

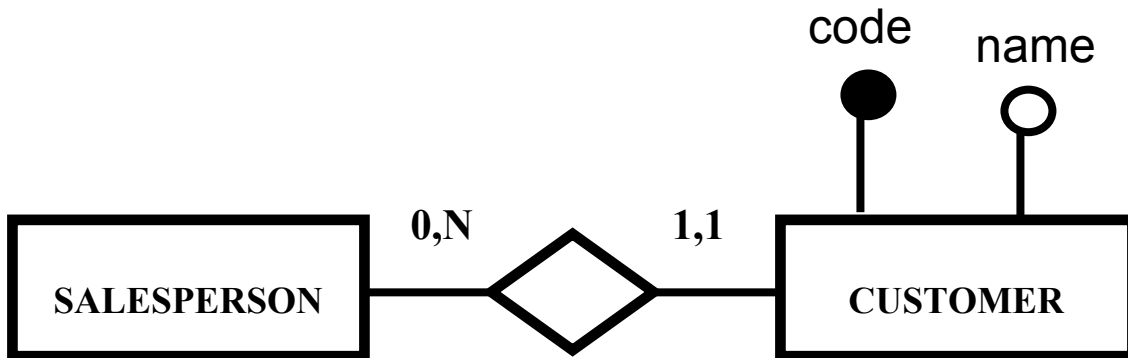
Use the “start” database for the first question (design).
Use the “stuffed animals” database (stufan) for all other questions

PART I: DESIGN QUESTION (2 POINTS)

ASSIGNMENT

Extend the start.mdb database with the Customer entity (see E-R diagram below).

- Implement the Customer entity (code is number and name is text).
- Implement the SalesPerson -- Customer relationship.
- Make sure that participation of Customer in the Customer-Salesperson relationship is mandatory.
- Make sure that participation of Salesperson in the SalesPerson-Customer relationship is optional (subset).



SOLUTION

The SalesPerson entity (table) is already implemented.

1. Implement the Customer Entity (Table)

Define the two attributes (code and name), their data types, and define code as primary key.

Field Name	Data Type	Description
code	Number	
name	Text	

Field Properties

General Lookup

The field description is optional. It helps you describe the field and is also displayed in the status bar when you select this field on a form. Press F1 for help on descriptions.

2. Implement the Salesperson-Customer Relationship

Define Salesperson as a Foreign Key attribute in Customer
Make sure that the data type of Customer.Salesperson (FK) and Salesperson.code (PK) are the same.

Field Name	Data Type	Description
code	Number	
name	Text	
Salesperson	Text	

Field Properties

General Lookup

Field Size: 255

Format:

Input Mask:

Caption:

Default Value:

Validation Rule:

Validation Text:

Required: Yes

Allow Zero Length: Yes

Indexed: No

Unicode Compression: Yes

IME Mode: No Control

IME Sentence Mode: None

Smart Tags:

Require data entry in this field?

2

3

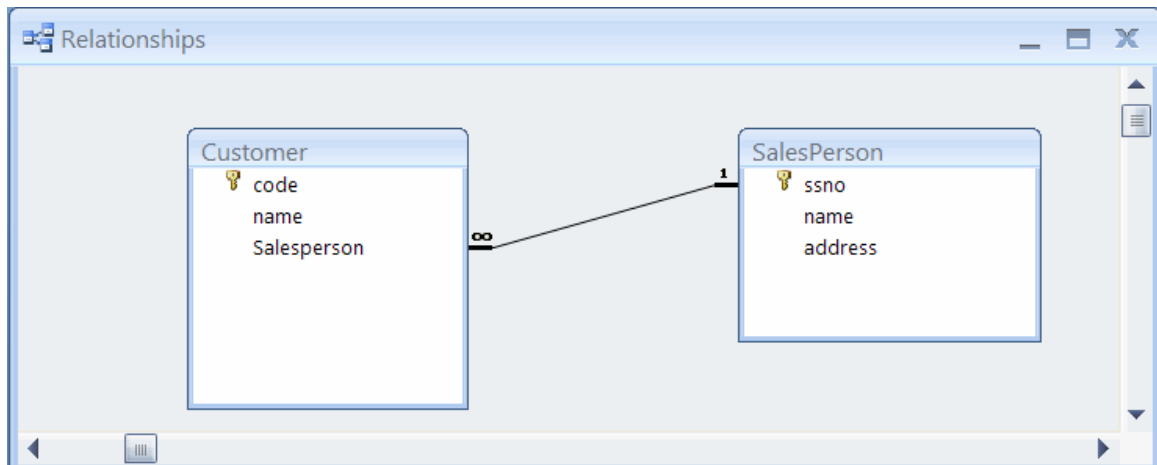
3. Make sure that participation of Customer in the Customer-Salesperson relationship is mandatory.

Define the property “Required” as “yes.” This implements mandatory participation of Customer in the Customer-Salesperson relationship (no bubble). There is a salesperson for each customer.

Save your table as “Customer.”

4. Make sure that participation of Salesperson in the Salesperson-Customer relationship is optional (subset).

Go to the relationship window and define the relationship between Customer and Salesperson.



The subset relationship is implemented by checking the “Enforce Referential Integrity” box.

PART II: QUERIES 1-10 (1 point queries)

QUERY 1

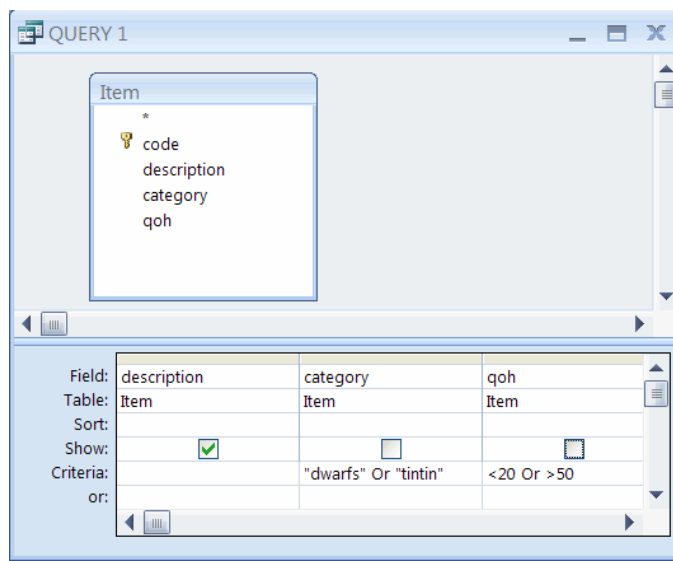
ASSIGNMENT

Generate a list of all products [Item] (description) that are either of category “dwarfs” or of category “tintin” and for which there are less than 20 or more than 50 in stock (qoh).

SOLUTION

Description
BashFul Large
BashFul Small
Captain Haddock
Grumpy Large
Grumpy Small
Happy Large
Happy Small
Sleepy Small
Sneezy Small
Snow White Large
Snow White Small

QUERY DESIGN



DISCUSSION

You need to determine a list of products (items) — attribute: description. You don't want all instances of the item table; only the instances that are either of category 'dwarfs' or of category 'tintin' and (at the same time) for which there are less than 20 or more than 50 in stock (qoh). Don't show the "category" and "qoh" attributes as part of the query results; they are used for the definition of the criteria only.

