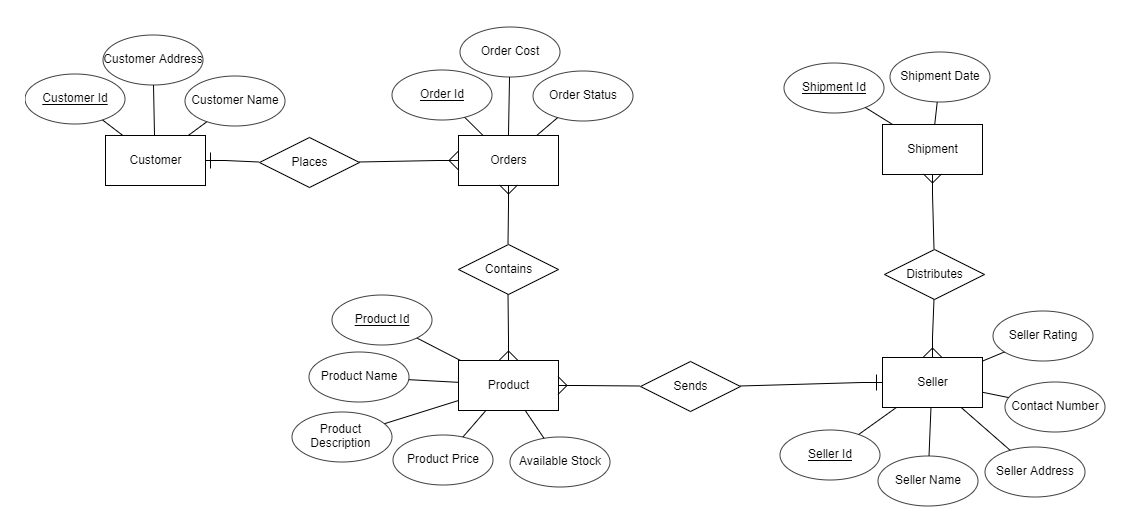
**Database Project**

**Kavya Kommineni**

I’ve considered Amazon **online shopping** database use case for this project which includes Customer, Orders, Product, Seller and Shipment entities with their corresponding relationships.

**Entity Relationship Diagram**

The Entity Relationship (ER) diagram for the above specified model is as follows:



**Entities:**

The entities for the model are: Customer, Orders, Product, Seller and Shipment.

**Attributes:**

**Customer**: Customer Id, Customer Address and Customer Name

**Orders:** Order Id, Order cost and Order Status

**Product:** Product Id, Product Name, Product Description, Product Price and Available stock

**Seller:** Seller Id, Seller Name, Seller Address, Contact Number and Seller Rating

**Shipment:** Shipment Id, Shipment Date

**Relationships:**

**Places:** It defines relation between the entities Customer and Orders. Customer can place an order.

**Contains:** It defines relation between the entities Orders and Product. An order contains the products.

**Sends:** It defines relation between the entities Product and Seller. The ordered products are being sent to the seller after the customer places an order.

**Distributes:** It defines relation between the entities Seller and Shipment. Seller distributes the products to the shipment.

**Cardinality:**

**Places:** One customer can place more than one order. So, it has one to many cardinality between customer and orders entities.

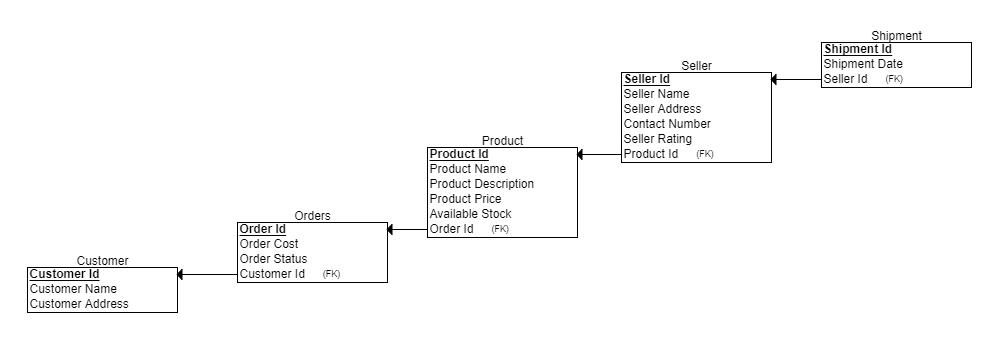
**Contains:** One order can contain many products and one product can be contained in many orders. So, it has many to many cardinality between orders and product entities.

**Sends:** Many products can be sent to the seller in an order. So, it has one to many cardinality between seller and product entities.

**Distributes:** A seller can distribute the products to different shipments. Shipment can receive many products from different sellers. So, it has many to many cardinality between Seller and Shipment entities.

**Conversion to Relations:**

The relational schemas as per the above ER diagram is as follows:



This schema contains all the relations which are equivalent to the entities in ER diagram. The fields in the schema are equivalent to the attributes of the entities in the ER diagram. The fields with the underline are the primary key for that relation. The Foreign keys are represented with (FK) for the corresponding key in the relation.

**SQL Schemas:**

The above relational schemas can be converted into SQL schemas with all the constraints and the keys as follows:

CREATE TABLE **Customer**

(

Customer\_Name VARCHAR(50) NOT NULL,

Customer\_Address VARCHAR(50) NOT NULL,

Customer\_Id VARCHAR(50),

PRIMARY KEY (Customer\_Id)

);

CREATE TABLE **Orders**

(

Order\_Id VARCHAR(50),

Order\_Cost REAL DEFAULT(0.00),

Order\_Status VARCHAR(50),

Customer\_Id VARCHAR(50) NOT NULL,

PRIMARY KEY (Order\_Id),

FOREIGN KEY (Customer\_Id) REFERENCES Customer(Customer\_Id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE **Product**

(

Product\_Id VARCHAR(50),

Product\_Name VARCHAR(50) NOT NULL,

Product\_Description VARCHAR(100),

Product\_Price REAL NOT NULL,

Available\_Stock INT NOT NULL,

Order\_Id VARCHAR(50) NOT NULL,

PRIMARY KEY (Product\_Id),

FOREIGN KEY (Order\_Id) REFERENCES Orders(Order\_Id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE **Seller**

