

23.1 Introduction: Reporting, OLAP, and data mining

In this lesson, we are going to briefly explore the ways in which multidimensional data can be queried and manipulated to provide answers to business questions. Of course, an in depth discussion of reporting, **on-line analytical processing** (OLAP) and **data mining** are well beyond the scope of the lesson. Instead, the objective here is simply to introduce terminology and some simple yet powerful decision support tools.

23.1.1 An example

To help sell its INTELLIGENT MINER data mining software, IBM uses the example of SAFEWAY PLC—the third largest chain of retail food stores in the United Kingdom. The SAFEWAY chain consists of more than 410 stores across the UK, 70,000 employees, and 25,000 product lines.¹ Point-of-sale scanners within the stores capture the details of about eight million transactions per week. This corresponds to a weekly addition to the data warehouse of roughly 500 MB.

¹ Source: IBM's web site and *DB2 magazine On-Line*, Vol. 2, No.1, Spring 1997.

The problem faced by organizations like SAFEWAY is that collecting data is one thing; knowing what to do with the data is another. For example, in an analysis of their scanner data, SAFEWAY management discovered that one particular type of cheese was not widely bought—indeed, it was ranked 209th within its product group. However, a second analysis using data mining software uncovered an interesting relationship: people who bought that particular brand of cheese also bought high-margin items (such as premium wines) on the same visit.

The lesson: a superficial level of data analysis led to the recommendation to de-list an unpopular product. However, a deeper analysis indicated that the company's most profitable customers were buying the product as a complement to high-margin products. Thus, it is possible that de-listing the cheese could have costly repercussions.

23.1.2 Tools for data analysis

The wine and cheese example begs the question: does one need hundreds of thousands of dollars worth of data mining software to perform “deep analysis” of data? The answer is: probably not (although it couldn't hurt). What



one does need, however, is plenty of high-quality data organized into an appropriate form for decision analysis (recall [Lesson 22](#)), some basic querying skills, and a solid understanding of how the business works.

23.2 Learning objectives

- exploit the dimensionality of data warehouse data to create crosstab reports
- use constraints and different GroupBy fields to drill down to finer granularity
- create more complex queries to answer specific business questions
- build a pivot table in EXCEL based on a star schema query
- gain experience with basic OLAP concepts such as pivoting and drilling down

23.3 Exercises

In this lesson, you will use the NORTHWIND TRADERS data warehouse you created in [Lesson 22](#) to create more complex queries and perform some *manual* data mining.

23.3.1 Exploiting dimensionality

➔ Modify the star schema query you created in [Section 22.3.6](#) to include *all* the fields from each dimension table.

➔ Save the resulting query as **qryStarSchema** and close it.



We will use **qryStarSchema** as the basis for other queries so that we do not have to keep re-creating the star schema relationships.

➔ Create a new query. When the “Show Table” dialog appears, click the **Queries** tab and select **qryStarSchema**.

➔ Save the query as **qrySalesAnalysis**.

➔ Use the aggregation techniques you learned in [Section 22.3.8](#) to answer the following question:

Which categories of products were selling in which countries in 1994.



If your copy of the NORTHWIND TRADERS database does not have many records from 1994, select a year for which there is more data.



➔ Set the **Format** property of the summed field to “currency”.

➔ Inspect the results, as shown in [Figure 23.1](#).

FIGURE 23.1: Create a query based on the star schema query.

1 Create a new query based on an augmented star schema query.

2 Use calculated field notation to rename the aggregated field to **TotalSale**.

Country	CategoryName	TotalSale
Austria	Beverages	\$13,146.00
Austria	Condiments	\$2,177.42
Austria	Confections	\$661.50
Austria	Dairy Products	\$284.16
Austria	Grains/Cereals	\$1,227.90
Austria	Produce	\$641.88
Austria	Seafood	\$1,031.60
Belgium	Beverages	\$441.60
Belgium	Confections	\$2,462.40
Belgium	Dairy Products	\$1,135.50
Belgium	Meat/Poultry	\$1,248.00
Belgium	Produce	\$1,019.20
Brazil	Beverages	\$2,000.42
Brazil	Condiments	\$1,117.80
Brazil	Confections	\$3,030.10

3 Use the **where** operator to constrain the results (**year** should not show in the results set).

4 Set the **Format** property of the **TotalSale** field to “currency”



If you do not like the default field name **SumOfTotalSale**, you may use a calculated field to rename the field within the query, for example:
TotalSale: SumOfTotalSale (see [Figure 23.1](#)).