QUERY 2

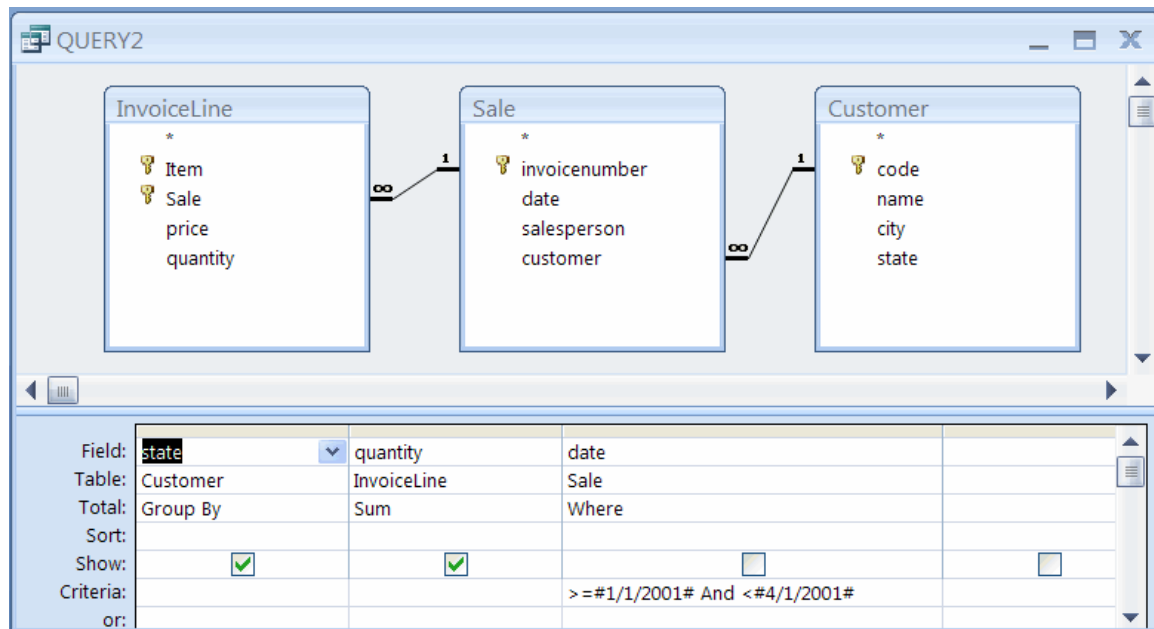
ASSIGNMENT

Total quantity of items sold per state in February and March

SOLUTION

State	Total Quantity Sold in February and March
AZ	80
DE	180
FL	86
IL	70

QUERY DESIGN



DISCUSSION

Where to find the information? State is information recorded for customers. The sales date is recorded as an attribute of the Sale entity. Quantity sold is information recorded per invoiceline. The inner-joins between Invoiceline, Sale and Customer generate a table of all invoicelines per customer. Where is used to restrict the sales (and thus the invoicelines) to the instances that took place in February and March. Finally, Group By groups the remaining instances per state and Sum adds the quantities per state.

QUERY 3

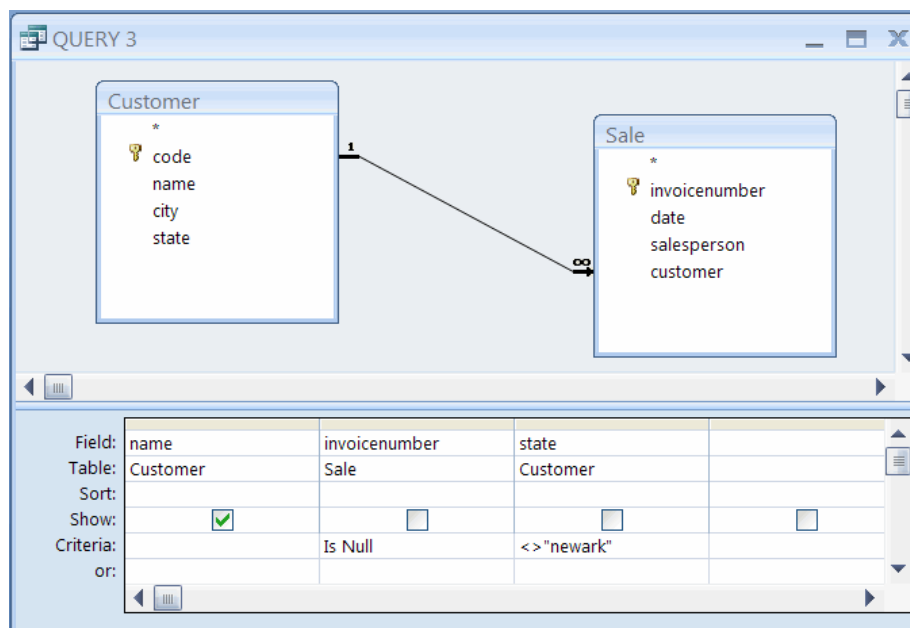
ASSIGNMENT

List of all potential customers living outside Newark; that is: customers living outside Newark that haven't bought anything from us yet.

SOLUTION

Name
Sidonia Lang

QUERY DESIGN



DISCUSSION

We use an **outer join**. The outer join relationship between Customer and Sale determines **ALL** customers and their participation in the relationship with sales. If a customer does not participate in the relationship then a null value will be shown for that Sale.invoicenum attribute. By selecting invoice numbers that are null, we select customers that do not participate in the relationship with sale (potential customers). We also use the attribute state to express that we only consider customers living outside of Newark. For the data currently in the database exclusion of the Newark customers does not change the result.

QUERY 4

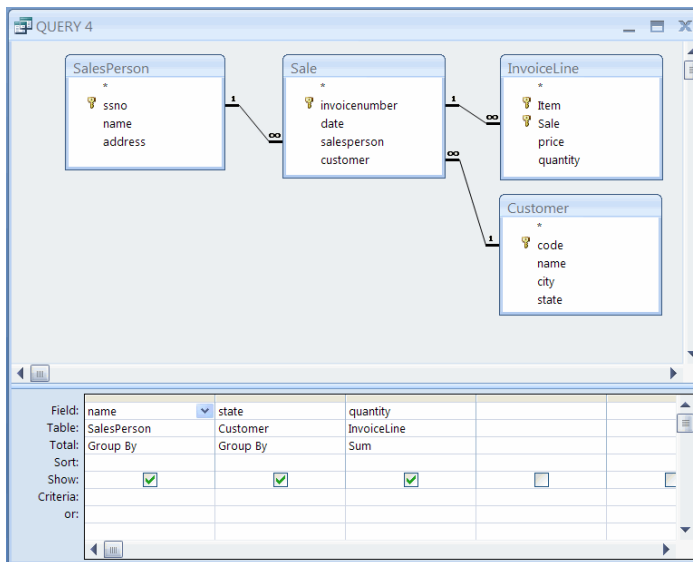
ASSIGNMENT

For each of the Salespeople, list the total number of items (quantity) sold per State.

SOLUTION

Name	State	TotalQuantitySold
Angela Lafrance	FL	26
Angela Lafrance	IL	70
Brad Watson	AZ	97
Stacy Smith	FL	10
Wesley Meckstroh	DE	228

QUERY DESIGN



DISCUSSION

You would like to know the total quantity sold per salesperson per state. The name of the salesperson can be found in the salesperson table. The state can be found in the customer table. Quantity sold can be found in the invoiceline table. Use inner-joins to connect salespeople with sales and sales with invoicelines and customers. Then Group By name and state and Sum quantity.

QUERY 5

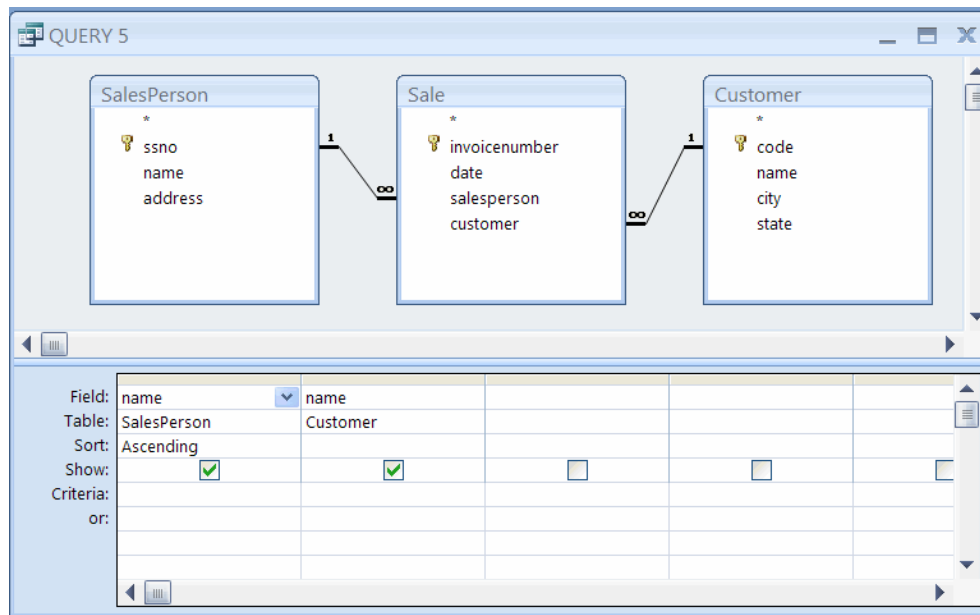
ASSIGNMENT

List of all Customers each SalesPerson has sold items to.

SOLUTION

Salesperson	Customer
Angela Lafrance	Asterix the Gaul
Angela Lafrance	Lucky Luke
Brad Watson	Cruella De Vil
Stacy Smith	Scar LeRoi
Wesley Meckstroh	Rosy Gobelijn
Wesley Meckstroh	Winnie Pooh

QUERY DESIGN



DISCUSSION

You need to define the path between Customer and SalesPerson (inner-joins). Use sale to link the two entities: SalesPerson-Sale-Customer. Select name from Salesperson and name from Customer. Make sure to set "Unique Values" to "Yes" in the Property Sheet.

QUERY 6

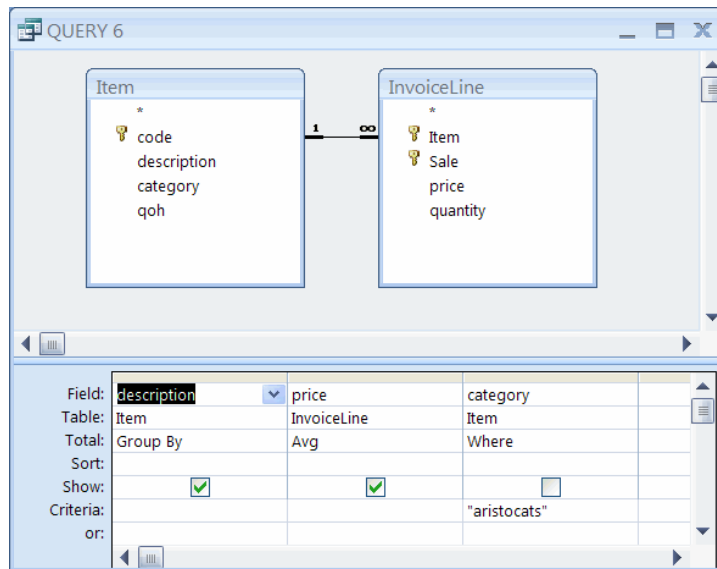
ASSIGNMENT

What is the average price we asked customers to pay for each of the ‘Aristocats’ Items?

SOLUTION

ITEM	AVERAGE PRICE
Berlioz	25
Duchess	23.5
Marie	25
Thomas	25
Toulouse	25

QUERY DESIGN



DISCUSSION

We calculate the average for all prices asked (invoiceline) — average price -- per item (description). Group By groups all instances of an item (description), for example all instances of Toulouse, which appear on an invoiceline. AVG calculates the average for all instances per (item) group — for example the average of all prices asked for Toulouse items. We only consider items of category “Aristocats.”

QUERY 7

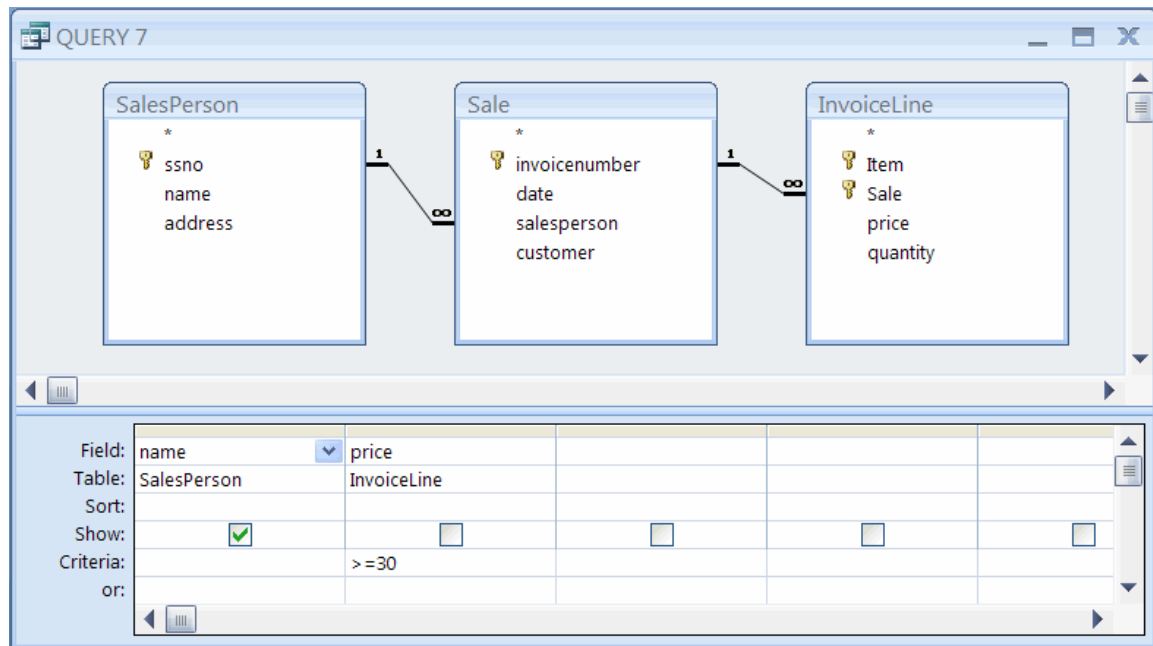
ASSIGNMENT

List of SalesPeople (name) that have sold at least one item of \$30 or more (price).