(

Seller\_Name VARCHAR(50) NOT NULL,

Seller\_Address VARCHAR(50) NOT NULL,

Contact\_Number VARCHAR(50) NOT NULL,

Seller\_Rating REAL NOT NULL,

Seller\_Id VARCHAR(50),

Product\_Id VARCHAR(50) NOT NULL,

PRIMARY KEY (Seller\_Id),

FOREIGN KEY (Product\_Id) REFERENCES Product(Product\_Id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE **Shipment**

(

Shipment\_Id VARCHAR(50) NOT NULL,

Shipment\_Date DATE NOT NULL,

Seller\_Id VARCHAR(50) NOT NULL,

PRIMARY KEY (Shipment\_Id),

FOREIGN KEY (Seller\_Id) REFERENCES Seller(Seller\_Id) ON DELETE CASCADE ON UPDATE CASCADE

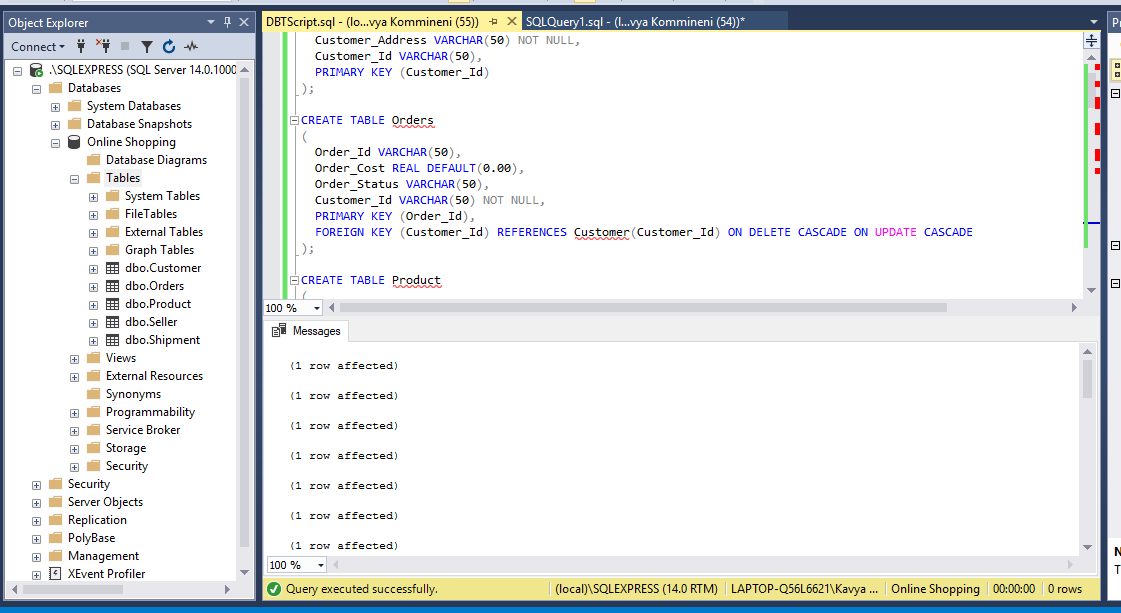
);

**Database Implementation Scripts:**

Here is the DBT Script I’ve used including all the schemas with all the constraints and the data to insert into the tables.



Below is the screenshot after executing the script.



Here are all the tables with the inserted data:

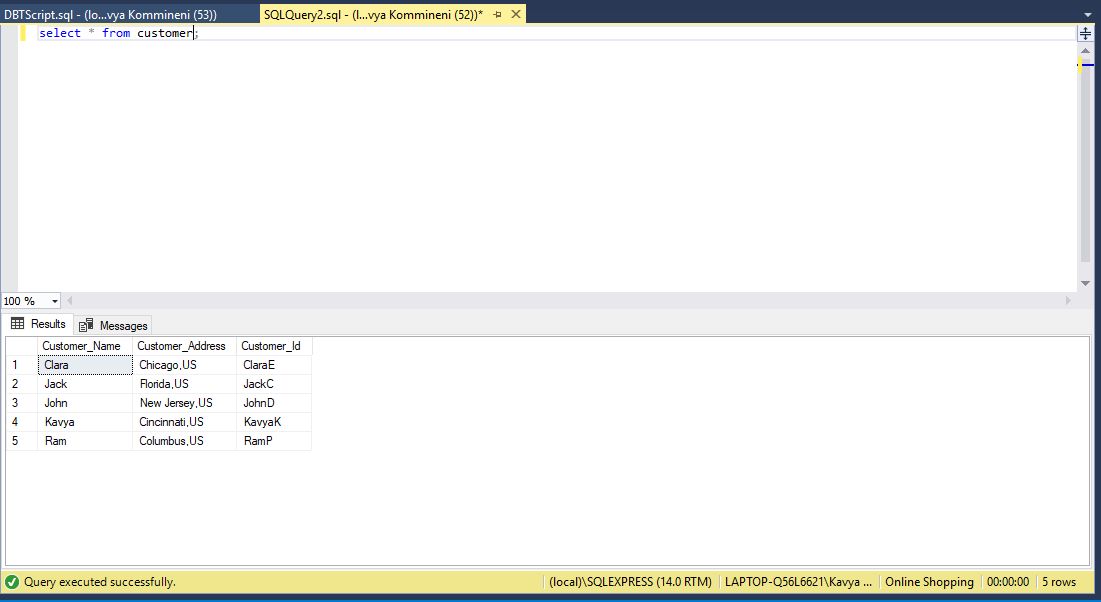
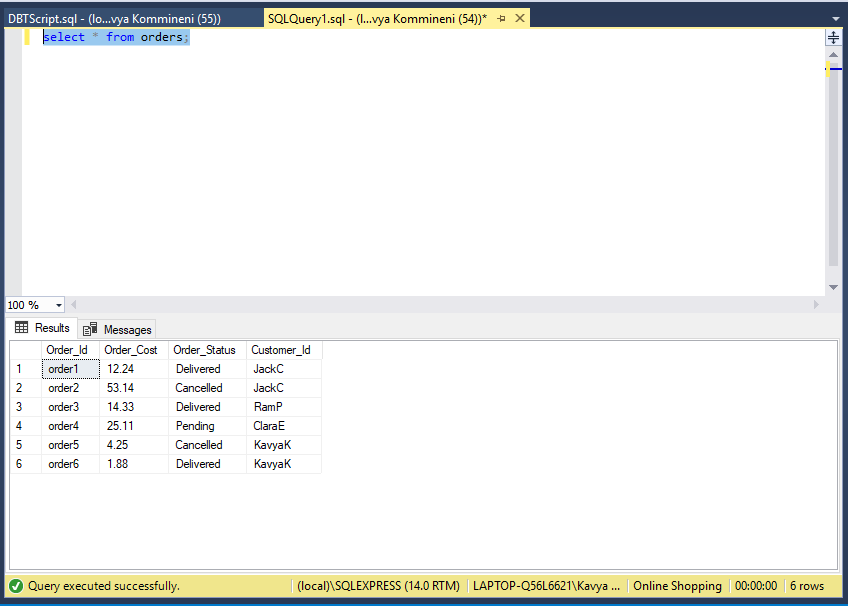
select \* from Customer;

