

## Session V

This week we will create Functions, Procedures and Reports. Sorry, but this session is long so give yourself time to complete!

The next set of aggregate functions is referred to as **Domain Aggregate Functions**. These functions provide statistical information about a field or domain. The domain is the set of values that a particular field has.

<i>Function</i>	<i>Description</i>
<b>DLookup()</b>	Returns the value in the specified field
<b>DMin()</b>	Returns the minimum value in the specified field
<b>DMax()</b>	Returns the maximum value in the specified field
<b>DFirst()</b>	Returns the value from the specified field from the first physical record
<b>DLast()</b>	Returns the value from the specified field from the last physical record
<b>DAvg()</b>	Returns the average of the values in the specified field
<b>DSum()</b>	Returns the sum of the values in the specified field
<b>DCount()</b>	Returns the count of the non null values in the specified field

The syntax for all the functions is as follows:

DFunction(“expression”, “domain”, “criteria”)

DCount (“Job Title”, “Faculty”, “‘Job Title’ = Instructor “)

In the above example each of the sections must be enclosed in double quotes (“).

Because Job Title is not one word it must be enclosed in single quotes.

You will find these functions extremely helpful in programming in Access.

### Reports:

<i>Section</i>	<i>Description</i>
<b>Report Header</b>	Contains information to be printed once at the beginning of the report. The report title is displayed in this section.
<b>Page Header</b>	Contains information to be printed at the top of each page. The column headings are displayed in this section.
<b>Detail</b>	Contains the records of the table. The field columns widths are the same as the column widths set in the table design.
<b>Page Footer</b>	Contains information to be printed at the bottom of each page such as the date and page number.
<b>Report Footer</b>	Contains information to be printed at the end of the report. The Report Footer section currently contains no data.

Don't include expressions in the Page Footer!

### Report Event Properties:

<i>Events</i>	<i>Description</i>
<b>Open Event</b>	The open event is the first event that occurs in a report, before the report begins printing or displaying.
<b>Close Event</b>	The close event occurs as a report is closing, before the deactivate event occurs.

<b>NoData Event</b>	If no records are retrieved in a report you can use the nodata event to cancel the report.
<b>Page Event</b>	The page event occurs immediately before the formatted page is sent to the printer.
<b>Error Event</b>	If an Access database engine error occurs when the report is formatting or printing, the error event is triggered. This error usually occurs if there is no record source for the report.
<b>Format Event</b>	The format event happens after access has selected the data to be included in the report section but before it formats or prints the data. With the format event, you can affect the layout of this section or calculate the results of data in the section, before the section actually prints.
<b>Print Event</b>	The code in the print event executes when the data formats to print in the section but before it actually gets printed.
<b>Retreat</b>	Occurs when Access must back up past one or more report sections on a page in order to perform multiple formatting passes.

### **Adding blank rows to a report:**

Some reports are easier to read if blank rows are inserted at certain intervals. Access does not provide a method to insert a blank row in the middle of the detail section. However, you can use VBA code and the properties of controls and sections on a report to insert blank rows.

To add blank rows to a report, you insert a blank unbound text box in the detail section of the report below the other text boxes, and name it "txtSpacer." Then you set the **Can Grow** and **Can Shrink** properties of the txtSpacer text box and the detail section to Yes. Remember that the Can Grow and Can Shrink properties allow controls and sections on a report to grow vertically and to shrink vertically when necessary. Wherever you want one of the blank row to print, txtSpacer needs to contain " ", which is a space between quotation marks. This causes the text box to print but nothing will be visible inside it. You can write VBA code to insert a blank row after a certain number of records. The VBA code tests which record is being laid out on the page. If you want a blank row every fifth record, for example, the code tests to see if a record number is evenly divisible by five. If it is divisible by five, then a blank row is inserted after it.

### **Using and If... Then... Else Statement:**

An **If... Then... Else Statement** conditionally executes a group of statements in VBA depending on the results of an expression. This type of conditional statement or control structure will be discussed in the future sessions.

A **Control Structure** is a series of VBA statements that work together as a unit. A **VBA statement** is a unit that expresses one kind of action, declaration, or definition in

complete syntax. A VBA Expression is a combination of keywords, operators, variables, and constants that yield a string, number, or object.

**Operators** are used to perform arithmetic calculations, perform comparisons, combined strings, and perform logical operations. You have been using the Ampersand (&) operator to create expressions in queries and reports that combine fields with other fields and strings of text.

The **mod operator** is arithmetic operator used to perform modulo division, which divides two numbers and returns only the remainder. For example, 10 mod 5 return 0 because 10 divided by five equals 2 with a remainder of 0; 10 mod three returns one because 10 divided by 3 equals 3 with the remainder 1.

### **Reading Assignments:**

#### **Writing Function Procedures**

Korol Pg. 86

Hands-On 4-1 **Writing a Function Procedure with Arguments**

Hands-On 4-2 **Executing a Function Procedure from the Immediate Window**

Hands-On 4-3 **Executing a Function Procedure from a Subroutine**

#### **Specifying the Data type for a Function's Result**

Korol Pg. 88

Hands-On 4-4 **Calling a Function from a Procedure**

#### **Passing Arguments by Reference and by Value**

Korol Pg. 90

Hands-On 4-5 **Passing Arguments to Subroutines and Functions**

#### **Using Optional Arguments**

Korol Pg. 92

Hands-On 4-6 **Using Optional Arguments**

#### **Converting Macros to VBA Code**

Korol Pg. 911 - 914

**Using the IsMissing Function** Korol Pg. 93 - 94

#### **Built-in Functions**

##### **Using the MsgBox Function**

Korol Pg. 94

Hands-On 4-7 **Formatting the Message Box**

Hands-On 4-8 **Using the MsgBox Function with Arguments**

#### **Returning Values from the MsgBox Function**

Korol Pg. 102

Hands-On 4-9 **Returning Values from the MsgBox Function**

### **Using the InputBox Function**

Korol Pg. 103

Hands-On 4-10 **Using the InputBox Function**

### **Converting Data Types**

Korol Pg. 106

Hands-On 4-11 **Converting Data Types**

**Saving Reports in the PDF or XPS File Format** Korol Pg. 766

**Event Properties** Korol Pg. 695 - 730

Professor's comments: Unfortunately, the project is long! However, as a programmer you will be expected to create numerous reports for management so I tried to give you a variety of areas to increase your competency.

If you find mistakes and you probably will please be patient and let me know so I can correct for other students. I will give you some additional points for pointing them out.

---

## **Project**

We have completed most of the switchboard forms for the MovieCam Technologies database, and now need to turn your attention to reports. Just as you can create a form template, you also can design the template for the reports in a database. A report template has already been created for the MovieCam database. The next step is for you to use that template to develop the various reports Amanda and the other managers such as Martin Woodward have requested. Such as a report on employees, categorized by supervisor. The report should contain data such as name, department, hourly rate, and whether the employee has a completed timecard. Each supervisor can then compare the report data to the timecards the employees turned in at the end of each week.

The product managers need a report that contains detailed information on the number of hours and the total dollars spent to date on each job. This report should include a summary of hours and dollars for each job.

Daniel Jenkins, human resources manager, has requested two reports. One report should list active employees, and the other should list terminated employees. Daniel will then compare these reports to the quarterly reports generated by the ACC 90 payroll module. Both of the reports for Daniel should contain employee number, employee name, job title, labor category, whether employees are hourly or salaried, and hourly rate of pay.

You must consider one more requirement before you design the databases reports. In a staff meeting earlier in the week, it was announced that MovieCam technology is being purchased by TechCam Inc., a large corporation located in Southern California. It is uncertain whether the MovieCam technology's name and logo will be retained once a buyout is completed. Amanda suggests that, from this point on, you design forms and reports that can be updated easily in the event the company name changes.

Reports are often the only component of a database that upper management or customers ever see. Most database administrators will agree that reports generally require a great deal of maintenance and ongoing development. That's because management typically request new reports or modification to existing reports on a regular basis. For this reason, you should strive for efficient report design. Report templates and report masters can help you achieve that efficiency.

### **Creating the Table to Contain the Company Name and Other Data:**

#### **Step 1**

Start access and open the Session 5 Movie database.

#### **Step2**

In the Ribbon select the **Create** tab and then click on **Table Design**.

#### **Step 3**

Create the following fields: **CompanyName**, **Address**, **City**, **State**, **ZipCode**, **Telephone**, and **Fax**. All the fields' data types are short text.

#### **Step 4**

Change the caption property for the **CompanyName** field to Company Name and the caption property for the **ZipCode** field to zip code.

**Step 5**

Set the input mask for the field **Telephone** and **Fax** to **telephone** and **ZipCode** to **zip code**.

**Step 6**

Switch to datasheet view.

A message box opens, asking if you want to save the table. Answer Yes and name the table **tblCorporate** and then click the **OK** button.

A second message box opens, asking if you want to define a primary key. You want Access to automatically add a primary key field.

Note: Earlier on you learned that the widely accepted naming conventions specifies plural names for tables. However, because this table will contain only one record, you use a single name.

**Step 7**

Click the **Yes** button.

**Step 8**

Open **tblCorporate** in datasheet view. The table appears with a new primary key field. Access automatically added a new field named ID to the table, assigned the field the Auto Number data type, and sets the field as a primary key. Press the **Tab** key to move the insertion point to the company name field.

**Step 9**

Add the record shown below to the table. As you begin to type the company name, notice that access assigns the value one to the primary key field named ID.

**Step 10**

Close the table.

tblCorporate								
ID	Company Name	Address	City	State	Zip Code	Telephone	Fax	Add New
1	MovieCam Technologies	1521 5th Street	Windsor	CA	94592	(707) 555-1111	(707) 555-1112	
* New)								

You will use the domain aggregate function, Lookup, to look up the report master objects.

In this example, the Dlookup function looks up and returns a company name field value from the tblCorporate table when the ID field value is equal to one. This is the expression that you will use in MovieCam reports. To change the company name or address on reports, all you have to do is change the data in the underlying table, and the changes are automatically reflected in the reports.

Before you complete the report master, you want to set the dmrptTemplate report as a default report template for Access.

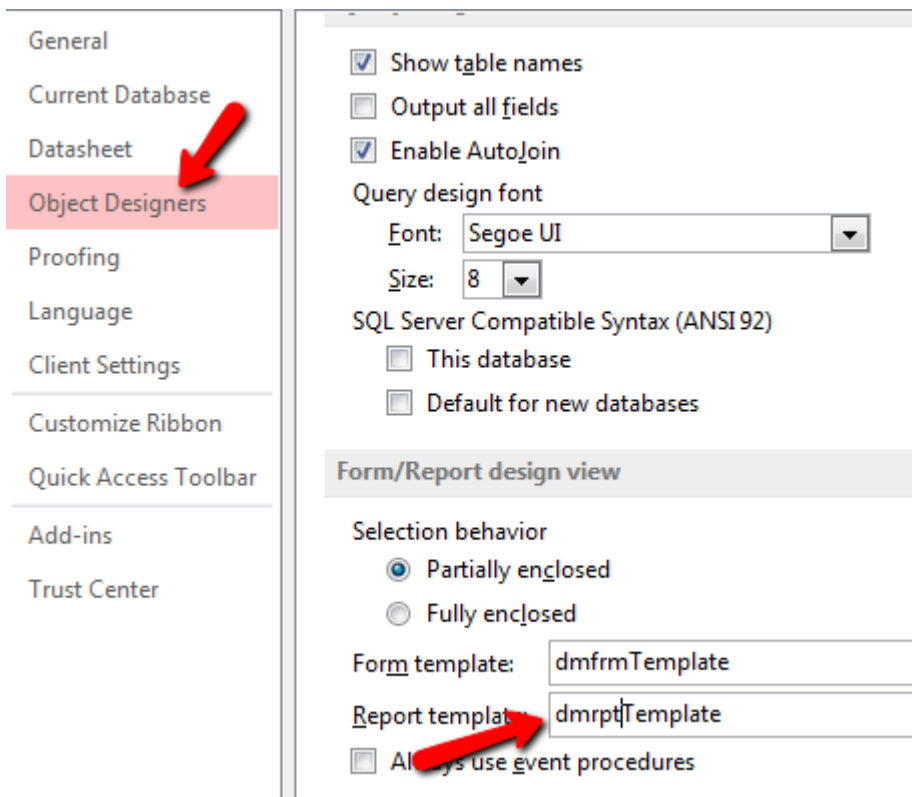
**Setting the dmrptTemplate report as the default report template:**

**Step 1**

Open the **File Tab** and select **Options** in the lower right corner of the drop down list box.

**Step 2**

Select **Object Designer** in the left side of the **Access Options** window:



### Step 3

Type in **dmrptTemplate** in the Report Template

### Step 4

Click OK to close out the access Options Window

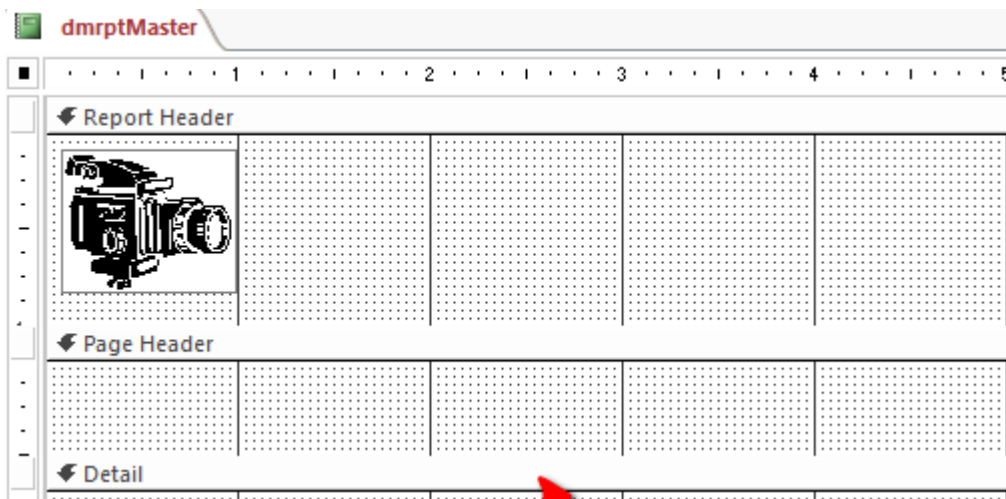
Now you will work on completing the report master.