SOLUTION

Name
Angela Lafrance
Brad Watson
Stacy Smith
Wesley Meckstroh

QUERY DESIGN



DISCUSSION

By using '>= 30' as criteria for price we select the instances of the InvoiceLine entity (table) for which the price is at least \$30. The inner-join between Sale and InvoiceLine then restricts the sales to sales that have at least one invoiceline with an item that has been sold for at least \$30. The inner-join between SalesPerson and Sale then determines the salespeople that have at least one sale with an item of at least \$30. Make sure to set "Unique Values" to "Yes" in the Property Sheet.

QUERY 8

ASSIGNMENT

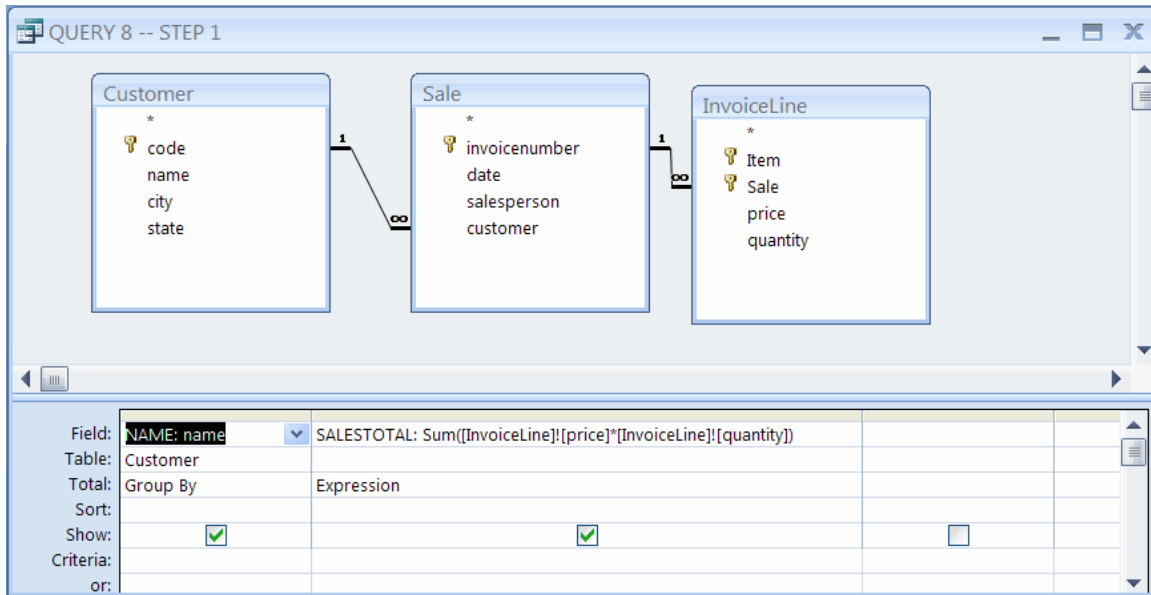
For all customers that have bought goods (items), list the total sale amount and the total amount of payments.

SOLUTION

NAME	SALESTOTAL	PAYMENTTOTAL
Asterix the Gaul	1220	600
Cruella De Vil	2595	1100
Lambik Wawa	1370	1350
Lucky Luke	1656	556
Rosy Gobelijn	2630	
Scar LeRoi	300	300
Winnie Pooh	3125	

QUERY DESIGN

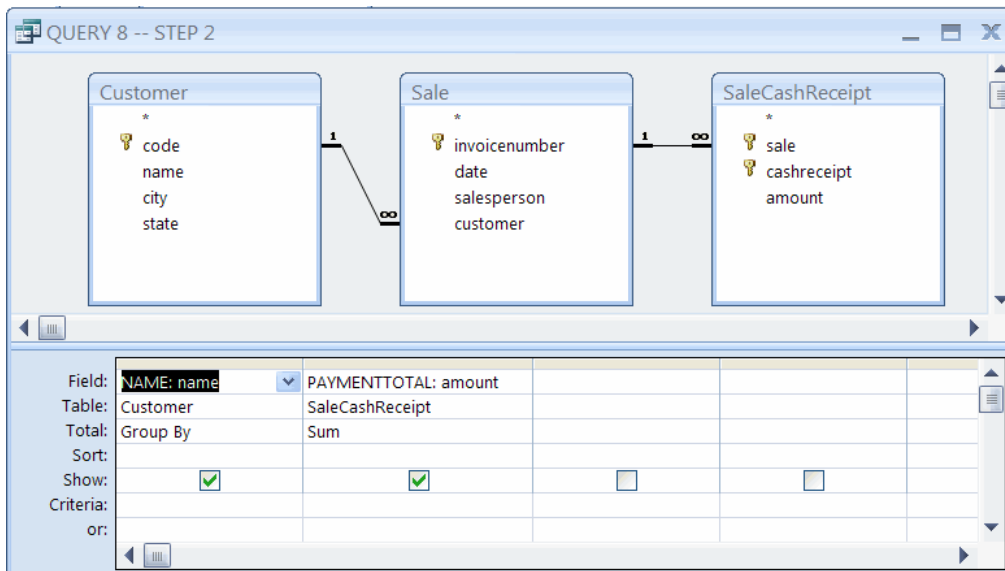
STEP-1 DESIGN (CUSTOMERTOTALSALES)



STEP-1 RESULTS

NAME	SALESTOTAL
Asterix the Gaul	1220
Cruella De Vil	2595
Lambik Wawa	1370
Lucky Luke	1656
Rosy Gobelijn	2630
Scar LeRoi	300
Winnie Pooh	3125

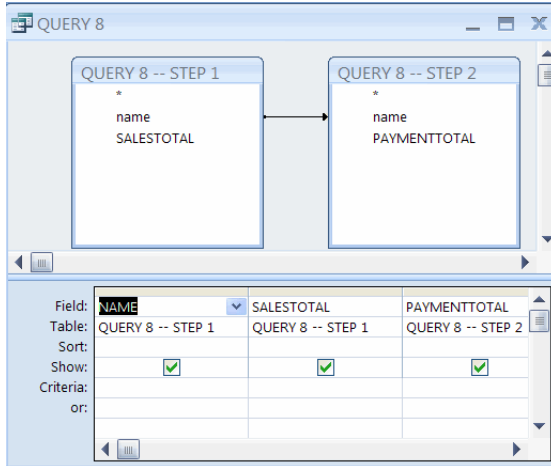
STEP-2 DESIGN (CUSTOMERTOTALPAYMENTS)



STEP-2 RESULTS

NAME	PAYMENTTOTAL
Asterix the Gaul	600
Cruella De Vil	1100
Lambik Wawa	1350
Lucky Luke	556
Scar LeRoi	300

QUERY 8 DESIGN



DISCUSSION

There are several ways to solve this problem. The solution presented here (see diagram next page) is a view hierarchy with 3 queries (2 levels).