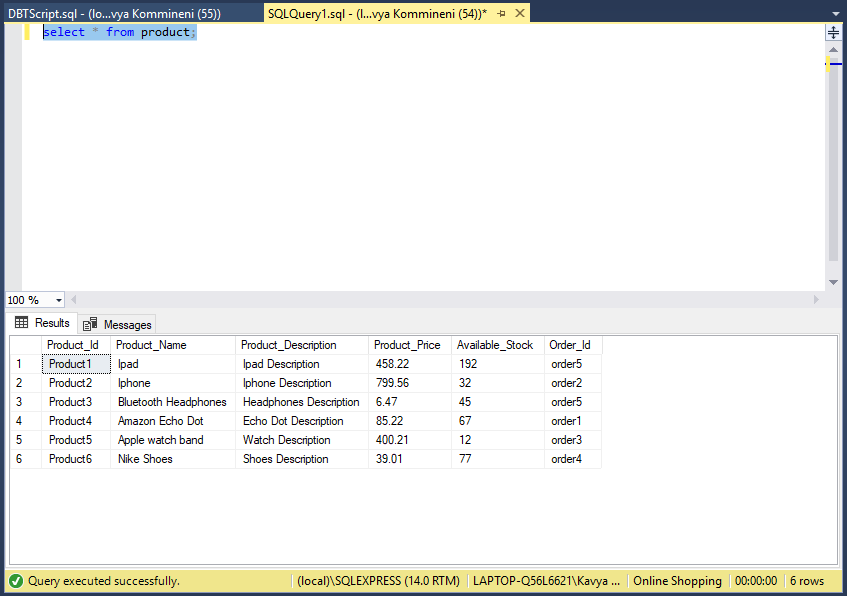
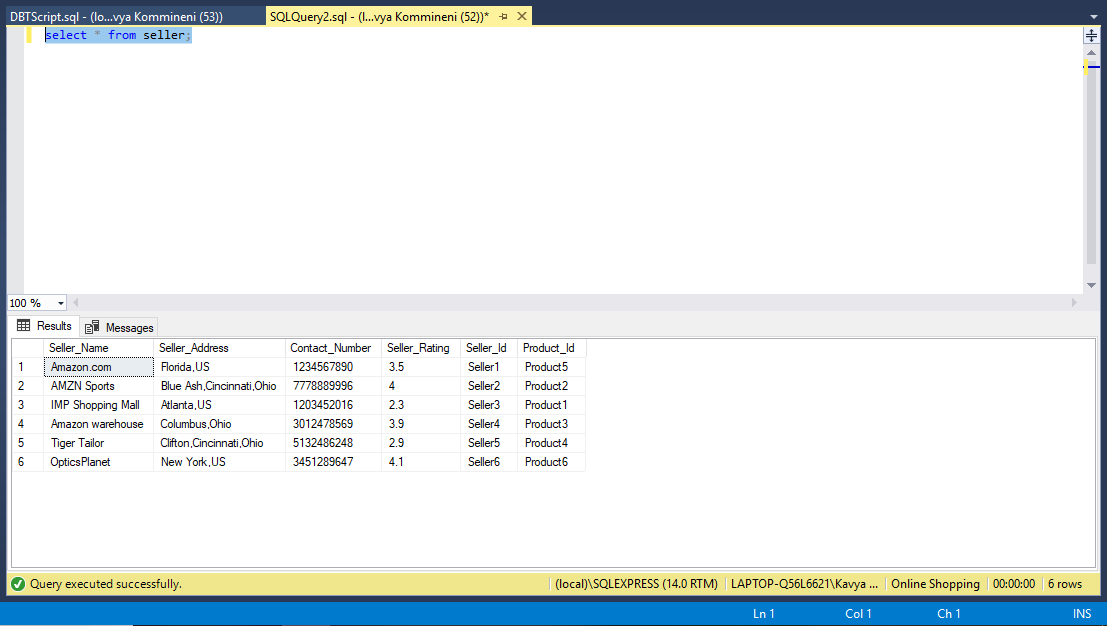
select \* from Orders;

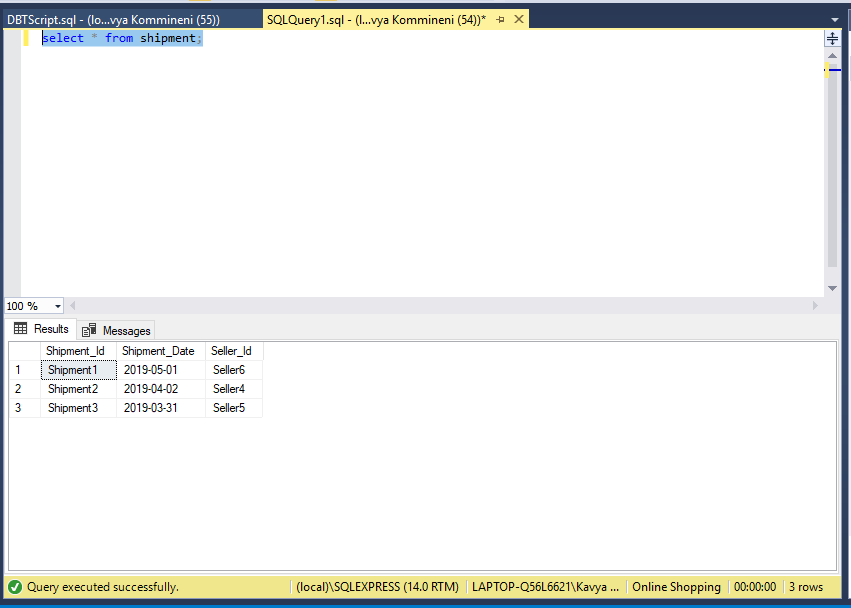
select \* from Product;

select \* from Seller;

select \* from Shipment;

