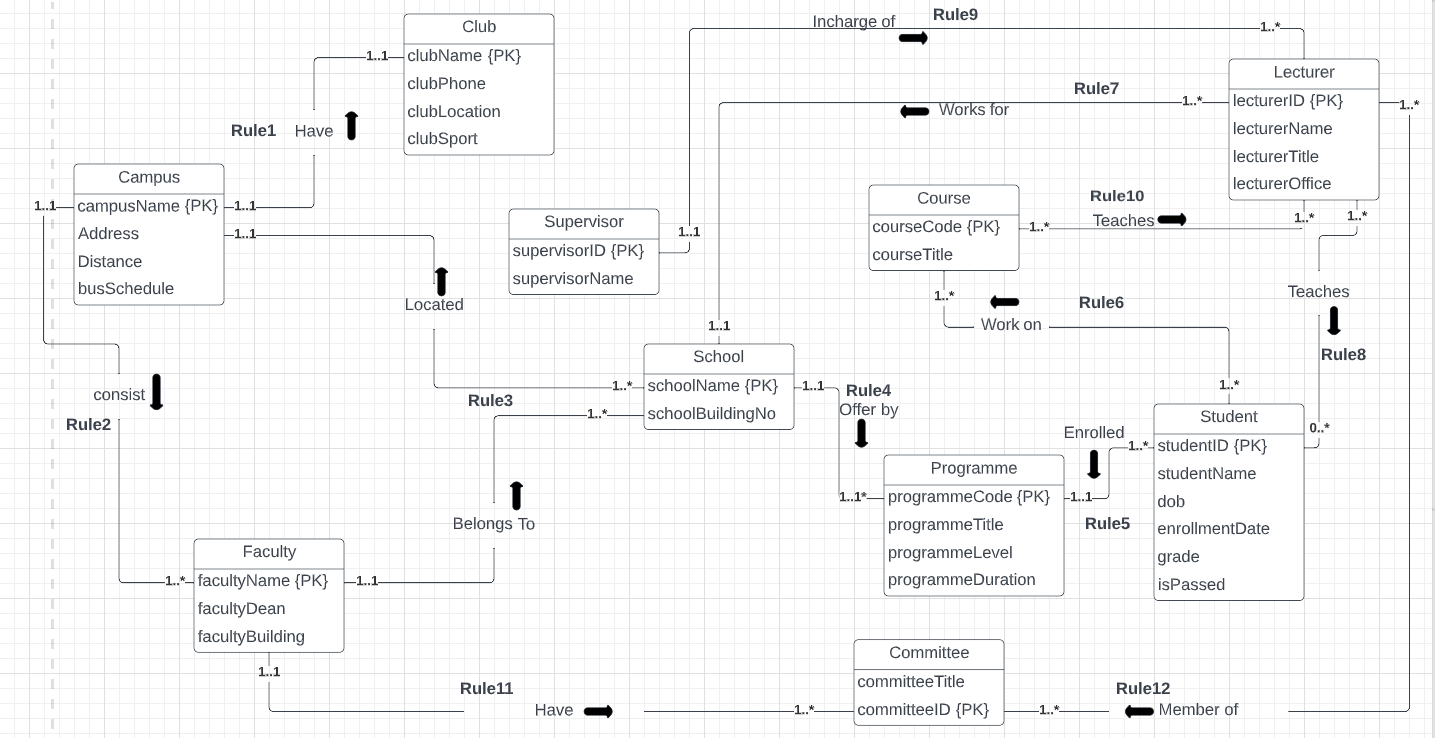
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Campus | Club | Faculty | School | Programme | Course | Student | Lecturer | Committee | Supervisor |
| campusName  Address  Distance  busSchedule | clubName  clubPhone  clubLocation  clubSport | facultyName  facultyDean  facultyBuilding | schoolName  schoolBuildingNo | programmeCode  programmeTitle  programmeLevel  programmeDuration | courseCode  courseTitle | studentID //PK  studentName  studentDOB  enrollDate  grade  isPassed | lecturerID  lecturerName  lecturerTitle  lecturerOffice | committeeTitle  committeeID | supervisorID  supervisorName |

Business Rules:

1. Each campus has one club
2. The University can consists of several faculties
3. A faculty can be divided into several schools. Each school belongs to one faculty only and is located on just one campus, but one campus can have multiple schools.
4. Each school offers different programmes, and each programme can be offered by only one school.
5. Each of the students is enrolled in a single programme of study.
6. Students can work in minimum one to multiple courses.
7. A lecturer is allowed to work for one school only. But a school can have multiple lecturers..
8. A lecturer can teach none at all to several students.
9. A supervisor can be in charge of several lecturers, but a lecturer, however, reports to only one supervisor.
10. A lecturer can teach many different courses. A course may also have been taught by many different lecturers.
11. Each faculty must have a number of committees with the same titles across the university,
12. The committee's members are all lecturers. A lecturer may be a member of several committees.

**ii.** A detailed ER diagram for the model, highlighting primary keys, multiplicity constraints, generalisation/specialisation, important attributes, and other pertinent details.

1. You must use the same notation scheme for the ER diagram as the textbook, and the ER diagram should be strictly in the sense the textbook uses.
2. The ER diagram should include, among others, representative attributes for all entity types, proper subclassing, and correct participation multiplicities for the relationship types. It should be meaningful and well designed and should also include all relevant and necessary aspects.



Question 4 - Selected Additional Exercises

**i.** Complete Question α in the Additional Exercises for Practical 5.

One case of database application in terms is about how a university runs. There are multiple entities on how the structure of the university looks like with different attributes containing its own primary keys and the multiplicity where they connect to other entities. There are many primary keys like the facultyName, campusName, clubName, supervisorID, schoolName, committeeID, lecturerID, courseCode, programmeCode and studentID.

**ii.** Complete Questions β and γ in the Additional Exercises for Practical 6.

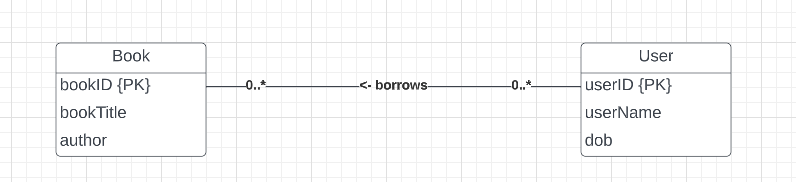
β: 

Choosing the hotelName and price attribute, the Hotel table joins with the Room table where the hotelNo in the Hotel table equals to hotelNo in the Room table, while the type of Room table is Double which is selected.

γ: σ hotelName, city (Hotel = ‘Paris’)

**iii.** Complete Question α in the Additional Exercises for Practical 7.

They are Physical, Logical and Conceptual designs.

Physical Design: 

Conceptual Design:

|  |
| --- |
| CREATE TABLE User (  userID numeric(5) PRIMARY KEY;  userName varchar(20);  dob datetime;  borrowID numeric(5) FOREIGN KEY;  )  CREATE TABLE Book (  bookID numeric(5) PRIMARY KEY;  bookTitle varchar(20);  author varchar(20);  userID numeric(5) FOREIGN KEY;  )  CREATE TABLE Borrow (  borrowID numeric(5) PRIMARY KEY FOREIGN KEY;  bookID numeric(5) FOREIGN KEY; |

Logical Design:

