CS1100: Data, Databases, and Queries

CREATING COMPLEX QUERIES WITH NESTED QUERIES

Nested Queries

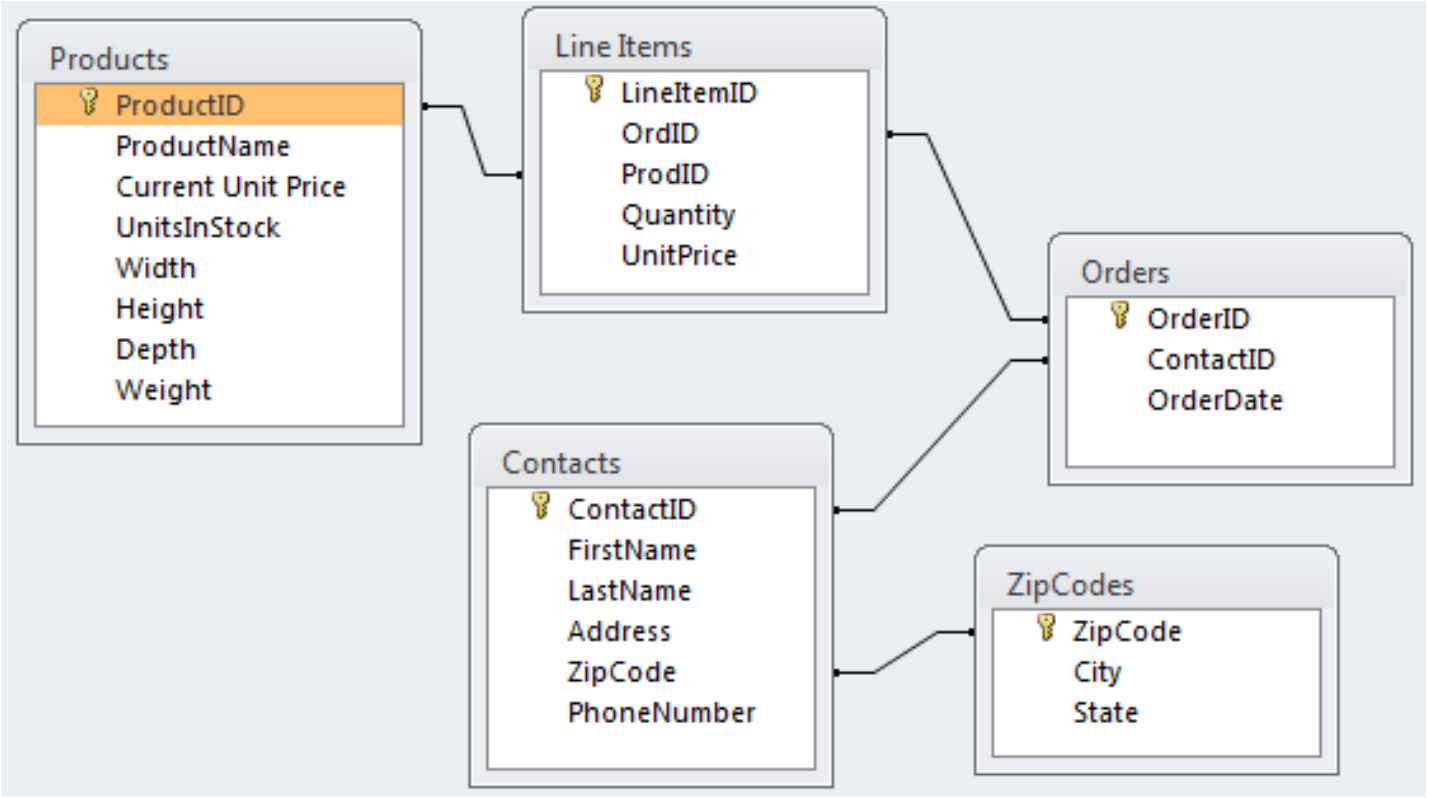
- Some complex queries must be based on the results of other queries.
- The result of a query is a *virtual* table, *i.e.*, something that looks and can be used like a table, but is not actually stored in the database as a table.
- A query can be used (as a table is used) in the design of a query.

Subqueries

- A subquery is a query statement that's nested inside of another query
- Sometimes need to use the results of a query as a field in another query, or as a criterion for a query field. Example:
  - How many orders have a total under $2,000?
  - To find the answer, first need to calculate order totals and filter out those that are $2,000 and over. Then do a count.

The Database Layout

- These are all of the tables in the database:



Where Does The Data Come From?

Order ***O0001*** ← Orders.OrderID

Customer Contact (Contacts)

Contact ID: C0004
Name: Colon, Nicholas
Address: 9020 N.W. 75 Street
Coral Springs, FL 33065 ← ZipCodes

Order Date: 4/15/1999 ← Orders.OrderDate

ExtendedPrice = Quantity * UnitPrice

Product ID	*Product Name*	*Quantity*	*UnitPrice*	*ExtendedPrice*
P0013	DVD Disks	1	$ 23.00	$ 23.00
P0014	HD Floppy Disks	4	$ 9.99	$ 39.96
P0027	Norton Anti-Virus	1	$ 115.95	$ 115.95