23.3.1.1 Creating a crosstab

As [Figure 23.1](#) illustrates, the standard layout of a query results set—with fields as columns and records as rows—is not ideal for viewing multidimensional data. A better way to present two-dimensional relationships is using a **crosstab query**.



➤ Return to design view and select **Query** → **Crosstab Query** from the main menu. A new “crosstab” row should appear in the query definition grid.

➤ In the “crosstab” row, select “Row Heading” under **Country** and “Column Heading” under **CategoryName**.

➤ Select “Value” under **TotalSale**.

➤ View the results, as shown in [Figure 23.2](#).

FIGURE 23.2: Use a crosstab query to organize two-dimensional data.

1 Select **Query** → **Crosstab Query** from the main menu to create a crosstab query.

2 Enter the row heading, column heading and value properties in the crosstab row.

3 Note the format of the crosstab: **Country** is the row heading, **CategoryName** is the column heading, and the intersection is the sum of sales for each country-category combination.



Instead of having field names as column headings, the crosstab query has *values* of the **CategoryName** field as column headings. The cells in the crosstab query show the total sales for each combination of country and product category.



Crosstabs are limited to two-dimensional relationships. Pivot tables (introduced in [Section 23.3.3](#)) introduce tricks for displaying additional information within the constraints of the two-dimensional format.

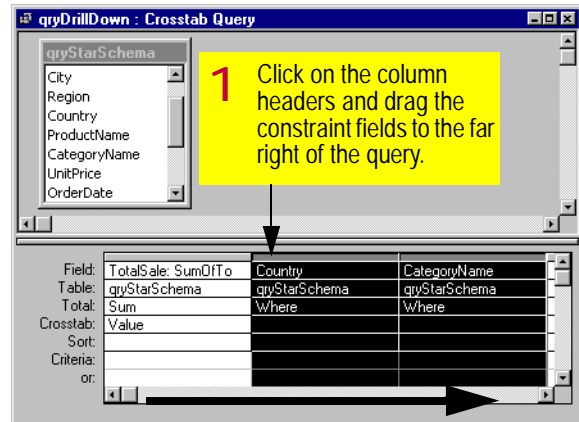
23.3.1.2 Drilling up and down

Taking a close look at the sales analysis query, you may notice that in Venezuela, sales of dairy products is high relative to the sales of products from other categories.¹ By adding some constraints and changing the fields used for grouping, it is possible to explore this issue in greater depth. This process is called **drilling down** to a finer level of granularity.

➔ Use the **File** → **Save As/Export** menu command to save a copy of the query as **qryDrillDown**.

➔ Use the grey bar above the **Country** and **CategoryName** fields in the query definition grid to drag the two fields to the far right of the query, as shown in [Figure 23.3](#).

FIGURE 23.3: Move your constraint fields to the far right of the query.



The order of the fields has no effect on the operation of the query. Moving the fields simply helps you keep track of which fields appear in the crosstab and which are merely used as constraints.

¹ The data in the NORTHWIND TRADERS database is presumably fictitious.



- Use the **where** operator in the totals row to specify constraints on the results set. The purpose of the constraints in this case is to limit the results to the sales of dairy products in Venezuela.
- Project the **city** field into the query. Set the “group by” and “row heading” options.
- Project the **productName** field into the query. Set the “group by” and “column heading” options.
- Run the query as shown in **Figure 23.4**.

FIGURE 23.4: Analyze the sales of dairy product in Venezuela.

