SQL Practical Exercise

Exercise 1 – Northwind Queries

* 1. Write a query that lists all Customers in either Paris or London. Include Customer ID, Company Name and all address fields.

Answer

To view Customers table, so I can deduce what columns are required for the query:

USE Northwind

SELECT \* FROM Customers;

From this, I deduced that the Customer ID, Company Name, Address, City, PostalCode and Country columns are required.

Query that that lists all Customers in either Paris or London including Customer ID, Company Name and all address fields:

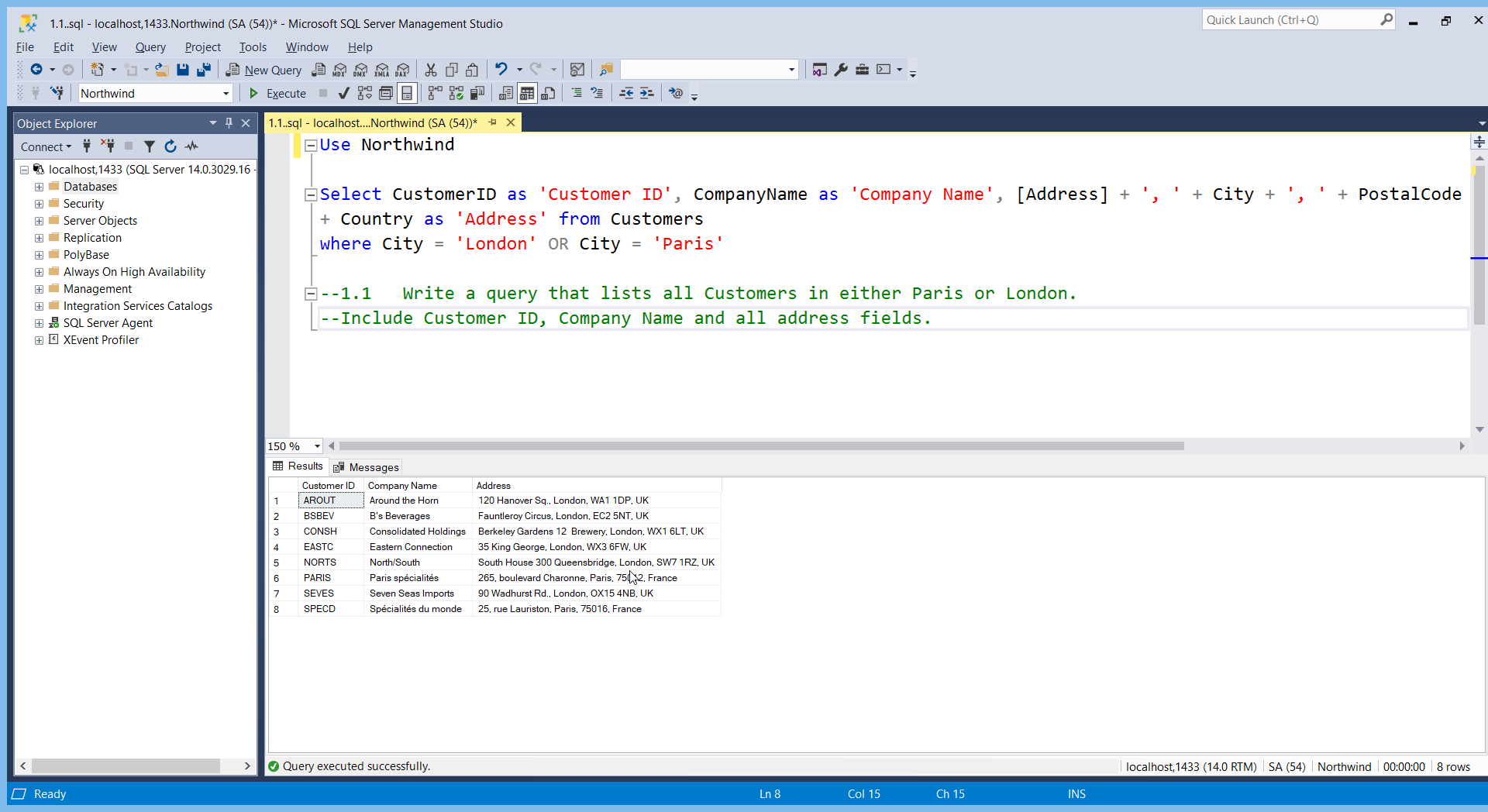
Use Northwind

Select CustomerID as 'Customer ID', CompanyName as 'Company Name', [Address] + ', ' + City + ', ' + PostalCode + ', ' + Country as 'Address'

from Customers

where City = 'London' OR City = 'Paris';

Table showing all the customers in either Paris or London:



* 1. List all products stored in bottles.

Answer

To view Product table, so I can deduce what columns are required for this question:

Use Northwind

Select \* from Products;

From this, I deduced that the “QuantityPerUnit” column is required to list all products that are stored in bottles.

Query to list products stored in bottles are:

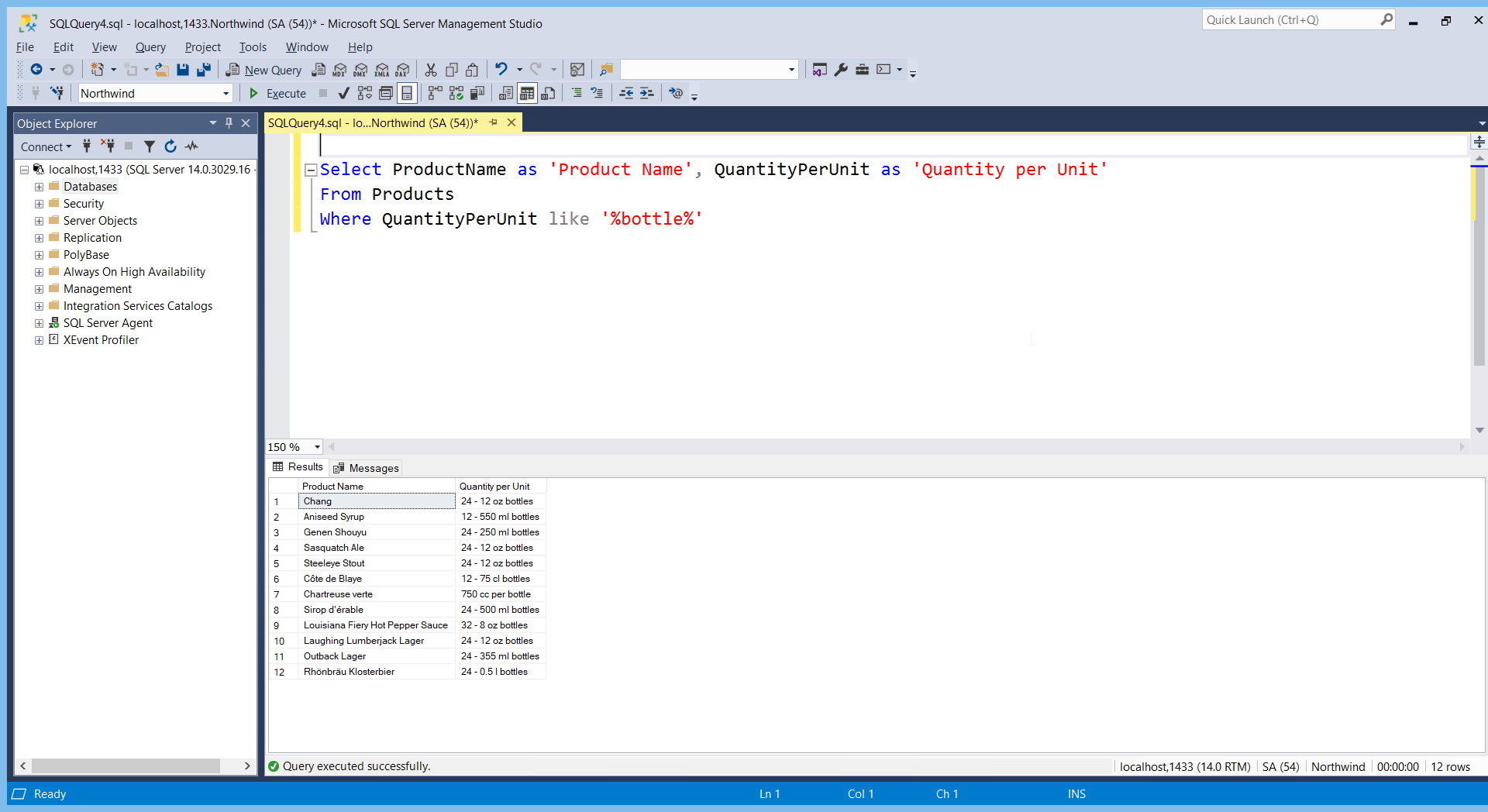
Use Northwind

Select ProductName as 'Product Name', QuantityPerUnit as 'Quantity per Unit'

From Products

Where QuantityPerUnit like '%bottle%';

The list:



* 1. Repeat question above but add in the Supplier Name and Country.

Looking at the products table, the supplier name and country is absent. So these columns need to be found. I looked at the Supplier table, which had information on both suppler name and country.

Query used to do this:

Use Northwind

Select \* from Suppliers;

So, the products table and the suppliers need to be joined, as well listing the products stored in bottles. Query to do this:

Use Northwind

Select ProductName as 'Product Name', QuantityPerUnit as 'Quantity per Unit', Suppliers.CompanyName as

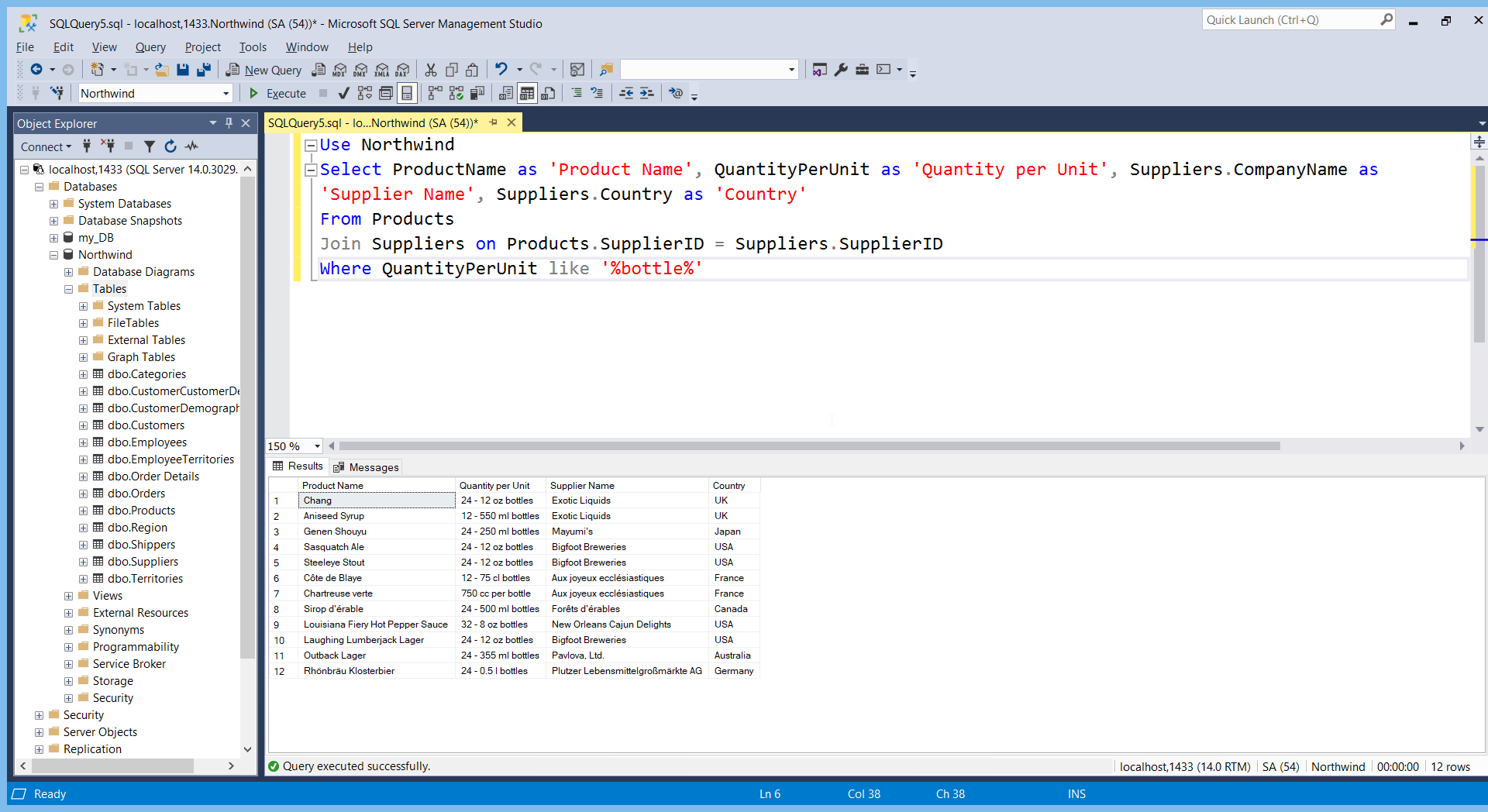
'Supplier Name', Suppliers.Country as 'Country'