LineItems

Order Total: *$ 178.91*

Total Order Amount

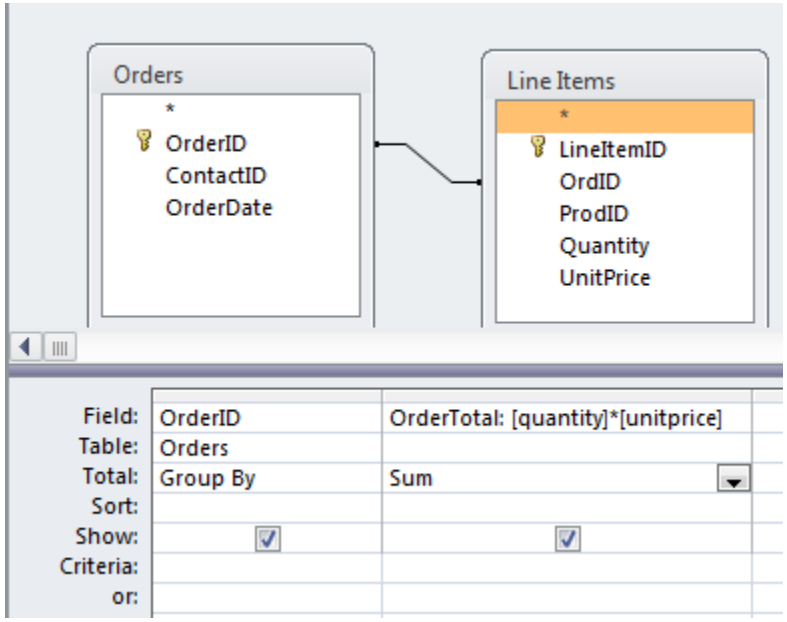
Example:

- How many orders were placed that had a total of less than $2000?

Step One: Find a total for each order

- What is the total for each order?

What is the total for each order?

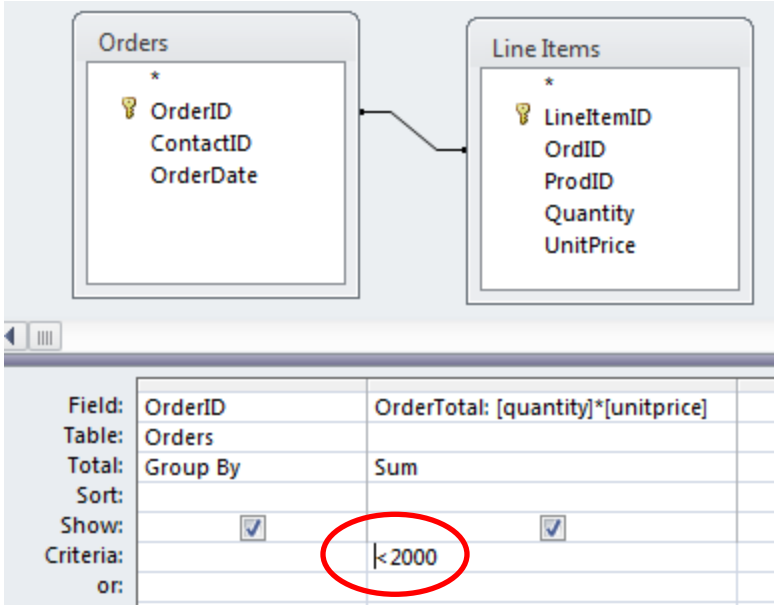


OrderID	OrderTotal
O0001	$178.91
O0002	$3,982.95
O0003	$4,183.95
O0004	$5,688.00
O0005	$5,055.90
O0006	$998.90
O0007	$209.80
O0008	$17,631.00
O0009	$499.00
O0010	$5,642.95
O0011	$739.80
O0012	$2,558.50
O0013	$4,535.40

Filter the order totals

- Which orders are for less than $2,000?

Which orders are for less than $2,000?



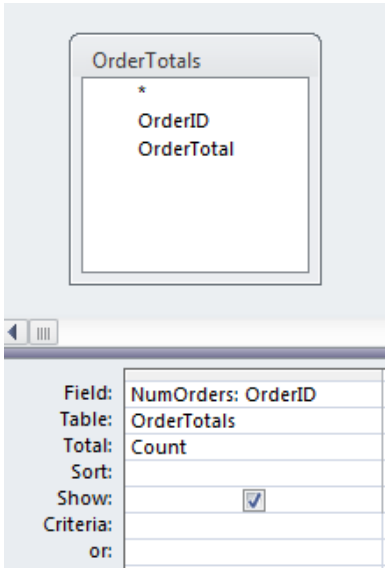
OrderID	OrderTotal
O0001	$178.91
O0006	$998.90
O0007	$209.80
O0009	$499.00
O0011	$739.80
O0014	$109.85
O0015	$249.90
O0016	$259.90
O0017	$79.90
O0018	$742.80
O0019	$249.75

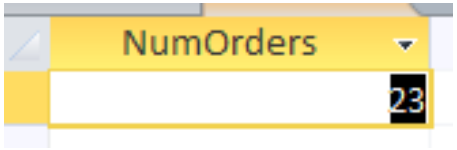
Step Two: Use the previous query as a subquery

- How many of these orders (orders less than $2,000) are there?

How many of these orders (orders less than $2,000) are there?

– This requires that we build a new query with the previous query as a subquery



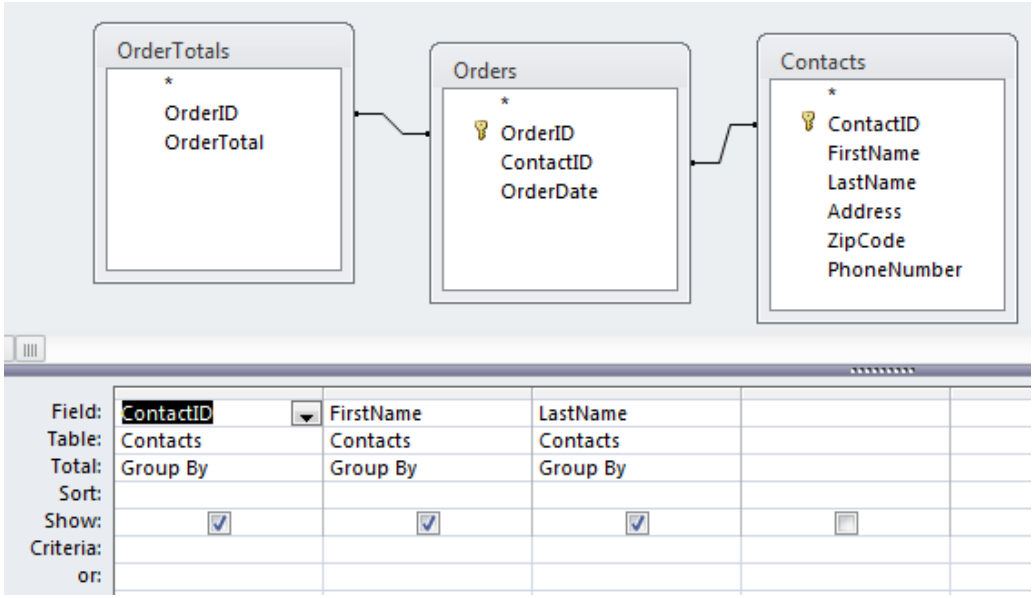


Query

- Who placed these orders less than $2,000?

