Subqueries

CMPE 2400

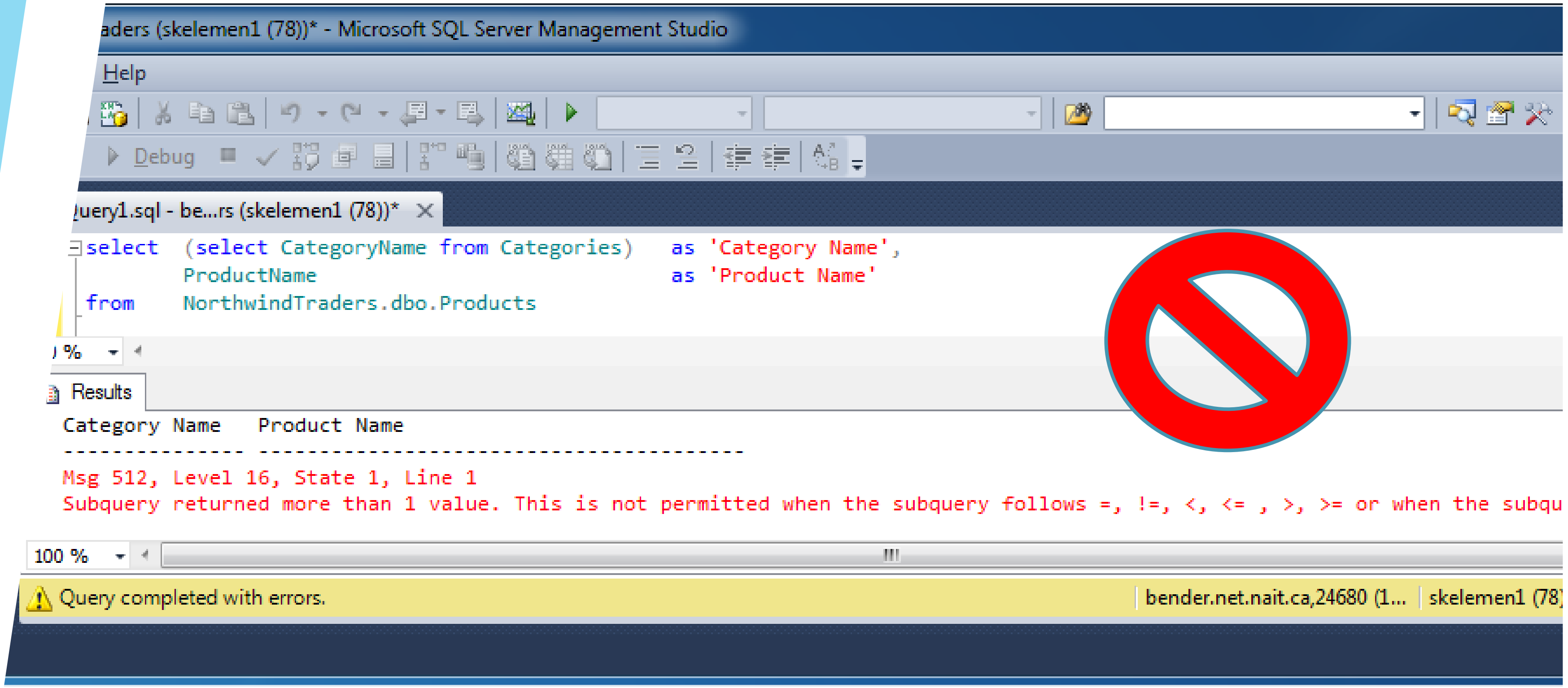
Databases

Subqueries

- A subquery is a query that is nested within a parent query.
- We generally use a subquery when we want to pull most data from one table and one item of data from a different table.
- Suppose we want the product Id, the product name, the price and the category name for a particular category of product.
  - While most of the required information is available from the product table, the category name has to be pulled from the category table

Subqueries

- The key to writing a proper subquery is to remember what is expected by the parent query in the spot in which the subquery is placed.
  - For example, if a subquery is to be included in a select list, the subquery must produce exactly one result, or the parent query will cease up.
  - Also make sure to remember that data types can also be mismatched even if the number of records returned by the subquery is acceptable.