From Products

Join Suppliers on Products.SupplierID = Suppliers.SupplierID

Where QuantityPerUnit like '%bottle%';

The list with supplier name and country is:



* 1. Write an SQL Statement that shows how many products there are in each category. Include Category Name in result set and list the highest number first.

Looking at the products table, there is no category name. So, the categories table and the product table must be joined. Then the categories must be grouped and counted to show how many products there are in each category. Finally, the numbers must be ordered in descending order to show the highest number.

The SQL statement is:

use Northwind

Select Categories.CategoryName as 'Category Name', count(Products.CategoryID) as 'Number of products'

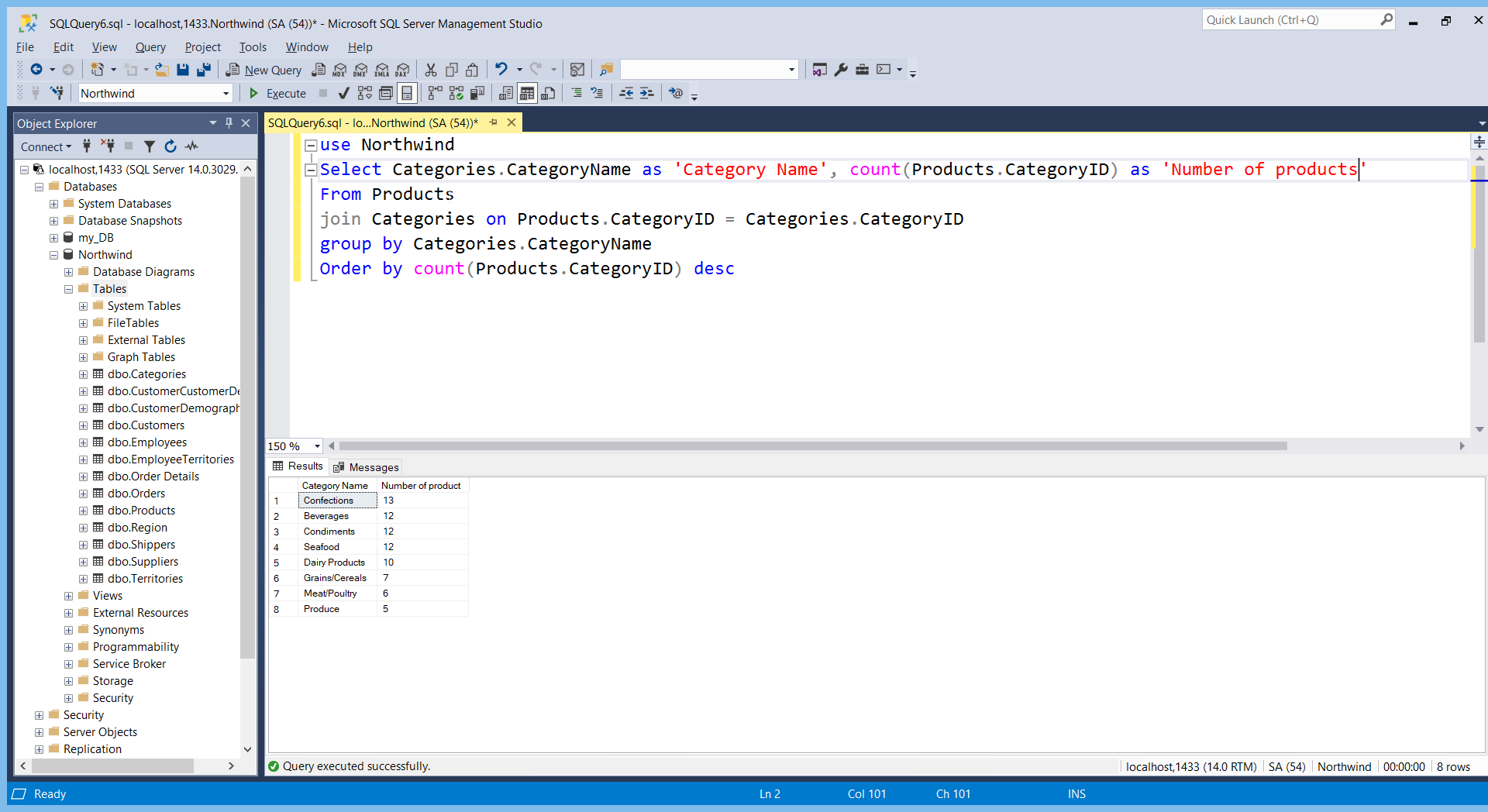
From Products

join Categories on Products.CategoryID = Categories.CategoryID

group by Categories.CategoryName

Order by count(Products.CategoryID) desc;

Result of this statement is:



* 1. List all UK employees using concatenation to join their title of courtesy, first name and last name together. Also include their city of residence.