City	Camembert Pi	Fløtemysost	Gorgonzola T	Gudbrandsdal	Mascarpone F
Barquisimeto	\$1,126.08			\$432.00	\$1,600.00
Caracas					
I. de Margarita	\$693.60		\$265.62		
San Cristóbal		\$645.00	\$250.00	\$1,569.60	\$128.00

1 Project finer-grained dimension fields into the query definition grid. In this example, we want to see each product in the “Dairy Products” category.

2 Use the **where** operator to constrain the results.

Field:	City	ProductName	TotalSale: SumOfTotal	Country	CategoryName
Table:	qryStarSchema	qryStarSchema	qryStarSchema	qryStarSchema	qryStarSchema
Total:	Group By	Group By	Sum	Where	Where
Crosstab:	Row Heading	Column Heading	Value		
Sort:					
Criteria:				"Venezuela"	"Dairy Products"
or:					

3 View the sales of dairy products in each city in Venezuela.

? Fields with **where** in the “total” row do not appear in the query results and are not used for grouping.



It is possible to use the detailed information to determine which city-product combinations are driving the sales of dairy products in Venezuela.

23.3.2 Manual data mining

In addition to manual drill down, it is possible to use query tools to manually explore more complex relationships in your data. An obvious relationship that one would expect to find in the context of organizations such as SAFEWAY PLC or NORTHWIND TRADERS is **complementarities** between products. Products are economic complements if they are worth more to the consumer together than individually. One means of estimating the strength of complementarity using sales data is to perform a **basket analysis**.

23.3.2.1 Counting the number of matches

In this section, you will create a new query to count the number of times that a pair of products appears in the same order. This query is a bit trickier since we are joining a table with itself.

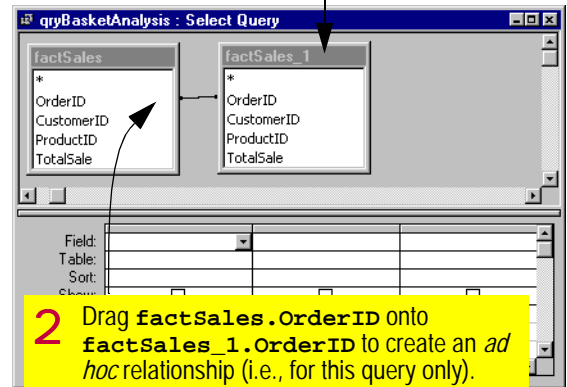
➔ Create a new blank query called **qryBasketAnalysis**.

➔ Project the **factSales** table into the query *twice*.

➔ Create an *ad hoc* relationship between **factSales.OrderID** and **factSales_1.OrderID**, as shown in Figure 23.5.

FIGURE 23.5: Create a query based on the fact table joined to itself.

1 Add the **factSales** table twice in the "Show Table" dialog. This creates a second copy called **factSales_1**.



2 Drag **factSales.OrderID** onto **factSales_1.OrderID** to create an *ad hoc* relationship (i.e., for this query only).



Think for a moment about the meaning of the join in Figure 23.5: The **factSales** table contains data at the granularity of individual order details. Thus every order detail in the **factSales** table is matched




with every order detail in the **factSales_1** table that has the same value for **OrderID**.

➔ Add the **dimProducts** table to the query twice and ensure the relationships between the table are set (see [Figure 23.6](#)).

➔ Enable the totals feature and project the **ProductName** field from both Products tables. These are the **GroupBy** fields.

➔ Project **factSales_1.ProductID** and set the **Count** operator in the “total” row.

 Any field from either table may be used for counting.

➔ Project **factSales_1.ProductID** into the query a second time and set its “total” row entry to **where**. Enter a criterion to ensure that the query does not count the number of time that a product appears with itself.

➔ Set the query to sort by the counted field in descending order. In this way, the products with the highest number of matches will sort to the top.

➔ Examine the results, as shown in [Figure 23.6](#).