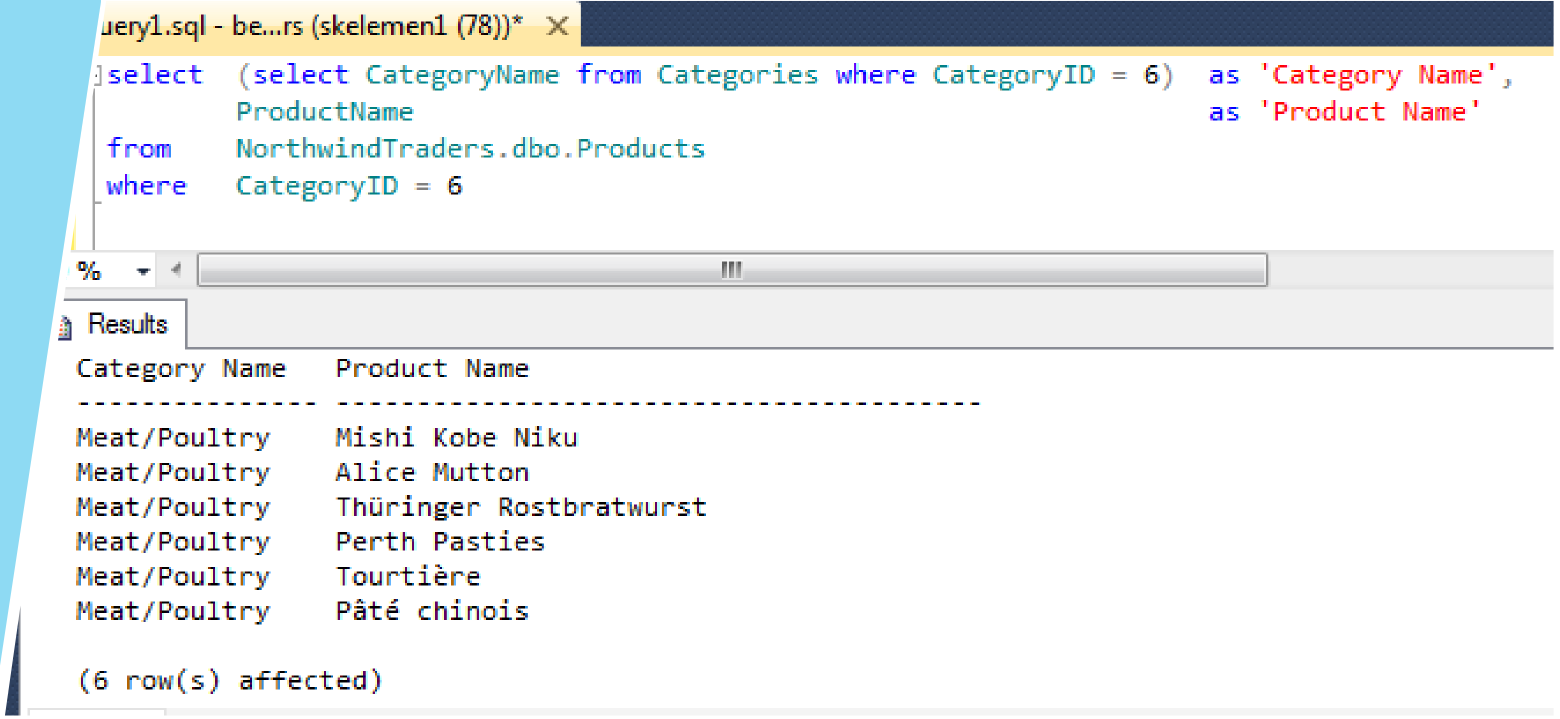

aders (skelemen1 (78))* - Microsoft SQL Server Management Studio
Help
Debug
uery1.sql - be...rs (skelemen1 (78))*
Results
100 %
Query completed with errors.
bender.net.nait.ca,24680 (1...
skelemen1 (78)

```
select  (select CategoryName from Categories)   as 'Category Name',
        ProductName                             as 'Product Name'
from    NorthwindTraders.dbo.Products
```

```
Category Name   Product Name
--------------- ----------------------------------------
Msg 512, Level 16, State 1, Line 1
Subquery returned more than 1 value. This is not permitted when the subquery follows =, !=, <, <= , >, >= or when the subqu
```

Subqueries

Busted on number of records mismatch.



```
select  (select CategoryName from Categories where CategoryID = 6)  as 'Category Name',
        ProductName                                                 as 'Product Name'
from    NorthwindTraders.dbo.Products
where   CategoryID = 6
```

