Excel Reporting with 1010data
Contents

Overview.................................................................................................................................................. 3
Start with a 1010data query.................................................................................................................. 5
Running your query using the Excel Add-in......................................................................................... 8
Adding a static input to a dashboard...................................................................................................... 12
Targeting the query results to the dashboard....................................................................................... 15
Formatting the results on the dashboard.............................................................................................. 17
Adding a simple chart to the dashboard.............................................................................................. 21
Creating a static drop-down in the dashboard...................................................................................... 25
Incorporating a dynamic drop-down in the dashboard......................................................................... 31
Adding another query to the dashboard............................................................................................... 41
Adding a button to run the queries........................................................................................................ 44
Charting the results dynamically on the dashboard.............................................................................. 49
Using 1010data libraries and blocks................................................................................................... 56
Hiding worksheets and locking the workbook....................................................................................... 60
1010data gives its users the ability to manipulate and analyze large amounts of data using a familiar spreadsheet interface. It is for that very reason that the 1010data GUI is known as the Trillion Row Spreadsheet. Many of our users also like to use Microsoft Excel for its widely-known functionality and its ability to create user-interactive dashboards. Because of that, we’ve made it very easy to create dashboards in Excel that can leverage the power and speed of data analysis using the 1010data engine. In fact, it is fairly simple to incorporate user-specified parameters in a query that is run on 1010data and have the results displayed directly in an Excel dashboard with charts and conditional formatting.

Let’s say we want to build a dashboard in Excel that displays the results of two 1010data queries. One simply aggregates total sales for a specified department and date range, while the other aggregates total sales by date for the same department. We want to give the user the ability to specify the start and end dates for the date range and to select the department for which they want to see total sales figures. In addition, we want to give the user the choice to aggregate either by group description or by brand. Finally, we want to see visualizations of the results in the dashboard using Excel’s charting capabilities.

When we’re finished with this tutorial, we want our dashboard to look something like the following:

Though this may look a little complex, it is created via a number of simple steps that this tutorial will guide you through:

• Start with a 1010data query that aggregates total sales for a specific department and date range
• Run the 1010data query from Excel using the 1010data Excel Add-in
• Add static inputs to the dashboard for the date range
• Target the results of the query to the dashboard
• Format the query results using basic and conditional formatting
• Add a simple chart to the dashboard
• Create a static drop-down to allow the user to select whether they want to aggregate on group description or brand
• Incorporate a dynamic drop-down to allow the user to select a particular department for the query
• Add another query that aggregates total sales by date for the same department
Add a button to run both queries from the dashboard
Chart the results of both queries dynamically on the dashboard
Incorporate the use of 1010data libraries and blocks
Hide the worksheets and lock the workbook

Let's build the dashboard one step at a time.

We'll use the 1010data Excel Add-in to run a 1010data query from Excel and download the results directly into an Excel worksheet. Let's start with a 1010data query.
Let's start off by building a query using 1010data's Trillion-Row Spreadsheet interface.

For our example, we'll use a Sales Detail table (retaildemo.salesdetail) that contains billions of rows of transactional data, and we'll incorporate information from a Product Master table (retaildemo.products) that contains detailed information about all of our products.

Let's say we want to calculate the total sales from 01/01/2011 to 03/31/2011 for items in department 19 and we want to group the results by the values in the Group Desc column of the Product Master table. So, our results would look something like:

To do this in the 1010data web interface, it would take just a few simple steps.

We'll start by selecting those rows in the Sales Detail table whose values in the Date column fall between 01/01/2011 and 03/31/2011:

Note: We specify the date using YYYYMMDD format.

In the Product Master table, we'll select those rows whose Department is 19.
In the Sales Detail table, we can then use the Link and Select dialog to link in only those rows we just selected in the Product Master. We’ll use the SKU column in each of the tables as the linking column, and we’ll add the suffix _prod to those columns from the Product Master table:

Finally, in the Sales Detail table, we can use the Tabulation dialog to compute the sum of extended sales (tot_sales), grouping by the values in the Group Desc column:
Click **View > Edit Actions (XML)**... to see the Macro Language code for our query:

We can then copy and paste this Macro Language code directly into the 1010data Excel Add-in.
Using the 1010data Excel Add-in, we can run a query on 1010data from Excel and have the results directly downloaded into an Excel worksheet.

The Macro Language code for our example 1010data query is:

```xml
<note type="base">Applied to table: retaildemo.salesdetail</note>
<sel value="between(date;20110101;20110331)"/>
<sel value="(dept=19)"/>
<link table2="retaildemo.products" col="sku" col2="sku" suffix="_prod" type="select">
  <sel value="between(date;20110101;20110331)"/>
</link>
<tabu label="Tabulation on Sales Detail" breaks="groupdesc_prod">
  <tcol source="xsales" fun="sum" name="tot_sales" label="Sum of Extended Sales" />
</tabu>
<sort col="tot_sales" dir="down"/>
```

To run this query in an Excel workbook, we can use the 1010data Excel Add-in.

**Note:** The 1010data Excel Add-in must be installed on your computer. (See [Installing the 1010data Excel Add-in](#) for more information.) Once installed, you want to make sure that you have enabled the advanced features of the 1010data Excel Add-in to perform the necessary steps in this tutorial.