### Completing the report master with the DLookup function:

#### Step 1

Open the dmrptMaster report in design view. The report looks like the one shown below:





### Step 2

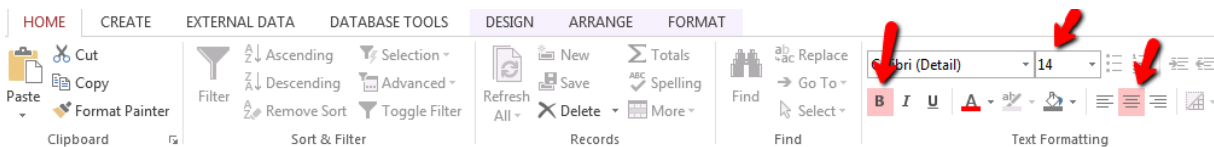
Select the Design tab in the Ribbon and click on **text box** and drag to create an unbound text box control approximately 4" wide and 3/10" tall in the Report Header section, located at the 1 1/2 "mark on the horizontal ruler and at the 1/4 "mark on the vertical ruler.

### Step 3

Type =DLookup("CompanyName","tblCorporate","ID=1") inside the **text box**, and then press the Enter key.

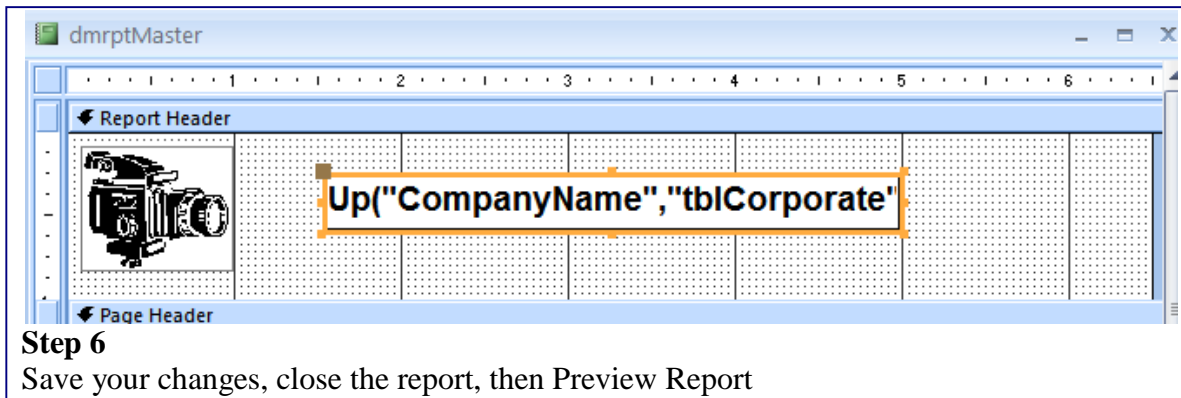
### Step 4

Change the font size to **14**, change the font style too **Bold**, and then change the text alignment to **Center**.



### Step 5

Right-click in the text box and choose **Special Effect** then the **Shadowed** button.



Now that the report master is complete, you can use it as a basis for new reports. To test the lookup function, you entered change the company name in tblCorporate to TechCam Incorporated, view the report, and then change it back again to MovieCam Technology.

Next we will work on Martin's employees-by-supervisor report.

The first report that Martin requested for the product managers is a list of employees grouped by supervisor. He asked that the supervisor's name and Department appear at the beginning of each group of employees that report to that supervisor. He also wants the group to print on a separate page.

The rptEmployees-By-Supervisor report will contain data from the employees table and the departments table. This report should contain a Group Header section based on the employee number of the supervisor, in a Group Footer section. Using the Group Header section to meet Martin's requirements poses some interesting challenges. If you set the header to group on the DeptNo field, the report will group employees by their respective departments. In some cases, the supervisors' Dept.Name is different from the employees. For example, Carolyn Valadez, supervisor in the Administrative and Accounting department, supervises Ann Garcia, who is in the Accounting department.

The other issue you need to resolve is how to show the supervisor's name in the reports Group Header section. If you add the FirstName and LastName fields from the tblEmployees table to the Group Header and Report Header, then each employee's name will appear in the header instead of only the supervisors.

If the records for supervisors were stored in a separate table, this would be a simple task. Because they are not, your solution is to create a query that actually joins the employees table to itself. Joining a table to itself in a query is referred to as a **Self - Join**.

A self - join can be either an inner or outer join. Remember from earlier sessions that an inner join is the default join type and is the one which all the records with equal joined field values are displayed. Because each employee has a supervisor, this is the type of join you will use for this query.

## Creating a Query with a Self-Join:

### Step 1

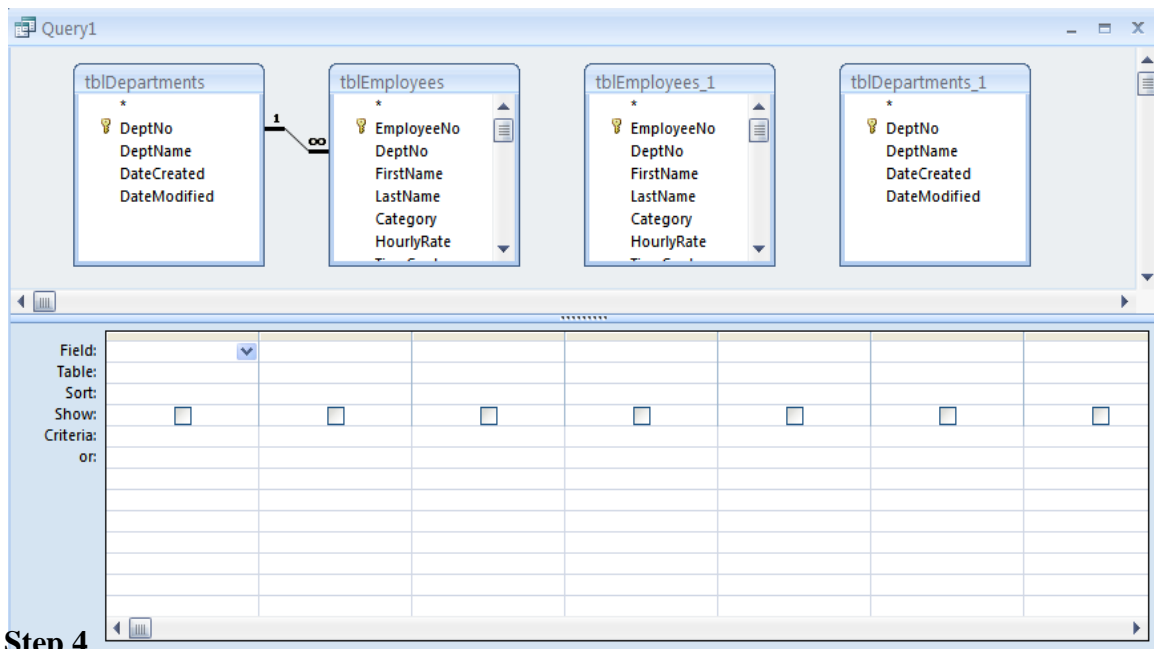
Create a new query in design view, click the **tblDepartments** table in the Show Table dialog box.

### Step 2

Click the **tblEmployees** table in the Show Table dialog box, and then click the **Add** button twice to add two copies of the tblEmployees table to the query window.

### Step 3

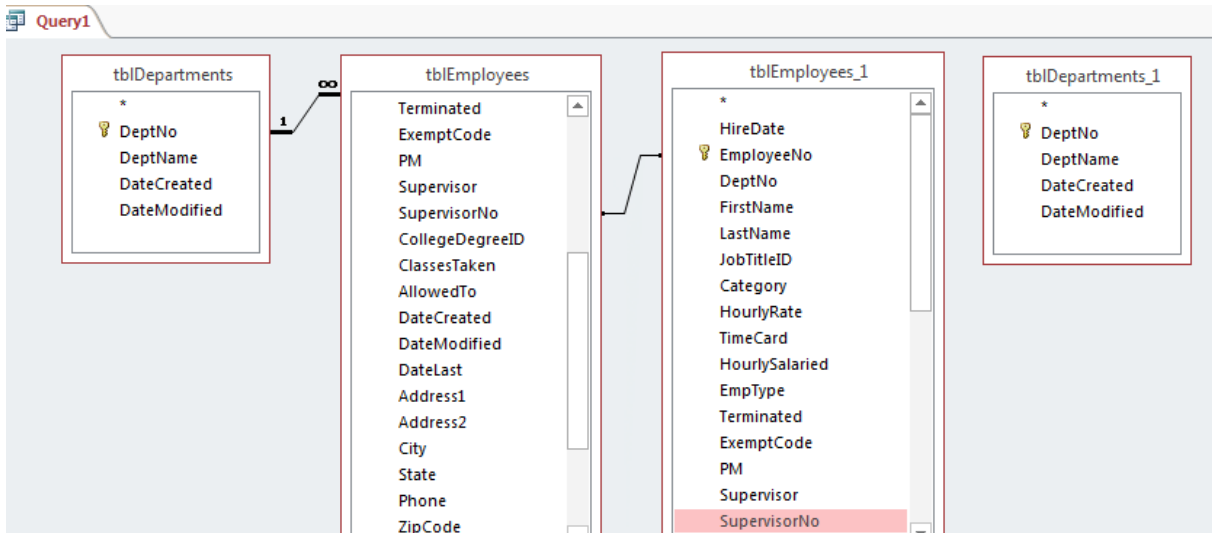
Click the **tblDepartments** table again, click the **Add** button, and close the Show Table dialog box. Now you have four tables in the query window, as shown below: the two tables on the left will be used to display data about the employees, and the two tables on the right will be used to display data about the supervisors.



### Step 4

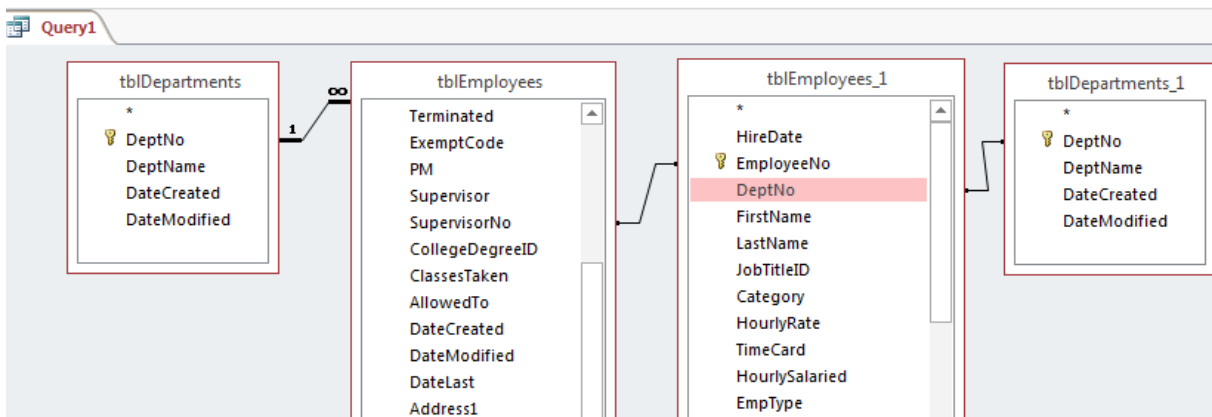
The only link should be between **tblDepartments.DeptNo** and **tblEmployees.DeptNo**. Delete any other links by selecting the link and pressing the Delete button.

In the far left **tblEmployees** table, click the **SupervisorNo** field (scroll down the field list, if necessary), and then drag it to the **EmployeeNo** field in the other **tblEmployees** table (**Employees\_1**) to create the join as shown below:



### Step 5

Drag the DeptNo field in the tblEmployees\_1 table to the DeptNo field and in the tblDepartments\_1 table (the far right departments table) create a join.



### Step 6

In this next step I notice that Access bombed out on me. So as to not lose your work save the query as **qryEmployeesBySupervisor** right now and then reopen in design view.

(See the problem comment below before beginning) Right-click the title bar of the field list for the tblEmployees\_1 table, click Properties on the shortcut menu, and then type tblSupersors in in the alias text box. Changing the name of the field list makes it clear which fields contain the supervisor data and which fields contain the employee's data when you rebuild the report.

(Problem: I noticed that Access 2007 didn't like my changing the Alias and then closing the properties sheet. If this problem occurs using Access 2016, then before closing Properties Sheet click on a table where the alias wasn't changed and then return and click on the table to make my changes then close the property sheet.)