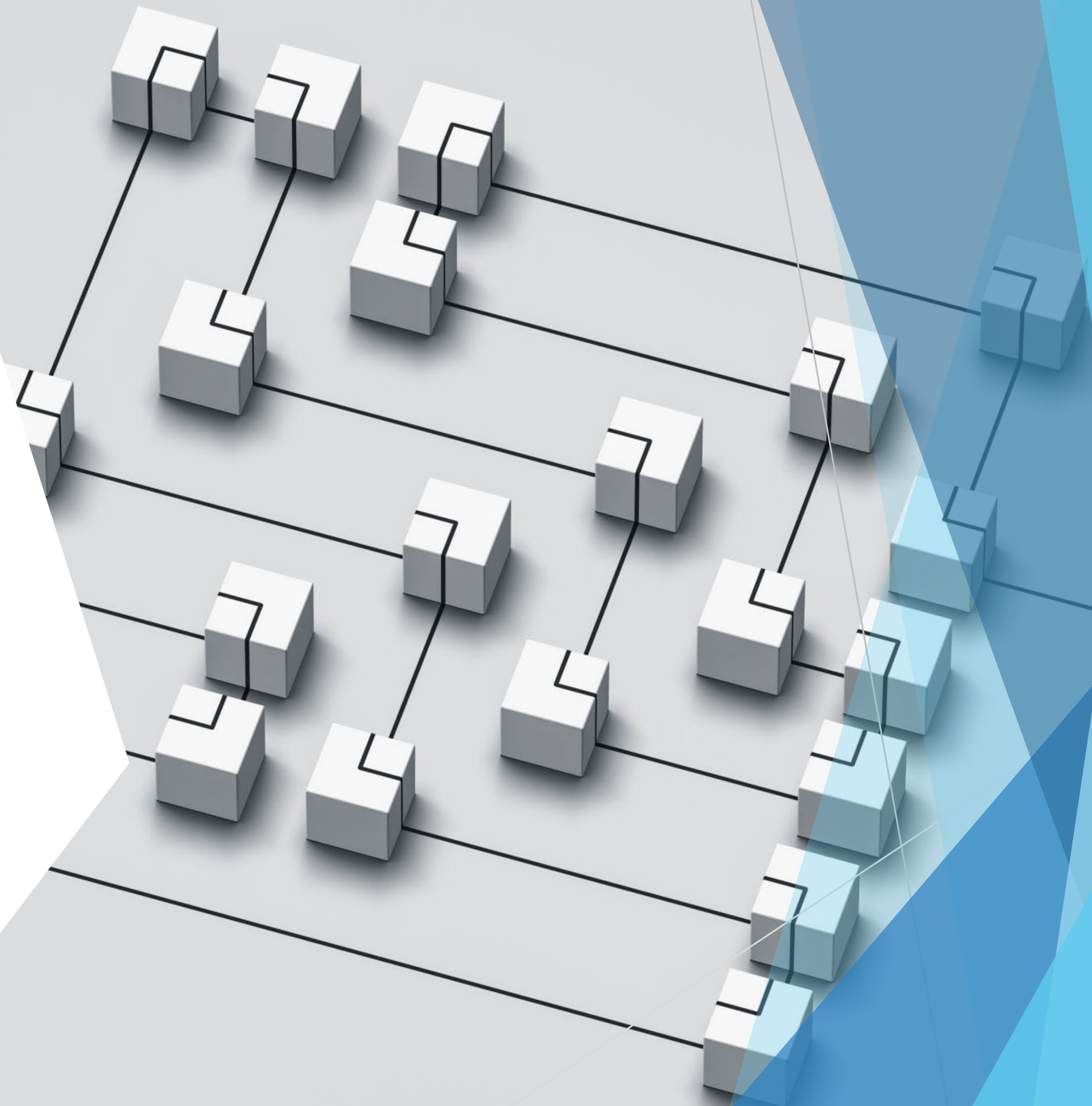
```
Category Name   Product Name
--------------- ----------------------------------------
Meat/Poultry    Mishi Kobe Niku
Meat/Poultry    Alice Mutton
Meat/Poultry    Thüringer Rostbratwurst
Meat/Poultry    Perth Pasties
Meat/Poultry    Tourtière
Meat/Poultry    Pâté chinois

(6 row(s) affected)
```

Subqueries

This one is contrived but at least it works…

Filtering with Subqueries

- Using a subquery is one way in which two entities in the database may interact.
- The results of a query which interrogates one entity, or table, may be used to filter the results obtained by querying a second entity from which we are actually interested in retrieving information.