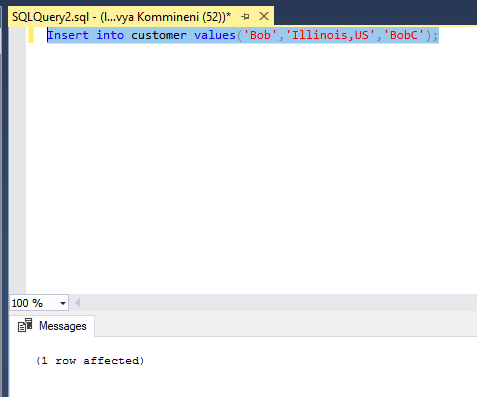


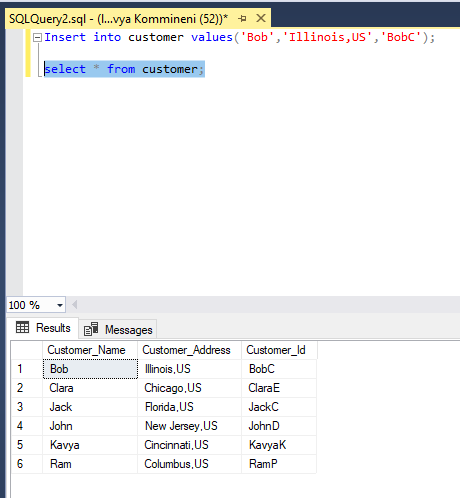
**Queries for Insert, Update and Delete:**

**For Customer Table:**

**Insert query:**

Insert into customer values('Bob','Illinois,US','BobC');

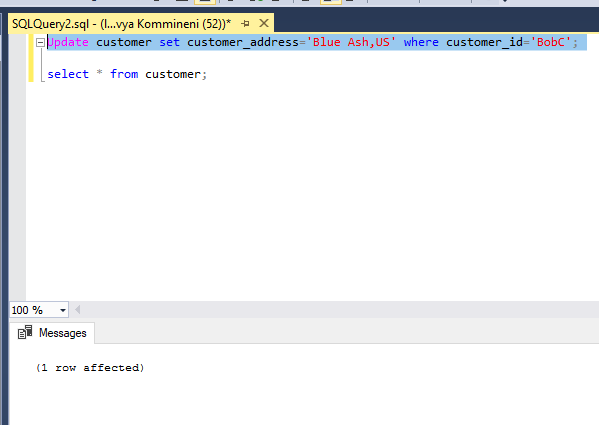


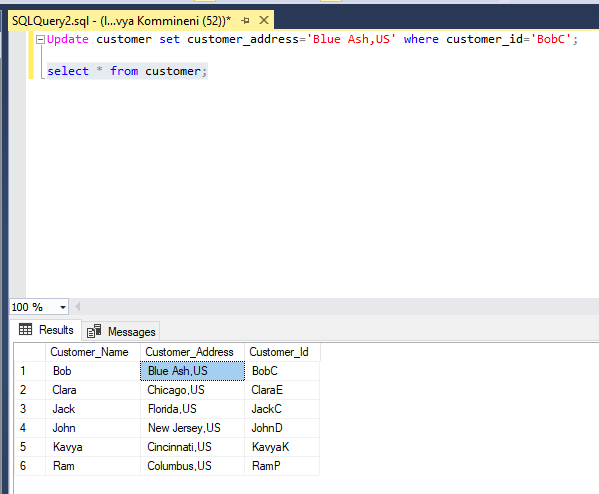


**Update query:**

Update customer set customer\_address='Blue Ash,US' where customer\_id='BobC';

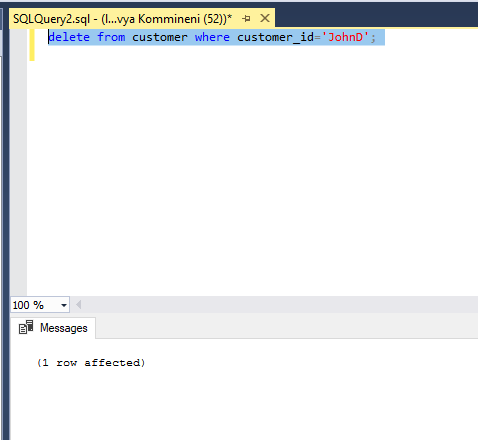
This query updates the address of the customer whose customer id is BobC

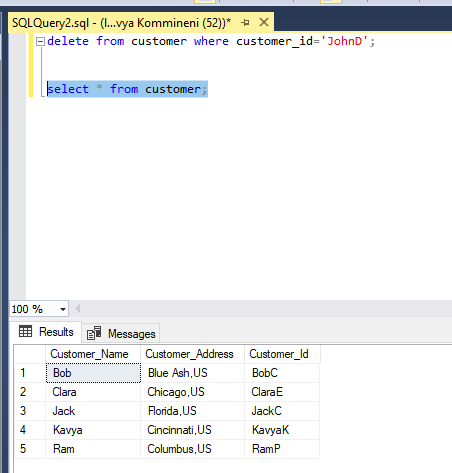




**Delete query:**

delete from customer where customer\_id='JohnD';