Query

- Who placed these orders less than $2,000?
  - Use **OrdersTotal** again as a subquery
  - Combine with **Orders** and **Contacts** to get customer information
  - Use *Group By* to remove duplicates



ContactID	FirstName	LastName
1	Benjamin	Lee
2	Eleanor	Milgrom
4	Nicholas	Colon
6	Jeffrey	Muddell
8	Serena	Sherard
9	Luis	Couto
10	Derek	Anderson

There are 15 contacts in total

Watch Out: Access Caches Queries

- Whenever Access executes a query it saves ("caches") the result.
- When that query is used as a subquery, Access uses the cached result instead of running it again.
- If you then update the subquery, Access does not automatically refresh the result for the query that uses the subquery.

Refreshing Queries

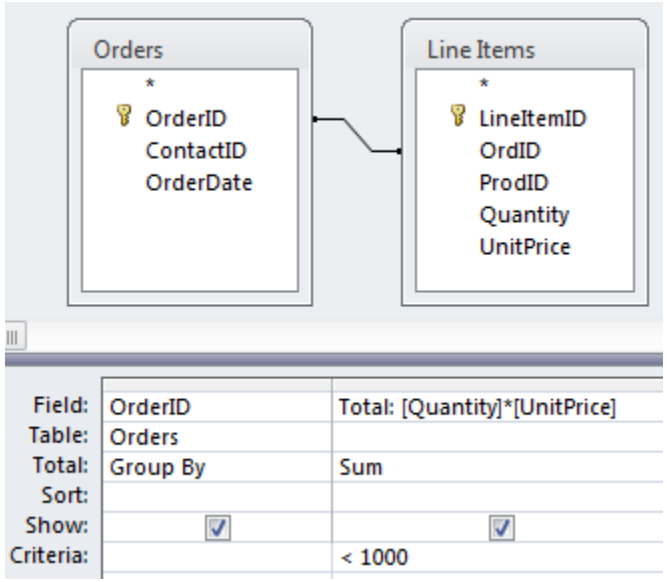
- To refresh all queries, you need to:
  - Close all queries
    - Click the right mouse button on the query tab and select “Close”
  - Load the query again
    - double-click on the query in the navigator
- This forces a refresh.

Step by Step

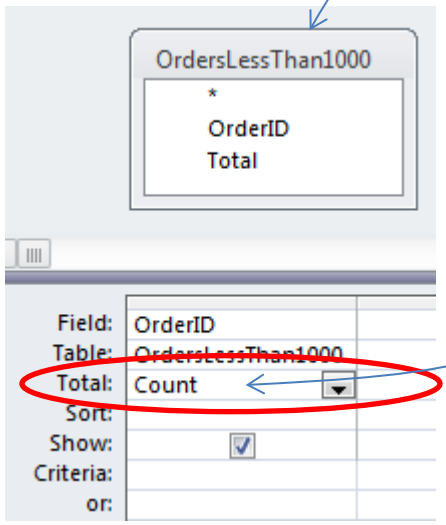
- How many orders are there that have a total value of less than $1000?
  - Create a query that finds all orders with a value less than $1000
  - Save the query under an appropriate name
  - Create another query based on the previous query that COUNTs all of the rows
- **CAUTION**: Do not modify or rename queries that are used as subqueries.

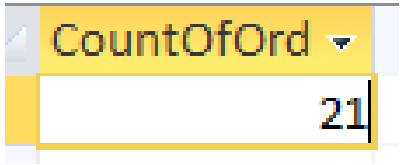
Example: Subquery

Subquery: OrdersLessThan1000



Note the use of the subquery and the aggregate *COUNT* function



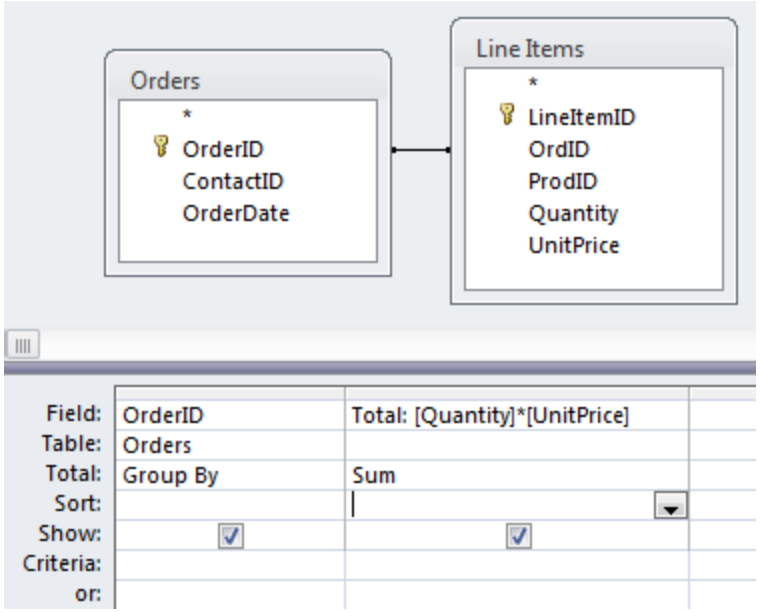


So there are 21 orders with a total less than $1000

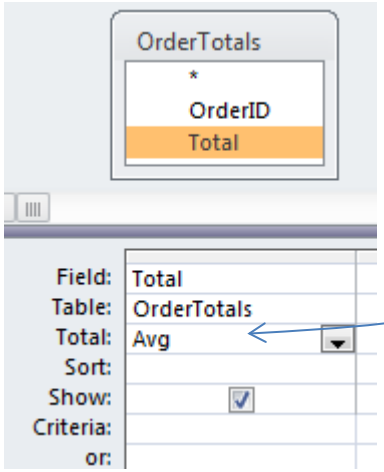
Example: AVG

- What is the average cost of all orders?