Filtering using in Subqueries

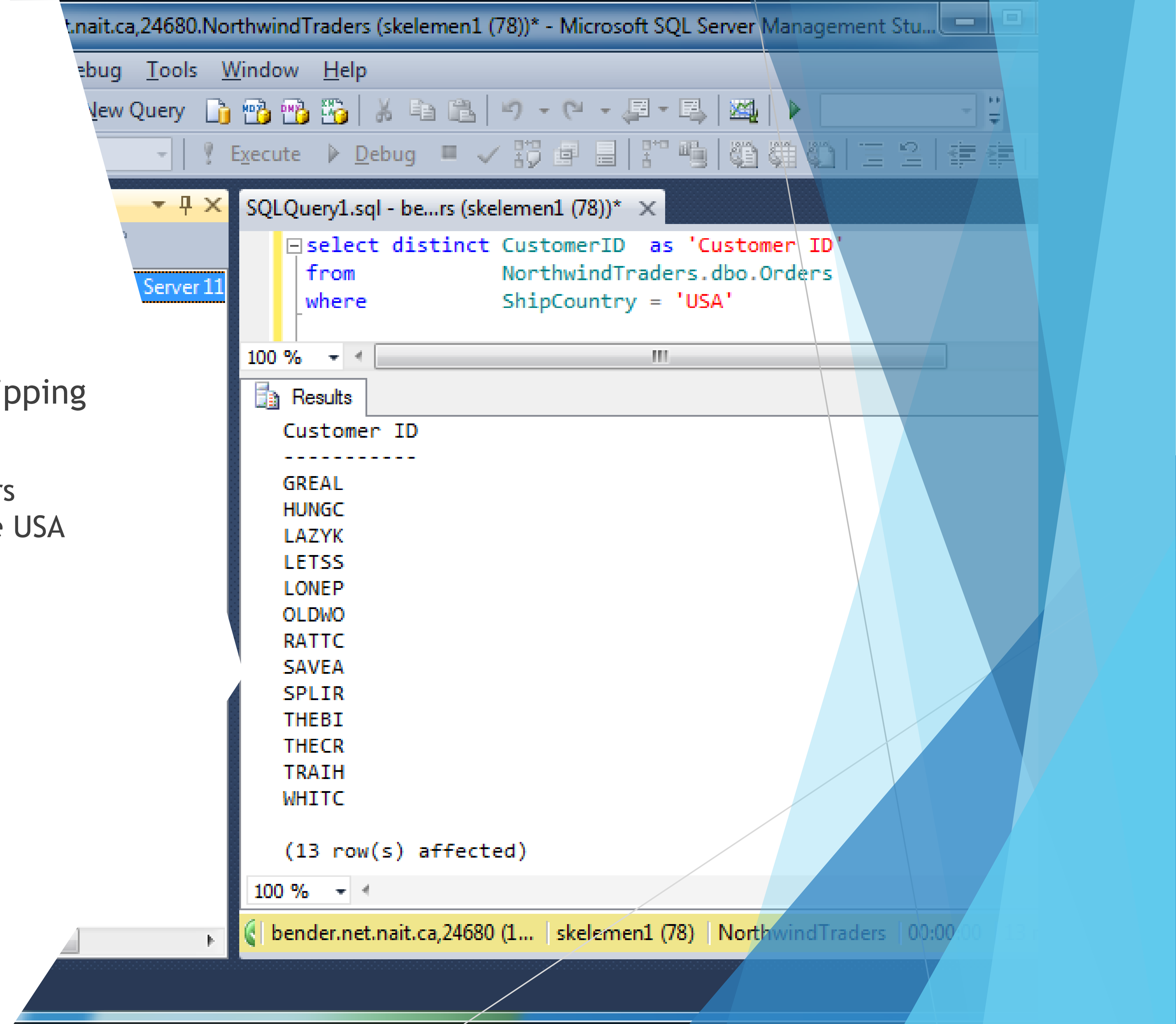
- The **in** operator is often used to create the subquery bridge between entities.
  - Recall that the **in** operator is used when creating filters, using the where clause, that compare a set of discrete values to a column or calculation.
  - A subquery returns a set of discrete values just like any other select statement.
  - Because the values retrieved by the subquery are discrete, the in operator is quite happy to use the returned results to filter the rows returned from the parent select statement.

Filtering using in Subqueries

- As the in operator works upon a *LIST* of values, not a multiple columned table of values, only one column may be chosen by the subquery to form that list.

Filtering using in Subqueries

- Suppose that you need a list of company names that have a shipping address within the USA.
  - First, a unique list of customers linked to orders shipped to the USA must be identified.



.nait.ca,24680.NorthwindTraders (skelemen1 (78))* - Microsoft SQL Server Management Stu...
ebug Tools Window Help
New Query
Execute
Debug
Server 11
SQLQuery1.sql - be...rs (skelemen1 (78))*
100 %
Results
bender.net.nait.ca,24680 (1...
skelemen1 (78)
NorthwindTraders
00:00:00

```
select distinct CustomerID  as 'Customer ID'
from            NorthwindTraders.dbo.Orders
where           ShipCountry = 'USA'
```