To answer this question, we need to look at the employees table. Query to do that is:

Use Northwind

Select \* from Employees

All the information required for the concatenation is in the table. The table needs to be filtered for UK employees by using the where section. The query for this is:

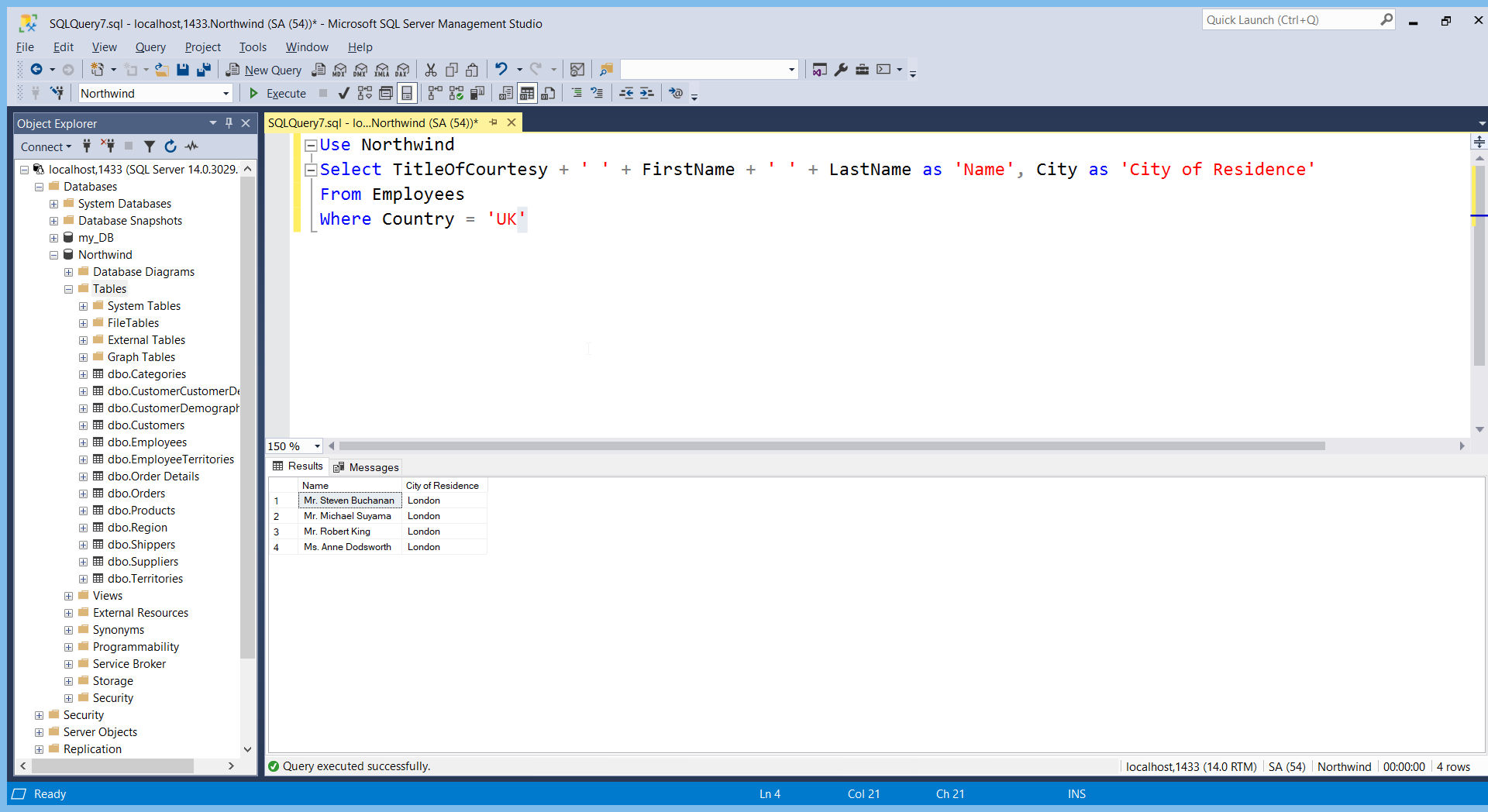
Use Northwind

Select TitleOfCourtesy + ' ' + FirstName + ' ' + LastName as 'Name', City as 'City of Residence'

From Employees

Where Country = 'UK'

The list is:



* 1. List Sales Totals for all Sales Regions (via the Territories table using 4 joins) with a Sales Total greater than 1,000,000. Use rounding or FORMAT to present the numbers.

The sales totals need to be calculated – this is done using summing the product of UnitPrice, Quantity and Discount in the order details table, whilst using the round function. The order details table need to be linked to the regions table, by 4 joins, where the order details is joined to the orders table, which is joined to the employee territories table, which is joined to the territories table, which is joined to the Regions table. Then the total sales need to be greater than a 1,000,000 using the having function, as it is an aggregate function.

Query is:

use Northwind

select ROUND(SUM((od.UnitPrice \* od.Quantity)\*(1-od.Discount)),1) as 'Sales Total', r.RegionDescription as 'Region'

from [Order Details] od

join Orders o on o.OrderID = od.OrderID

join EmployeeTerritories e on e.EmployeeID = o.EmployeeID

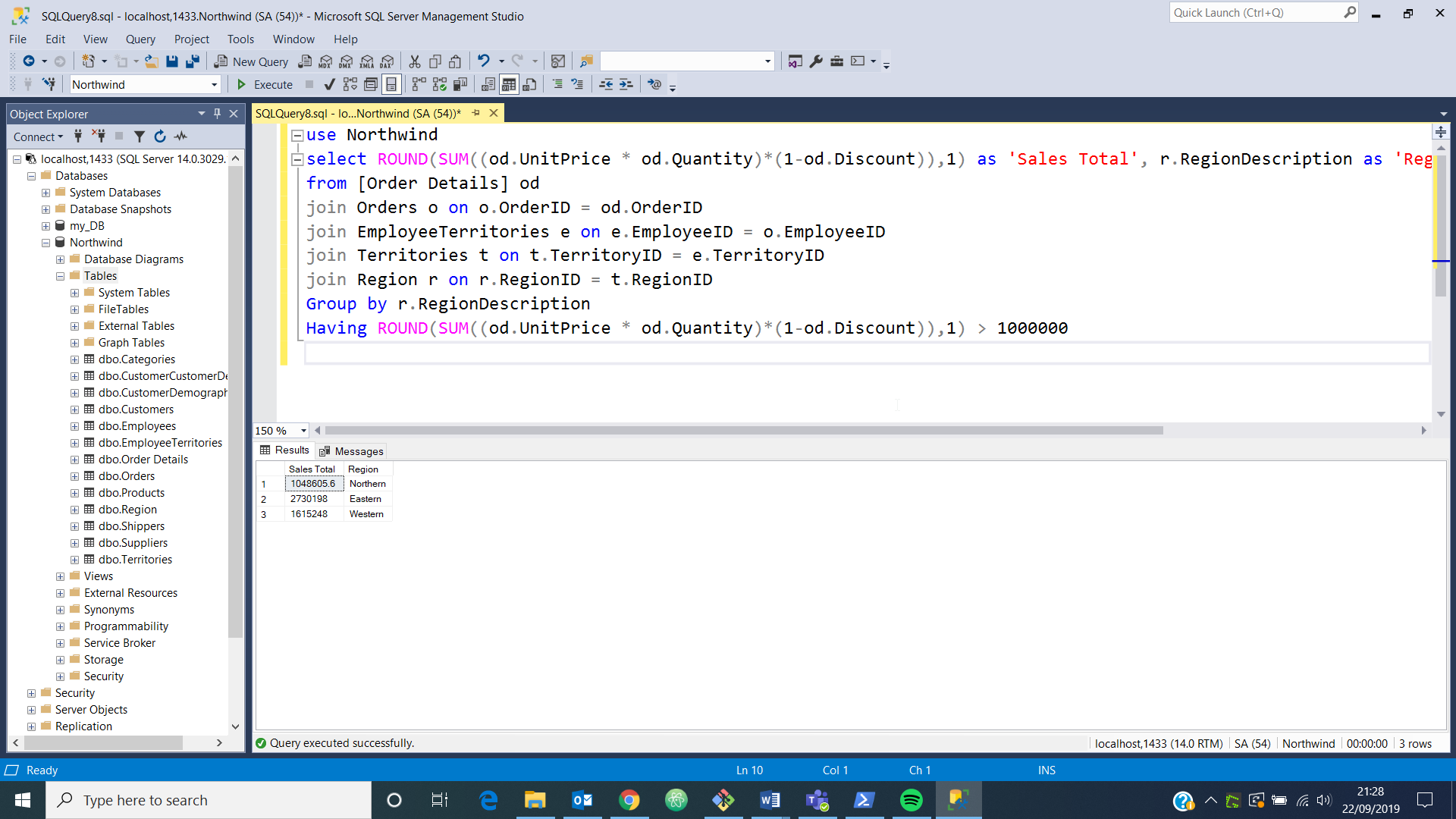
join Territories t on t.TerritoryID = e.TerritoryID