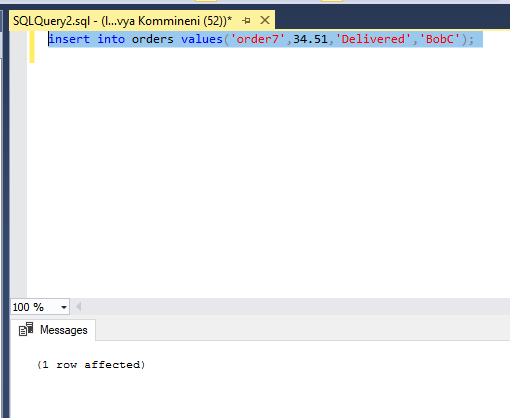


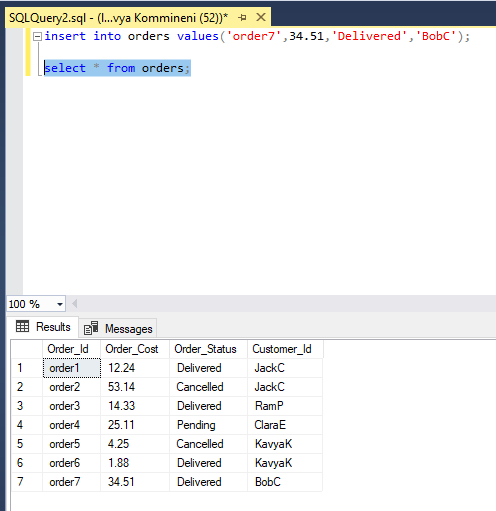


**For Orders Table:**

**Insert query:**

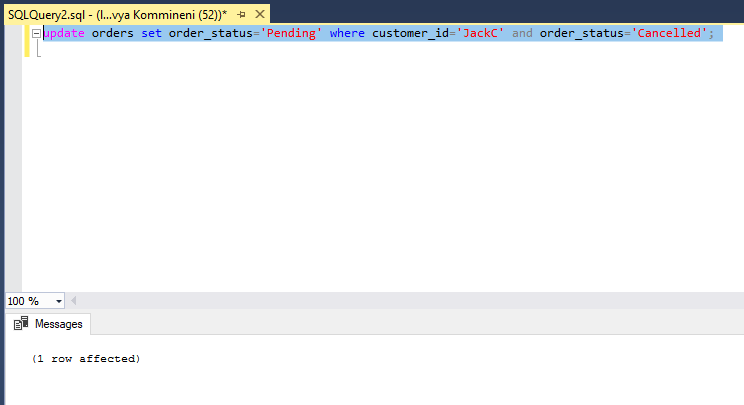
insert into orders values('order7',34.51,'Delivered','BobC');

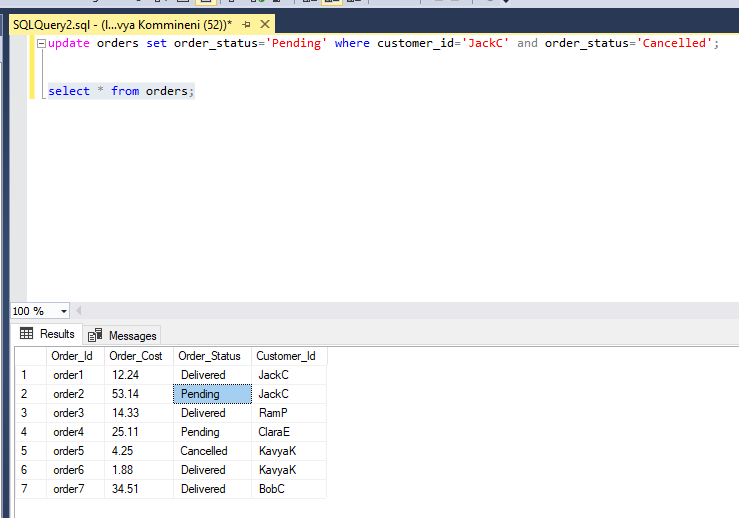




**Update query:**

update orders set order\_status='Pending' where customer\_id='JackC' and order\_status='Cancelled';

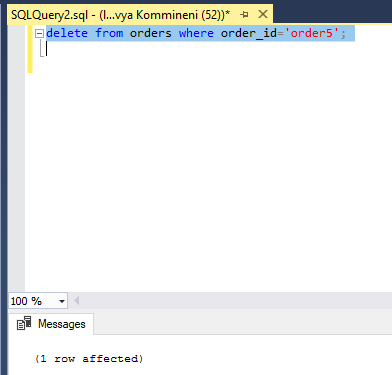


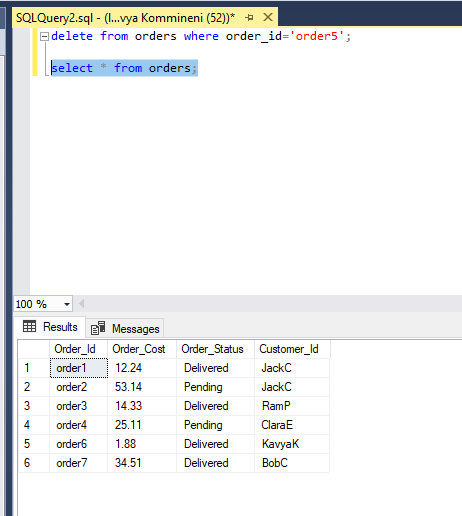


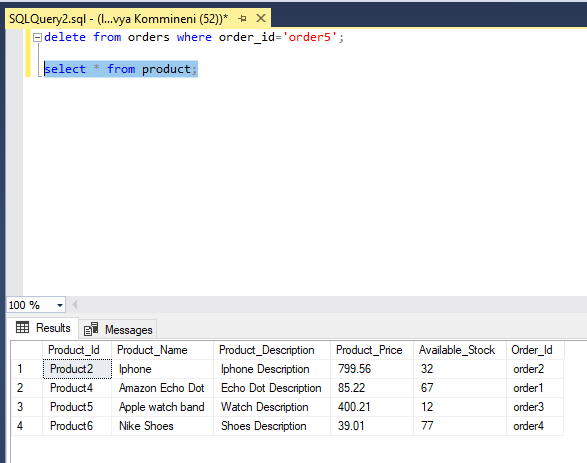
**Delete query:**

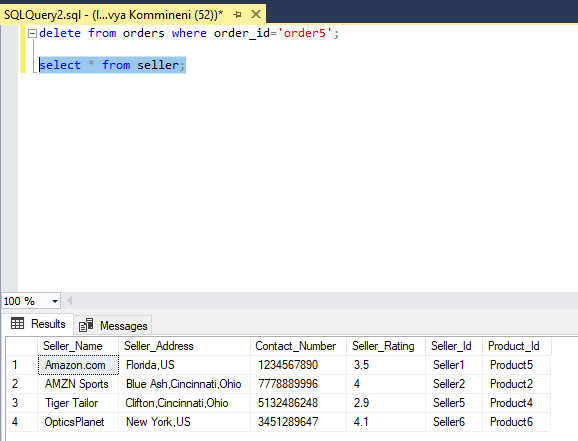
delete from orders where order\_id='order5';

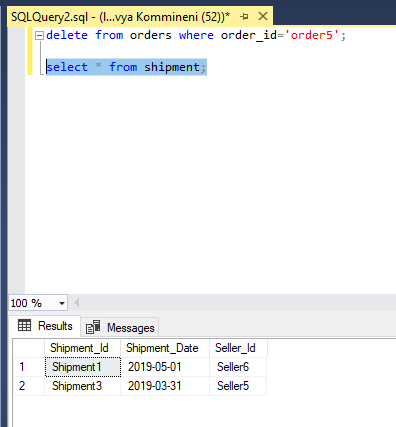
This query deletes the row in the table whose order id is “order5”. It also deletes the corresponding rows in other tables as well where order\_id is the foreign key in that table.