### Step 7

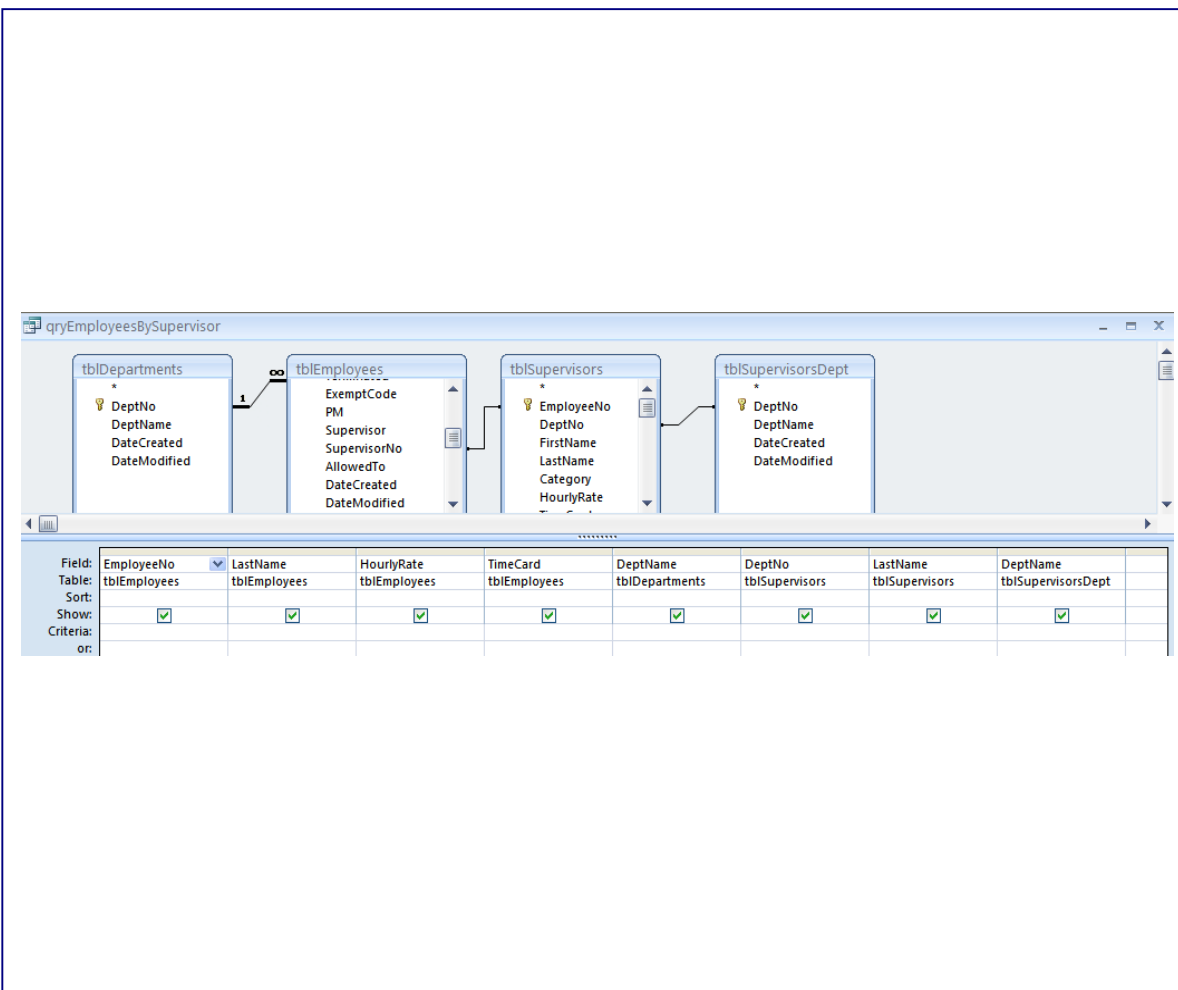
Click the title bar of the tblDepartments\_1 table to display properties, type tblSupervisors Dept in the alias property text box, and then close the property sheet.

### Step 8

Add the **EmployeeNo**, **LastName**, **HourlyRate**, and **TimeCard** fields (in this order) from the **tblEmployees** table to the design grid, and then add the **DeptName** field from the **tblDepartments** table to the design grid.

### Step 9

Add the **DeptNo** and **Last Name** fields from the **tblSupervisors** table to the design grid, and then add the **DeptName** field from the **tblSupervisorsDept** table to the design grid. Scroll to the right so you can see the column containing the last field you just added. Your Query should appear similar to the one below: depending upon where you stopped scrolling within the various field list and if you resize any of the field lists or the query window.



**Step 10**

Run the query to test it. There should be 13 records in the query results.

**Step 11**

Save and close the query.

**Creating the Employees by Supervisor Report:****Step 1**

Right-click on **dmrptMaster** report, click **Copy** on the shortcut menu, right-click on an empty area and click **Paste** on the shortcut menu. The Paste As dialog box opens.

**Step 2**

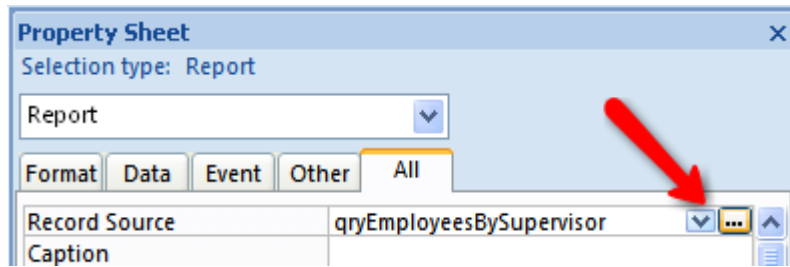
Type **rptEmployeesBySupervisor** in the report name text box, and then press the **OK** key. Recall that because the OK button is highlighted by default, pressing the **Enter** key is just like clicking the OK button.

**Step 3**

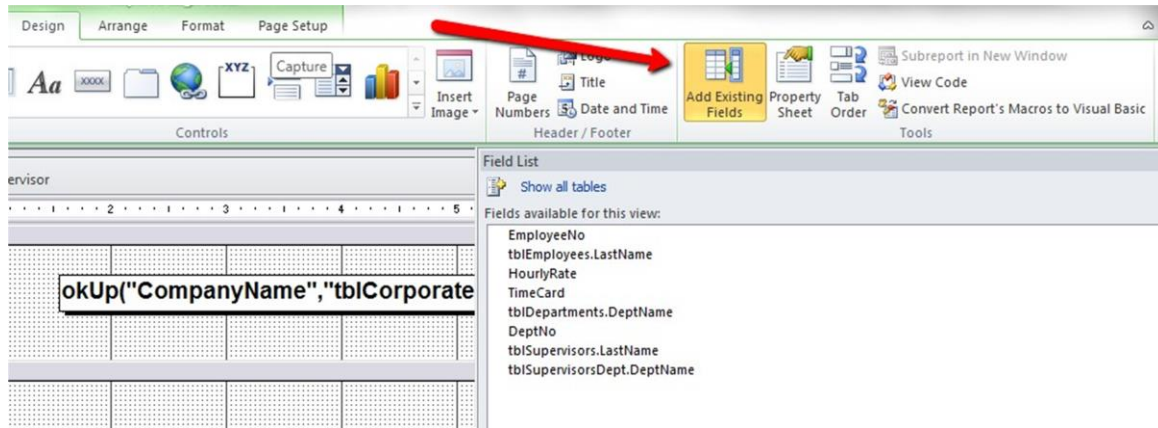
Open the new report in design view, and then Right-click to select its property sheet.

**Step 4**

Click the **All** tab if necessary, click the **Record Source** list arrow, and then select **qryEmployeesBySupervisor**. Close the Property sheet.



In the Design tab in the Ribbon click on **Add Existing Fields**. Note that any fields with the same name are preceded by the table name.

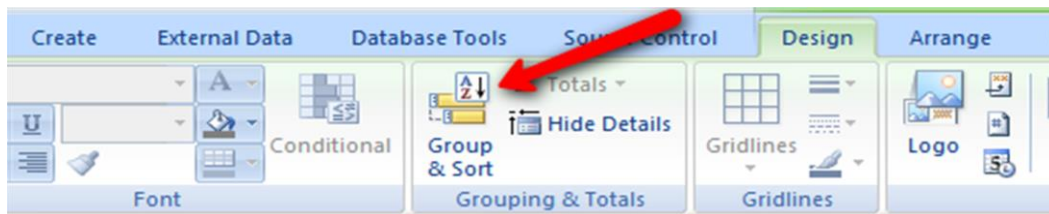


### Step 5

Close the Field List

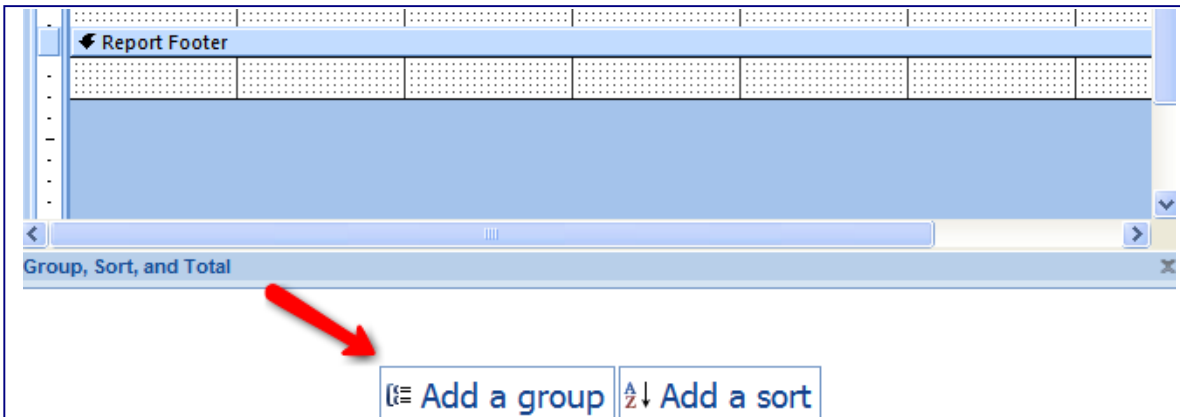
### Step 6

Click the **Group & Sort** button in the Ribbon.



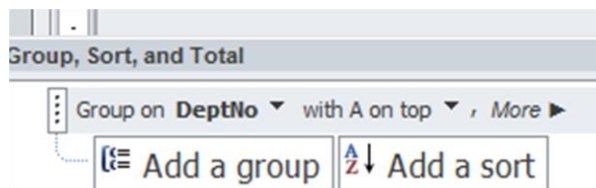
### Step 7

Click the **Add a Group** button (See below).



### Step 8

Click the **DeptNo**. This is the department number of the supervisor!



### Step 9

In the Group properties section of the Sorting and Grouping dialog box, change both the Group Header and Group Footer properties for the DeptNo group to **with a header section** and **with a footer section**. Do not change any other properties. Access adds a Group Header section, named DeptNo Header, and a DeptNo Group Footer section, named DeptNo Footer, to the report.

The Group On property lets you group on prefix characters, such as the first letter in a company name, rather than on the entire field.

The group interval property is used to indicate the number of characters you want to group on if you select the Prefix characters setting for the Group On Property.

The Keep Together property is used to keep a group together so it prints on one page. Don't change the Keep Together property because in a future step, you will set each group in this report to begin on a new page.

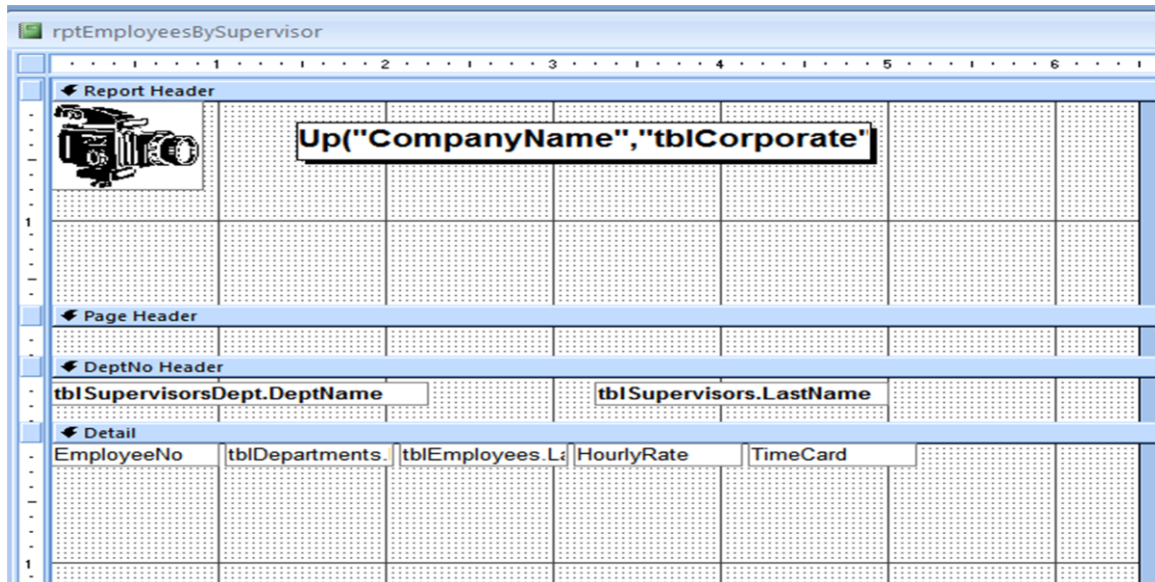


Close the **Sorting And Grouping** dialog box by clicking on the icon . (Be careful not to delete what you have just done)



### Step 10

Using the following image as a guide, from the Field List drag the **tblSupervisorDept.DeptName** field and the **tblSupervisor.LastName** field to the DeptNo header section, and then drag the **EmployeeNo**, **tblDepartments.DeptName**, **Employees.LastName**, **HourlyRate**, and **TimeCard** fields (in that order) to the detail section.



### Step 11

Close the field list at this time. Leave the report open in Design view.

You can use the **Can Grow** and **Can Shrink** properties of sections or controls on printed or previewed forms and reports. Setting the Can Grow property to Yes enables the section or control to grow vertically so that all the data is visible. This is useful when the data in a few records is much longer than the data in the majority records. Setting the Can Shrink property to Yes shrinks the section or control vertically to avoid printing blank lines.

When you set the Can Grow property to Yes for control in a section, Access automatically sets that report sections Can Grow property to yes. However, if you change the Can Grow property of the section back to No, this section does not, in turn, alter the Can Grow property of any controls residing within that section. If a control in a section is set to be able to grow the entire section needs to be able to grow as well, because if it does not, then the control can grow to print more data than the section can display. If the controls in the section in which they reside are both set to grow or expand, then as controls grow vertically to display all of their data, the section grows to accommodate this. By default, Access automatically changes the section's Can Grow property to Yes if a control within that section is set to grow. This is the only scenario

where Access automatically updates the Can Grow property of a section based on a setting changing for a control within that section. This automated property setting behavior does not apply for the Can Shrink property of the section.

The only time you would not want a section to grow is when you allow the controls within it to grow is if the finished report absolutely must fit within a certain page count or perhaps it has been designed to match some pre-existing form, such as a medical or government form. In such situations, you may now want to set controls to expand either, but rather size them to take up a fixed amount of space, just as the entry spaces for information on a preprinted form.

You need to make some formatting changes to the reports design, and change the Can Grow property for the tblDepartments.Name field to Yes so that it wraps to two lines. Note: I think it's a good idea to make up a backup copy of your report at this time. In case you mess up you can start again without losing any more work.

### **Formatting the employees by supervisor report:**

#### **Step 1**


Rt-click the Page Header in the design grid and select the header/footer option this will remove the Page Header. Size the DeptNo Header section so that it is approximately 1" tall,

#### **Step 2**

Add a **label** below the MovieCam Technology text box in the Report Header, type **Employees By Supervisor**, change the font size to **14**, and change the alignment to **Center**. Expand the label control to be approximately as wide as the text box above it, and resize the label control vertically, if necessary, to ensure that all the text in the label is clearly visible.