join Region r on r.RegionID = t.RegionID

Group by r.RegionDescription

Having ROUND(SUM((od.UnitPrice \* od.Quantity)\*(1-od.Discount)),1) > 1000000

The list of sales totals for all sales regions:



* 1. Count how many Orders have a Freight amount greater than 100.00 and either USA or UK as Ship Country.

The count function needs to be used to count how many freights there are. Conditions of the ship country being UK or USA as well as Freight being greater than 100.00 are added also.

The query for this is:

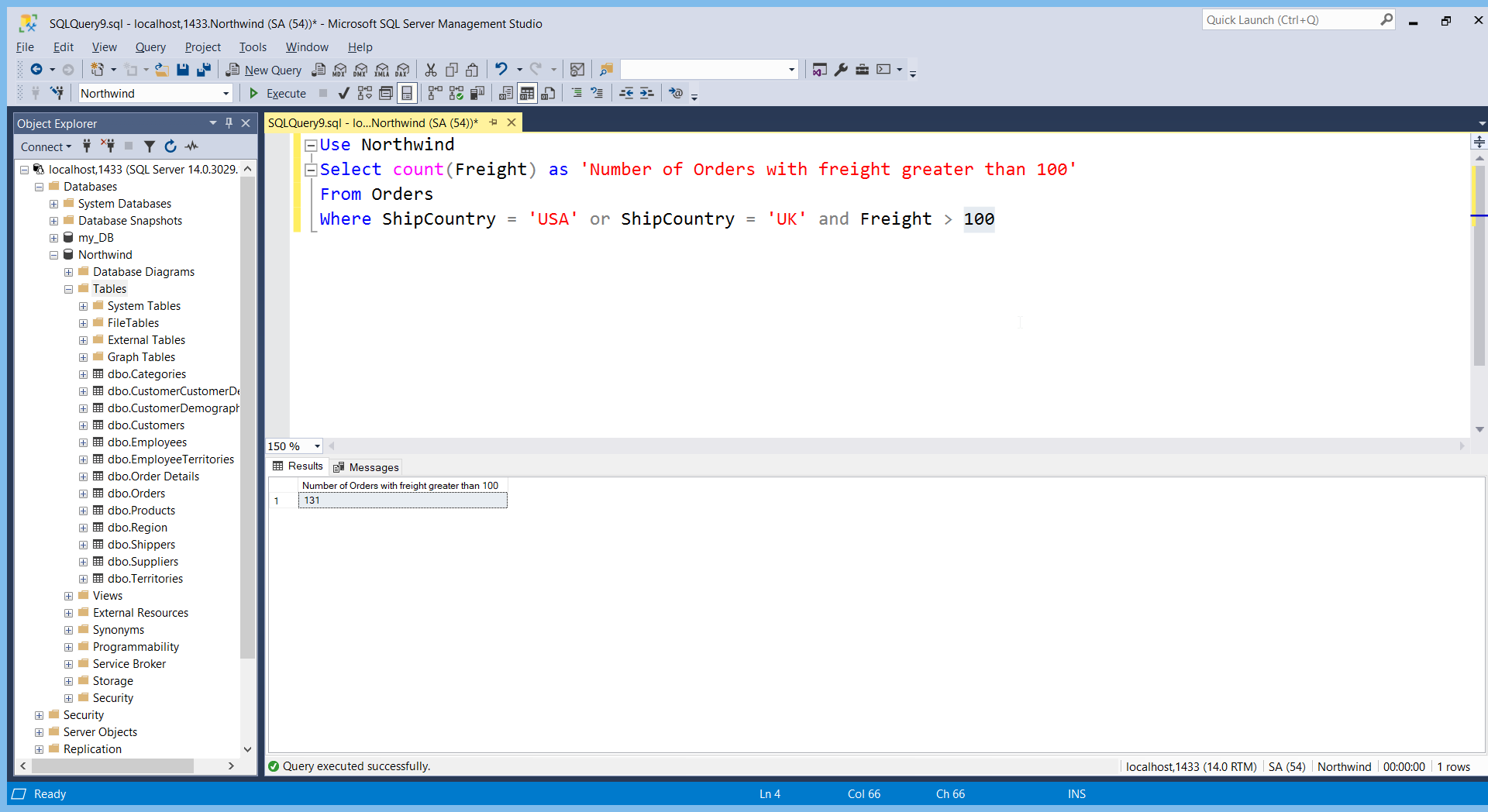
Use Northwind

Select count(Freight) as 'Number of Orders with freight greater than 100'

From Orders

Where ShipCountry = 'USA' or ShipCountry = 'UK' and Freight > 100

Result of this query is:



* 1. Write an SQL Statement to identify the Order Number of the Order with the highest amount of discount applied to that order.