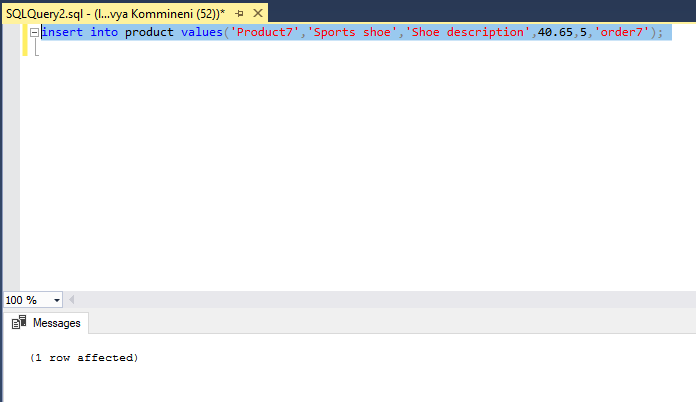


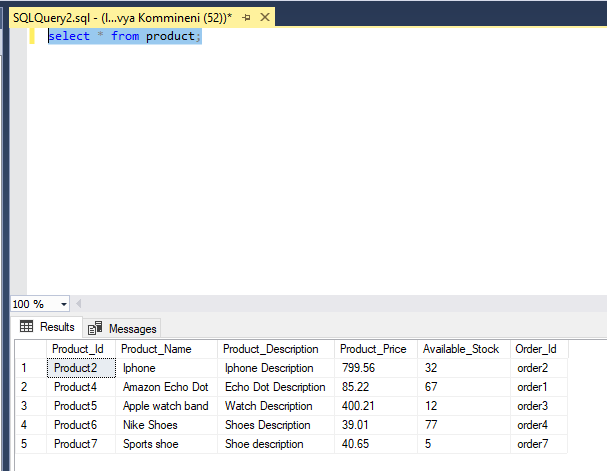


**For Product Table:**

**Insert query:**

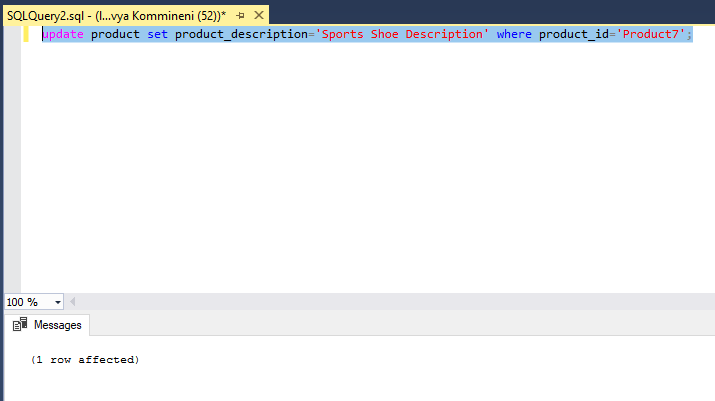
insert into product values('Product7','Sports shoe','Shoe description',40.65,5,'order7');

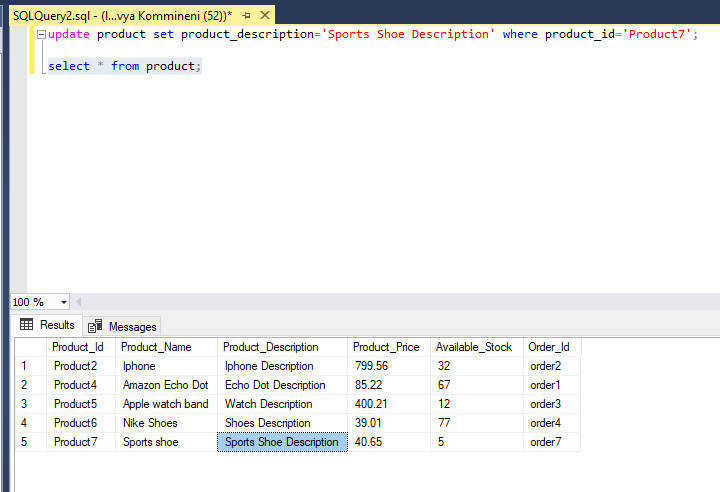




**Update query:**

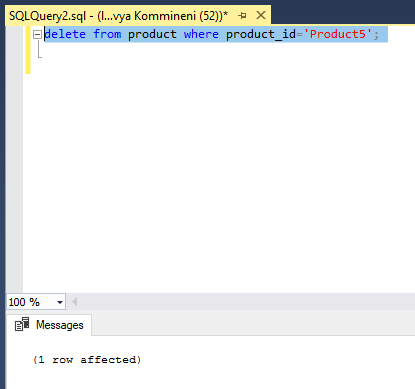
update product set product\_description='Sports Shoe Description' where product\_id='Product7';