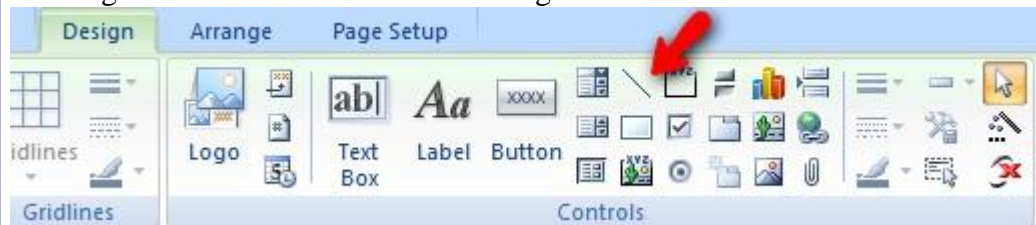
#### **Step 3**

Drop the Detail section title bar to allow the placement of labels. Insert field labels in the DeptNo Header section as shown below, pressing **Ctrl + Enter** to create a new line in a label. Make the labels the same width as the text boxes they represent, and line them up manually as close as you can as shown below: you are inserting the labels in the DeptNo Header section instead of the page Header section because, in a future step, you will set each group to begin on a new page.

tblSupervisorsDept.DeptName Header					
tblSupervisorsDept.DeptName			tblSupervisors.LastName		
Employee No	 Employee Departments	Employee Last Name	Hourly Rate	Time Card	
Detail					
EmployeeNo	tblDepartments.	tblEmployees.L:	HourlyRate	TimeCard	

#### Step 4

Click the **line** button in the Design tab in the ribbon, press and hold the **Shift** key, and drag below the labels to create a straight horizontal line.

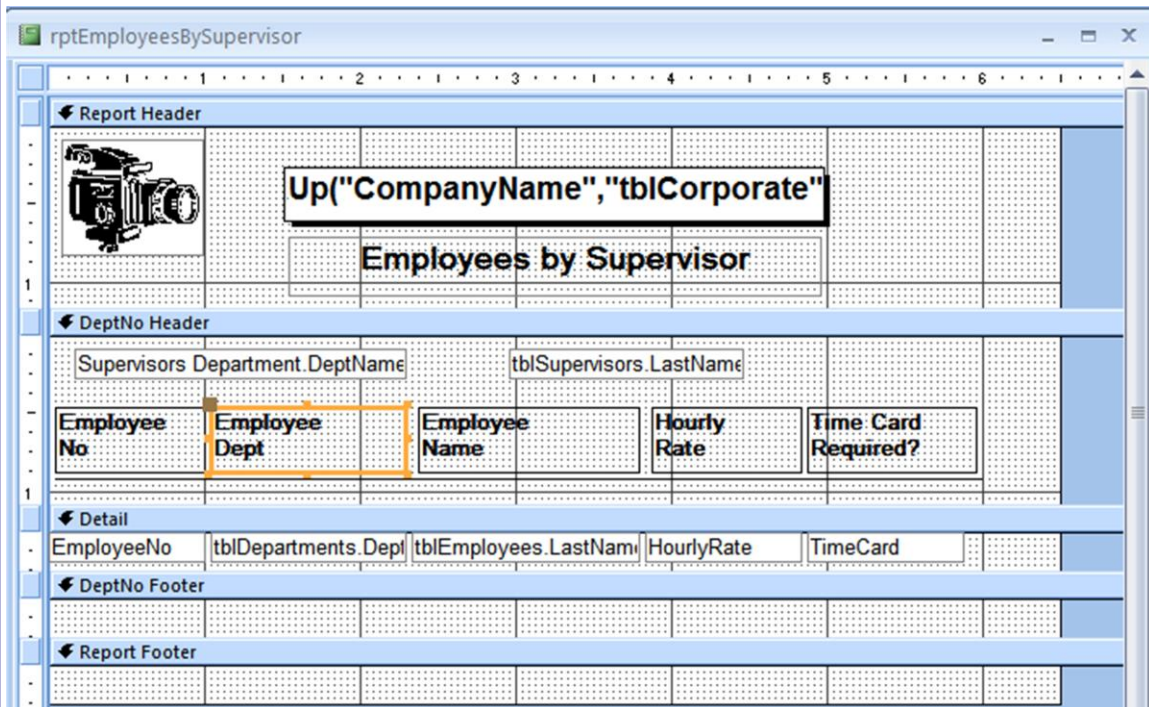


#### Step 5

Align the line to the bottom of the labels control. The line should start at the left edge of the Employee No label and end at the right edge of the Time Card Required label. This will present a more pleasing appearance and help to visually set the labels off from the data that will print on the report.

#### Step 6

Scroll to the bottom of the report window so you can more easily work with the detail section, drag the detail section of the report up so that it is approximately a 1/4" tall, and then adjust the vertical placement of the text box controls in the detail section as shown below: (there should be no space between the text box controls and the detail section bar.)



#### Step 7

Left align the labels in the DeptNo Header section with the text boxes in the detail section, and adjust their width to be the same as each of their corresponding text boxes. (If you have not already done so) Change the font size of the labels in the DeptNo Header section to **9**. The label should already be bold because default labels in the template are in

bold

### Step 8

Center the alignment of the **Employee No** and **TimeCard Required?** Labels and their corresponding text boxes. The rest of the labels and text boxes should be aligned left.

### Step 9

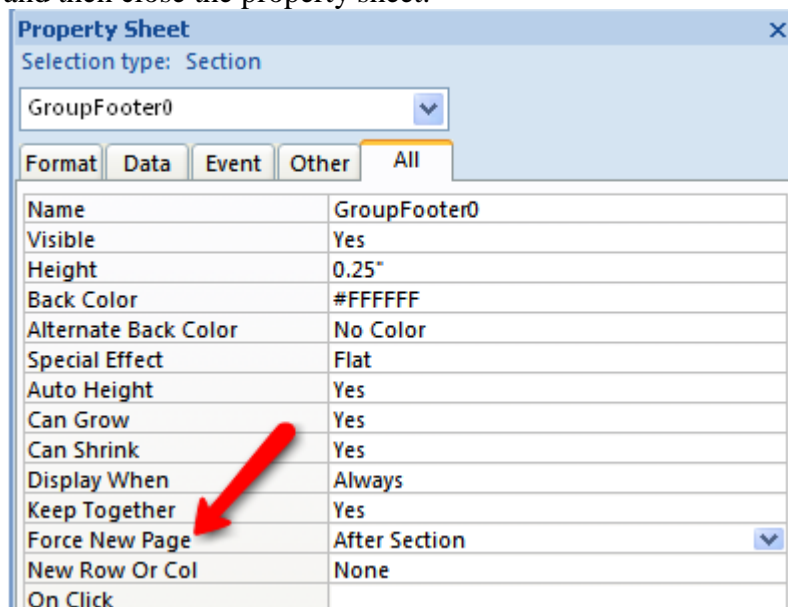
Click the **tblDepartments.DeptName** text box in the detail section of the report, open the properties sheet for the text box control, and then change Can Grow property to **Yes**.

### Step 10

Size and aligned the **tblSupervisorsDept.Dept.Name** and **tblSupervisors. Last name** field in the DeptNo Header section as shown below: and then apply the **Bold** style to both fields. Leave the font size at 10. Note that you could also use a property sheet to set the width for the tblSupervisorDept.DeptName control to **2.25** “and the width of the tblSupervisors.LastName to **1.75**”.

### Step 11

Click the **DeptNo Footer** section, change the Force New Page property to **After Section**, and then close the property sheet.



The screenshot shows the 'Property Sheet' window for 'GroupFooter0'. The 'All' tab is selected. A red arrow points to the 'Force New Page' property, which is set to 'After Section'. Other properties like 'Can Grow' and 'Keep Together' are also visible.

Property Sheet	
Selection type: Section	
GroupFooter0	
Format Data Event Other All	
Name	GroupFooter0
Visible	Yes
Height	0.25"
Back Color	#FFFFFF
Alternate Back Color	No Color
Special Effect	Flat
Auto Height	Yes
Can Grow	Yes
Can Shrink	Yes
Display When	Always
Keep Together	Yes
Force New Page	After Section
New Row Or Col	None
On Click	

### Step 12

Create a text box in the DeptNo footer section, and set its control source property to =**count ([EmployeeNo])**, or simply type the expression into the text box.

### Step 13

Create a label to the left of the text box, and then type **Total Employees** in the label control, size and alignment of controls as shown below:

rptEmployeesBySupervisor

Report Header

Up("CompanyName","tblCorporate")

Employees by Supervisor

DeptNo Header

Supervisors Department.DeptNam tblSupervisors.LastName

Employee No Employee Dept Employee Name Hourly Rate Time Card Required?

Detail

EmployeeNo tblDepartments.Dept tblEmployees.LastName HourlyRate TimeCard

DeptNo Footer

Total Employees =Count([EmployeeNo])

Report Footer

#### Step 14

Preview the report. (Adjust the size of the fields to show all the data) Use the navigation buttons at the bottom to move between pages. Note that the logo, company name, and title appear only on the first page. This is because these items are in the Report Header section of the report.

Problem? If you preview the report and there are blank pages, it may be because of the width of the report.

#### Step 15

Return to design view, and then save your changes.

Because the report pages will be distributed to various supervisors, the data in the Report Header section should be moved to the DeptNo Header section; and it will appear on each page of the report. As a result, you will no longer need the Report Header section. Also, you need to include criteria in the query so that only active employees are included in the report.

All the test records in the tblEmployees table include active employees. However, the final data will include terminated employees. You need to add the field and criteria to the query to designate only active employees in this report area.

## Changing the Query of the Report from Design View:

### Step 1

Click the **report selector** (the box in the upper left corner of the report at the intersection of the vertical and horizontal rulers) to select the report, and then open its property sheet.

### Step 2

Click the **Build** button for the record source property to open the **qryEmployeesBySupervisor** query in the Query Builder window.

### Step 3

Scroll to display the first empty column in the design grid, and then add the **Terminated field** from the **tblEmployees** table field list to the empty column.

### Step 4

Click the **Criteria** row in the Terminated column, type **No**, and then close the query.

### Step 5

Click the **Yes** button to save the change, returning focus to the Report design, and close the Property Sheet.

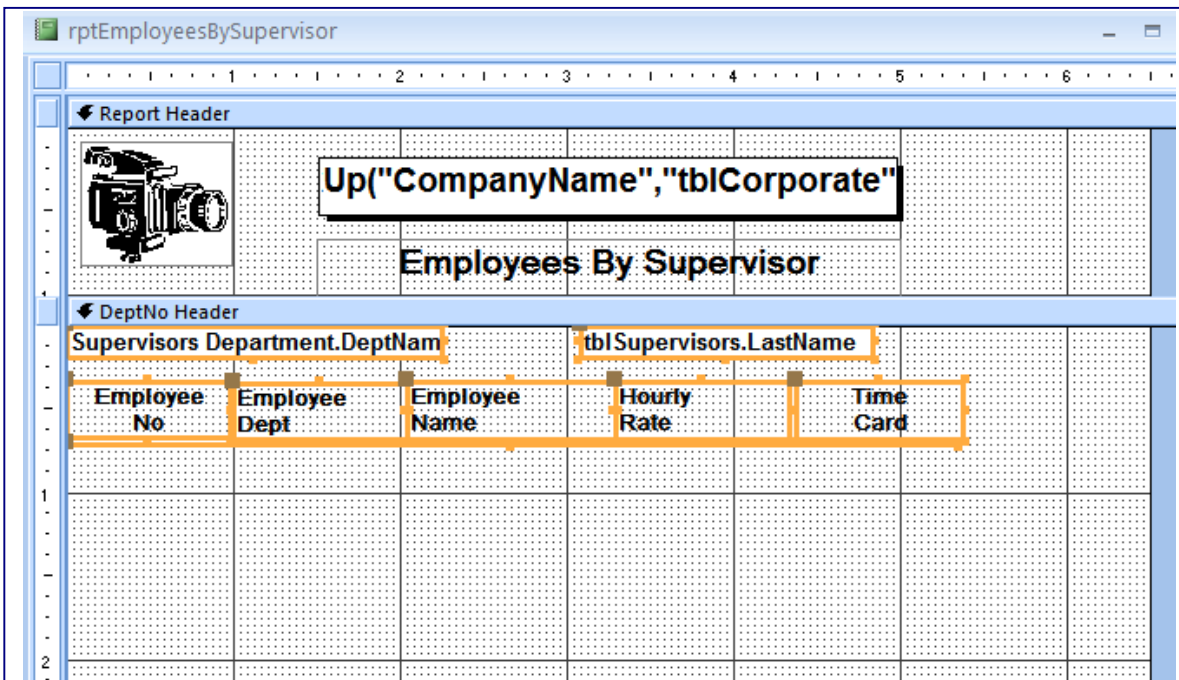
Now you'll move the controls from the Report Header section to the DeptNo Group Header section, and remove the Report Header from the report. You need to resize the DeptNo Header section and move its existing controls down first.

## Moving the Controls and Eliminate the Report Header Section:

### Step 1

Increase the height of the DeptNo Header section so that is approximately 2" tall, and then select all the controls in the section, as shown below: be sure to select the horizontal line below the labels, as well.





### Step 2

Drag all the controls to the bottom of the section, In the Report Header select all of the controls and then **cut** them either by using Ctrl + X

### Step 3

Right-click an empty area in the DeptNo Header section, and then click **Paste** on the shortcut menu.

### Step 4


Right-click on the Report Header bar, and then click **Report Header/Footer** to **deselect** it. Your report you look like the one shown below:



rptEmployeesBySupervisor

1 2 3 4 5 6

DeptNo Header

 Up("CompanyName","tblCorporate")

Employees By Supervisor

tblSupervisorsDept.DeptName tblSupervisors.LastName

Employee No Employee Dept Employee Name Hourly Rate Time Card Required?

Detail

EmployeeNo tblDepartments. tblEmployees.L: HourlyRate TimeCard

DeptNo Footer

Total Employees =Count([EmployeeNo])

### Step 5

Click the **Logo Control** in the DeptNo Header section, open the Properties Sheet, and then change the Size Mode property to **Stretch**, if necessary. It might have changed back to clip when you cut and pasted the control.