Subquery: OrderTotals

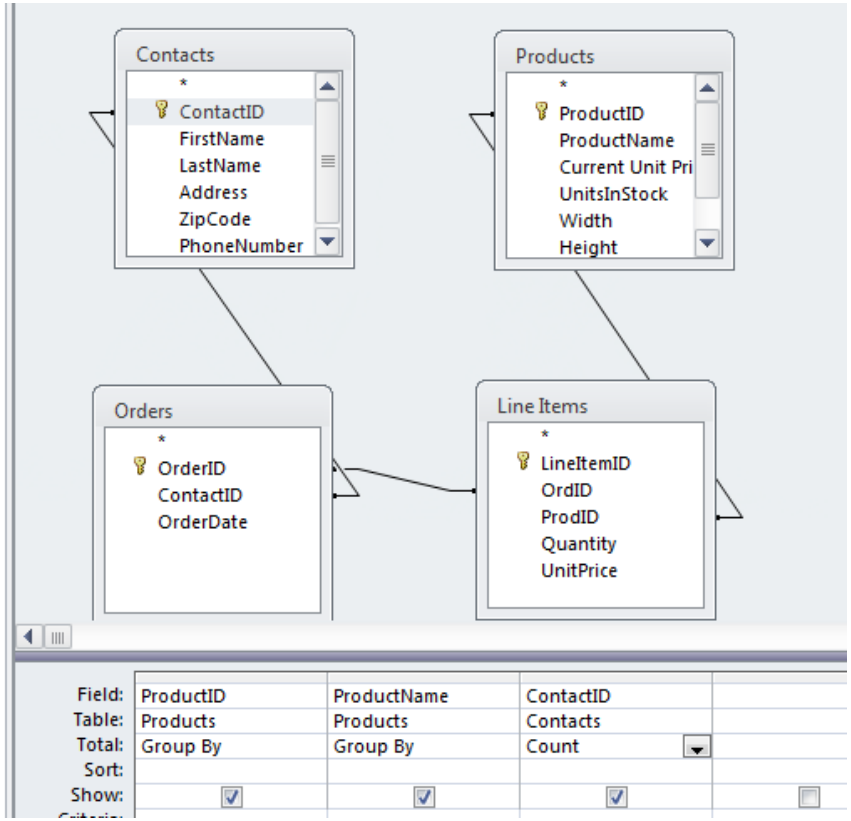


Note the use of the subquery and the aggregate AVG function



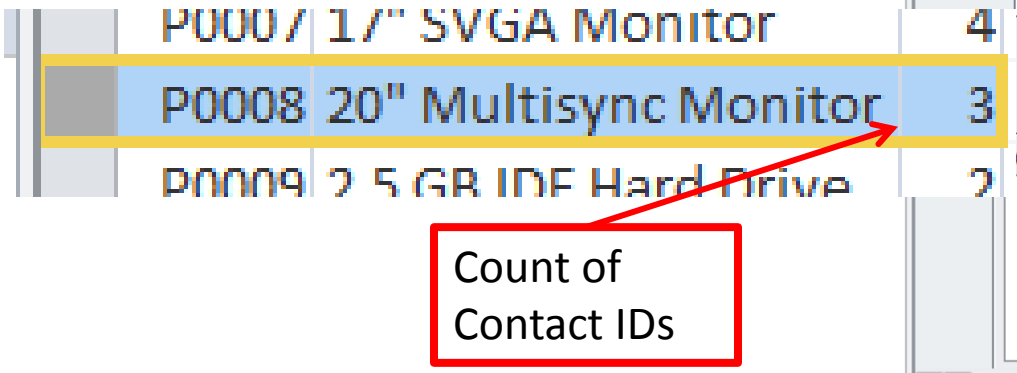
Example

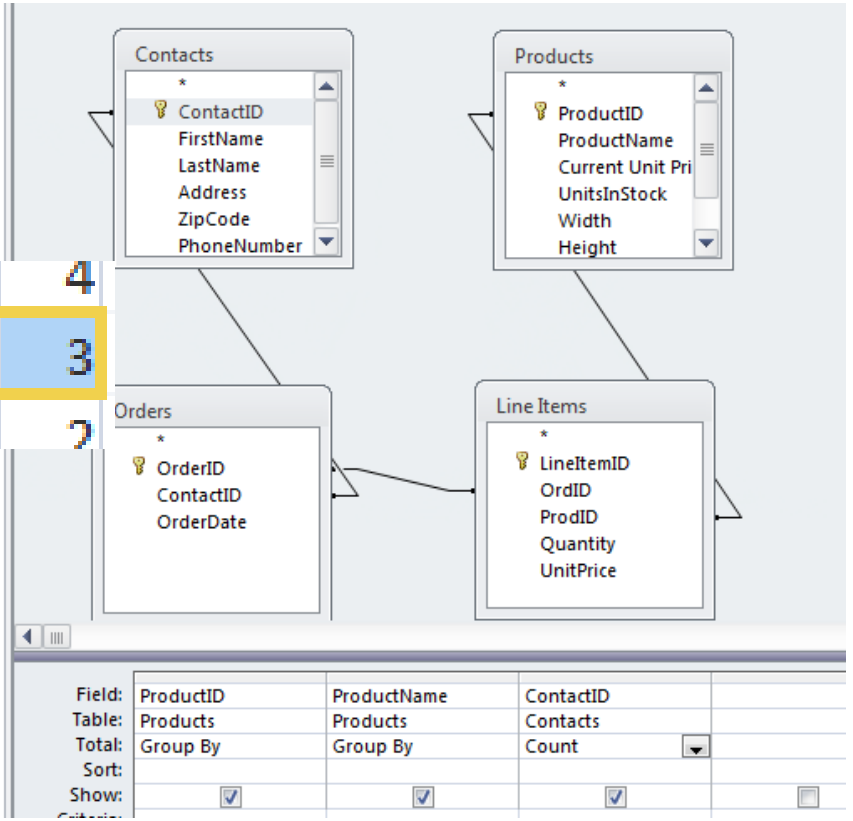
- How many different people bought each product?
- Attempted query