There are two things to notice about this query:

1. Each pair of items appears twice since the query counts the number of times Product₁ appears with Product₂ and the number of times Product₂ appears with Product₁.
2. The measure used—number of orders containing both products—is not particularly useful since the volume sold varies with each product.

In the next section, you will create a new query that reports the *relative frequency* with which each product appears with every other product.

23.3.2.2 Getting the total number of orders for each product

To calculate relative frequency, you must first determine the number of orders in which each product appears.

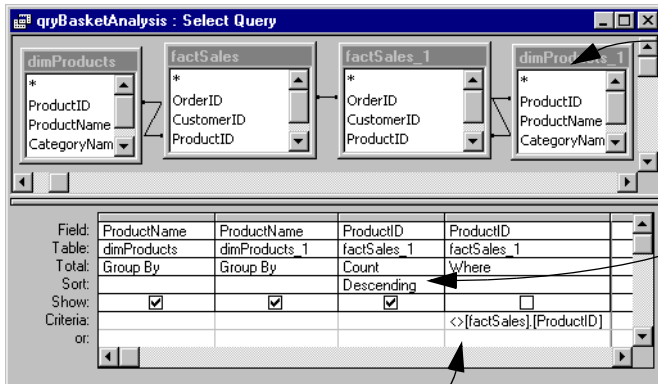
➔ Create a new query called **qryOrdersPerProduct** based on the **factSales** table.

➔ Use the **GroupBy** and **Count** operators to count the number times each product appears in an order.

➔ Verify the resulting query, as shown in [Figure 23.7](#).



FIGURE 23.6: View the results of the basket analysis query.



1 Include the **dimProducts** table twice to provide the names of the products.

2 Set the sort order so that the largest number of matches appear first.

3 Include a criteria to exclude cases in which a product matches itself.

? You will have two entries for each pair of products in this query.

dimProducts_1.Prod	dimProducts_1.Prod	CountOf
Sirop d'érable	Sir Rodney's Scones	8
Sir Rodney's Scone	Sirop d'érable	8
Gorgonzola Telino	Pavlova	7
Pavlova	Gorgonzola Telino	7
Camembert Pierrot	Fløtemysost	6
Fløtemysost	Camembert Pierrot	6
Camembert Pierrot	Pavlova	6
Nord-Ost Matjesheri	Tourtière	6
Gorgonzola Telino	Mozzarella di Giovanni	6
Tourtière	Nord-Ost Matjesherin	6
Pavlova	Tarte au sucre	6
Mozzarella di Giova	Gorgonzola Telino	6
Pavlova	Camembert Pierrot	6

➔ Save and close **qryOrdersPerProduct**.

➔ Create a new query based on **qryBasketAnalysis** and **qryOrdersPerProduct** (see Figure 23.8).

23.3.2.3 Calculating relative frequency

➔ Open **qryBasketAnalysis** in edit view and add **factSales.ProductID** to the far left of the query definition grid. Save it and close the query.

? When creating complex queries, it is sometimes useful to exploit the fact that you can base queries on other queries.



FIGURE 23.7: Count the number of times each product appears in an order.

1 Use the **GroupBy** operator to count the number of records associated with each unique value of **ProductID**.

2 Verify the results.

? Any field can be used for counting.

The screenshot shows a SQL Server Enterprise Manager interface. At the top, a query window titled 'qryOrdersPerProduct : Sel...' displays a table with two columns: 'ProductID' and 'TotalOrders'. The data rows are as follows:

ProductID	TotalOrders
1	38
2	44
3	12
4	20
5	10
6	12
7	29
8	13
9	5

Below the query window is a 'factSales : Select Que' window showing a list of fields: OrderID, CustomerID, ProductID, and TotalSale. At the bottom, a query design grid is visible with the following fields:

Field:	ProductID	TotalOrders	OrderID
Table:	factSales	factSales	
Total:	Group By	Count	
Sort:			
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Criteria:			
or:			

➔ Project both **ProductName** fields into the query.