### Step 6

Close the property sheet, save your changes, and then preview the report.

### Step 7

The bottom portion of some text in the department name and last name controls in the DeptNo header section is not entirely visible. Return to design view to slightly increase the height of those two controls so nothing is cropped.



## Movie Cam Technologies

### Employees By Supervisor

Information Systems		Tyson		
Employee No	Employee Dept	Employee Name	Hourly Rate	Time Card Required?
900	Information Syste	Tyson	\$34.00	No
400	Information Syste	Elvy	\$12.90	No
Total Employees		2		

Now that you have completed the report for the product managers, you want Martin to take a look at it, so you decide to e-mail it to him in a **Portable Document Format** (PDF) because you are not sure if he has Access installed on his computer. Open the report in Print Preview and Click on the PDF or XPS button. Save the report as a PDF file and attach to an email. This way he can look at the report and then e-mail you a response at his convenience.

#### Using the No Data Event:

The report showing terminated employees, which Daniel Jenkins requested, has already been created. It is named rptTermEmployees and uses a qryTermEmployees query for its record source. Although records for terminated employees will be stored in the final database, the sample records that you are working with do not include data on terminated employees; so this report contains no records. In the future, terminated employees might be archive out at year end, so at times there will be no records in this report.

You will create a macro that uses the NoData print event which, as you have learned, stops the printing or previewing of a report that contains no data. This macro will be saved in the global macro and can be used by any of the reports in the database.

### Changing the NoData Event:

#### Step 1

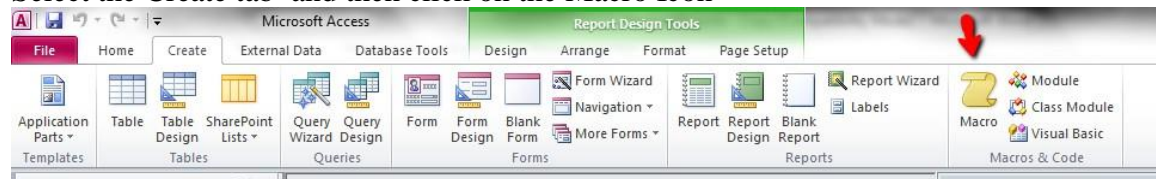
Open the **rptTermEmployees** report in Print Preview, and then maximize the report window and zoom in on the report. There are no records in the report and in the Report Footer section the count of the total number of employees is 0.

#### Step 2

Switch to design view, and then open the property sheet for the report.

#### Step 3

Select the Create tab and then click on the Macro Icon



#### Step 4

Click the arrow or the drop down box **Add New Action**, select **SubMacro**. Name the submacro **Report\_NoData**.

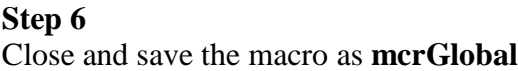
Submacros are like subroutines in a programming module. You could also create submacros that perform related functions within the same macro.

#### Step 5

Click the arrow below in the drop down box **Add New Action**, select **Message Box**, In the Message Type **This report has no data**

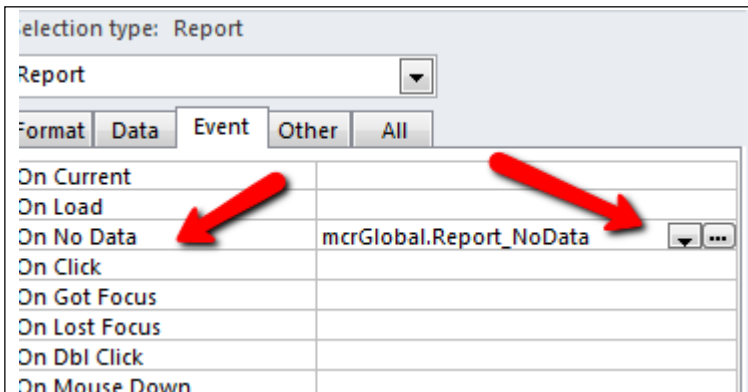
#### Step 6

Make sure that beep is set to **yes**



Open **rptTermEmployees** in design view, select the properties sheet for the report

Select the event tab and in the **On No Data** event use the drop down box to select **mcrGlobal.Report\_NoData**. The macro mcrGlobal has the submacro Report\_NoData as part of the name.



### Step 10

Close the property sheet, save your changes, close the report, select the **rptTermEmployees** report, and then click **Preview** to test the macro. The message box that you created opens, with the information icon and the text indicating that the report has no data. Note that the title bar at the message box displays “Microsoft Office Access” because you do not specify a title for the box.

### Step 11

Click the **OK** button to acknowledge the error message. The macro cancels the previewing event.

You decide that you would rather use VBA code in place of macros whenever possible. Macros might be simpler to write, but VBA code executes faster, allows for error handling and is a better way to automate the user interface.

You need to convert the macro you created to respond when no data was found for report into VBA code as a function. This is a good example of when you would create a function and place it in a standard module. Just as you created one macro that can be called from any report if the NoData event was triggered, you will convert this macro to one VBA function that can be called by any report. This way, you do not have to place the same VBA code in every single report class module. Rather you write the code once and update or maintain it from a single location: a standard module.

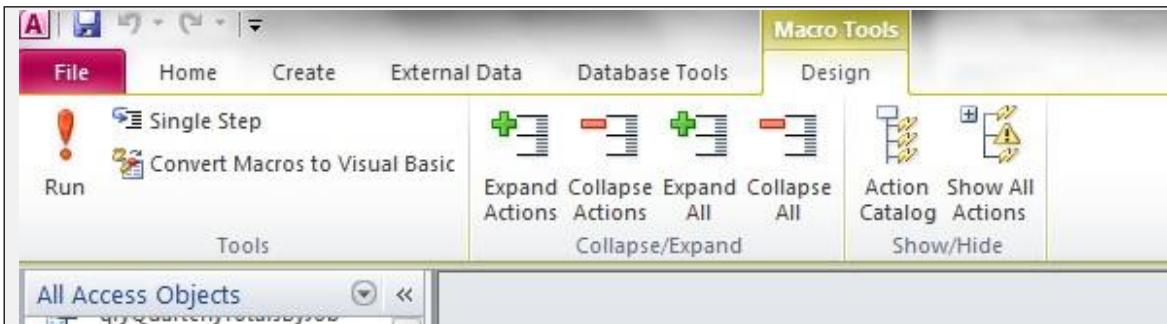
### Converting the Macro to VBA:

#### Step 1

In the Navigation Pane open **mcrGlobal macro** in Design View

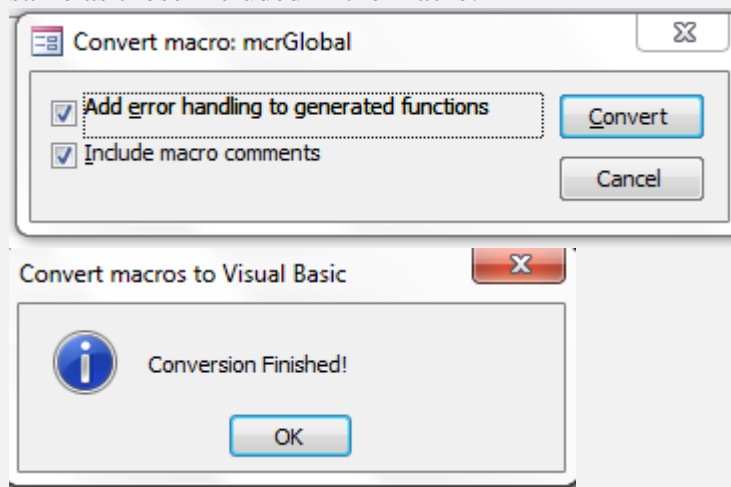
#### Step 2

In the Design Tab click on the **Convert Macros To Visual Basic** button.



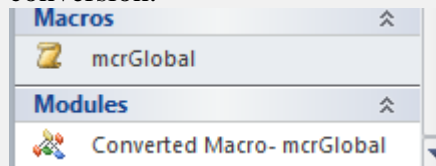
### Step 3

In the convert macro dialog box, be sure that the **Add error handling to generate functions** and **Include macro comments** checkboxes are selected (see below.) These options add some generic error handling to the function and add descriptions that are the same as those included in the macro.



### Step 4

Click the **Convert** button, and a conversion finished dialog box displays. Click **OK** the converted macro appears under the list of modules in the Navigation Pane with Converted Macro: followed by the name of the macro. Click **Design** to view the results of the conversion.



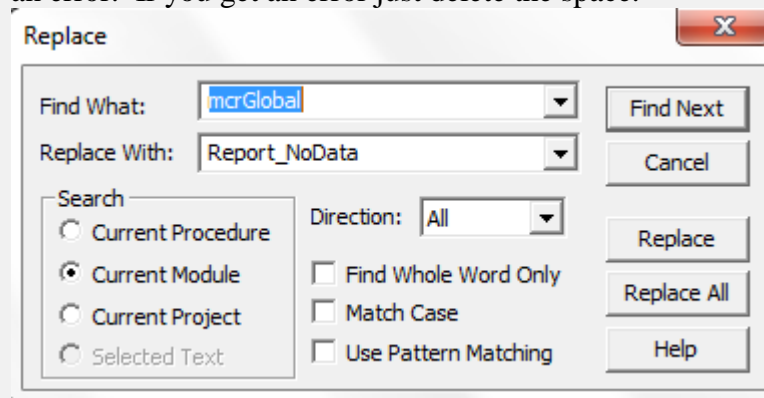
### Step 5

Make sure the converted macro is displayed in the Project Explorer window. You can view the Project Explorer window by right clicking **Converted Macro-mcrGlobal** and select **Design View**.

### Step 6

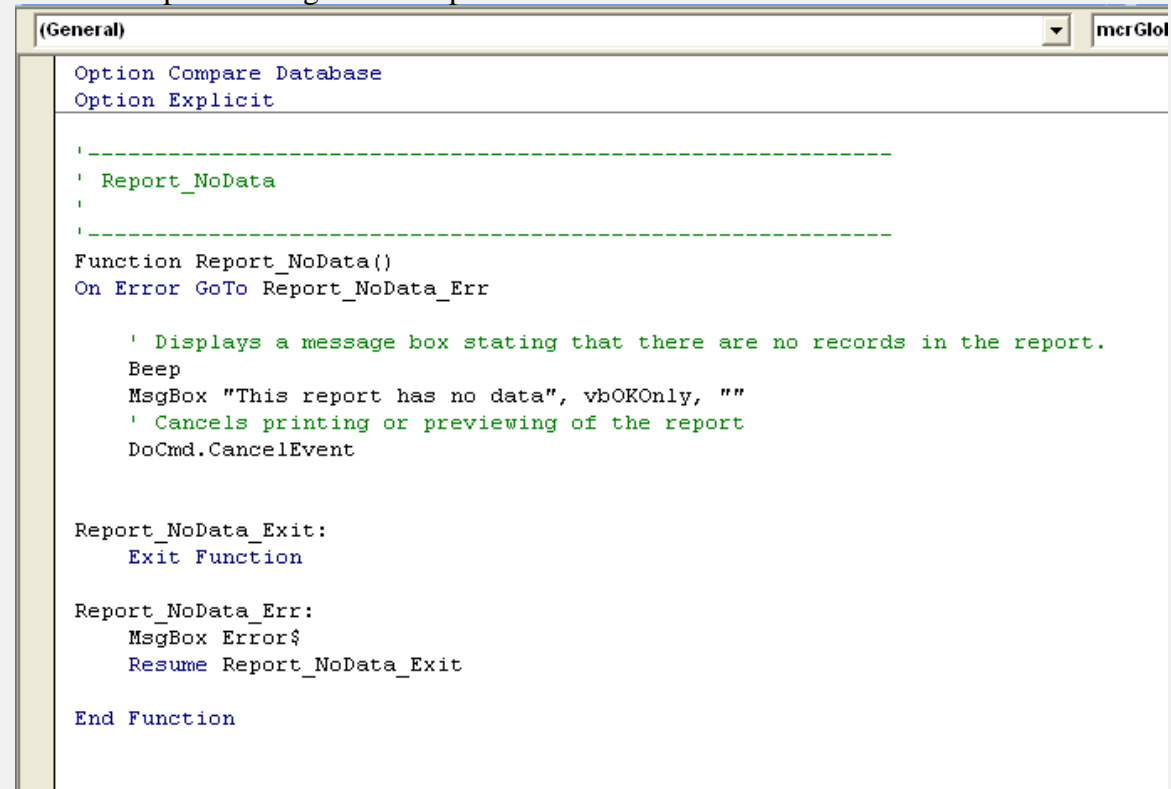
Click **Edit** on the menu bar, click **Replace**, type **mcrGlobal** in the Find What text box, type **Report\_NoData** in the Replace With text box, click the **Replace All** button, and then click the **OK** button and the message box tells you six replacements were made. This search-and-replace procedure eliminates the mcrGlobal string of text, which can

look confusing in the code. You can copy mcrGlobal\_Report\_NoData from the code in Project Explorer if necessary. Make certain there is no space after NoData or it will cause an error. If you get an error just delete the space.



### Step 7

Close the replace dialog box. The procedures should look like the one shown below:



### Step 8

Click the **Debug** and then click **Compile MovieCam** to compile the code. Compiling checks for syntax errors and speeds up the processing of your code. Save your changes and then close the Visual Basic window.

### Step 9

Right-click the **Converted macro- mcrGlobal** module (in the Navigation Pane), click **Rename**, type **mdlReportFunctions**, and then press **Enter** key. This name better describes the type of function contained in this module area

Next you need to test the function. Before you delete the mcrGlobal macro, you will change the NoData event property of the rptTermEmployees report to the function name instead of the macro name, and then preview the report to test the function.

#### Assigning the VBA function to an event property:

##### Step 1

Open the **rptTermEmployees** report in design view, open the property sheet for the report, and then click the **Event** tab if necessary.

##### Step 2

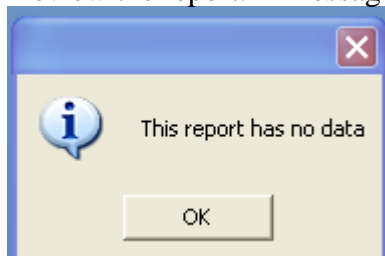
In the property sheet select the **Event** tab then find the **On no data** event and replace the **mcrGlobal.Report\_NoData** with **=Report\_NoData()** and then close the property sheet.