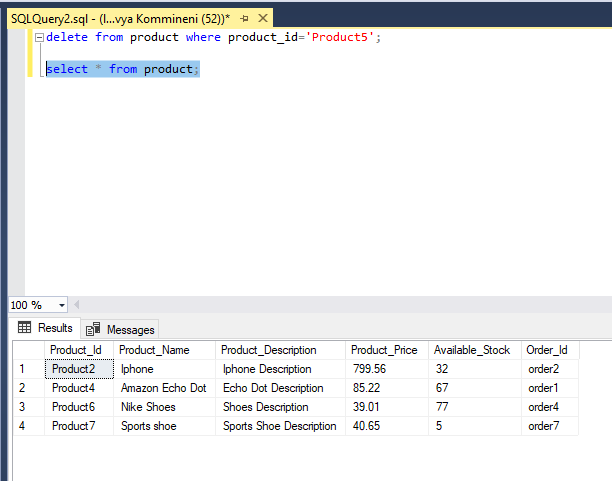


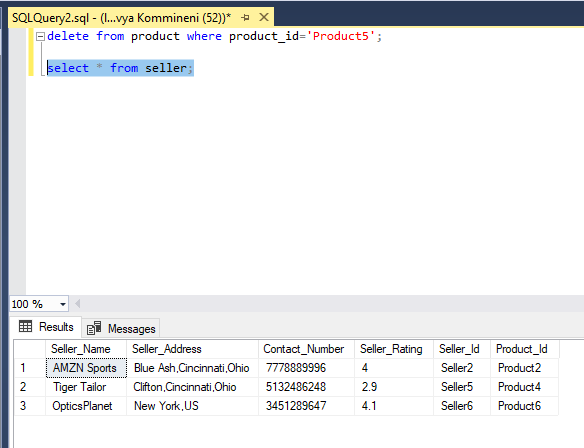


**Delete query:**

delete from product where product\_id='Product5';



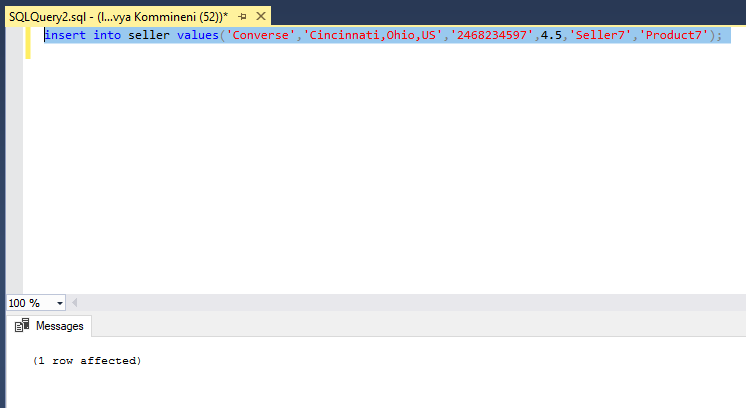


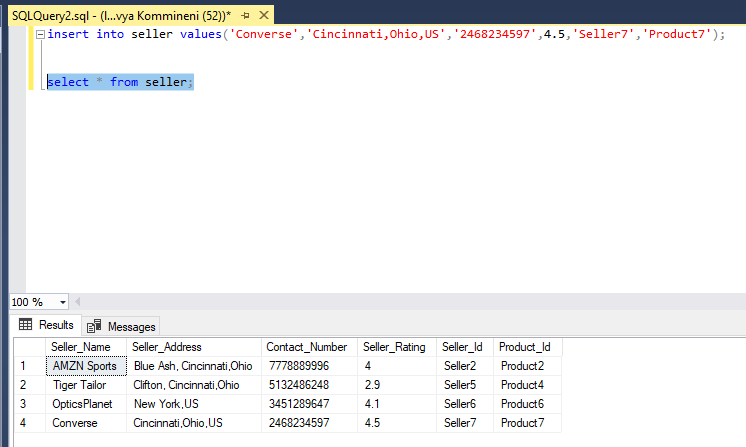


**For Seller table:**

**Insert query:**

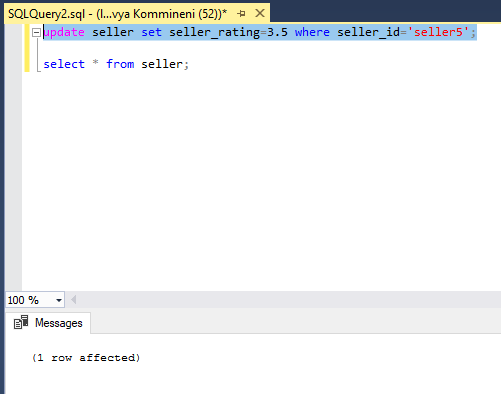
insert into seller values('Converse','Cincinnati,Ohio,US','2468234597',4.5,'Seller7','Product7');

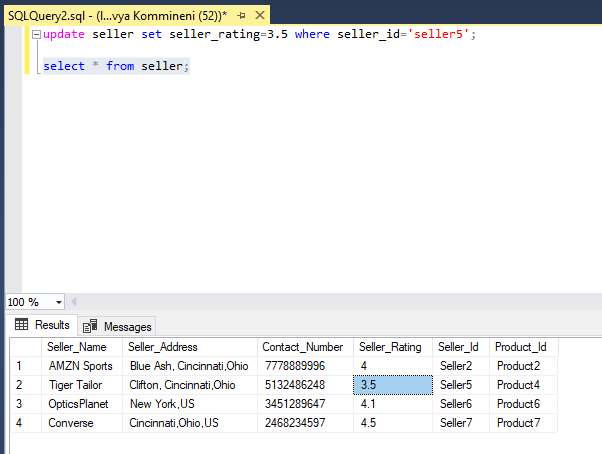




**Update query:**

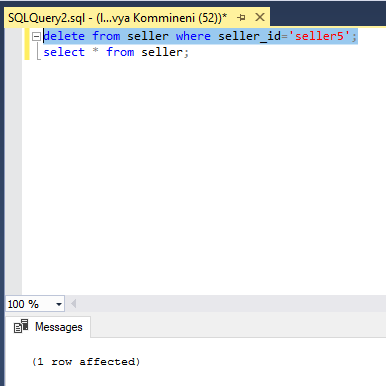
update seller set seller\_rating=3.5 where seller\_id='seller5';