To keep the semantics of the query clean, ensure that **dimProducts.ProductName** rather than **dimProducts_1.ProductName** is at the far left of the query definition grid.

➔ Create a calculated field called **Frequency** which divides the number of matches by the number of orders for the product.

➔ Right-click the calculated field and set its **Format** property to "percent".

➔ Sort on **Frequency** in descending order.

➔ Verify the results, as shown in [Figure 23.8](#).

Frequency refers to the percentage of orders containing **Product₁** (in the first column) that also contain **Product₂** (in the second column). For example, 40% of the orders that contain "Mishi Kobe Niku" also contain "Röd Kaviar".



As in automatic data mining, interpretation requires a solid underlying knowledge of the business and the ability to construct a story around the putative relationship.

This permits to you decompose the problem into manageable chunks.

➔ Create an *ad hoc* relationship between **qryBasketAnalysis.ProductID** and **qryOrdersPerProduct.ProductID**.



FIGURE 23.8: Calculate the percentage of orders containing the product in first column that also contain the product in the second column.

1 Join the original basket analysis query with the orders-per-product query on the **ProductID** field.

2 Divide the number of matching orders by the total number of orders for the product.

3 Set the format of the field to "percent" and sort in descending order.

40% of the orders containing Mishi Kobe Niku also contain Röd Kaviar.

dimProducts.ProductName	dimProducts_1.ProductName	Frequency
Mishi Kobe Niku	Röd Kaviar	40.00%
Chocolade	Ipoh Coffee	33.33%
Gravad lax	Tarte au sucre	33.33%
Sirup d'érable	Sir Rodney's Scones	33.33%
Genen Shouyu	Camembert Pierrot	33.33%
Chef Anton's Gumbo Mix		30.00%
Mascarpone Fabioli		26.67%
Sir Rodney's Marmalade		25.00%
Aniseed Syrup		25.00%
Grandma's Boysenberry Spre		25.00%
Louisiana Hot Spiced Okra		25.00%
Louisiana Hot Spiced Okra	Gorgonzola Telino	25.00%



Spurious relationships can and do occur in data. Good judgement must be used to distinguish real gems of new knowledge from noise.

sales. This would allow you to perform the type analyses performed by SAFEWAY described [Section 23.1.1](#). This is left as an exercise.

To make the query in [Figure 23.8](#) even more useful, you could order the results by the total



23.3.3 Exploring the data using pivot tables

The data shown in [Figure 23.8](#) might be better shown as matrix using the crosstab functionality of ACCESS. However, a more flexible way to view any type of multidimensional data is to use the pivot table feature of MICROSOFT EXCEL.

23.3.3.1 Preliminaries

The most elegant way to use the pivot table feature is to have EXCEL access the data in the data warehouse directly. In this section, you will set up an ODBC link to the augmented **qryStarSchema** query you created in [Section 23.3.1](#).

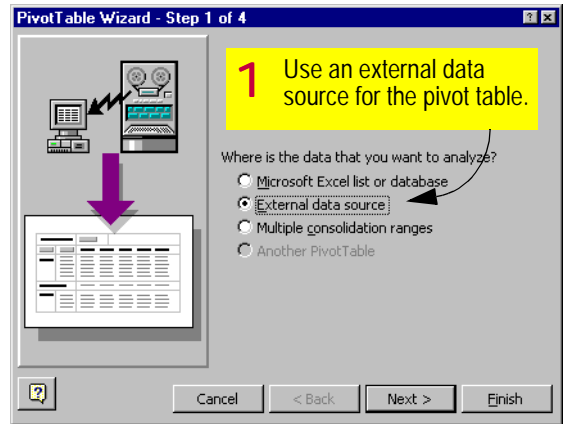
➔ Create a new EXCEL workbook and save it as **OrderEntryPivot.xls**.

➔ Select **Data** → **Pivot Table Report** from the main menu.

➔ When prompted, select "external data source" and "get data" as shown in [Figure 23.9](#).