1. We start (**step 1**) with the definition of a query that determines all customers that have bought goods and determine the total amount of sales for each of these customers. A customer who bought products is a customer who participates in the relationship with sales. For the definition of the query we a) select the customer, sale, and invoiceline tables, b) the inner-joins between these three tables are automatically defined; the inner-join between customer and sale determines the customers that have bought goods, c) we need to determine the sales total per customer, we sum (Sum) the invoice line totals (p*q) for all customers (Group By). The result table contains the names of 7 customers and their sales totals.
2. Next (**step 2**), we determine the total payments per customer. We use the tables customer, sale, and salecashreceipt to determine that information. The inner-join between customer and sale restricts the customers in the result table to customers that have bought goods, the inner-join between sale and salecashreceipt further restricts the customers in the result table to customers that have made a payment. The result table contains the name of 5 customers and their payment totals. Rosy Gobelijn and Winnie Pooh have not made any payment yet.
3. Finally we would like to create a result table that lists the name of all customers that have bought goods (there are 7), the sales totals for all 7 customers and the payment made by all 7 customers. However, the customerstotalpayments table contains only five customers. We can use an outer join to include all customers that have bought goods (and thus all customers in the customerstotalsales table). The result table contains seven customers and blanks for Rosy Gobelijn and Winnie Pooh in the paymenttotal column.

QUERY 9

ASSIGNMENT

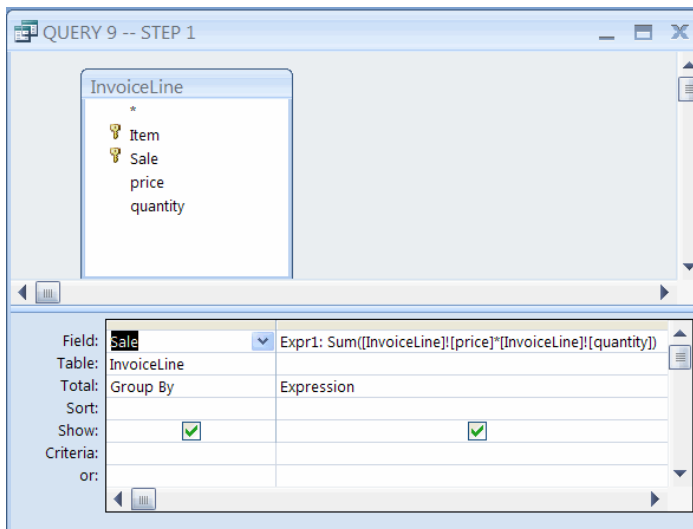
Show payment information for sales with invoice numbers 1, 2, 3 and 4. Show invoice number, invoice total, the cash receipts (remittance advice numbers) that pay for the invoices, and the amount of these cash receipts used to pay for the invoices.

SOLUTION

invoicenumber	invoice total	cashreceipt	amount
1	600	1	600
2	400	6	400
3	300	2	300
4	600	3	300
4	600	4	300

QUERY DESIGN

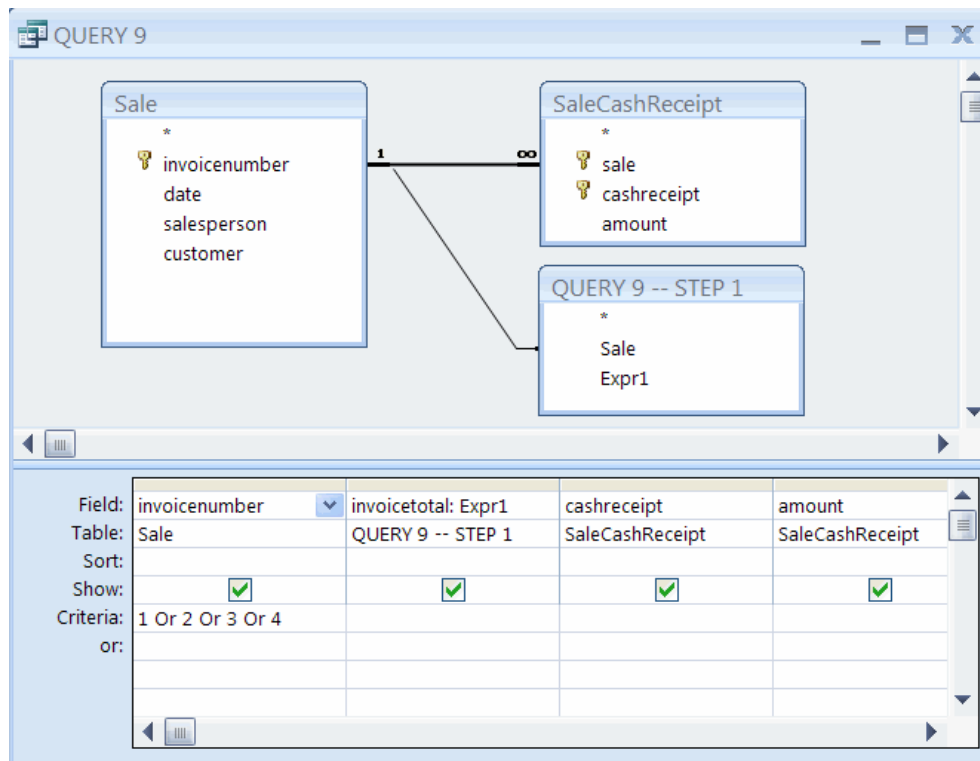
STEP-1 DESIGN (InvoiceTotal)



STEP-1 RESULTS

Sale	Total
1	600
2	400
3	300
4	600
5	556
6	750
7	200
8	1400
9	3125
10	1100
11	620
12	595
13	1610
14	20
15	1020

QUERY 9 DESIGN



DISCUSSION

First (**step 1**) create a query that determines the invoice total for each of the sale instances (invoicetotal). Then, create a query that shows for each invoice: invoiceno, the invoice total (inner join with the invoicetotal query) and information of the payments for each of the invoices — remittance advice number and amount. You need to further restrict the output table to the invoices 1,2,3 and 4. This can be done by putting ‘1 or 2 or 3 or 4’ in the criteria box.