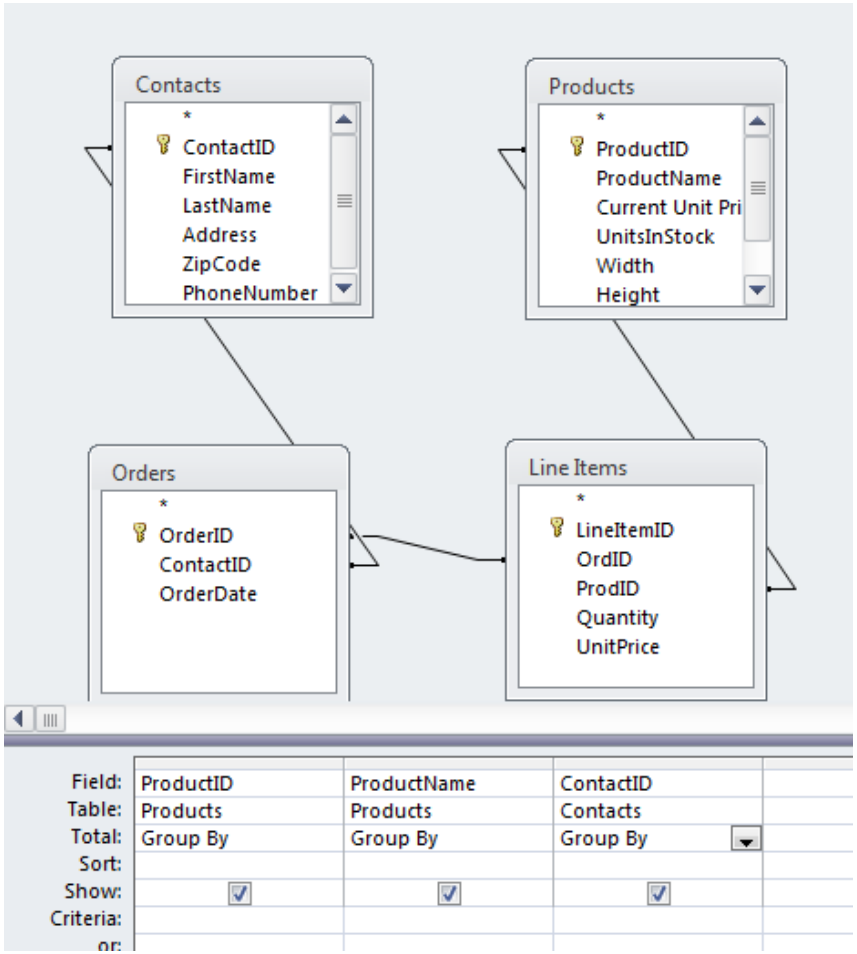
Example: Counting Unique Occurrences

- How many *different* people bought each product?
- Attempted query





But

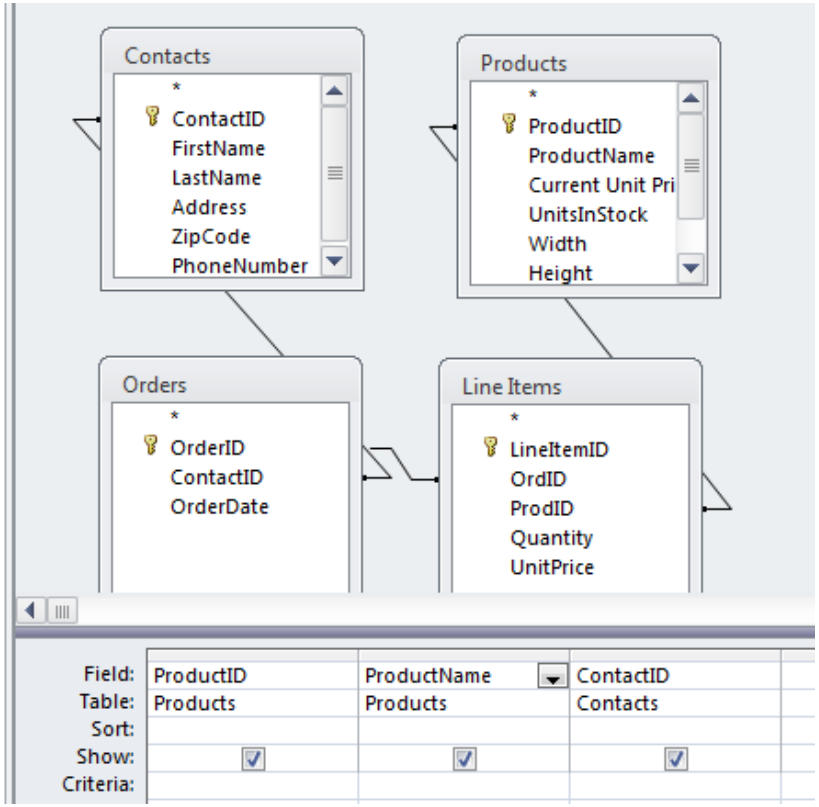


Grouping by Contact IDs shows 2 different contacts ordered Product #8

P0007	17" SVGA Monitor	18
P0008	20" Multisync Monitor	2
P0008	20" Multisync Monitor	14
P0009	2.5 GB IDE Hard Drive	4

Why?