⚠ If you did not install MICROSOFT QUERY, you will have to re-run the setup program from your OFFICE CD-ROM and add it. See [Section 9.3.3](#) for additional information.

FIGURE 23.9: Create a new pivot table within EXCEL using external data.



2 Click *Get Data* to continue.



23.3.3.2 Creating an ODBC connection

- From the “choose data source” dialog, select “new data source”.
- In the “create new data source” dialog, give your data source connection an easy-to-read name, such as “Order Entry Data Warehouse”.
- Since your data warehouse is a MICROSOFT ACCESS database file, choose the appropriate driver, as shown in [Figure 23.10](#).
- Press the **Connect** button to continue.

In the next dialog, you are asked to enter specific information about the ACCESS file that you are using as a data source. If you were using (say) a client/server ORACLE database instead, you would get a dialog tailored to making a client/server connection. The sequence of dialogs is shown in [Figure 23.11](#)

- In the “ODBC MICROSOFT ACCESS setup” dialog, press the **Select** button.
- Navigate to your **OrderEntryWarehouse.mdb** file and press **OK**.

- In the “Create New Data Source” dialog, select **qryStarschema** as the default “table”.
- Select **OK** twice.
- From the “query wizard—choose columns” dialog, click on **qryStarschema** and press the > button to include all the fields in the pivot table, as shown in [Figure 23.12](#).
- Since there is no need to worry about sorting or filtering the data at this stage, press **Next** until you encounter the **Finish** button.

23.3.3.3 Creating the pivot table

You are now back to the EXCEL pivot table wizard. Although you may not realize it, what you just did is set up a “file DSN” ODBC connection called “Order Entry Data Warehouse”.



- To view or modify your new DSN, you can open the WINDOWS control panel, double click the data sources icon, and select the “file DSN” tab. We discussed the use of file DSNs for ODBC in [Section 8.3.3](#).
- Press **Next** to move on to Step 3 of the pivot table wizard.



- ➔ Drag the **SumOfTotal** “chicklet” into the “data” area in the center of pivot table.
- ➔ Drag **Quarter** into the “column” area and **CategoryName** and **ProductName** into the “row” area.
- ➔ Finally, drag **Country** into the “page” area. The result should resemble [Figure 23.13](#).
- ➔ In Step 4, indicate where you want the new pivot table to be located and press **Finish**.

FIGURE 23.11: Specify the location of the source ACCESS database.