QUERY 10

ASSIGNMENT

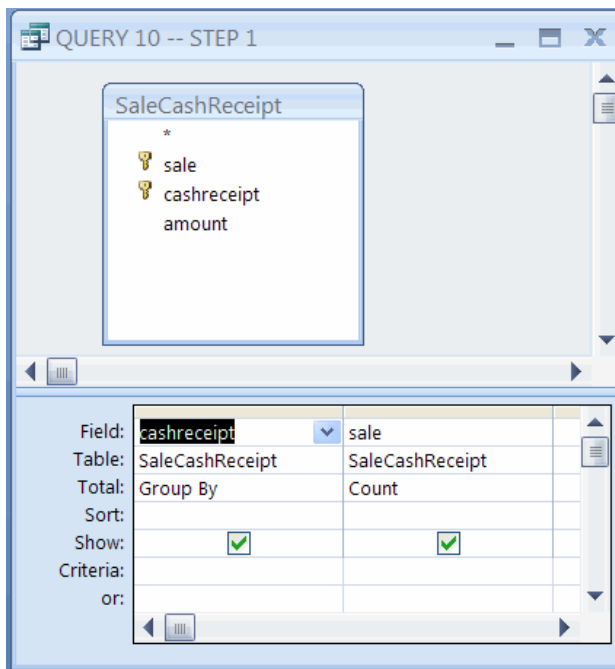
List all cash receipts that pay for two or more invoices.

SOLUTION

cashreceipt
6

QUERY DESIGN

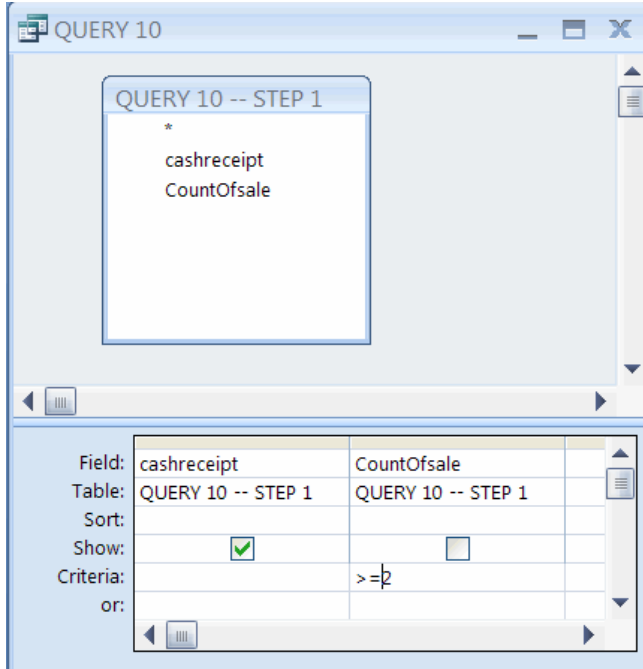
STEP-1 DESIGN (Number of Invoices a Cash Receipt Pays For)



STEP-1 RESULTS

cashreceipt	number of sales
1	1
2	1
3	1
4	1
5	1
6	3
7	1

QUERY 10 DESIGN



DISCUSSION

We are going to use a view hierarchy and two queries to solve the problem (Note: this problem can be solved by one query). The first query uses aggregation -- Group By cash receipt and Count for how many invoices a payment (check) pays. The second query then selects all cash receipts that pay for 2 or more invoices (≥ 2).

PART III: QUERIES 11-14 (2 point queries)

QUERY 11

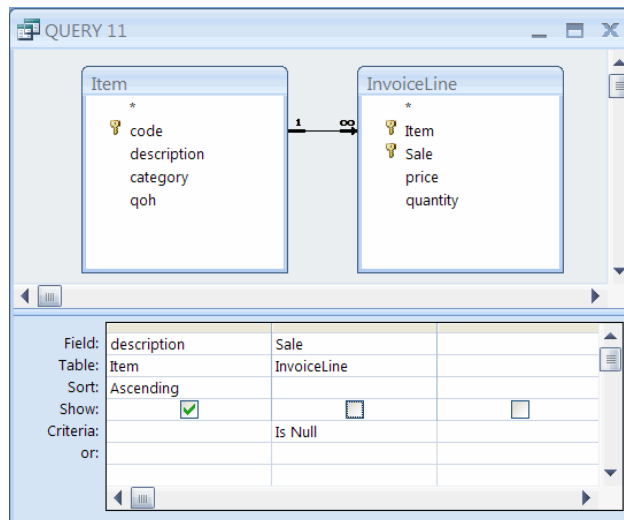
ASSIGNMENT

List of items (descriptions) that haven't been sold yet.

SOLUTION

description
BashFul Large
BashFul Small
Grumpy Small
Happy Large
Sleepy Small
Snow White Small
Snowy

QUERY DESIGN



DISCUSSION

This problem can be solved by an outer join. First select the item and invoiceline tables, an inner-join exists between these two tables. An item that has been sold has to appear on an invoiceline. Next, define an outer join. Include all instances of item, even when they don't participate in the relationship. Then select description (attribute of Item) and Sale (attribute from InvoiceLine). The instances with no value (null) for Sale represent the items that haven't been sold yet. Select these items by putting "is null" in the criteria box.

QUERY 12

ASSIGNMENT

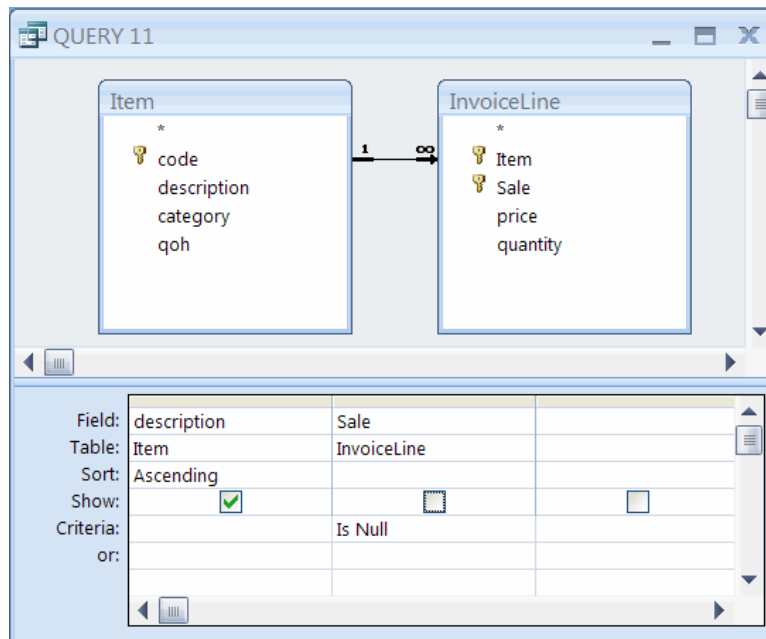
Determine the total number of different (!) items bought by each Customer.