In Microsoft Excel, from the Add-Ins menu, click **1010data > Add New Q-Sheet**. This will open a **q-sheet** in a new tab labeled **_1010q Sheet**, which is where we will specify the query we want to run on 1010data.

Copy the Macro Language code for the 1010data query and paste it into the section of the q-sheet labeled **1010data Macro Code**.
We need to enter some other information in the q-sheet:

- For the **Query Description**, enter "Total Sales by Department".
- In the **To be Applied to Table** field, we need to specify the full path to the table this query should be applied to. For this example, we'll enter `retaildemo.salesdetail`.
- In the **Result Destination** field, enter `Sheet1!$A$1` so that the results will be pasted in cell A1 in the **Sheet1** worksheet.
- From the **Column Headers** drop-down, select "Column Names".

Our q-sheet should look similar to the following:

Let's run the query and see our results. From the **Add-Ins** menu, click **1010data > Run Queries > In Active Workbook** (or press Ctrl+Q).

**Note:** If you have not logged into 1010data via the Excel Add-in, you will be presented with the following dialog:
Enter your 1010data credentials and click Secure Login.

As the query runs on 1010data, you’ll see a 1010data Query Progress dialog, which shows the percentage of total operations completed within the current query. When the query finishes running, you should see a dialog saying: Query completed successfully. Click OK to dismiss this dialog.

To see the results, click the Sheet1 tab:

You can see that the results of this query have been downloaded into Excel and are the same as what we got when we ran this query directly on 1010data.

**Note:** It’s a good idea to save your workbook as you progress through this tutorial. Click File > Save As and navigate to the folder where you want to save your work. Be sure to select Excel Macro-Enabled Workbook (*.xlsm) from the Save as type drop-down, as we will be incorporating VBA macros later in this tutorial.
Adding a static input to a dashboard

Our original query had date values hard-coded for the row selection. Let's add a little more flexibility by allowing the user to enter a date range using static inputs.

We'll create a dashboard where the user can enter the start and end dates for the date range, and then we'll use those values in the query.

We can use Sheet2 for our dashboard (or create a new worksheet, if necessary). Let's rename it "Dashboard" and change the background color to a light grey.

Enter text in the cells so that your dashboard looks similar to the following:

You can see in this example that we will allow the user to enter a start date in cell B3 and an end date in cell B7. Below the inputs, we specify that we want the dates in the format YYYYMMDD, which is the date format used by 1010data.

Note: There are no restrictions as to what the user can enter in these cells. These are simply cells in which the user can type in any values. Also, though we are specifying that we want the user to enter dates in the YYYYMMDD format for this example, you might want to allow the user to enter dates in a more recognizable format (e.g., MM/DD/YYYY). In this case, you would need to do some type of transformation on these input values to put them in the YYYYMMDD format. This date transformation could be done in Excel before the values are passed to the query, or within the 1010data query itself. (See Date Transformations for examples on how to do such a transformation within the 1010data query.)

Let's also name the cells that contain the start and end dates so that it will be easier to reference them in our q-sheet. We'll give the cell B3 the name startdate and the cell B7 the name enddate.

The q-sheet essentially sends whatever Macro Language code it contains to 1010data to process and return the results. Because of that, we can convert any row in the q-sheet that contains our macro code into an Excel formula that references the contents of other cells. Then, when the Excel Add-in sends that query to 1010data, it will contain the values in the referenced cells.

In our query, the date selection was done using the between(X;Y;Z) function:

```
<note type="base">Applied to table: retaildemo.salesdetail</note>
<sel value="between(date;20110101;20110331)"/>
<link table2="retaildemo.products" col="sku" col2="sku" suffix="_prod" type="select">
  <sel value="(dept=19)"/>
</link>
<tabu label="Tabulation on Sales Detail" breaks="groupdesc_prod">
  <tcol source="xsales" fun="sum" name="tot_sales" label="Sum of `Extended Sales"/>
</tabu>
<sort col="tot_sales" dir="down"/>
```
Let's go over to our q-sheet and change the hard-coded date values in our query to reference the new input values, `startdate` and `enddate`.

To reference input values from within the 1010data query, we will need to change the line in our macro code into a formula. We do this by:

- removing any blank spaces from the start of the line
- inserting an `=` at the beginning of the line
- enclosing the macro code in double quotes
- preceding any double quote within the macro code with another double quote so that Excel will not interpret any of them as the end of the formula
- replacing each hard-coded value with a reference to its corresponding input cell using the syntax: 
  `"&[CELL_REFERENCE]&"` (e.g., `"&B3&"`)

Let's go through those steps for our example. There are no blank spaces at the start of the line, so we can move to the next step and add an `=` to the beginning:

```excel
=\<sel value="between(date;20110101;20110331)"/>
```

Next, let's enclose the macro code in double quotes:

```excel
=\"<sel value="between(date;20110101;20110331)"\"/>
```

Then, we'll precede each double quote within the macro code with another double quote so that Excel will not interpret any of them as the end of the formula:

```excel
=\"<sel value=""between(date;20110101;20110331)""\"/>
```

Our last step is to replace each hard-coded value with a reference to its corresponding input cell, so we'll change the start date value to a reference to the `startdate` cell and the end date to a reference to the `enddate` cell:

```excel
=\"<sel value=""between(date;"&startdate&";"&enddate&")""\"/>
```

Now, let's test it out!