- Let's remove the Total field and Ungroup.



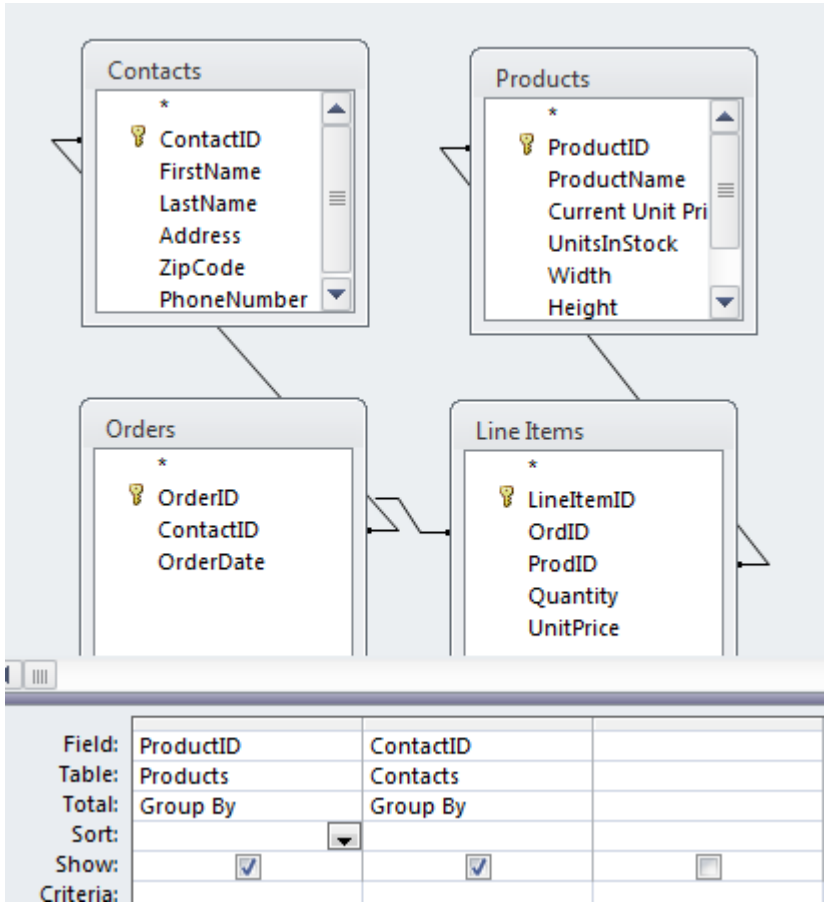
	P0007	17" SVGA Monitor	18
	P0008	20" Multisync Monitor	2
	P0008	20" Multisync Monitor	2
	P0008	20" Multisync Monitor	14
	P0009	2.5 GB IDE Hard Drive	4

Why?

- Let's remove the total field.
- Someone ordered it twice
  - **We have duplicates**
- Group by removes duplicates but doesn't count.
- Count aggregates but doesn't remove duplicates.
- So what do we do?

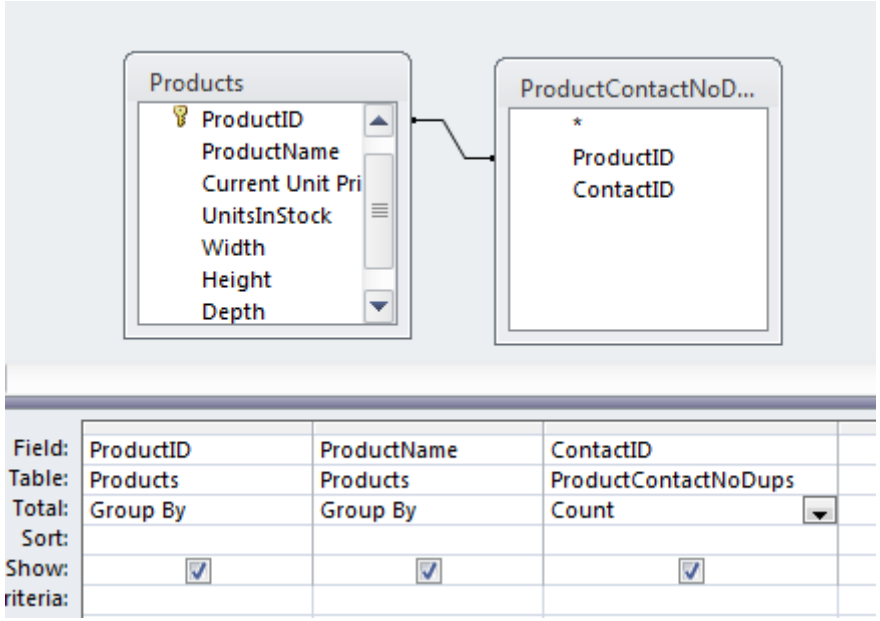
Solution

- Use subquery to perform grouping
- **ProductContactNoDups**



Solution

- Use subquery to perform grouping, then count



P0007	17" SVGA Monitor	4
P0008	20" Multisync Monitor	2
P0009	2.5 GB IDE Hard Drive	2

Access Queries

TRY FOR YOURSELF…

Question 1

- Which contacts placed three or more orders?

Question 2

- How many contacts placed three or more orders?

Question 3

- Which states had contacts that placed 3 or more orders?