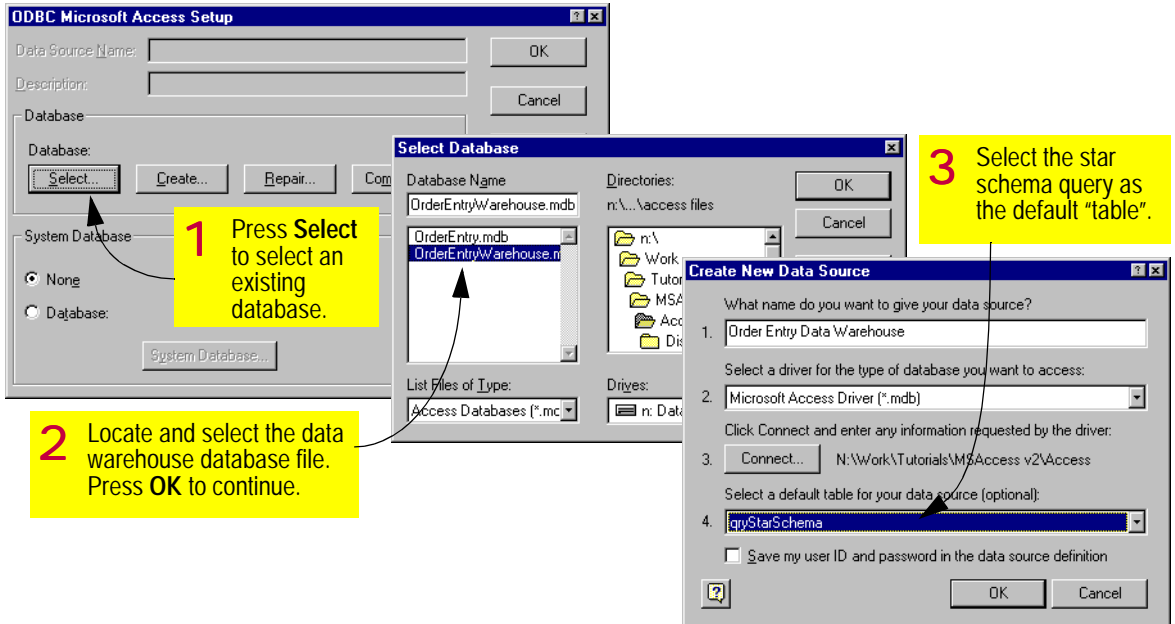
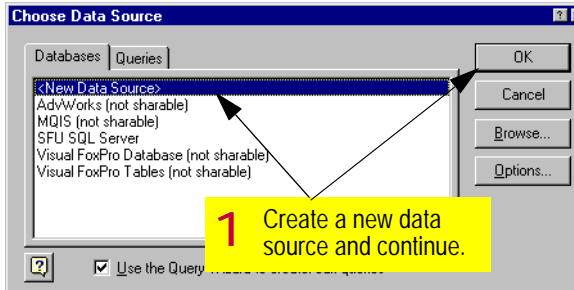
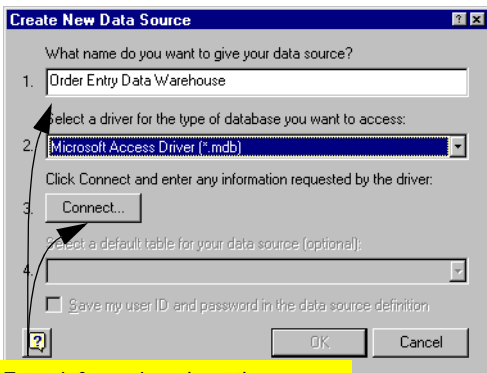




FIGURE 23.10: Create a new data source.



1 Create a new data source and continue.



2 Enter information about the connection and press **Connect**.

23.3.3.4 Altering the format

Before playing with the pivot table, we should format the data cells to show currency:

- Right-click anywhere on the pivot table and select “wizard”. This returns you to Step 3 of the wizard.
- Double-click the **Sum of SumOfTotal** chicklet in the “data” area.
- Press the Number button on the right side (as shown in [Figure 23.14](#)) and select “currency”.
- Press **Finish** to return to the pivot table.

23.3.3.5 Using your pivot table

The best way to learn about pivot tables is to play with one. Clearly, what you have at this stage is similar to a crosstab query. However, unlike a crosstab, you can easily explore your multidimensional data by pivoting around the fact in the middle. For example:

- Drag the **CategoryName** chicklet off the pivot table and drop it. The field should disappear from the pivot table.



Remember, you can always right-click on the pivot table to get back to Step 3 of

FIGURE 23.12: Select the fields from **qryStarSchema** to include in the pivot table.

1 Verify that your new data source appears in the list and press **OK** to continue.

2 Select **qryStarSchema** and press the **>** button to include all the fields in the pivot table.

As discussed in [Lesson 9](#), you can use EXCEL and an existing ODBC connection to access virtually any database. Use **Control Panel** → **Data Sources** to set up or modify data sources manually.

What columns of data do you want to include in your query?

Available tables and columns:	Columns in your query:
<input type="checkbox"/> qryPivotSource	CompanyName
<input type="checkbox"/> qryQuestion	City
<input type="checkbox"/> qrySalesAnalysis	Region
<input type="checkbox"/> qryStarSchema	Country
<input type="checkbox"/> Shippers	ProductName
<input type="checkbox"/> Suppliers	CategoryName
	UnitPrice
	OrderDate

Preview of data in selected column:

Buttons: **Preview Now**, **< Back**, **Next >**, **Cancel**

the wizard. In this way, you can easily change the dimension fields that appear in the row, column, and page areas.