##### Step 3

Save your changes and then close the report.

##### Step 4

Preview the report. A message box like the one shown below opens:



##### Step 5

Click the **OK** button. Note that there is no title in the message box. This is because you did not enter a title in the macro steps, so when you converted this to VBA, the title was left blank as well.

Daniel Jenkins also requested a report on all active employees, similar to the rptTermEmployees report. The report has been designed, but you still need to add an expression to count the number of employees. This report has no grouping levels, so only the Report Footer section can contain the expression. However, the Report Footer prints on the last page of the report, and Daniel wants to have a number of employees shown on each page of the report:



When you include a calculated expression in a Report Footer or a Group Footer, Access knows to which records the expression applies, and does the calculation for the expression before laying out the pages of the report in memory. If you try to calculate an expression in a Page Footer, an error results because Access does not know which records to include in the calculation of the expression until after the page has been laid out. To overcome this limitation, you can create a macro or write VBA code to perform the calculation.

You can use the print event to run the code that will perform tabulations for a text box contained in a Page Footer. A report section recognizes the print event after this section has been laid out, but before this section is printed or displayed in print preview.

To calculate a count of employees on each page, you can place an unbound text box in the Page Footer section. The unbound text box can tabulate the total number of employees on the page as Access lays out the records. A new page starts after Access recognizes the print event as it lays out the Page Header. So you will use the Print event of the Page Header to initialize the unbound text box. You will set the value of the unbound text box to zero before any records are added, and then increment the value of the text box by one for each record that is laid out.

The print event of the detail section is recognized each time a record is laid out in the detail section, but before the record is printed. You'll use this event to run a line of code that increments the value of the text box by one for each record in the detail section. By the time all the records are laid out in a page of the report, the value in the text box will be equal to the total number of records on that page.

You'll use the code builder, which enables you to write VBA code, to create two event procedures. Remember that an event procedure is a group of statements that executes when an event occurs. You will also create these event procedures in both report modules and form modules. A report module is saved as part of a report and contains one of many procedures that apply specifically to that report.

Now you will create a report module to count the number of employees per page in the report.

### **Creating a Report Module for a Page Record Count:**

#### **Step 1**

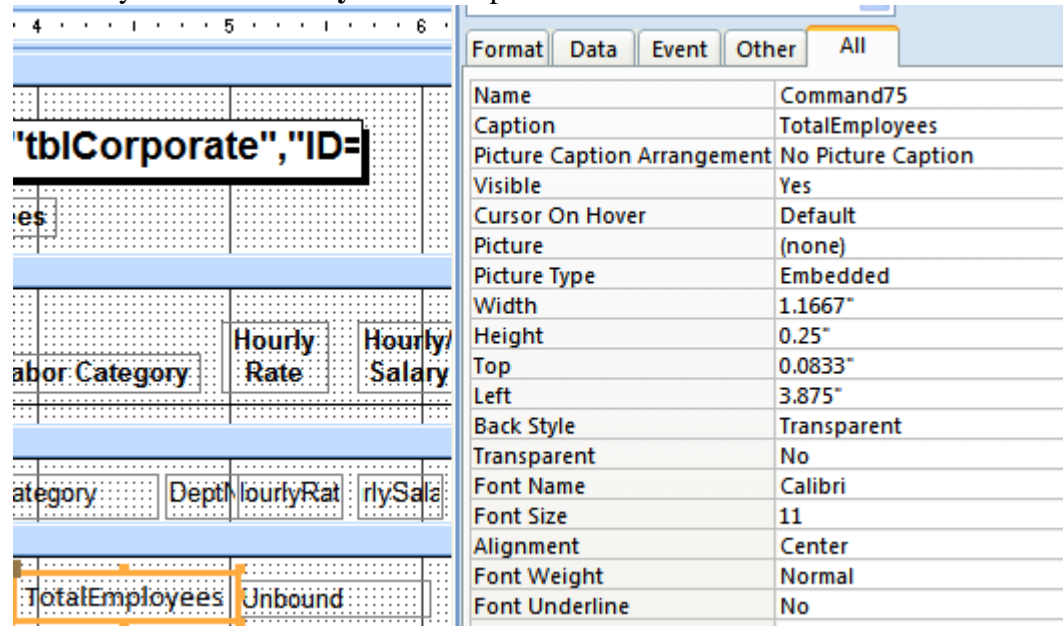
Open the **rptActiveEmployees** report in design view, select the **Text Box** icon in the **Design** section of the Ribbon, and then click in the Page Footer section at the 5" mark on the horizontal ruler to create an unbound text box control.

#### **Step 2**

Open the properties sheet for the new control, click the **All** tab, and then change the name property to **txtCount**. Leave the property sheet open.

### Step 3

Insert a label control to the left of the text box, and then type **Total Employees**, if necessary set the **Back Style** to Transparent as shown below:



### Step 4

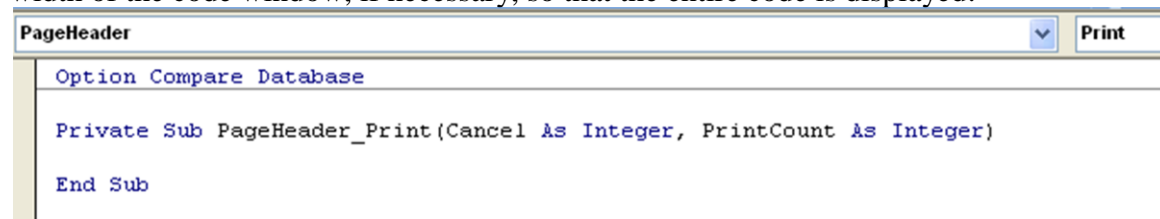
Select both the label and text box and change the height property of both the label and text box to **0.225"**. **Right-click** and set the align property to **Top**.

### Step 5

Click the **Page Header** section band, and then click the **Event** tab on the properties sheet.

### Step 6

Click the **Build** button on the **On Print** property, click **Code Builder** in the Choose Builder dialog box, and then click the **OK** button. The code window opens, and the first and last line of the event procedure are created automatically (shown below). Increase the width of the code window, if necessary, so that the entire code is displayed.



### Step 7

Press the tab key to indent a line containing the insertion point, and then type **txtCount = 0**. This statement sets the txtCount text box, which actually resides in the Page Footer section equal to zero at the beginning of each new page before the detail section is laid out for that page. Note that it is a standard practice to indent lines of code; you will learn more about code formatting in later sessions.

### Step 8

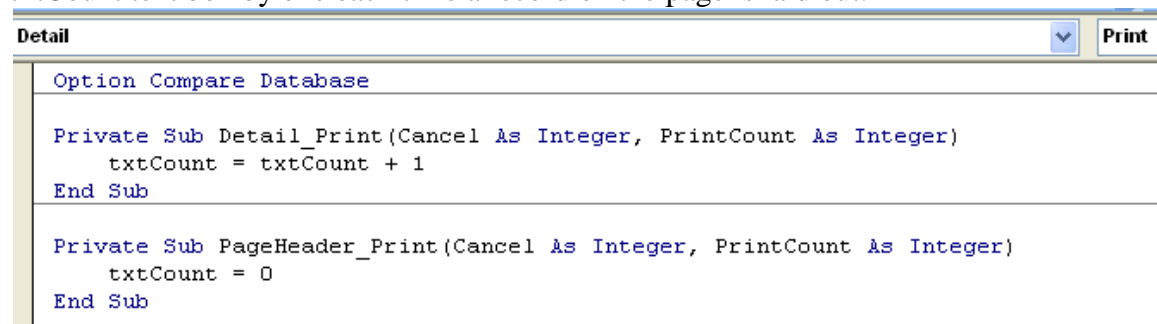
Compile the code, save your changes, and then returned to the report window in design view.

### Step 9

Click the detail section band, click in the **On Print** text box in the property sheet, click its **Build** button, click **Code Builder**, and then click the **OK** button.

### Step 10

Press the **Tab** key to indent the line containing the insertion point, and then type **txtCount = txtCount + 1** as shown below. The statement increments the value of the txtCount text box by one each time a record on the page is laid out.



```
Detail Print
Option Compare Database

Private Sub Detail_Print(Cancel As Integer, PrintCount As Integer)
    txtCount = txtCount + 1
End Sub

Private Sub PageHeader_Print(Cancel As Integer, PrintCount As Integer)
    txtCount = 0
End Sub
```

### Step 11

Compile the code, save your changes, and then close the Visual Basic window.

### Step 12

Close the property sheet, preview the report, scroll down to see the contents of expression, and then close the report.

You need to create one last report for the product managers. It will show a detailed listing of hours by job, and a summary of hours by quarter of the year.

The design of the report has been started, and the report is named rptHoursByJob. The reports record source is the qryHoursByJob query. This query contains a list of hours for each job by each employee. The report groups records on the JobID field. A page break has been inserted after the group footer section so each job number begins on a new page. The product managers want a line item number for each entry on the report to serve as a point of reference.

Records are not numbered in the report; they simply appear in the order determined by the underlying query or table (the record source) if no sorting or grouping has been specified elsewhere in the report. Settings applied in the sorting and grouping dialog box overrides the sort order of the records in the record source. To create a reference number for each record, you can insert a text box and use the Running Sum property to increment the text boxes value.

You use the Running Sum property to calculate record –by- record totals or group -by - group totals in a report. You can specify that the text box displays a running total, and you can set the range over which to accumulate the values.

When you set the Running Sum property to Over Group, the Running Sum value accumulation until another group is encountered.

### **Numbering each Detail Section Line in the Report:**

#### **Step 1**

Open the **rptHoursByJob** report in design view.

#### **Step 2**

Make sure that the Design tab is selected in the Ribbon and click on the **Text Box** button.

#### **Step 3**

Click to the left of the timecard date control in the detail section of the report to create a new unbound text box.

#### **Step 4**

Make sure the property sheet is open, click the **All** tab, change the name property to **txtLineItem**, set the control source property to = **1**, and then change the Running Sum property to **Over Group**.

#### **Step 5**

Close the properties sheet and then print preview the report. Each group should have numbered line items and the number should start over for each group. Each group begins on a new page, but whenever a group extends to more than one page, the numbering will continue for that group until the next group is encountered or until the end of the report is reached.


#### **Step 6**

Return to design view, size the txtLineItem text box so that it is the same height (0.225”) as the other controls in the detail section of the report, and then reposition the text box so it is aligned with the other controls, as shown below. You can use the sizing and aligning shortcut menu items on the property sheet to accomplish this step.

**rptHoursByJob**

1 2 3 4 5

Page Header

 ("CompanyName","tblCorpora

Hours By Job  
Detail and Quarterly Totals

JobID Header

JobID Model

Date Name

Hours Worked Total Dollars

Detail

=1 TimeCardDate LastName FirstNam hrsWorked TotalDollars

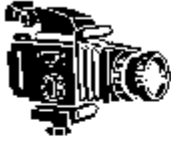
JobID Footer

### Step 7

Add a label to the JobID header section, positioning it just below the JobID label. Type **Line Item** in the label box. Align the label with the left edge of the txtLineItem control you just added in the detail section. Size the height and a line the bottom of the line item label to match the other labels in the header section

### Step 8

Save your changes, and then preview the report. Maximize the report window, if necessary. Advanced page 5 and zoom in, if necessary. By scrolling a bit to the right and down slightly, page 5 of the final report should look like the one shown below:



## Movie Cam Technologies

### Hours By Job Detail and Quarterly Totals

99562	ZTSN1022			Hours Worked	Total Dollars
	Date	Name			
1	11/03/2014	Gold	Ernest	4	\$68.00
2	11/03/2014	Cauldwell	Gloria	9	\$352.98
3	11/03/2014	Cook	Alan	7	\$137.20
4	11/03/2014	Woodward	Martin	10	\$393.30
5	11/03/2014	Woodward	Martin	5	\$196.65
6	11/03/2014	Arquette	Thomas	7	\$416.50
7	11/03/2014	Arquette	Thomas	10	\$595.00

### Step 9

Close the Print Preview and then Close the Design View saving your changes.

The next step is to modify the report so that it sums the hours worked on the job during a specified quarter of the year. The quarterly totals can be calculated in a query, and then the query can be used to create a sub report on the main report.

### Subreports:

A subreport is a report that is inserted in another report. When you combine reports, one must be the main report. The main report can be balance to an underlying table, query, or SQL statement, or can be unbound. An unbound main report serves as a container for unrelated subreports that you want to combine. A bound report must contain a field in its record source that is also a field in its subreport(s).

A main report can contain several levels of subreports; there is no limit to the number of subreports a main report can contain, but there is a limit to what may be practical.

Subreports are often used to show summary data that is related to the detailed data in the main report. In this way, subreports are similar to subforms and main forms. For example, when you view the page in the rptHoursByJob report for a specific job, you can include a subreport that has a summary of hours and dollars spent on the job during a calendar quarter.

The subreport will be based on a query you create that calculates total hours and dollars by quarter for each job. You will use the DatePart function in an aggregate query to perform these calculations.

## Create a totals query for the subreport:

### Step 1

Create a new query in design view.

### Step 2

Add the following tables to the query window in this order: **tblEmployees**, **tblTimeCards**, **tblHours**, and **tblJobs**, and then close the show table dialog box.