Maximum and Minimum

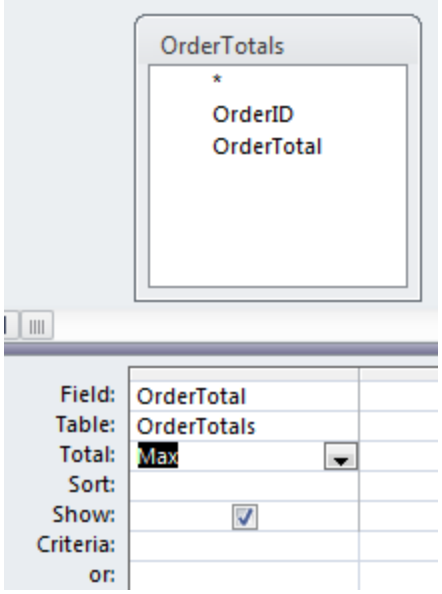
- What is the maximum amount of any order that was ever placed?
- What is the minimum amount of any order that was ever placed?
- Which order was the minimum order?
- Who placed the minimum (smallest) order?

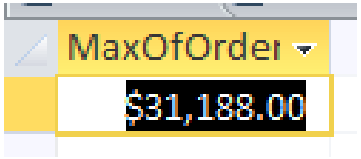
Query

- What is the maximum amount of any order that was ever placed?

Query

- What is the maximum amount of the any order that was ever placed?
  - Use **OrdersTotal** again as a subquery
  - Apply the *MAX* aggregate function



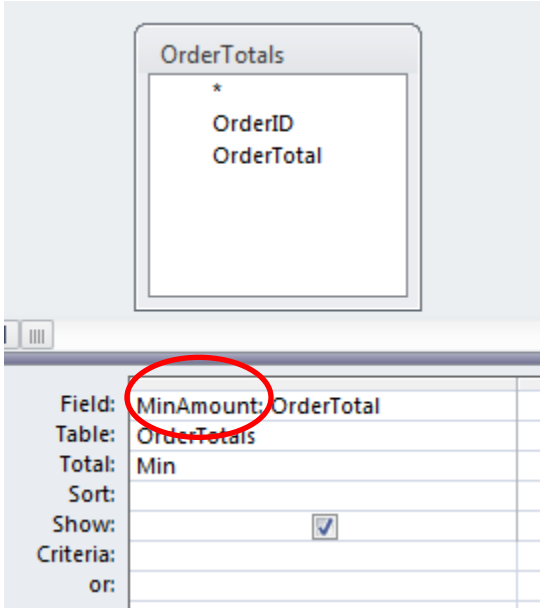


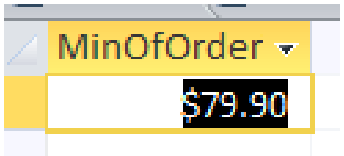
Query

- What is the minimum amount of any order that was ever placed?

Query

- What is the minimum amount of any order that was ever placed?
  - Use **OrdersTotal** again as a subquery
  - Apply the *MIN* aggregate function



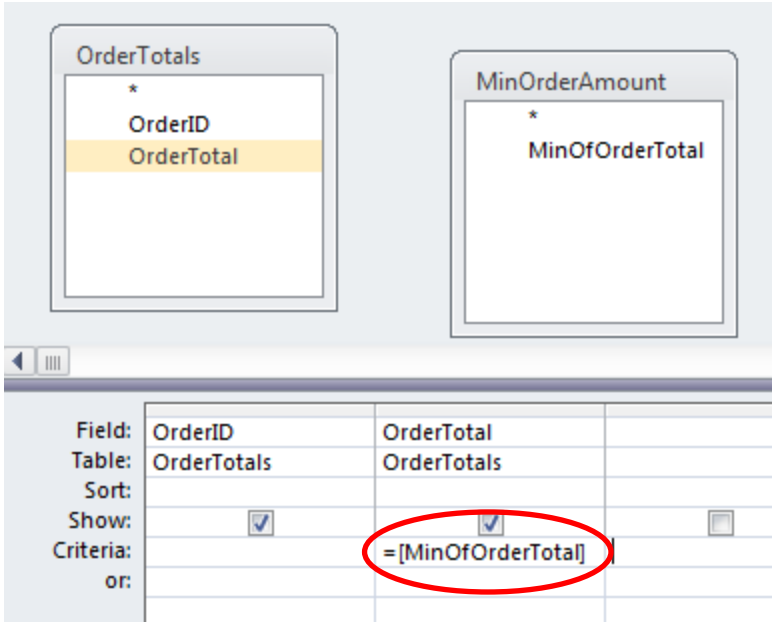


Query

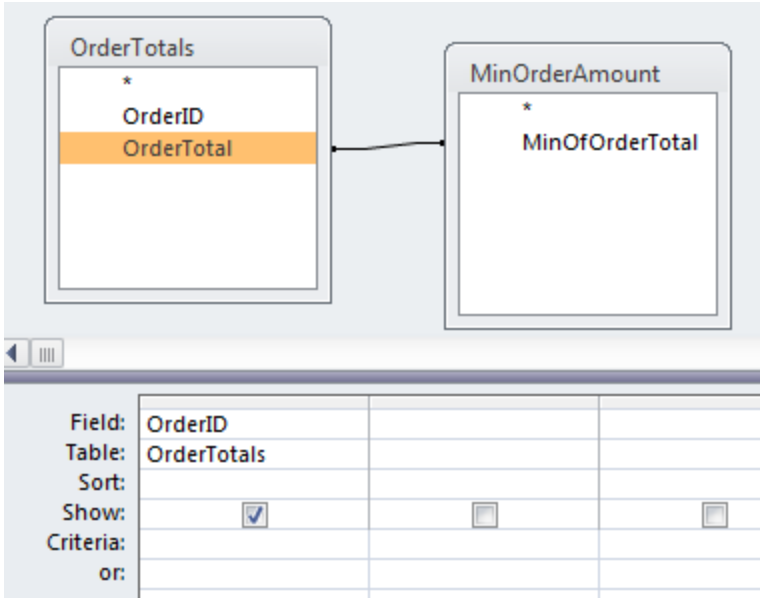
- Which order was the minimum order?