Change the date range values on the **Dashboard** worksheet so that the start date is 04/01/2011 and the end date is 06/30/2011. (Remember, we want these input values in the form `YYYYMMDD`.)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sales Report</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Date Range:</td>
<td>[20110401]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>start</td>
<td>[YYYYMMDD]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>[20110630]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>end</td>
<td>[YYYYMMDD]</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To run the query, click **1010data** > **Run Queries** > **In Active Workbook** (or press Ctrl+Q).

Then to see the results, click the **Sheet1** tab. The values displayed now show the total sales for each department for the new date range:
It's not very convenient to have to go to one worksheet to enter input values and a different one to view the results. It would be better if we could do everything from our dashboard. In the next step, we'll see how to target the results of our query to the dashboard itself.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupdesc_prod</td>
<td>tct_sales</td>
</tr>
<tr>
<td>SOFT DRINKS</td>
<td>22685213</td>
</tr>
<tr>
<td>SOFT DRINKS SINGLES</td>
<td>5275938</td>
</tr>
<tr>
<td>ENERGY DRINKS</td>
<td>3380641</td>
</tr>
<tr>
<td>NEW AGE BEVERAGE</td>
<td>3265687</td>
</tr>
<tr>
<td>SPARKLING/ENHANCED WATER</td>
<td>2305238</td>
</tr>
<tr>
<td>SODA MIXERS</td>
<td>473634.9</td>
</tr>
<tr>
<td>UNSCANNED BEVERAGE</td>
<td>18379.35</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
</tbody>
</table>
Targeting the query results to the dashboard

Let’s set a result destination for our query results on the dashboard next to our input values.

Let’s say we want our result values to be placed starting at cell D2 on the Dashboard worksheet:

![Sales Report](image)

Click the _1010q Sheet tab to modify the q-sheet. The Result Destination is a reference to the top left cell of the range where the query results will be pasted. This can be an address or a defined name. Let’s create a defined name for cell D2 on the Dashboard worksheet.

On the Formulas tab, click Define Name. In the New Name dialog, enter query_results for the Name and enter the following for Refers to: =Dashboard!$D$2, then click OK.

Note: If, for whatever reason, you need to modify a defined name that you have created, you can access it by clicking the Name Manager button on the Formulas tab and editing it within the Name Manager dialog. In addition to the names that you have defined, you will notice that there are other names that are needed by the Excel Add-in to work properly, which should not be altered in any way.

In the Result Destination field on the q-sheet, enter query_results.

Since we will most likely run this query more than once, let’s make sure that the contents from the results of any previous run of the query are cleared before we paste the results of the current query. To do this, click Define Name. In the New Name dialog, enter query_results_range for the Name and enter the following for Refers to: =Dashboard!$D$2:$E$500, then click OK.

Note: This assumes that our result data will be less than 500 rows, which is probably safe for this example; however, you would want to specify a range that makes sense for you.

In the Clear range before pasting field on the q-sheet, enter query_results_range.

We would also like the column labels to be displayed at the top of the result columns instead of the column names, so let’s select "Column Labels" from the Column Headers drop-down.
Our q-sheet should now look similar to the following:

![Excel Sheet Image]

Now, let's test it out. To run the query, press Ctrl+Q.

Then, click the **Dashboard** tab to see the results:

![Dashboard Image]

You can see that the results have been pasted in our dashboard right next to our input values. In the next section, we'll see how to apply basic and conditional formatting to these results.
Formatting the results on the dashboard

You can apply basic and conditional formatting in Excel on the results that you got from running your 1010data query.

Basic Formatting

Since this is a dashboard, we can add a little formatting to make it more visually appealing. So, for our query results, let's change the cells containing the column headers to be bold, and let's center the text in those cells both horizontally and vertically.

We'll also change the format of the **Sum of Extended Sales** column to appear as dollar amounts. Right-click the column heading in Excel and click the $ from the context menu:

The values in that column will now be formatted correctly:
These formats will be retained every time the query is run. We can verify that by changing the dates and running the query again. Change the start date to 20110701 (07/01/2011) and the end date to 20110930 (09/30/2011). To run the query press Ctrl+Q (or click 1010data > Run Queries > In Active Workbook). When the query finishes running, you should see the following results:

You can see that the values in the Sum of Extended Sales column have changed to reflect the new date range, and the formatting has been kept intact.

**Conditional Formatting**

We can also incorporate conditional formatting for the results from our 1010data query within our Excel workbook. Let's say we want the format the range of values in the Sum of Extended Sales column using a color scale. On the Home tab, click the Conditional Formatting button, and select Manage Rules... from the menu. The Conditional Formatting Rules Manager dialog is presented:
Let's create a rule to apply conditional formatting using a color scale:

1. Click the New Rule... button.
2. In the New Formatting Rule dialog, change the Format Style to 3-Color Scale, then click OK.
3. In the Applies to field, enter =query_results_range, so that the rule will be applied to any of the values that appear within the range we defined for our results earlier.
4. Click Apply to create the new rule.