### Step 3

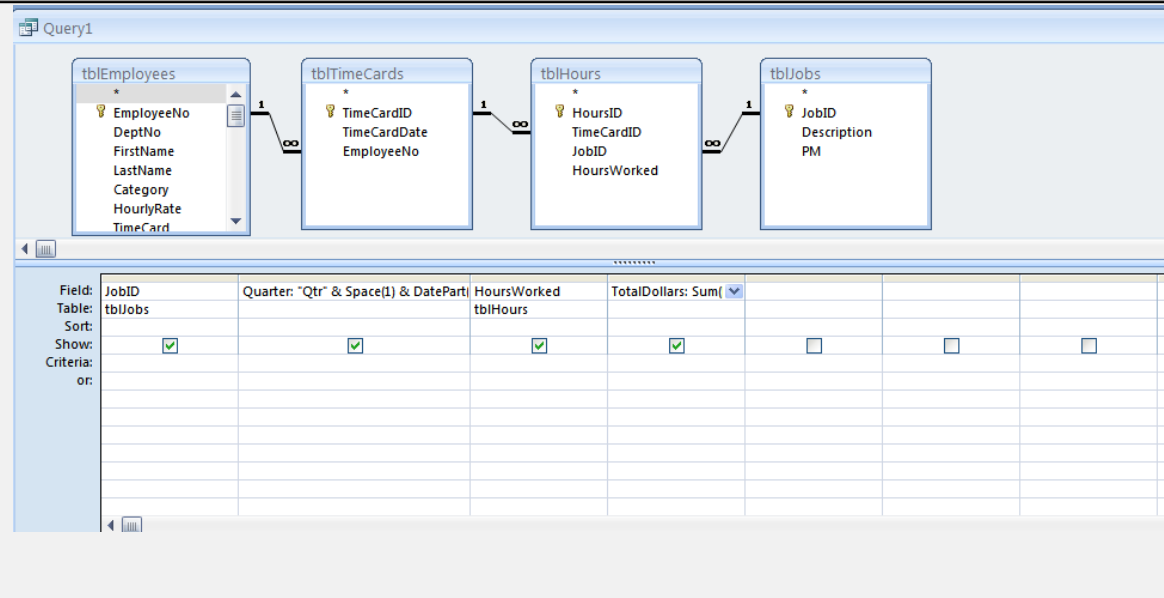
Add the **JobID** field in the Jobs field list to the design grid, click the field text box of the second column in the design grid, and then type **Quarter: "Qtr" & space (1) & DatePart ("q", [TimeCardDate])**. This formula joins the abbreviation Qtr (for quarter) to the DatePart function that returns the quarter of the year in the TimeCardDate field. The quarter of the year value will be an integer value from 1 to 4.

### Step 4

Add the **HoursWorked** field in the tblHours field list to the third column of the design grid, right-click the third column, click **Properties** on the shortcut menu, and then change the Format property to **Fixed**. Leave the property sheet open.

### Step 5

Click the **Field** text box of the fourth column in the design grid, type **TotalDollars: Sum([HoursWorked] \* [HourlyRate])** and then change the format property to **currency**. This expression will calculate the total dollars by quarter for each job. Close the property sheet. The query window should look like the one shown below.



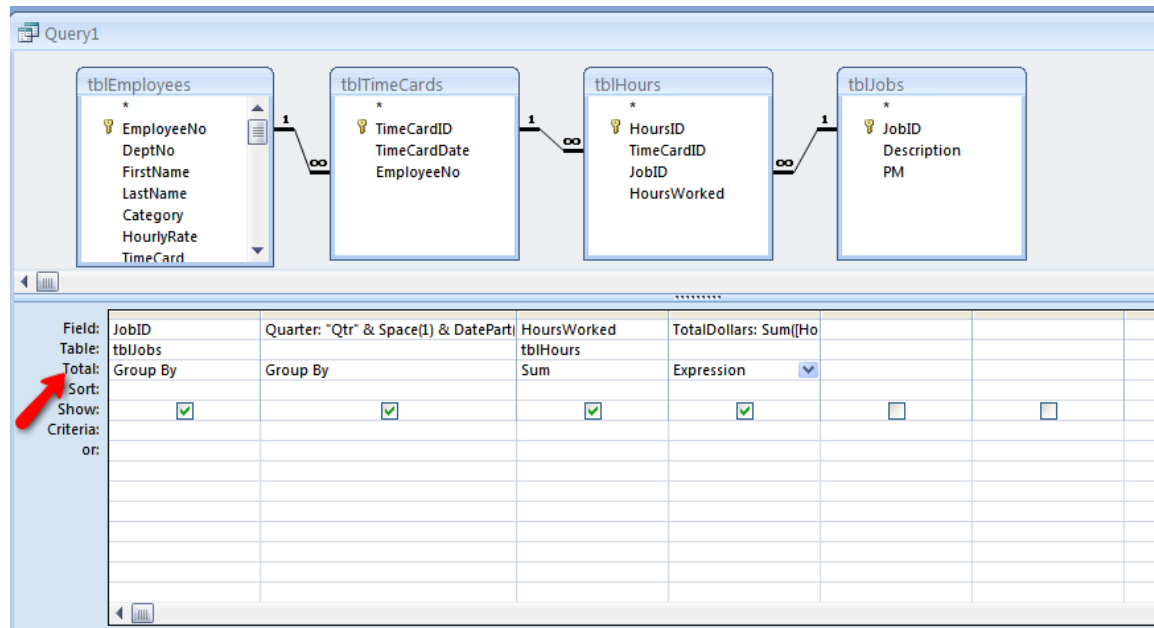


### Step 6

Click the **Totals** button on the Show Hide section of Ribbon. The Total Row in the design grid indicates that totals will be entered in the report.

### Step 7

Click the **list arrow** for the HoursWorked field in the Total Row, clicked **Sum**, and then change the Total cell for TotalDollars to **Expression**. The design grid should look like the one shown below:



### Step 8

Run the query check the results. The query results include 4 records and 4 columns.

Only one jobID was used to demonstrate the use of this query which is listed for each quarter in which work was done for that job, and the total hours and total dollars for that job are displayed by a quarter.

### Step 9

Save the query as qryQuarterlyTotalsByJob and then close the query.

Now you will create the subreport and add it to the main report. The Subforms/Subreport button on the Ribbon allows you to create a subreport, but an easier way to create one is to drag the query object from the Navigation Pane to the desired section of the report.

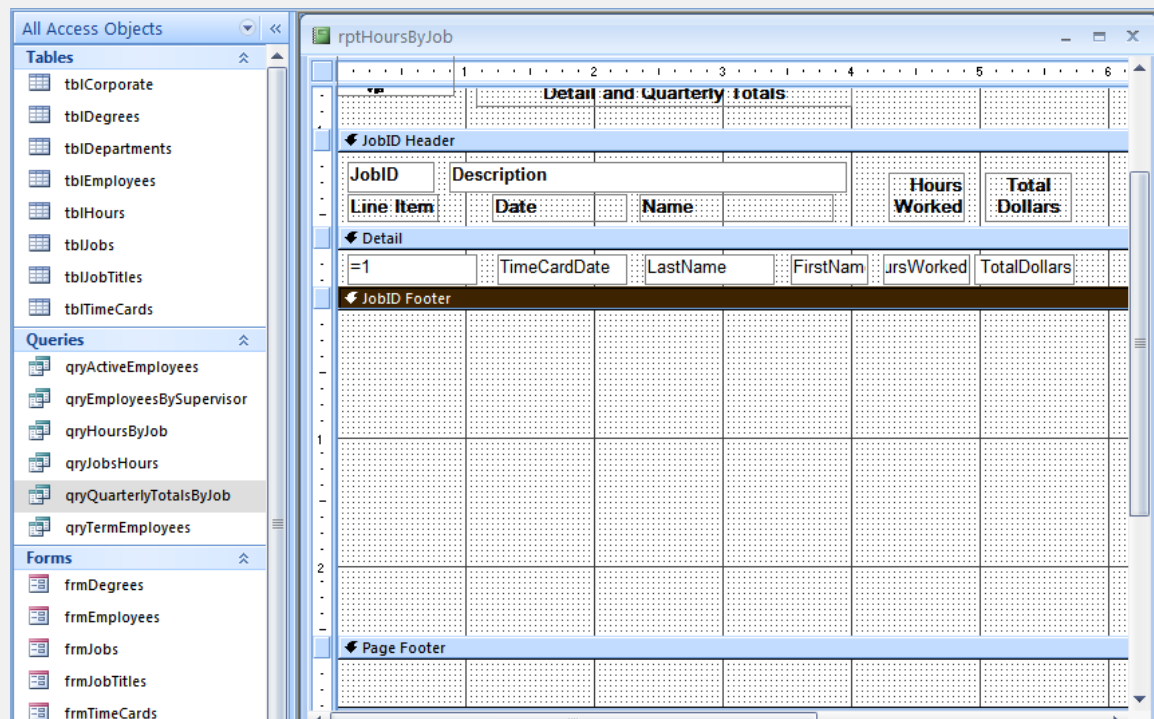
Access will activate a wizard to insert a subreport and use the selected query as the underlying record source. After you've inserted the subreport, you can make formatting changes to it in design view.

### **Creating the subreport:**

#### **Step 1**

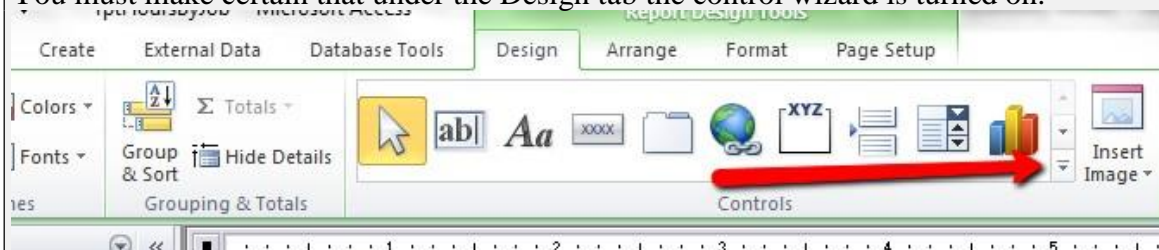
Open the **rptHoursByJob** report in design view, drag the Page footer section to the 2 ½" vertical mark, scroll up or down as needed so you can see as much of the JobID footer

section as possible, remove any labels from the JobID Footer. You can see the Navigation Pane and the report window side-by-side as shown below:



## Step 2

You must make certain that under the Design tab the control wizard is turned on.



## Step 3

Click the **qryQuarterlyTotalsByJob** query in the Navigation Pane to select it and drag to the upper left corner of the JobID footer section of the report. The first subreport Wizard dialog box opens as shown below:

**SubReport Wizard**

Would you like to define which fields link your main form to this subform yourself, or choose from the list below?

☐ Choose from a list.
 ☒ Define my own.

Form/report fields: Subform/subreport fields:

None

Cancel < Back Next > Finish

#### Step 4

Click the **Define my own** option button so that you can identify the field(s) on which the main report and the subreport are linked.

#### Step 5

Click the **Form/report fields** list arrow, click **JobID**, click the **Subforms/subreport fields** list arrow, and then click **JobID** as shown below:

**SubReport Wizard**

Would you like to define which fields link your main form to this subform yourself, or choose from the list below?

☐ Choose from a list.
 ☒ Define my own.

Form/report fields: Subform/subreport fields:

JobID	JobID

Show qryQuarterlyTotalsByJob for each record in qryHoursByJob using JobID

Cancel
 < Back
 Next >
 Finish

### Step 6

Click the **Next** button, type **srptQuarterlyTotalsByJob** as the name of the subreport, and then click the **Finish** button.

### Step 7

If necessary, resize the report window so it is easier to make changes to the report design, and then delete the **srptQuarterlyTotalsByJob** label above the subreport object in the JobID Footer section. If you place a subreport in the upper left corner of this section, you may have to reposition the subreport temporarily in order to select and delete the label. The label may only be partially showing above the subreport.

### Step 8

Reposition the subreport back to the upper left hand corner of the JobID Footer section, leaving just a small amount of space from the edge. Notice that the Wizard also placed the subreport in such a way as to expand the report width, so you will resize both the report and subreport.

### Step 9

Reduce the subreport width to 2 ½ “, and then reduce the width of the main report back to 6 ½ “.

### Step 10

Save your changes and close the report.

Next, you will format the subreport and add expressions to the Report Footer section that sum the hours worked and total dollars.

### Completing the Subreport:

#### Step 1

Open the **srptQuarterlyTotalsByJob** report in design view.

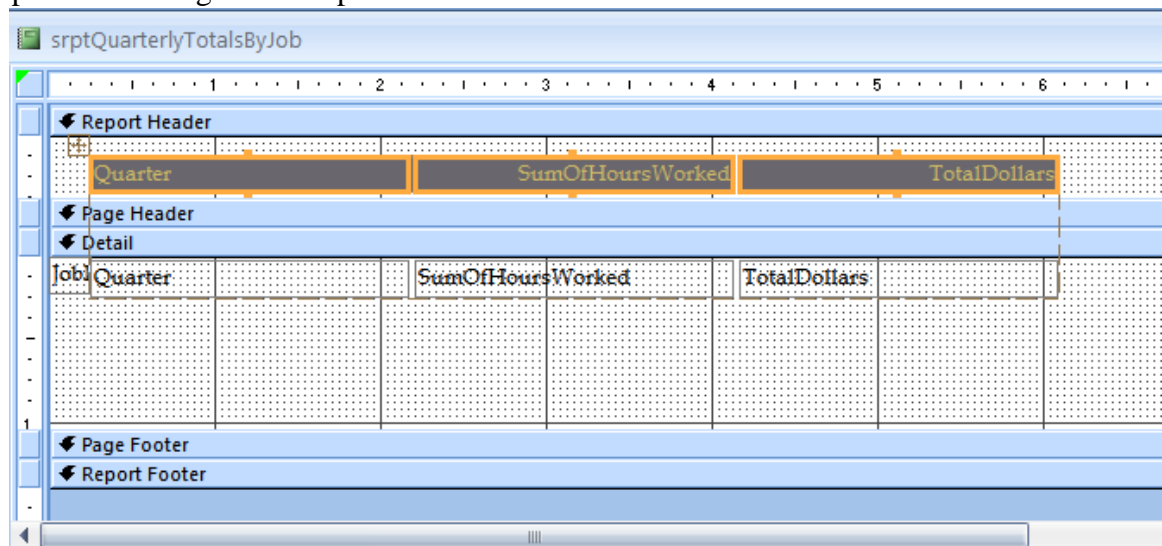
#### Step 2

Select the label JobID and then **click Delete**. Make certain only the label for jobID is selected.

Readjust and resize the JobID text box to approximately ¼” **wide**. Open the Property Sheet and set the visible property **No**.