The order number with the highest discount can be filtered from the orders details table using the top function. The SQL statement is:

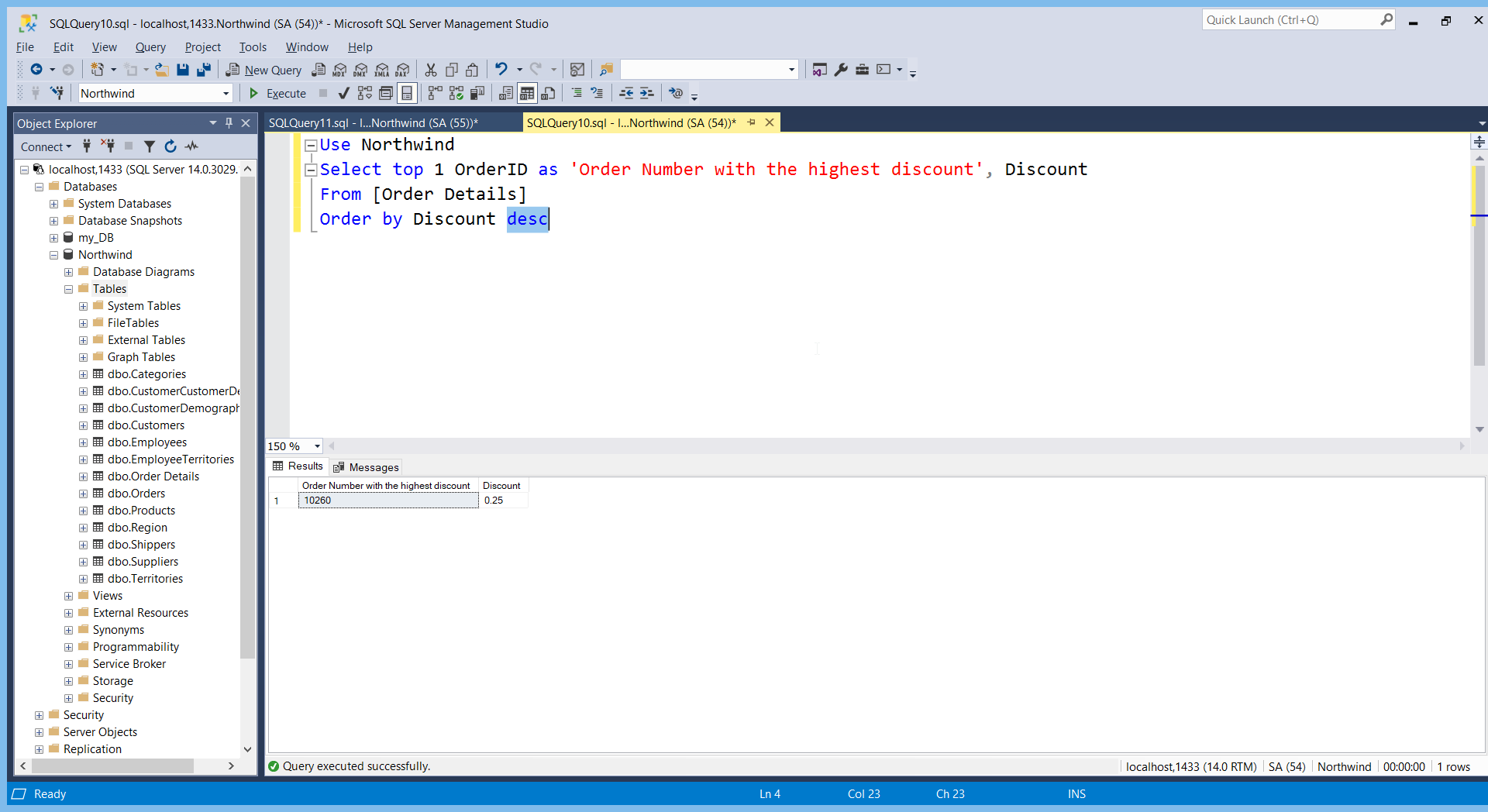
Use Northwind

Select top 1 OrderID as 'Order Number with the highest discount', Discount

From [Order Details]

Order by Discount desc

Result of the SQL statement:



Exercise 2 – Create Database Schema (20 marks)

2.1 Design and build a suitable set of fully normalised tables to store the following information:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Course Name** | **Start Date** | **End Date** | **Academy Name** | **Room Name** | **Trainer** | **Spartan** |
| BA-Test | 2018-01-15 | 2018-03-02 | Richmond | Room 1 | Tim Cawte | Adam Smith |
| BA-Test | 2018-01-15 | 2018-03-02 | Richmond | Room 1 | Tim Cawte | John Williams |
| BA-Test | 2018-01-15 | 2018-03-02 | Richmond | Room 1 | Tim Cawte | Nick Willis |
| BA-Test | 2018-01-15 | 2018-03-02 | Richmond | Room 1 | Tim Cawte | Jenny Jones |
| BA-Test | 2018-01-15 | 2018-03-02 | Richmond | Room 1 | Tim Cawte | Katie Prince |
| BA-Test | 2018-01-15 | 2018-03-02 | Richmond | Room 1 | Tim Cawte | Peter Brown |
| Engineering | 2018-01-22 | 2018-03-03 | Richmond | Room 3 | Richard Gurney | Mo Khan |
| Engineering | 2018-01-22 | 2018-03-03 | Richmond | Room 3 | Richard Gurney | Juan Carlos |
| Engineering | 2018-01-22 | 2018-03-03 | Richmond | Room 3 | Richard Gurney | Jan Miller |
| Engineering | 2018-01-22 | 2018-03-03 | Richmond | Room 3 | Richard Gurney | Kyle Carpenter |
| Engineering | 2018-01-22 | 2018-03-03 | Richmond | Room 3 | Richard Gurney | Alvarao Carlos |
| Engineering | 2018-01-22 | 2018-03-03 | Richmond | Room 3 | Richard Gurney | Margaret Baker |
| Engineering | 2018-01-22 | 2018-03-03 | Richmond | Room 3 | Richard Gurney | Oti Mwase |

This information can be split into 6 parts: information on Spartans, Trainers, Rooms, Academies, Courses and dates. This makes up 6 of the 7 normalised tables. The final table required is the table required to join the course information with the Spartans taking the course.