Let's also create a rule to show cells that have negative values with a red fill color:

1. Click the New Rule... button.
2. From the Select a Rule Type list, select Format only cells that contain.
3. Under Edit the Rule Description, make sure the first drop-down is Cell Value, change the second drop-down to less than, and enter 0 in the third drop-down.
4. Click the Format button.
5. In the Format Cells dialog, click the Fill tab.
6. Under Background Color, click the red box, then click OK.
7. In the New Formatting Rule dialog, click OK.
8. In the Conditional Formatting Rules Manager dialog, in the Applies to field, enter =query_results_range.
9. Click OK.

The results of our query are now formatted with a color scale:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Sales Report**

**Date Range:** 20110701

<table>
<thead>
<tr>
<th>Group Desc</th>
<th>Sum of Extended Sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOFT DRINKS</td>
<td>$ 23,198,912.50</td>
</tr>
<tr>
<td>SOFT DRINKS SINGLES</td>
<td>$ 5,458,703.31</td>
</tr>
<tr>
<td>NEW AGE BEVERAGE</td>
<td>$ 3,727,620.43</td>
</tr>
<tr>
<td>ENERGY DRINKS</td>
<td>$ 3,532,010.72</td>
</tr>
<tr>
<td>SPARKLING/ENHANCED WATER</td>
<td>$ 2,549,205.85</td>
</tr>
<tr>
<td>SODA MIXERS</td>
<td>$ 516,283.07</td>
</tr>
<tr>
<td>UNSCANNEB Beverage</td>
<td>$ 11,925.32</td>
</tr>
</tbody>
</table>
Our dashboard now shows us the total sales in a certain department within a specified date range and groups those results by the values in the **Group Desc** column in the **Product Master** table. These results are also conditionally formatted using a color scale.

Let's add a simple chart to the dashboard to visually represent these results.
Adding a simple chart to the dashboard

Leverage Excel's charting capabilities to visually represent the results from your query.

Let's combine Excel's easy-to-use charting functionality with the power and speed of running queries on 1010data. Let's create a simple bar chart that shows the total sales for each group returned by our query.

We'll start by creating variables that will represent the x-axis and y-axis values for the chart.

For the x-axis, we will specify the list of groups returned by our query. Go to the Dashboard worksheet, then on the Formulas tab, click Define Name. In the New Name dialog, enter total_sales_category for the Name. For the value of this variable, we'll specify the range of cells containing the groups returned by our query. For our example, in the Refers to field, we'll enter the following:

=Dashboard!$D$3:$D$9

Click OK to create the variable.

Now let's create a variable for the y-axis, which will correspond to the range of cells containing the summarization results for each group. Click Define Name and in the New Name dialog, enter total_sales_sum for the Name. In the Refers to field, we'll enter the following:

=Dashboard!$E$3:$E$9

Click OK to create the variable.

Now that we've created our variables, we can create our chart. Click in any empty cell in the dashboard. Then, on the Insert tab, click Column and select the first 2-D Column chart (Clustered Column):
Position the chart where you want it to appear in the dashboard and resize it as desired:

Now let's incorporate the x-axis and y-axis variables we created earlier.

Let's start with the y-axis:

1. Right-click anywhere on the chart and click **Select Data...** from the context menu.
2. In the **Select Data Source** dialog, click the **Add** button under **Legend Entries (Series).**
3. In the Edit Series dialog, enter Total Sales for the Series name.
4. In the Series values field, enter Dashboard!total_sales_sum, which is the name of the y-axis variable we created earlier.
5. Click OK.

And now, for the x-axis:
1. In the Select Data Source dialog, under Horizontal (Category) Axis Labels, click the Edit button.
2. In the Axis Labels dialog, enter Dashboard!total_sales_category, which is the name of the x-axis variable we created earlier.
3. Click OK.

The Select Data Source dialog should look similar to the following:

Click OK.

We now have a bar chart showing the total sales for each group:

which appears on our dashboard:
The chart will dynamically update when the query is run with different parameters. For instance, if we change the start date and the end date and then run the q-sheet, the dashboard (including the chart we just created) will update with the results.

**Note:** When we created the x-axis and y-axis variables for our chart, we specified a set range of cells for each (e.g., =Dashboard!$D$3:$D$9), so that our chart displayed the extended sales for the seven groups in department 19. This is fine if we know the results of our query will always be the same number of rows, or if we know that we just want to chart the top seven items returned by our query. But what if our query returns more than seven values and we want our chart to display all of them? For example, in the next section, we're going to give the user the choice to summarize extended sales either by the group description or the brand. Since there are more than seven brands in our data set, not all of them will be displayed as our chart is currently defined. Later in this tutorial, we'll see how to modify this chart to be more dynamic so it can handle a varying number of rows returned by the query.
Creating a static drop-down in the dashboard

Let's create a drop-down to give the users of our dashboard the ability to select, from a static set of values, the column by which they want to group the results of the tabulation.

Currently, our query tabulates the sum of extended sales, grouping by the values in the Group Desc column in the Product Master table. Let's make it so that the user can select to group the results of the tabulation either by the group description or by the brand.

We're going to need a place to list the values that will populate the static drop-down. We can use Sheet3 for our drop-down values (or create a new worksheet, if necessary). Let's rename it "Dropdowns".