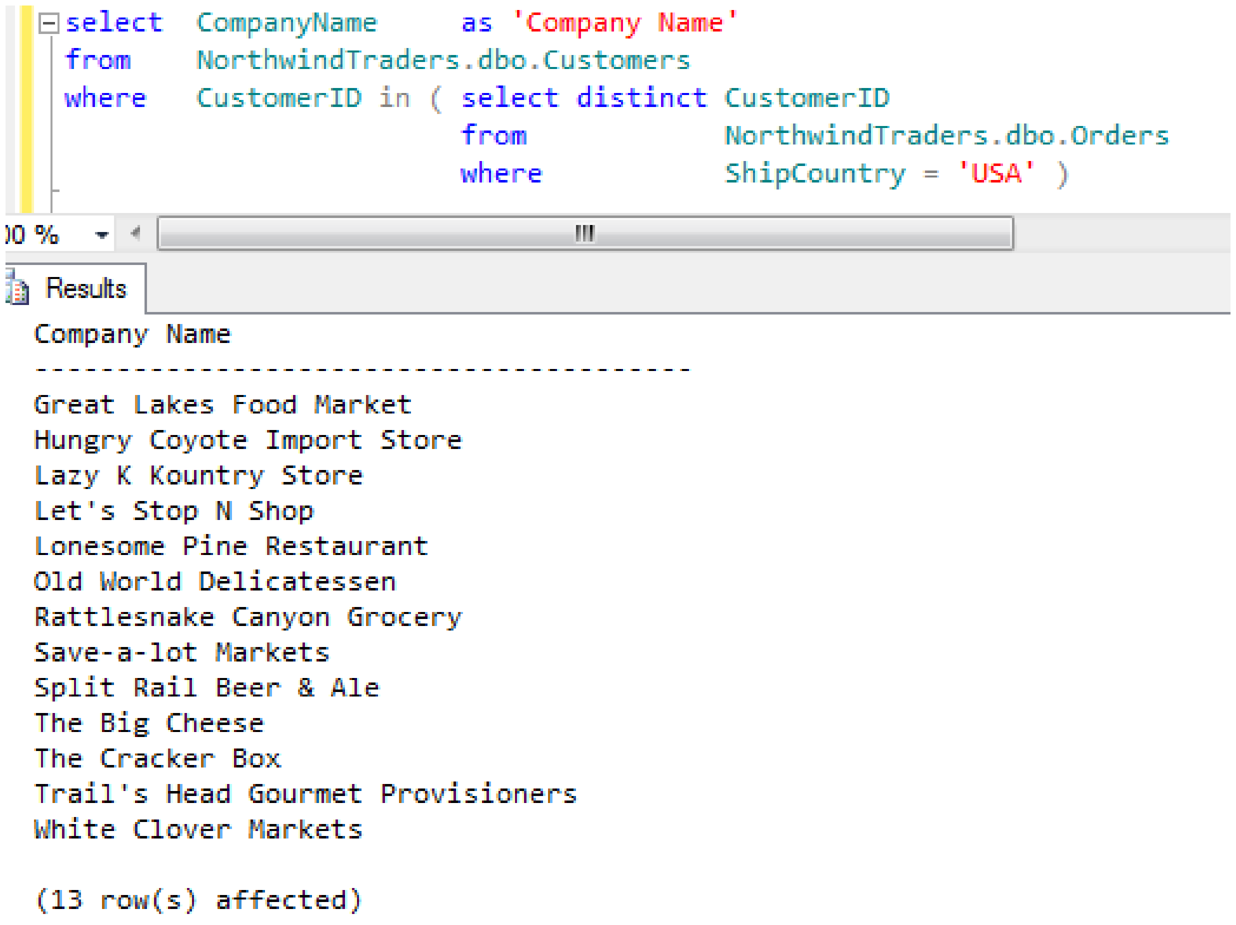
```
Customer ID
-----------
GREAL
HUNGC
LAZYK
LETSS
LONEP
OLDWO
RATTC
SAVEA
SPLIR
THEBI
THECR
TRAIH
WHITC

(13 row(s) affected)
```

Filtering using in Subqueries

- The Company Name is stored in the customer table, so we may use the in operator, and embed the previous as a subquery.
  - Note that, quite conveniently, the CustomerID provides a link between the two tables.
  - Also, column aliasing will only be required on the outermost query.

Filtering using in Subqueries

- Voila! Company names for orders shipped to the USA.



```
select  CompanyName      as 'Company Name'
from    NorthwindTraders.dbo.Customers
where   CustomerID in ( select distinct CustomerID
                        from            NorthwindTraders.dbo.Orders
                        where           ShipCountry = 'USA' )
```