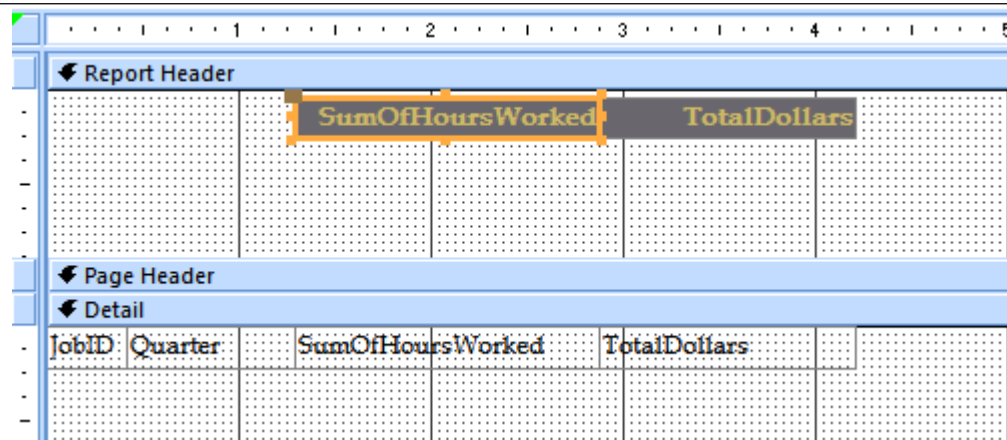
The main report and subreport are linked by the JobID field, so this control must remain on the report; however, it does not need to be visible on the subreport when it is viewed or printed. The JobID field is located already in the JobID header section of the main report.

Pull up the remaining files and place them to the left of the JobID field.



#### Step 3

Delete the **Quarter** label in the Report Header section because the abbreviation "Qtr" is included in the expression in the detail section. The labels may already be bold because the subreport was created using the SubReport Wizard; therefore, the font style and color applied to the subreport are the same as the font style and color selected for the last report you created using the Report Wizard.

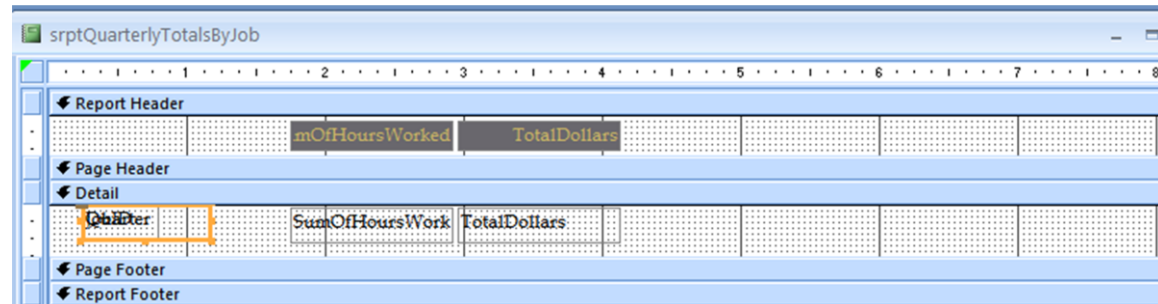


#### Step 4

Move the **Quarter** text box control so it covers the JobID text box (you can use the align features to align the left edges of both controls). The JobID field will not appear on the subreport now because the visible property has been set to No, so covering it with a Quarter text box has no negative effect on the report.

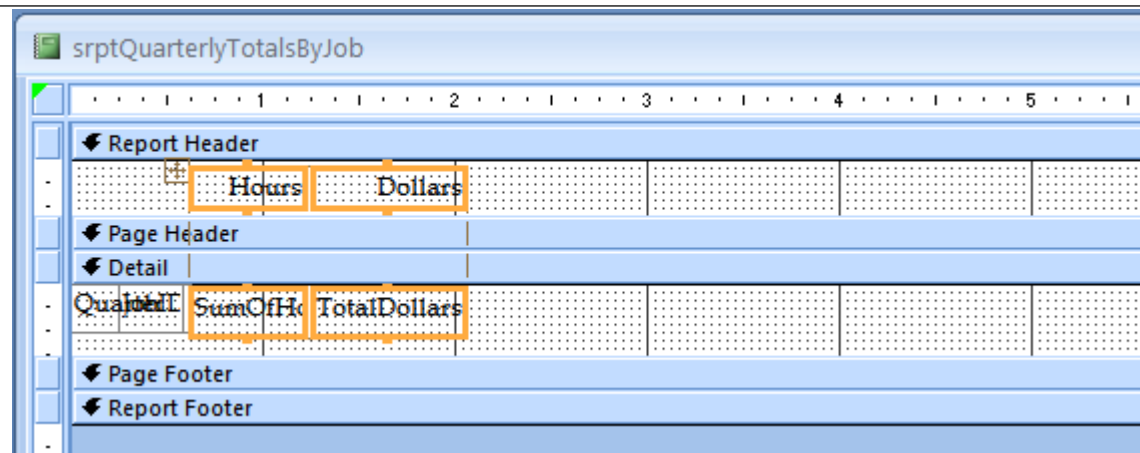
#### Step 5

Using the figure below as a guide, rename, size, and reposition the labels and the Report Header section, and then size and reposition the text boxes in the Detail section. You need to create the labels for Hour and Dollars.



#### Step 6

Using the figure below as a guide, rename, size and reposition the labels in the “Report Header section, and then size and reposition the text boxes in the Detail section. It might be easier to click on the **Total Dollars** text box and then double-click on the **Format Painter** button and then click on each of the two labels above. This will set the format as shown below. All you have to do now is change the name in the labels.



### Step 7

Size the Report Footer section of the subreport to approximately ¼" tall, and then add two text boxes to it: one beneath the SumOfHoursWork control and the other beneath the TotalDollars control.

### Step 8

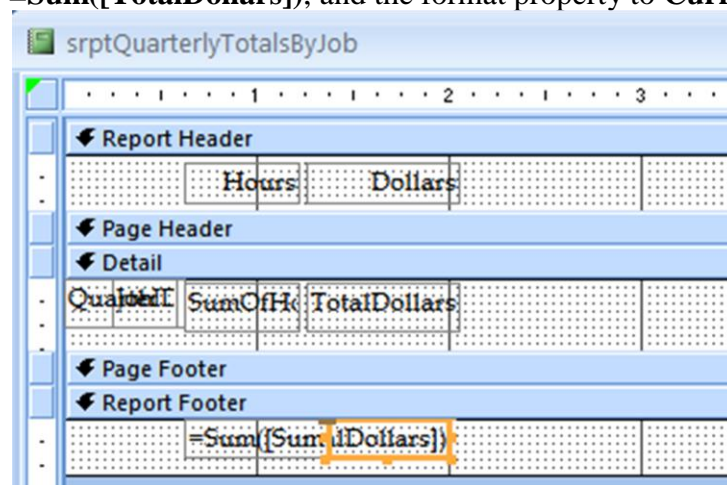
If necessary, delete any labels that Access may have been added with the text boxes.

### Step 9

Click the text box control you inserted beneath the SumOfHoursWork control, open the property sheet, and then change the name property to **txtTotalHoursWorked**. Click the **txtTotalHoursWorked** text box, and then enter **= Sum ([SumOfHoursWorked])** as the control source property. Change the format property to **Fixed**.

### Step 10

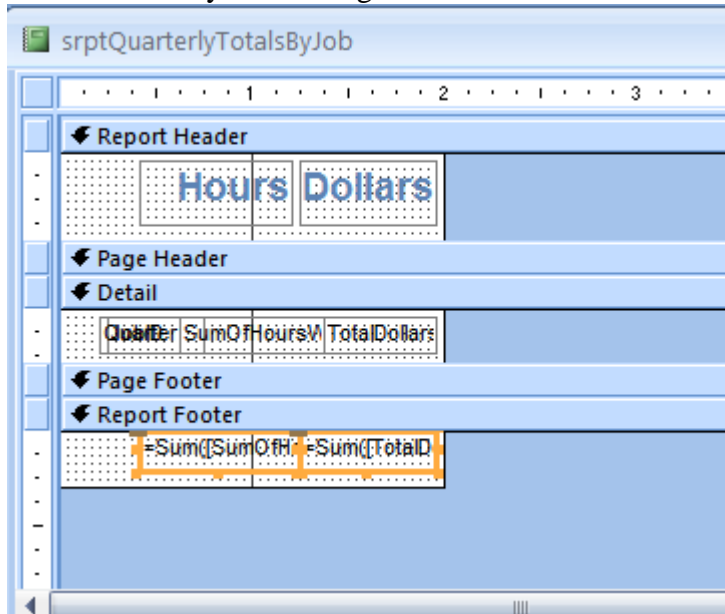
Click the text box control you inserted beneath the total dollars control, and then change the name property to **txtTotalDollars**, the control source property to **=Sum([TotalDollars])**, and the format property to **Currency**. Set the text align to **Right**.



### Step 11



Using the image below as a guide, resize the controls and align them with those in the Detail section of the report. Select all labels and text boxes for all sections of the report (except for the Report Header which you can choose whatever font you desire), change the font to **Arial** and the font size to **8**, change the font color to **black** if necessary, and then size the report to be **2" wide**. Deselect the controls and then close the properties sheet. Your report should look as below: It might be easier to work with the labels and the text boxes by deselecting the link between the two.



### Step 11

Save your changes, and close the report.

Next you'll size the subreport on the main report.

### Sizing the Subreport on the Main Report:

#### Step 1

Open the **rptHoursByJob** report in design view, and then maximize the report window.

#### Step 2

Scroll down enough so you can see as much of the JobID footer section as possible. Size the subreport control on the main report to approximately 2" tall and reposition it in the JobID footer section, as necessary, to approximate its position as shown below. The top of the subreport should be about a 1/3" from the top of the section.


The screenshot displays the Microsoft Access Report Designer interface. The main report, titled 'Hours By Job', is in the design view. It features a 'Page Header' section with a label containing a concatenated name field: `("Name", "tblCorporate", "Comp")`. Below this is a 'JobID Header' section with fields for 'JobID', 'Description', 'Date', 'Name', 'Hours Worked', and 'Total Dollars'. The 'Detail' section contains fields for 'TimeCardDate', a concatenated name field `= [Lastname] & ", " & [FirstN]`, 'rsWorked', and 'TotalDollars'. The 'JobID Footer' section is currently being edited to insert a subreport. The subreport, titled 'Hours Dollars', has its own 'Report Header' section with a label `Hours Dollars`, a 'Page Header' section, a 'Detail' section with fields for 'Quarter', 'SumOfHours', and 'TotalDollars', and a 'Page Footer' section. The main report's 'Page Footer' section is also visible at the bottom.

### Step 3

Insert an unbound label control above the subreport control, type **Quarterly Hours And Dollars**, set the label height property to at least **0.25"**, widen the label to match that of the subreport (approximately 2.5"), and then **center** the label text within the label control. Save the report.

### Step 4

Preview the report, (if necessary you may have to reduce the width of the report) move to the second page to see the subreport, and scroll as needed to view the subreport as shown below:

 <div> <b>Movie Cam Technology</b> </div> <div> Hours By Job Detail and Quarterly Totals </div>		
Quarterly Hours and Dollars		
	Hours	Dollars
Qtr 1	16.00	\$709.96
Qtr 2	26.00	\$860.09
Qtr 3	14.00	\$268.48
Qtr 4	131.00	\$3,019.00
	187	\$4,857.53

Next we need to insert page numbers in the report.

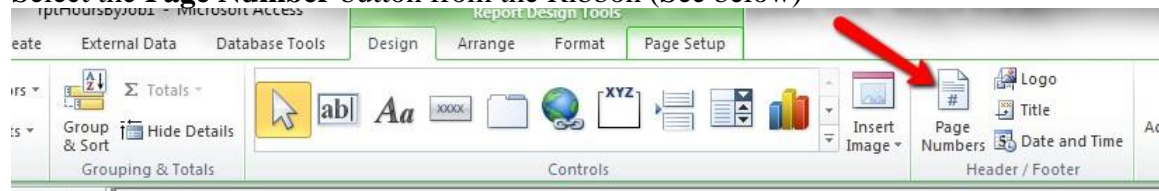
Inserting Page Numbers in the Report:

### Step 1

Return to the rptHoursByJob report in Design view, and maximize the report window, if necessary.

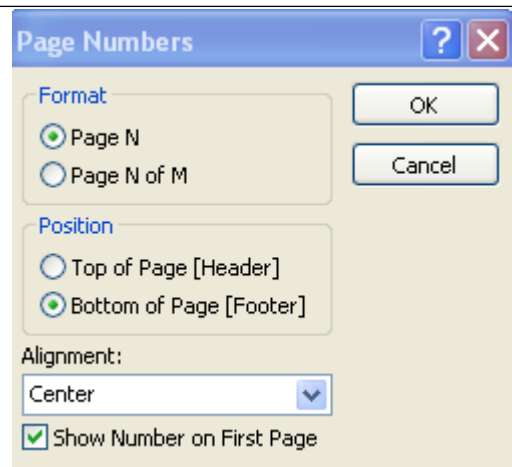
### Step 2

Select the **Page Number** button from the Ribbon (See below)



### Step 2

In the Page Numbers dialog box select **Page N** in the format section, select **Bottom of Page** in the Position section, select **Center** under Alignment and then click the **OK** button.



### Step 3

Close and save the report and then reopen in Print Preview.

## Adding Alternate Row Shading:

### Step 1

Return to design view for the rptHoursByJob report, click the **Detail section** band if necessary. Open the properties sheet and select the **Format** tab.

### Step 2

Click in the **Alternate Back Color** row and select **Alternate Row**. You can click on the **Build** button and select a specific color or different degrees of grey as well.

### Step 3

Close the report and property sheet, and save your changes.

### Step 4

Open the Report in print preview to view the alternate row design.

### Step 5

Close the report, close the Session 5 movie database, and then exit Access.



## MovieCam Technologies

### Hours By Job Detail and Quarterly Totals

90000 Model 18 Gimbal

Line Item	Date	Name		Hours Worked	Total Dollars
1	01/13/2007	Woodward	Martin	12	\$471.96
2	02/10/2007	Arquette	Thomas	4	\$238.00
3	04/21/2007	Cook	Alan	7	\$137.20
4	05/26/2007	Combs	Todd	9	\$127.89
5	06/23/2007	Arquette	Thomas	10	\$595.00
6	07/28/2007	Gerardo	Juan	6	\$154.80
7	08/18/2007	Cook	Todd	8	\$142.00

Martin is pleased with the report. The subreport provides a summary data he needs, and the detail data is included and is numbered and formatted so that it is easy-to-read.