In the Dropdowns worksheet, enter the following information:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Drop-down Values</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Aggregate by Drop-down</td>
<td>Group</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Brand</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Let's give this range of values a defined name. On the Formulas tab, click Define Name. In the New Name dialog, enter aggregate_by_values for the Name and enter the following for Refers to: =Dropdowns!$B$3:$B$4, then click OK.

Now let's go back to the Dashboard tab and create the static drop-down.

In the Dashboard worksheet, add the label "Aggregate by:" next to where you want the drop-down to appear.

Click the cell that you want to contain the drop-down. In our example, that would be cell B12. Let's give that cell the defined name aggregate_by_dropdown.

From the Data tab, click Data Validation. In the Data Validation dialog, select List from the Allow dropdown, and enter =aggregate_by_values in the Source field:
Then click OK in the Data Validation dialog to finish creating the drop-down.

Now, if you click the newly created drop-down, you should see the two values, Group and Brand, in the list:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Date Range:</td>
<td>20110701</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>start</td>
<td>YYYMMDD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Date Range:</td>
<td>20110630</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>end</td>
<td>YYYMMDD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Aggregate by:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Since we want the tabulation in our query to group by the item we select from the drop-down, we need to associate the choice in the drop-down with its corresponding column name. For instance, if someone chooses Brand from the drop-down, we want the query to specify the column brand_prod for the breaks attribute to the <tabu> operation. Right now, it's hard-coded to be groupdesc_prod:

```xml
<note type="base">Applied to table: retaildemo.salesdetail</note>
<sel value="between(date;20110101;20110331)"/>
<link table2="retaildemo.products" col="sku" col2="sku" suffix="_prod" type="select">
  <sel value="(dept=19)"/>
</link>
<tabu label="Tabulation on Sales Detail" breaks="groupdesc_prod">
  <tcol source="xsales" fun="sum" name="tot_sales" label="Sum of Extended Sales"/>
</tabu>
<sort col="tot_sales" dir="down"/>
```

So let's go back to the Dropdowns worksheet and add another column that will associate the drop-down choices with their corresponding column names:
Since our query needs to reference a single cell (similar to what we did in Adding a static input to a dashboard on page 12), we also need somewhere to store the choice that the user makes from the drop-down. Let's add one more column to the Dropdowns worksheet with a cell that will hold that value:

So, for our example, whatever choice the user selects from the drop-down will be stored in cell D3. Let's give that cell the defined name aggregate_by_selected

Let's add the functionality to populate that cell with the user's selection. We can use the Excel function VLOOKUP() to help us do that. Let's add the following formula to cell D3:

=VLOOKUP(aggregate_by_dropdown,'Dropdowns'!B3:C4,2,FALSE)

This says that for the value in the aggregate_by_dropdown, search the first column in the table defined by the range 'Dropdowns'!B3:C4, and return the value in the second column (which we indicate by setting the third parameter to 2), and only return if there is an exact match, which we specify by setting the last parameter to FALSE. Keep in mind that there will always be an exact match since we only have a finite number of defined choices in the drop-down.

So, if the user selects Brand from the drop-down in the Dashboard worksheet, the VLOOKUP function will search the values in the Drop-down Values column on the Dropdowns worksheet and will return the value in the second column, which is brand_prod:
Now that we have the column name, we can insert it into our query, just like we did in *Adding a static input to a dashboard* on page 12.

The line that we need to change in our macro code is:

```
<tabu label="Tabulation on Sales Detail" breaks="groupdesc_prod">
  <tcol source="xsales" fun="sum" name="tot_sales" label="Sum of `Extended Sales" />
</tabu>
```

To reference input values from within the 1010data query, we will need to change the line in our macro code into a formula. We do this by:

- removing any blank spaces from the start of the line
- inserting an `=` at the beginning of the line
- enclosing the macro code in double quotes
- preceding each double quote within the macro code with another double quote so that Excel will not interpret any of them as the end of the formula
- replacing each hard-coded value with a reference to its corresponding input cell using the syntax: `"&[CELL_REFERENCE]&"` (e.g., `"&B3&"`)

Let's go through those steps for our example. Click on the `_1010q Sheet` tab so that we can modify the macro code in our q-sheet.

Since there are no blank spaces at the start of the line, we can move to the next step and add an `="` to the beginning:

```
="<tabu label="Tabulation on Sales Detail" breaks="groupdesc_prod">
```

Next, let's enclose the macro code in double quotes:

```
=""<tabu label="Tabulation on Sales Detail" breaks="groupdesc_prod">
```

Then, we'll precede each double quote within the macro code with another double quote so that Excel will not interpret any of them as the end of the formula:

```
="""<tabu label="Tabulation on Sales Detail" breaks="groupdesc_prod">
```

Our last step is to replace the hard-coded `groupdesc_prod` value with a reference to the `aggregate_by_selected` cell, which contains the value the user selected from the drop-down:

```
=""""<tabu label="Tabulation on Sales Detail" breaks=""&aggregate_by_selected&"">
```

**Note:** There are three sets of double quotes around `&aggregate_by_selected&`. Although this may look odd, it is correct syntax for this Excel formula to work properly.

We can see the line in our macro code now consists of the formula we just created and that it resolves to `brand_prod` within the *1010data Macro Code* section of the q-sheet:
Now let's test this out. Click 1010data > Run Queries > In Active Workbook (or press Ctrl+Q).