SOLUTION

Name	NumberOfDifferentItems
Asterix the Gaul	3
Cruella De Vil	6
Lambik Wawa	6
Lucky Luke	3
Rosy Gobelijn	3
Scar LeRoi	1
Winnie Pooh	5

QUERY DESIGN

STEP-1 DESIGN (Items Bought by a Customer)



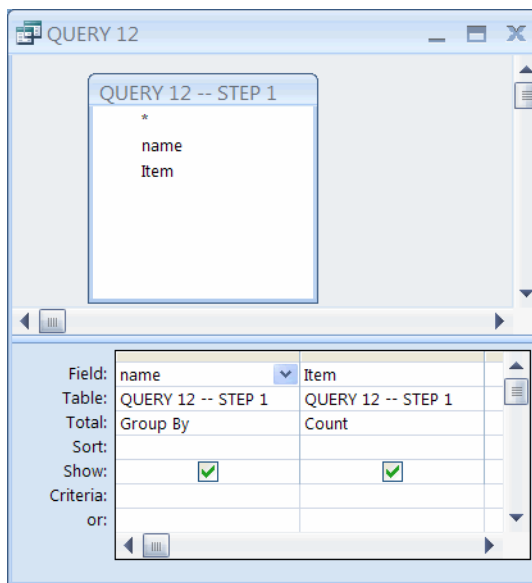
STEP-1 RESULTS

Name	Item
Cruella De Vil	DOCL
Cruella De Vil	GRUML

Name	Item
Lambik Wawa	SNES
Scar LeRoi	DOCL
Asterix the Gaul	SLB
Lucky Luke	DOCL
Lucky Luke	TINL
Lambik Wawa	MAR
....	

(28 records)

QUERY 12 DESIGN



DISCUSSION

Different steps and thus a view hierarchy are needed to solve this problem (See next page for the view hierarchy).

We first (**step 1**) determine the items bought by a customer. We use the tables customer, sale and invoiceline for this query. Invoiceline contains the item. If a customer buys the same item more than once, the item will be listed more than once in the result table. Since we are only interested in the different items bought by a customer, we set “Unique Values” to “Yes” in the Property Sheet.

We then use the first query as input to determine (count) the different items bought by a customer. We Group By customer and count (Count) the different items.

QUERY 13

ASSIGNMENT

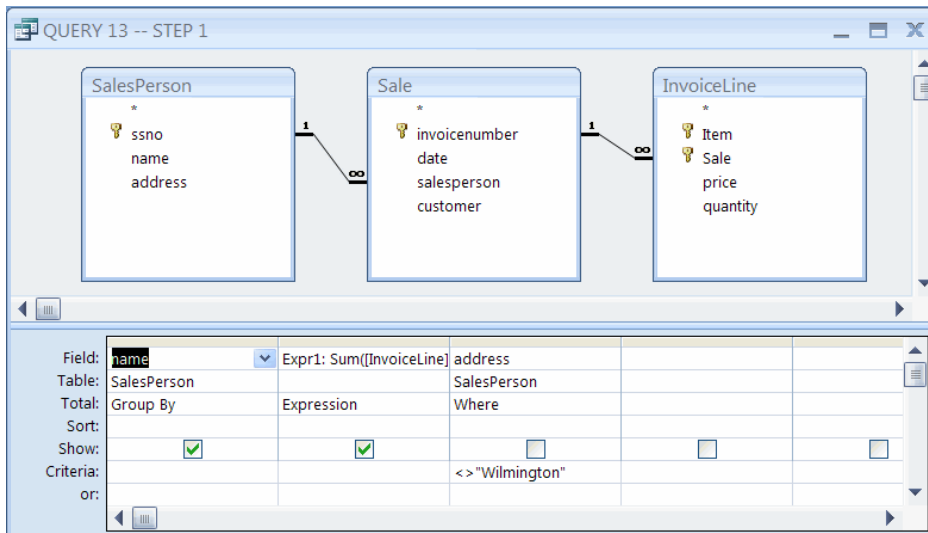
We would like to give an award to the salesperson with the highest sales total (\$) who lives outside Wilmington. What is the name of this SalesPerson?

SOLUTION

Name
Brad Watson

QUERY DESIGN

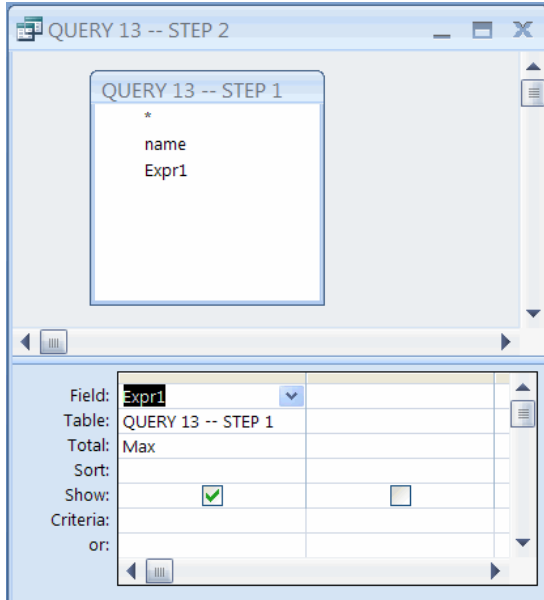
STEP-1 DESIGN (Sales Total for Sales People living outside Wilmington)



STEP-1 RESULTS

Name	Sales Total
Angela Lafrance	1776
Brad Watson	2595
Stacy Smith	300

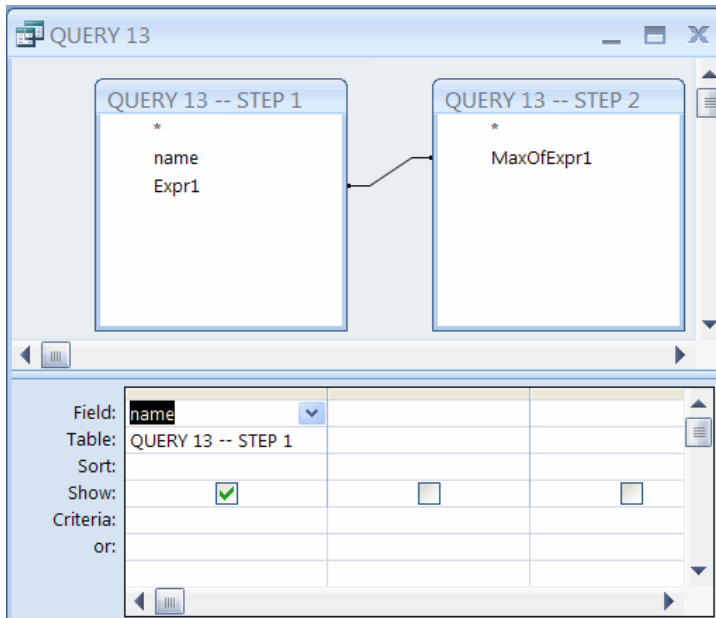
STEP-2 DESIGN (Highest Sales Total)



STEP-2 RESULTS

Highest Sales Total
2595

QUERY 13 DESIGN



DISCUSSION

Different steps are needed to solve this problem and we are going to use a view hierarchy (see next page).

We first (**step 1**) determine the sales total for all salespeople living outside of Wilmington. We do this by summing all the invoiceline totals ($p*q$) (Sum) per salesperson (Group By). We further use Where to eliminate the salespeople that live in Wilmington.

Next (**step 2**), we determine the highest sales total — what is the highest salestotal for a salesperson living outside of Wilmington (max salestotal).