2.2 One that contains all statements required to create and re-create this database. DDL only.

This is the SQL statement required to build this database.

Create database [Sparta Academy DB]

Create Table Trainers

(

TrainerID INT NOT NULL identity primary key,

[Trainer Name] VARCHAR(80),

[Trainer Type] VARCHAR(24)

);

Create Table Courses

(

CourseID INT NOT NULL identity primary key,

[Course Name] VARCHAR(80),

);

Create Table Academies

(

AcademyID INT NOT NULL identity primary key,

[Academy Name] VARCHAR(80),

);

Create Table Rooms

(

RoomID INT NOT NULL identity primary key,

[Room Name] VARCHAR(80),

Capacity INT,

AcademyID INT foreign key references Academies(AcademyID)

);

Create Table Spartans

(

SpartanID INT NOT NULL identity primary key,

[Spartan Name] VARCHAR(80),

TrainerID INT foreign key references Trainers(TrainerID)

);

Create Table Dates

(

DateID INT NOT NULL identity primary key,

[Start date] date,

[End date] date,

CourseID INT foreign key references Courses(CourseID)

);

Create Table [Course Participants]

(

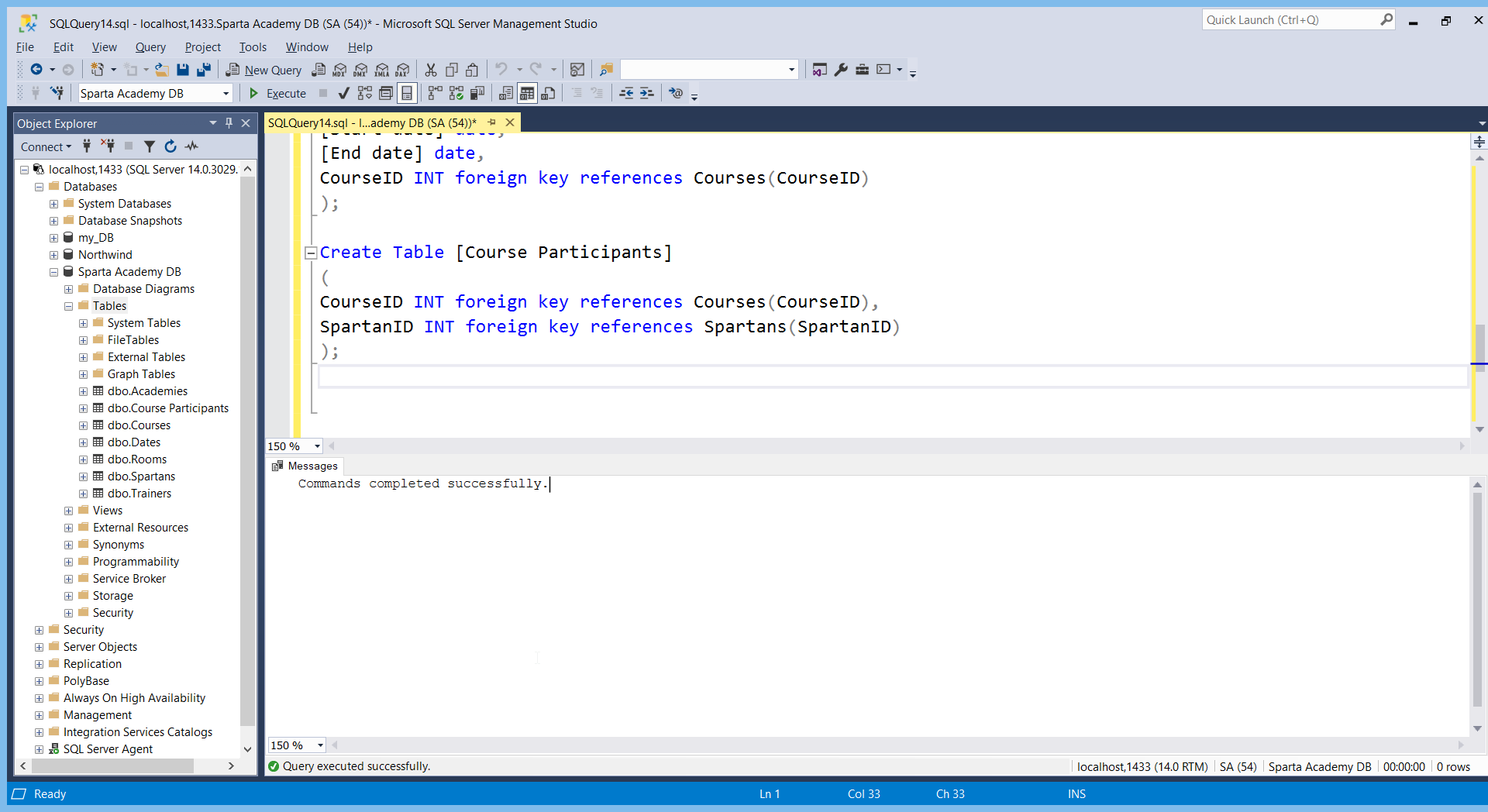
CourseID INT foreign key references Courses(CourseID),

SpartanID INT foreign key references Spartans(SpartanID),

AcademyID INT foreign key references Academies(AcademyID)

);

The result of these statements are that the tables are formed as shown in the screenshot:



2.3 Add more sample data to include all current trainers and at least one TA (Training Assistant) and Spartans on your current course.