When the query completes, click the Dashboard tab to see that the results show the sum of extended sales for each brand:
The results we see are for department 19 only, since that was the department that was hard-coded in our original query. The next step will show how to add a dynamic drop-down that allows the user to choose any department.
Incorporating a dynamic drop-down in the dashboard

The contents of dynamic drop-downs can change based on selections in the dashboard or the results of queries, which means your dashboard will always be up to date even as the data changes. This is historically something very hard to do in Excel-based reporting and a big differentiator with using 1010data and Excel.

Let's take a look at how to create a dynamic drop-down. Our original query calculated the sum of sales for one particular department, which was hard-coded into the query. Let's say we want to give the user of our dashboard the ability to select the department from a list of all the departments in our Product Master table. We could go through that table and list each department individually like we did in the previous step, Creating a static drop-down in the dashboard on page 25, but since our Product Master table has 59 different departments, that would be a bit time consuming and could introduce errors (for instance, if a department number was entered incorrectly). It would be easier (and less susceptible to error) if we ran a simple query on the Product Master table to find out the list of departments in the table and then used those results as the items in the drop-down. Let's create a dynamic drop-down to do just that.

We'll need to write a query that gives us a list of all the departments in the Product Master table, and we'll need to put the results of that query in a place that our dashboard can access. We'll also need to create the actual drop-down on the dashboard and populate it with the results of our department list query.

Since we already have a worksheet that we use for holding the values in our Aggregate by drop-down, let's use that same worksheet to hold the values for our dynamic drop-down. Click on the Dropdowns tab and enter a label similar to the one shown in cell A7 below. Then click in the cell where you want the results pasted, which in our example is B7, and give it a defined name. Let's call it dept_list.

Now that we have a destination for the results of our query, let's create the query.

We'll need to create another q-sheet, since each q-sheet can only contain one query.

From the Add-Ins menu, click 1010data > Add New Q-Sheet. This will open a q-sheet in a new tab labeled _1010q Sheet (2). Let's rename this _1010q Departments, so it's a little more descriptive.

Note: It is important that the worksheet begins with the prefix _1010q, which is used to identify it as a q-sheet to the Excel Add-in.
Let's paste the following query into the **1010data Macro Code** section, which will give us the list of departments from the **Product Master** table:

```xml
<note type="base">Applied to table: retaildemo.products</note>
<tabu label="Tabulation on Product Master" breaks="deptdesc,dept">
  <break col="deptdesc" sort="up"/>
  <tcol source="deptdesc" fun="cnt" label="Count"/>
</tabu>
<colord cols="deptdesc,dept"/>
```

We need to enter some other information in the q-sheet:

- For the **Query Description**, enter "Department List".
- In the **To be Applied to Table** field, we need to specify which table this query should be applied to. For this q-sheet, we'll enter `retaildemo.products`.
- In the **Result Destination** field, enter `dept_list` so that the results of our query will be pasted in the **Dropdowns** worksheet.
- From the **Column Headers** drop-down, select "None", which is the default. We don't need any column headers since the results of the query are going to be used to populate the drop-down.

Finally, before we paste the results of the query, let's clear the contents from any previous run. Create a defined name for the range of cells where the results will be pasted. On the **Formulas** tab, click **Define Name**. In the **New Name** dialog, enter `dept_list_range` for the **Name** and enter the following for **Refers to**: `=Dropdowns!$B$7:$C$500`. 
Note: This assumes that our result data will be less than 500 rows, which is probably safe for this example; however, you would want to specify a range that makes sense for you.

Then, in the _1010q Departments q-sheet, we can enter `dept_list_range` in the Clear range before pasting field.

Our q-sheet should look similar to the following:

To run the q-sheet, click 1010data > Run Queries > In Active Workbook, or press Ctrl+Q.

To see the results, click the Dropdowns tab. You can see that the results of the query have been pasted in the cells that we specified:
Let's create the drop-down on our dashboard and use these values to populate it. Click on the Dashboard tab and add a label where you want the drop-down to appear:

Click the cell that will contain the drop-down. In our example, that would be cell B15. Let's also give it the defined name: department_dropdown.
From the Data tab, click Data Validation. In the Data Validation dialog, select List from the Allow drop-down. For the Source field, you will need to use the following Excel OFFSET function:

=OFFSET(dept_list_range,0,0,COUNTA(dept_list_range),1)

The first parameter specifies the range of cells corresponding to the values in your query-generated drop-down list (which we named dept_list_range), and the second and third parameters tell the OFFSET function to start the list with the top left value in that range. The fourth parameter uses the Excel COUNTA function to calculate the number of non-empty cells in the range, which in essence specifies the length of the list. The last parameter tells the OFFSET function that we just want one column of data from the results.

Dynamic drop-downs require this OFFSET function in conjunction with the COUNTA function to automatically account for the varying length of the list returned from our query. So, our Data Validation dialog will look something like:

Click OK in the Data Validation dialog to finish creating the drop-down.