Query

- Which order was the minimum order?
  - Requires the previous query and **OrdersTotal** as subqueries
  - Can use one of two approaches



Approach 1: Use Criteria



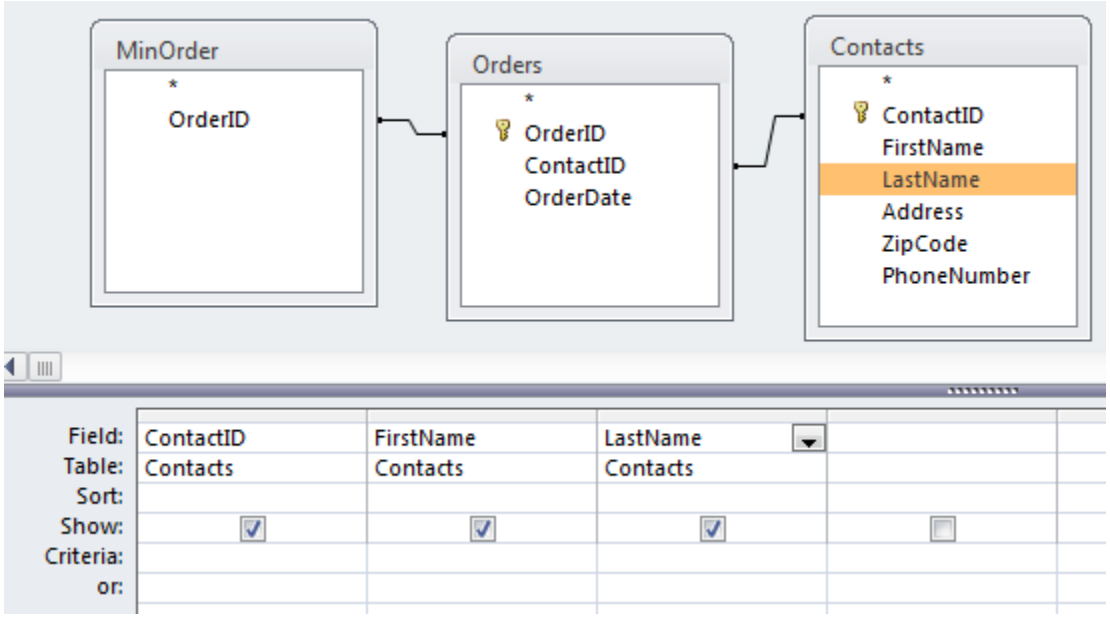
Approach 2: Create Relationship

Query

- Who placed the smallest order?

Query

- Who placed the smallest order?
  - Use previous query as subquery
  - Combine with **Orders** and **Contacts** table



ContactID	FirstName	LastName
11	Lauren	Center

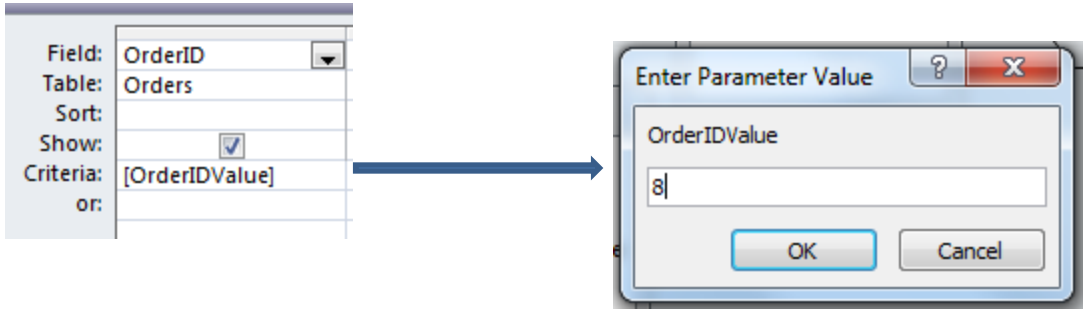
Try this:

- Which orders contained a line item for the least expensive product (based on CurrentUnitPrice)?

- Who ordered the least expensive product?

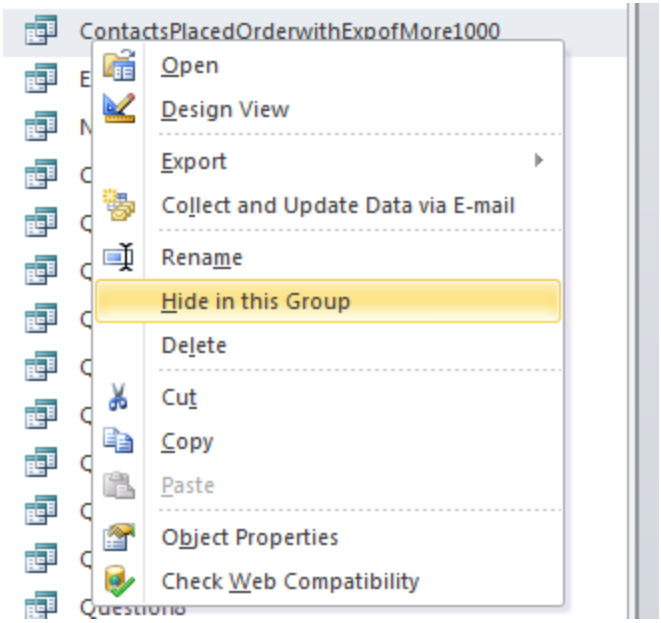
Parameterized Query

- To allow user input for a query value:
  - specify a variable that has a name different from any of the field names



Hiding Subqueries

- Queries (and tables) can be “hidden”:
  - Right-click on query in navigation panel
  - Select “Hide in this Group”



Unhiding Queries

- Click anywhere in the Query Explorer
- Select “Navigation Options…”
- Check “Show Hidden Objects”

- Now all hidden queries are visible and can be unhidden.

Summary

- Group By removes duplicate rows where the Group By values are the same
- Aggregate functions apply to groups or entire tables depending how they are used
- Subqueries are necessary to build complex queries