All tables were filled with data. These are the SQL statements to show the input of data into these tables:

Use [Sparta Academy DB]

SET IDENTITY\_INSERT Academies ON

INSERT INTO Academies(AcademyID, [Academy Name])

VALUES (1, 'Richmond'), (2, 'London'), (3, 'Birmingham')

SET IDENTITY\_INSERT Academies OFF;

SET IDENTITY\_INSERT Trainers ON

INSERT INTO Trainers(TrainerID, [Trainer Name], [Trainer Type])

VALUES (1, 'Richard Gurney', 'Trainer'),

(2, 'Tim Cawte', 'Trainer'),

(3, 'Filipe Paiva', 'Trainer'),

(4, 'Samira Parkinson', 'Training Assistant')

SET IDENTITY\_INSERT Trainers OFF;

SET IDENTITY\_INSERT Courses ON

INSERT INTO Courses(CourseID, [Course Name])

VALUES (1, 'BA-Test'), (2, 'Engineering'), (3, 'DeveOps')

SET IDENTITY\_INSERT Courses OFF;

SET IDENTITY\_INSERT Rooms ON

INSERT INTO Rooms(RoomID, [Room Name], Capacity, AcademyID)

VALUES (1, 'Room 1', 20, 1), (2, 'Room 2', 20, 1), (3, 'Room 3', 20, 1)

SET IDENTITY\_INSERT Rooms OFF;

SET IDENTITY\_INSERT Dates ON

INSERT INTO Dates(DateID, [Start date], [End date], CourseID)

VALUES (1, '2018-01-15', '2018-03-02', 1), (2, '2018-01-22', '2018-03-03', 2), (3, '2019-09-09', '2019-11-15', 3)

SET IDENTITY\_INSERT Dates OFF;

SET IDENTITY\_INSERT Spartans ON

Insert into Spartans(SpartanID, [Spartan Name], TrainerID)

Values (1, 'Mo Khan', 1),

(2, 'Juan Carlos', 1),

(3, 'Jan Miller', 1),

(4, 'Kyle Carpenter', 1),

(5, 'Alvarao Carlos', 1),

(6, 'Margaret Baker', 1),

(7, 'Oti Mwase', 1),

(8, 'Adam Smith', 2),

(9, 'John Williams', 2),

(10, 'Nick Willis', 2),

(11, 'Jenny Jones', 2),

(12, 'Katie Prince', 2),

(13, 'Peter Brown', 2),

(14, 'Francis Thevipagan', 3),

(15, 'David Naim', 3),

(16, 'Miles Eastwood', 3),

(17, 'Rory Sokes', 3),

(18, 'Sharik Gurung', 3),

(19, 'Sam Forrester', 3),

(20, 'Vishnu Jeyarathnam', 3),

(21, 'Jack Wallace', 3),

(22, 'Moustapha Akanmu', 3)

SET IDENTITY\_INSERT Spartans OFF;

Use [Sparta Academy DB]

Insert into [Course Participants](CourseID, SpartanID, AcademyID)

Values (2, 1, 1), (2, 2, 1), (2, 3, 1), (2, 4, 1), (2, 5, 1), (2, 6, 1), (2, 7, 1), (1, 8, 1), (1, 9, 1), (1, 10, 1), (1, 11, 1), (1, 12, 1), (1, 13, 1), (3, 14, 2), (3, 15, 2), (3, 16, 2), (3, 17, 2), (3, 18, 2), (3, 19, 2), (3, 20, 2), (3, 21, 2), (3, 22, 2);

3.1 Produce a report similar to the above table (see 2.1) from all 7 tables using one SQL statement (use FORMAT for the dates). (10 marks)

To produce a report similar to the above table, all tables must be joined and the titles must be selected from the tables.

Use [Sparta Academy DB]

Select [Course Name], D.[Start date], D.[End date], A.[Academy Name], R.[Room Name],

T.[Trainer Name], S.[Spartan Name]

from Courses C

Join [Course Participants] CP on CP.CourseID = C.CourseID

Join Dates D on D.CourseID = C.CourseID

Join Spartans S on S.SpartanID = CP.SpartanID

Join Trainers T on T.TrainerID = S.TrainerID

Join Academies A on A.AcademyID = CP.AcademyID

Join Rooms R on R.AcademyID = A.AcademyID

3.2 Repeat 3.1 above but replace the Spartan Name column with Spartan Initials.

The Sparta Name is changed to Spartan Initials by replacing the alias accordingly. The left function is used to show the first letter of the name.

Use [Sparta Academy DB]

Select [Course Name], D.[Start date], D.[End date], A.[Academy Name], R.[Room Name],

T.[Trainer Name], left(S.[Spartan Name],1) as ‘Spartan Initials’

from Courses C

Join [Course Participants] CP on CP.CourseID = C.CourseID

Join Dates D on D.CourseID = C.CourseID

Join Spartans S on S.SpartanID = CP.SpartanID

Join Trainers T on T.TrainerID = S.TrainerID

Join Academies A on A.AcademyID = CP.AcademyID

Join Rooms R on R.AcademyID = A.AcademyID

3.3 Add a new column after End Date headed “Check Date” which uses the DATEADD function to add 8 weeks to the Start Date column for BA Test courses and 12 weeks for any others. (5 marks)

A new column needs to be added to the Dates table entitled ‘Check Date’, where a dateadd function of 2 months is added on the condition that the course ID is BA-Testing (= 1) and 3 months when it is not (<>1).

Here is the query for this:

Use [Sparta Academy DB];

Alter Table Dates

Add [Check Date] Date;

Update Dates

Set [Check Date] = Dateadd(mm, 2, [Start Date])

Where CourseID = 1

Update Dates

Set [Check Date] = Dateadd(mm, 3, [Start Date])

Where CourseID <> 1;

The “Check Date” column needs to be included after “End Date” in the final report so the query should look like this:

Select [Course Name], D.[Start date], D.[End date], D.[Check Date], A.[Academy Name], R.[Room Name],

T.[Trainer Name], S.[Spartan Name]

from Courses C

Join [Course Participants] CP on CP.CourseID = C.CourseID

Join Dates D on D.CourseID = C.CourseID

Join Spartans S on S.SpartanID = CP.SpartanID

Join Trainers T on T.TrainerID = S.TrainerID

Join Academies A on A.AcademyID = CP.AcademyID

Join Rooms R on R.AcademyID = A.AcademyID

Exercise 4 – Add Constraints (10 marks)

4.1 Add Primary Keys and Foreign Keys to the Sparta Academy database, where needed. (If not already included). (8 marks)

Primary and foreign keys included already.

4.2 Add constraints for other tables including Trainer Type ('T' Trainer or 'A' Training Assistant) and Employee Type ('T' Trainer, 'S' Spartan) and set a maximum capacity for Rooms to 25. (2 marks)

As the column “Trainer types” is present in the Trainers table and the column “Capacity” is present in the Rooms table, the constraints mentioned will added to these tables.

Alter Table Trainers

Add Constraint [Trainer Type] Check ([Trainer Type] = 'T' OR [Trainer Type] = 'A')

Alter Table Rooms

Add Constraint Capacity Check (Capacity <= 25)