Now, if you click the newly created drop-down, you should see the list of all the departments returned by the query in the _1010q Departments q-sheet:
Since we want our main query to perform the row selection based on the item we select from the Department drop-down, we need to associate the choice in the drop-down with its corresponding department value. For instance, if someone chooses "DAIRY DELI" from the drop-down, we want the query to specify department 20 in the value attribute to the <sel> operation. Right now, it's hard-coded to be 19:

```
<note type="base">Applied to table: retaildemo.salesdetail</note>
<sel value="between(date;20110101;20110331)"/>
<link table2="retaildemo.products" col="sku" col2="sku" suffix="_prod" type="select">
    <sel value="(dept=19)"/>
</link>
<tabu label="Tabulation on Sales Detail" breaks="groupdesc_prod">
    <tcol source="xsales" fun="sum" name="tot_sales" label="Sum of Extended Sales"/>
</tabu>
<sort col="tot_sales" dir="down"/>
```

So let's go back to the Dropdowns worksheet and choose a cell that will contain the department value associated with the selected item in the Department drop-down (similar to what we did in Creating a static drop-down in the dashboard on page 25). For our example, we'll use cell D7. Let's give that cell the defined name department_selected.
Let's add functionality to populate that cell with the department number that corresponds to the item the user selects from the Department drop-down. We can use the Excel function \( \text{VLOOKUP()} \) to help us do that. Let's add the following formula to cell D7:

\[
=\text{VLOOKUP}(\text{department\_dropdown},\text{dept\_list\_range},2,\text{FALSE})
\]

This says that for the value in the cell named `department\_dropdown`, search the first column in the table defined by the range `dept\_list\_range`, and return the value in the second column, and only return if there is an exact match (which there will always be since we populated the drop-down from this list). So, if the user selects "DAIRY DELI" from the drop-down in the Dashboard worksheet, the \( \text{VLOOKUP}() \) function will search the values in the *Drop-down Values* column on the Dropdowns worksheet until it finds a match, and then return the value in the *Column Values* column, which for our example is 20.
The last step is to reference the selected value in our original query. This process is essentially the same as what we did in Adding a static input to a dashboard on page 12 and Creating a static drop-down in the dashboard on page 25.

The line that we need to change in our macro code is:

```xml
<note type="base">Applied to table: retaildemo.salesdetail</note>
<sel value="between(date;20110101;20110331)"/>
<link table2="retaildemo.products" col="sku" col2="sku" suffix="_prod" type="select">
  <sel value="(dept=19)"/>
</link>
<tabu label="Tabulation on Sales Detail" breaks="groupdesc_prod">
  <tcol source="xsales" fun="sum" name="tot_sales" label="Sum of Extended Sales"/>
</tabu>
<sort col="tot_sales" dir="down"/>
```

To reference input values from within the 1010data query, we will need to change the line in our macro code into a formula. We do this by:

- removing any blank spaces from the start of the line
- inserting an `=` at the beginning of the line
- enclosing the macro code in double quotes
- preceding any double quote within the macro code with another double quote so that Excel will not interpret any of them as the end of the formula
- replacing each hard-coded value with a reference to its corresponding input cell using the syntax: `"&[CELL_REFERENCE]&"` (e.g., `"&B3&"`)

Let's go through those steps for our example. Click on the _1010q Sheet tab so that we can modify the macro code in our q-sheet.

First, remove any blank spaces from the start of the line:

```xml
<sel value="(dept=19)"/>
```

Then, add an `="` to the beginning of the line:

```xml
= <sel value="(dept=19)"/>
```
Next, let’s enclose the macro code in double quotes:

`="<sel value="(dept=19)"/>"`

Then, we’ll precede each double quote within the macro code with another double quote so that Excel will not interpret any of them as the end of the formula:

`="<sel value=""(dept=19)""/>"`

Our last step is to replace the hard-coded department 19 value with a reference to the `department_selected` cell, which contains the value the user selected from the drop-down:

`="<sel value=""(dept=\&department_selected\&)""/>"

**Note:** You must include the set of double quotes around `&department_selected&` for this Excel formula to work properly.

Now we can see the line in our macro code consists of the formula we just created and that it resolves to 20 within the **1010data Macro Code** section of the q-sheet:

![Image](image_url)

Now let’s test this out. Click **1010data > Run Queries > In Active Workbook** (or press Ctrl+Q).

When the query completes, you can see that the results show the sum of extended sales for each brand in department 20:
You can also see that the cell for the last item in the list, which contains a negative value, appears with a dark red fill color, which we specified in **Conditional Formatting** on page 18.
Adding another query to the dashboard

In addition to aggregating total sales by either group description or brand for a specified department and date range, we would also like to aggregate the total sales by date for the same department and date range and display those results in our dashboard.