- ➔ Drag the **Country** chicklet down beside **Quarter** but then drag **Quarter** to the page area (at the top-left of the pivot table).

- ➔ Click on the **Quarter** drop-down list and select "Q4". This limits the data in the table to orders placed in the fourth quarter.

- ➔ Double-click on any fact in the table to drill down on specific order detail records from the underlying **qryStarSchema** data source.



FIGURE 23.13: Drag fields to the appropriate location on the pivot table grid.

1 Drag **sumOfTotal** onto the data area. The other areas "pivot" around the data area.

2 Drag **Quarter** onto the column area.

3 Drag **CategoryName** and **ProductName** onto the row area.

4 Drag **Country** onto the page area.

Construct your PivotTable by dragging the field buttons on the right to the diagram on the left.

? The dimension and fact data available for use in your pivot tables are shown as "chicklets" (like the chewing gum).



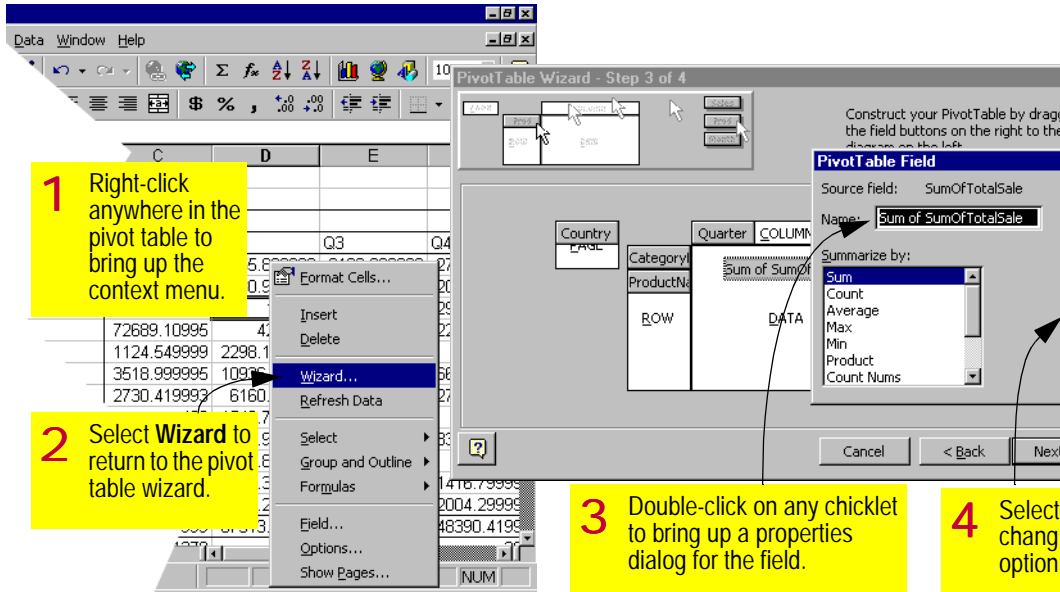
This drill down feature is a bit clumsy since you create a new worksheet every time you double-click a number. You can right-click on the sheet's name tab to delete it once you have examined the detailed data.

23.4 Discussion: Commercial OLAP tools

There are a number of commercial OLAP tools on the market (e.g., ARBOR SOFTWARE'S ESSBASE, COGNOS POWERPLAY, and ORACLE'S EXPRESS SERVER) that are designed to facilitate interactive analysis with large amounts of multidimensional data.

Such products have clear advantages over EXCEL in terms of scalability and ease of use. For

FIGURE 23.14: format the pivot table data as currency.



example, in EXPRESS SERVER, you simply double-click on any value or bar chart to drill down to a finer level of granularity. The software dynamically re-queries the data warehouse, thereby eliminating the manual process you performed in [Section 23.3.1.2](#). Of course, EXCEL has this capability in the virtue of costing much less than \$3,000 per user.

23.5 Application to the assignment