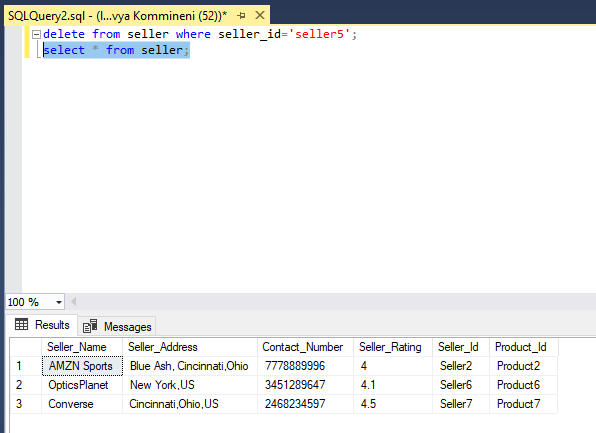


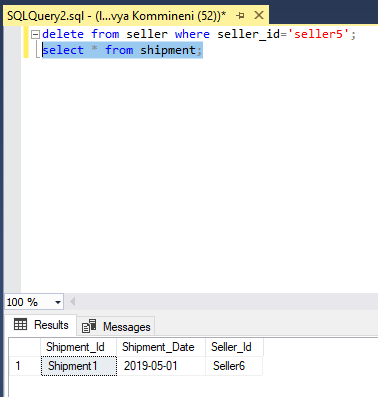


**Delete query:**

delete from seller where seller\_id='seller5';



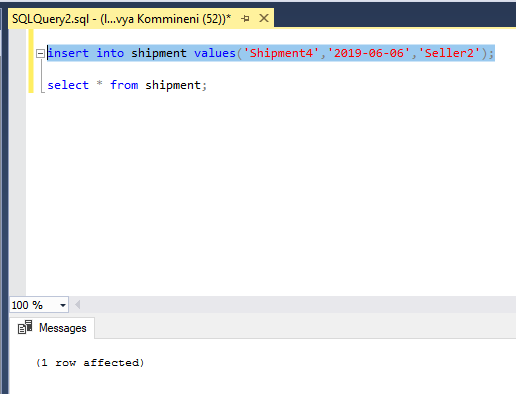


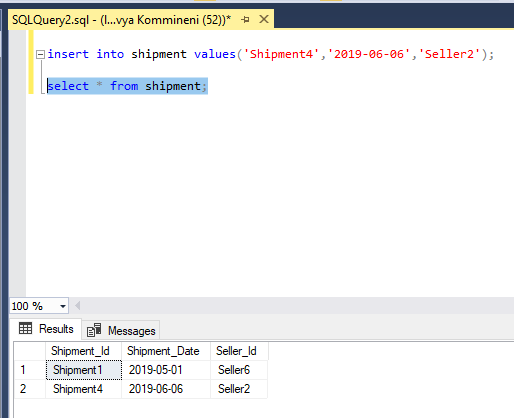


**For Shipment table:**

**Insert query:**

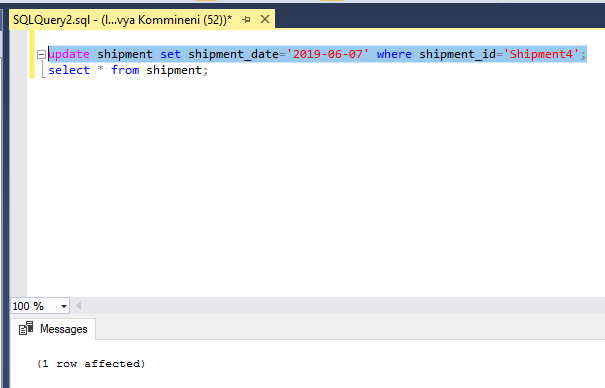
insert into shipment values('Shipment4','2019-06-06','Seller2');

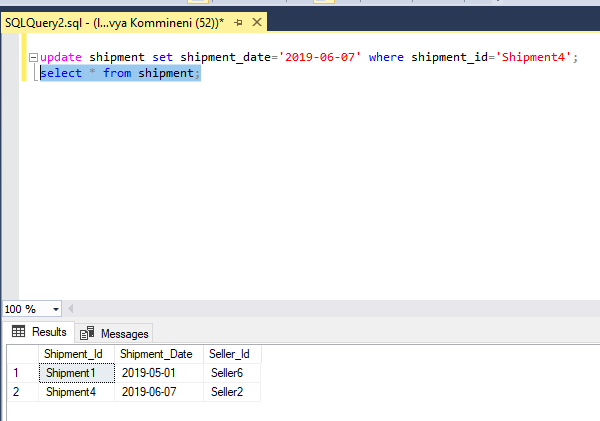




**Update query:**

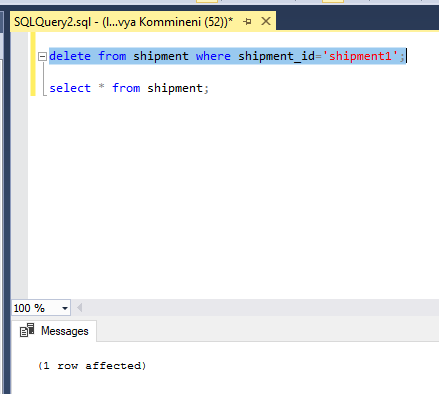
update shipment set shipment\_date='2019-06-07' where shipment\_id='Shipment4';

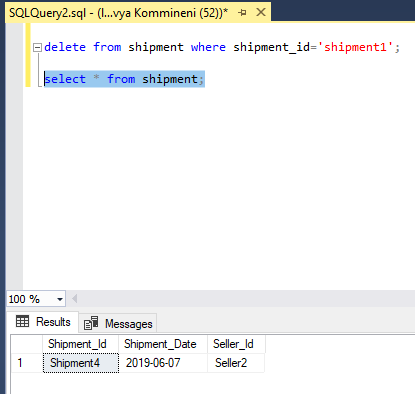




**Delete query:**

delete from shipment where shipment\_id='shipment1';





**Java Application which connects to the database using JDBC:**

Here is the screenshot of the output which is giving all the details of the customer table in the Java application. I’ve used a statement in the application to retrieve all the details from the Customer table. I’ve also a new row into the Customer table. Retrieved and displayed the data of Customer table after inserting the new row. I’ve added the appropriate connector jars to the project to connect to the database.

Below is the Java application I’ve developed to connect to the database and perform actions on the Customer table.



To connect to the database, I’ve used the JDBC in which I’ve provided the JDBC URL string as follows:

Where, **Jdbc:sqlserver** is the server I am using,

**localhost**: As I am running the application on my local machine, I am using localhost

**1410**: This is the port on which SQL is running in my local machine

**databaseName=Online Shopping**: This represents the database name to which I am trying to connect in SQL server

"jdbc:sqlserver://localhost:1410;" +

"databaseName=Online Shopping;integratedSecurity=true;"