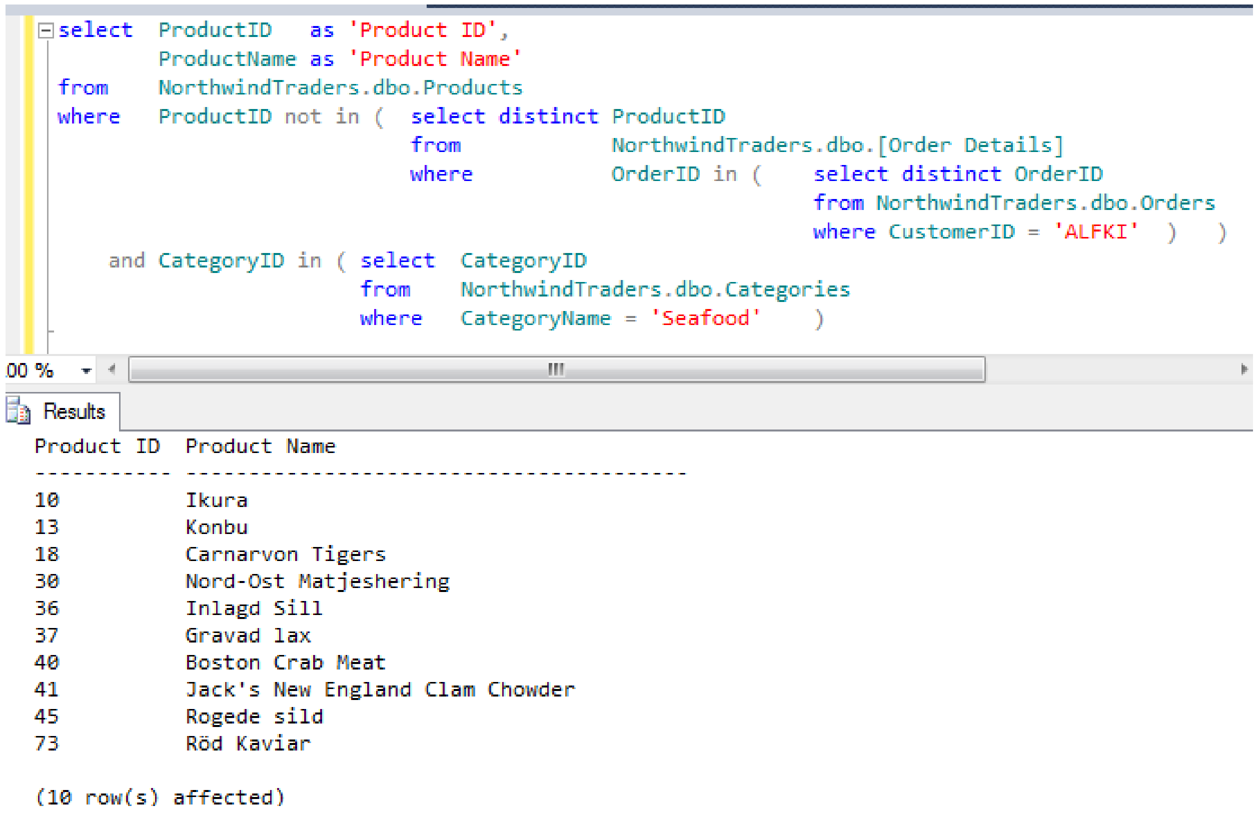
```
Company Name
----------------------------------------
Great Lakes Food Market
Hungry Coyote Import Store
Lazy K Kountry Store
Let's Stop N Shop
Lonesome Pine Restaurant
Old World Delicatessen
Rattlesnake Canyon Grocery
Save-a-lot Markets
Split Rail Beer & Ale
The Big Cheese
The Cracker Box
Trail's Head Gourmet Provisioners
White Clover Markets

(13 row(s) affected)
```

Filtering using in Subqueries

- Just to show a glimpse of the possibilities, realize:
  - Double nesting, or more, is perfectly legal.
  - Negative logic works nicely.
  - Combinational logic will still work just fine.
    - Remember that the in operator is merely a comparison operator in the where clause.
    - Eg. Make a list of meat products that have *never* been sold to the company with the CustomerID *ALFKI*

Filtering using in Subqueries

- Retrieve a list of seafood products that have *never* been sold to the company with a customer ID of ALFKI.



```
select  ProductID   as 'Product ID',
        ProductName as 'Product Name'
from    NorthwindTraders.dbo.Products
where   ProductID not in (  select distinct ProductID
                            from            NorthwindTraders.dbo.[Order Details]
                            where           OrderID in (    select distinct OrderID
                                                            from NorthwindTraders.dbo.Orders
                                                            where CustomerID = 'ALFKI'  )   )
    and CategoryID in ( select  CategoryID
                        from    NorthwindTraders.dbo.Categories
                        where   CategoryName = 'Seafood'    )
```

```
Product ID  Product Name
----------- ----------------------------------------
10          Ikura
13          Konbu
18          Carnarvon Tigers
30          Nord-Ost Matjeshering
36          Inlagd Sill
37          Gravad lax
40          Boston Crab Meat
41          Jack's New England Clam Chowder
45          Rogede sild
73          Röd Kaviar

(10 row(s) affected)
```

Exercise 1

- Write a query to display the first name and last name of all authors from the Publishers database who write business books