Finally, we have to determine the name of the salesperson. We define an inner join between the max sales total (step 2) and the sales totals in the first query (step 1). This constraint (matching values) selects the row with the highest sales total in query 1 and we can then determine the name of the salesperson living outside Wilmington with the highest salestotal.

QUERY 14

ASSIGNMENT

Name of the salesperson with the highest average invoice total in March.

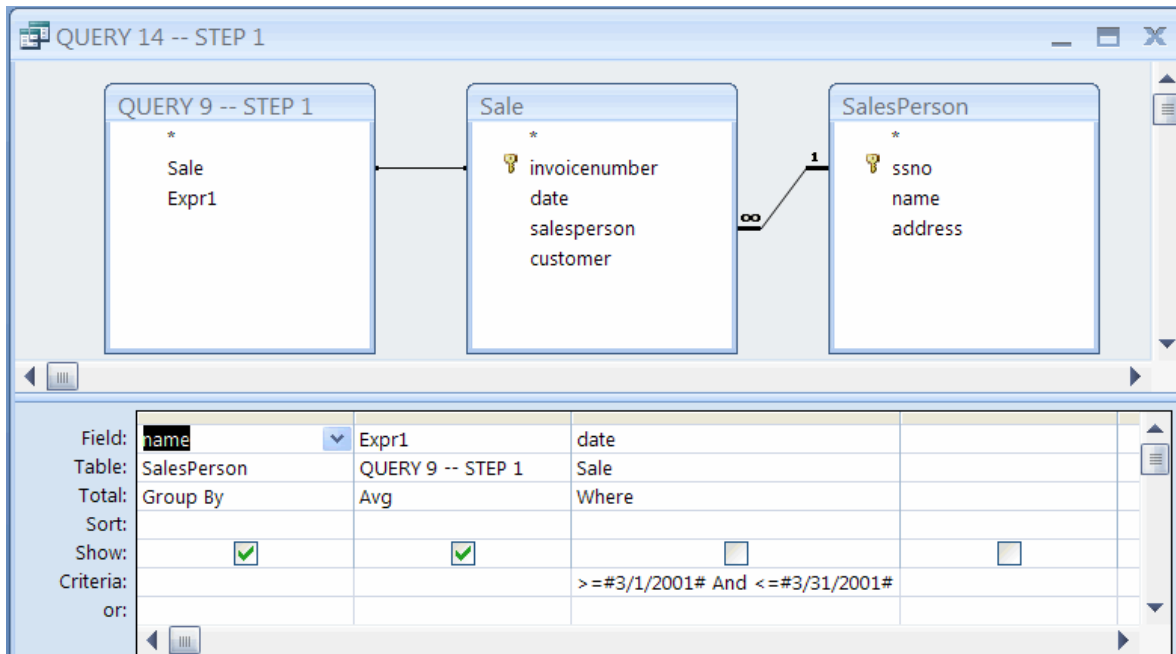
SOLUTION

Name
Wesley Meckstroh

QUERY DESIGN

STEP-1 DESIGN AND RESULTS – (See Step-1 Query 9: Invoice Total)

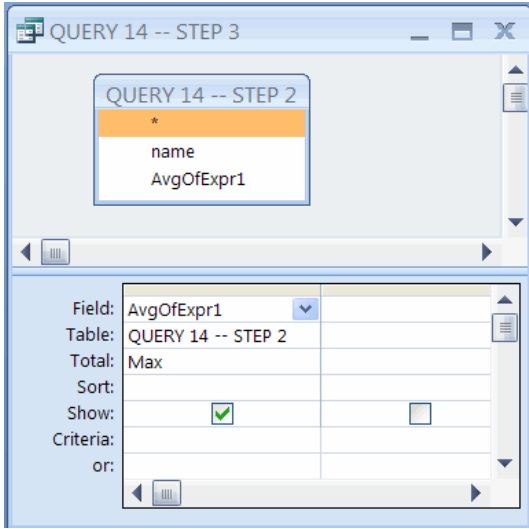
STEP-2 DESIGN (Average Invoice Total in March per SalesPerson)



STEP-2 RESULTS

Name	AverageInvoiceTotalMarch
Angela Lafrance	588
Brad Watson	1400
Wesley Meckstroh	3125

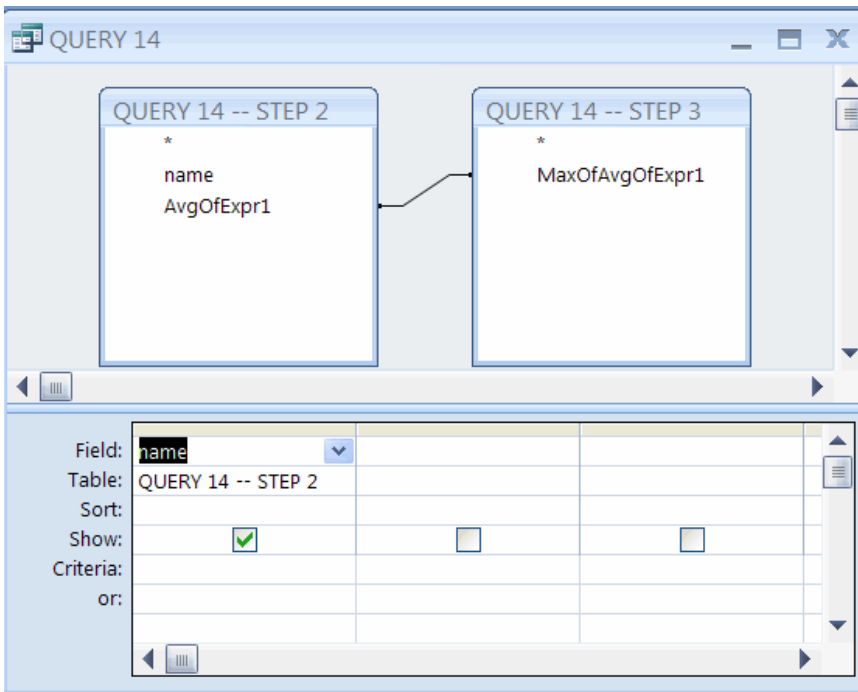
STEP-3 DESIGN (Highest Average Invoice Total)



STEP-3 RESULTS

Highest Average Invoice Total
3125

QUERY 14 DESIGN



DISCUSSION

Different steps are needed to solve this problem and we are going to use a view hierarchy (see next page).

We first (**step 1**) determine the total for each invoice (see Query 9). Next (**step 2**), we determine the average March invoicetotal per salesperson. We use AGGregation to accomplish this — AVG to calculate the average invoicetotal (per salesperson) and Where to limit the instances of sale to March sales. The result table shows 3 salespeople and their average invoicetotal for March. Next (**step 3**), we determine which of these average invoice totals is highest; this is done by AGGregation/Max. Finally, we determine the name of the salesperson. We use an inner join to link the maximum amount (step 3) with the amounts in query 2 (step 2). The inner join expresses a constraint. Only the salesperson for which the average invoice total in March is equal to (=) the maximum average invoice total in March will be selected. That salesperson is Wesley Meckstroh.