Filtering using exists Subqueries

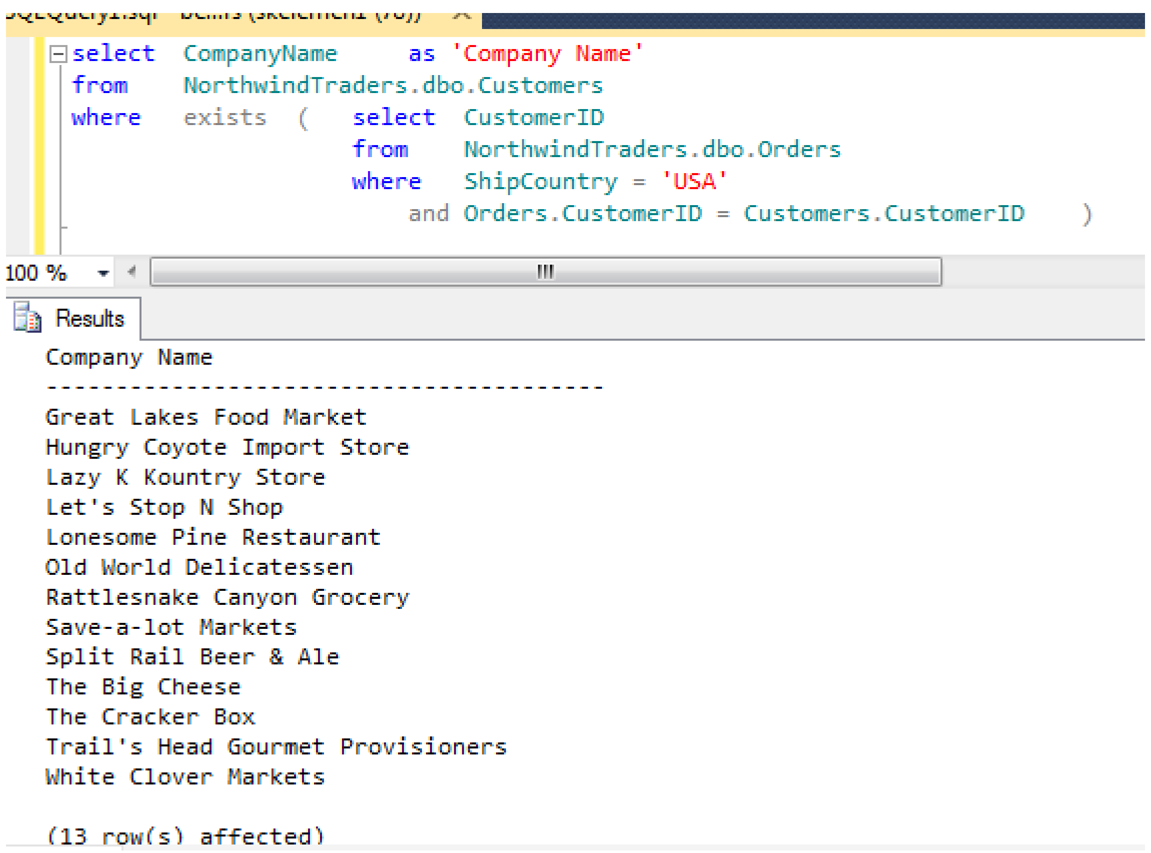
- The **exists** operator is a Boolean operator that tests for the existence of at least one record from the contained subquery.
- **exists** is sometimes more efficient to use than **in**
  - Consider the operation of exists to be similar to a nested for loop in C#. Each parent query table record is sequentially compared to the child query table records, with a true returned on the first match.

Filtering using exists Subqueries

- In the following example, the information from the *parent* query is used in the *subquery*
  - The subquery results are based on the information from the current row of the parent query
- The *child* select can see the *parent* select information
- The parent select can only see the results of the child select, but not the individual column values
  - You may look upon this behaviour as somewhat akin to scoping in C#. Any variables created in a scope are local to that scope and are invisible to any encapsulating scopes.

Filtering using exists Subqueries

- Provide a list of company names in the USA who have made orders

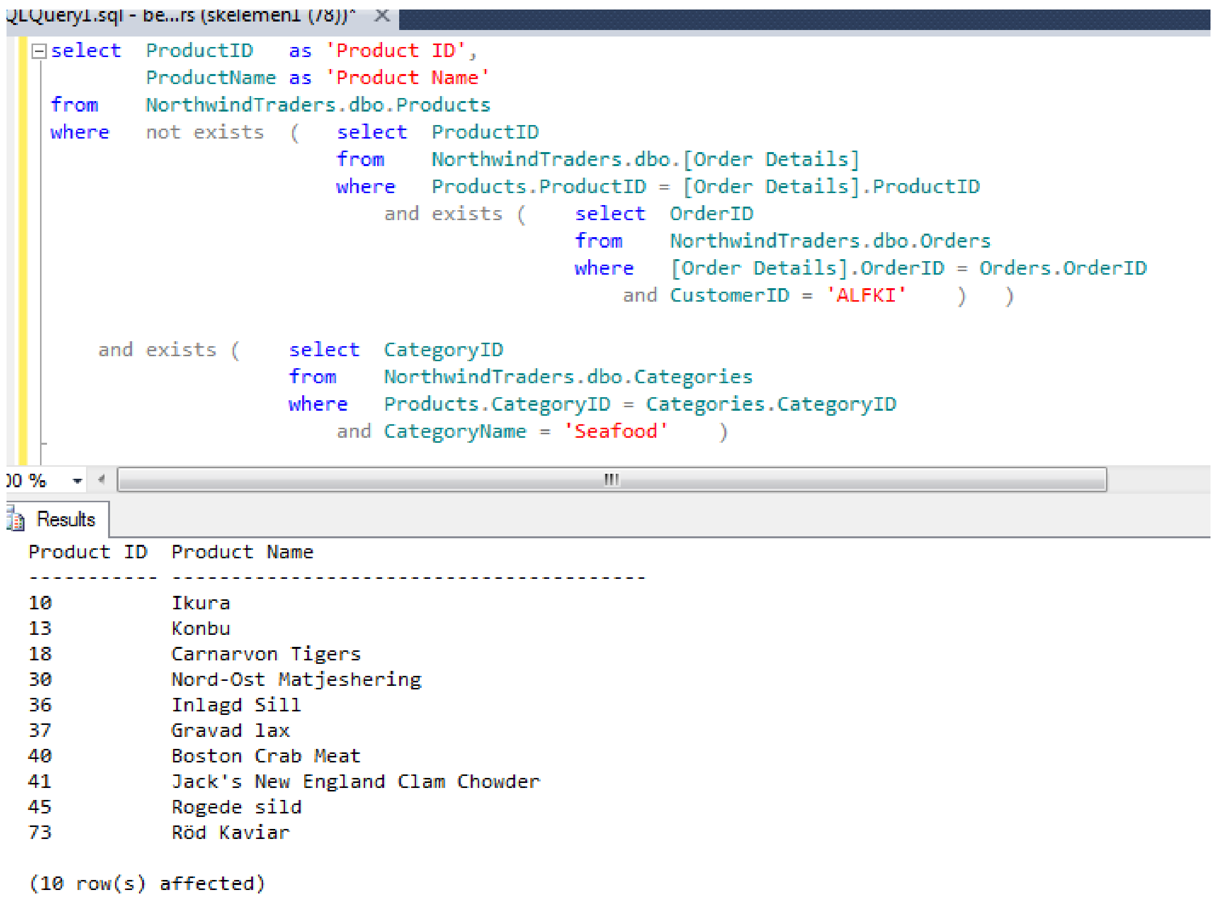


```
select  CompanyName     as 'Company Name'
from    NorthwindTraders.dbo.Customers
where   exists  (   select  CustomerID
                    from    NorthwindTraders.dbo.Orders
                    where   ShipCountry = 'USA'
                        and Orders.CustomerID = Customers.CustomerID    )
```

```
Company Name
----------------------------------------
Great Lakes Food Market
Hungry Coyote Import Store
Lazy K Kountry Store
Let's Stop N Shop
Lonesome Pine Restaurant
Old World Delicatessen
Rattlesnake Canyon Grocery
Save-a-lot Markets
Split Rail Beer & Ale
The Big Cheese
The Cracker Box
Trail's Head Gourmet Provisioners
White Clover Markets

(13 row(s) affected)
```

Filtering using exists Subqueries

- Retrieve a list of seafood products that have *never* been sold to the company with a customer ID of ALFKI



```
select  ProductID   as 'Product ID',
        ProductName as 'Product Name'
from    NorthwindTraders.dbo.Products
where   not exists (    select  ProductID
                        from    NorthwindTraders.dbo.[Order Details]
                        where   Products.ProductID = [Order Details].ProductID
                            and exists (    select  OrderID
                                            from    NorthwindTraders.dbo.Orders
                                            where   [Order Details].OrderID = Orders.OrderID
                                                and CustomerID = 'ALFKI'    )   )

    and exists (    select  CategoryID
                    from    NorthwindTraders.dbo.Categories
                    where   Products.CategoryID = Categories.CategoryID
                        and CategoryName = 'Seafood'    )
```

```
Product ID  Product Name
----------- ----------------------------------------
10          Ikura
13          Konbu
18          Carnarvon Tigers
30          Nord-Ost Matjeshering
36          Inlagd Sill
37          Gravad lax
40          Boston Crab Meat
41          Jack's New England Clam Chowder
45          Rogede sild
73          Röd Kaviar

(10 row(s) affected)
```

Performance Consideration: IN vs EXISTS

The EXISTS clause is much faster than IN when the subquery results is very large. Conversely, the IN clause is faster than EXISTS when the subquery results is very small.

Also, the IN clause can't compare anything with NULL values, but the EXISTS clause can compare everything with NULLs.

Reference: Burleson Consulting

Exercise 2

- Write a query using **exists** subquery to display the first name and last name of all authors who have written a business book