We have already created the framework to easily add this new query to our dashboard. Basically, we need to perform the following steps:

- Determine where on the dashboard we want our results to go
- Create a new q-sheet for our new query and target the results to the dashboard

Determine the result destination on the dashboard

First let’s figure out where we want our results to go. If we look at our dashboard, we can put our results right next to the other aggregation if we move our chart over to the right a bit:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sales Report</td>
<td>Brand</td>
<td>Sum of Extended Sales</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Date Range</td>
<td>20110701</td>
<td>KIRK</td>
<td>$4,293,391.09</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>20110702</td>
<td>YOPRAIT</td>
<td>2,069,325.38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>20110703</td>
<td>MINUTE</td>
<td>1,671,911.94</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>20110704</td>
<td>TILLANOOK</td>
<td>1,445,131.47</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>20110705</td>
<td>DANON</td>
<td>1,332,479.99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>20110706</td>
<td>HILL</td>
<td>1,026,403.62</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>20110707</td>
<td>JELLO</td>
<td>977,468.63</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>20110708</td>
<td>PILS</td>
<td>951,185.62</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>20110709</td>
<td>TROPICANA</td>
<td>695,868.73</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>20110710</td>
<td>DOLE</td>
<td>594,173.28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>20110711</td>
<td>DANON</td>
<td>599,712.84</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>20110712</td>
<td>TROP</td>
<td>226,526.49</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>20110713</td>
<td>PILLSBURY</td>
<td>221,656.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>20110714</td>
<td>KRAFT</td>
<td>187,871.48</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>20110715</td>
<td>NSTL</td>
<td>130,872.32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>20110716</td>
<td>BROWN</td>
<td>128,956.86</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>20110717</td>
<td>MONTEREY</td>
<td>101,118.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: Since the first query will return results of varying length, it makes more sense to place the results of the second query next to, not beneath, the first query in the dashboard.

Let’s create a defined name for where the results of our query will be placed. On the Formulas tab, click Define Name. In the New Name dialog, enter query_2_results for the Name and enter the following for Refers to: =Dashboard!$G$2, then click OK.

Let’s also create a defined name for the range to be cleared before pasting the results from a subsequent query: Click Define Name. In the New Name dialog, enter query_2_results_range for the Name and enter the following for Refers to: =Dashboard!$G$2:$H$500, then click OK.

Note: This assumes that our result data will be less than 500 rows, which is probably safe for this example; however, you would want to specify a range that makes sense for you.

Create a new q-sheet

Now let’s create a new q-sheet for the query we want to run on 1010data. From the Add-Ins menu, click 1010data > Add New Q-Sheet. This will open a q-sheet in a new tab labeled _1010q Sheet (2).

We need to enter some other information in the q-sheet:

- For the Query Description, enter “Total Sales by Date”.
- In the To be Applied to Table field, enter retaildemo.salesdetail, since we want this query to be applied to the same table as our first query.
• In the **Result Destination** field, enter `query_2_results` so that the results will be placed where we decided on our dashboard.
• From the **Column Headers** drop-down, select "Column Labels".
• In the **Clear range before pasting** field, we'll enter: `query_2_results_range`.

In the **1010data Macro Code** box, enter the following Macro Language code:

```xml
<note type="base">Applied to table: retaildemo.salesdetail</note>
="<sel value=""between(date;"&startdate&";"&enddate&")"/>'"</note>
<link table2="retaildemo.products" col="sku" col2="sku" type="select" suffix="_prod">="<sel value=""(dept="&department_selected&")"/>'"</link>
<willbe name="date_formatted" value="date" format="type:date" label="Date"/>
<tabu label="Tabulation on Sales Detail" breaks="date_formatted">  
	break col="date_formatted" sort="up"/>
	tcol source="xsales" fun="sum" name="tot_sales" label="Sum of`Extended`Sales"/>
</tabu>
```

**Note:** The lines in bold are Excel formulas that reference the start and end dates (`startdate` and `enddate`) as well as the department number (`department_selected`) that the user provides on the dashboard. (See [Adding a static input to a dashboard](#) on page 12 and [Creating a static drop-down in the dashboard](#) on page 25 for more details on how to reference input values within 1010data Macro Language code.)

Our new q-sheet should look similar to the following:

![Excel screenshot](image)

**Note:** The lines containing the Excel formulas display the values of the cells that they are referencing. For example, department number 20 appears for the reference to `department_selected` in cell B16 within the **1010data Macro Code** section.

Now let's test this out. Click **1010data > Run Queries > In Active Workbook** (or press Ctrl+Q).

You can see the results from both queries next to one another in the dashboard:
Note: You can format the column headings and the dollar amounts, which appear in the new Sum of Extended Sales column, so that the look across the dashboard is consistent. (See Basic Formatting on page 17 for details.)

It would be even more convenient if we could run the queries directly from our dashboard. The next step will show how to add a button to the dashboard that will allow you to do just that.
We want to be able to run our queries by simply clicking a button on the dashboard.

Click on the Dashboard tab. Let's add the button somewhere around cell B18 on that worksheet:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sales Report</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Date Range:</td>
<td>20110701</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>start YYYYMMDD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>20110930</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>end YYYYMMDD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Aggregate by:</td>
<td>Brand</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Department:</td>
<td>DAIRY DELI</td>
<td></td>
</tr>
</tbody>
</table>

On the Developer tab, click the Insert button to bring up the Form Controls and then click the icon in the top left corner, which corresponds to the Button (Form Control).

**Note:** If the Developer tab does not appear in your ribbon, click File > Options > Customize Ribbon. Under the Customize the Ribbon list on the right side of the dialog, select Developer and then click OK. The Developer tab should now appear in the ribbon.
On the dashboard, drag across the area where you want the button to appear.

In the **Assign Macro** dialog, enter "RunQueries" for the **Macro name** and then click the **New** button:

This will bring up the **Microsoft Visual Basic for Applications** window, which will allow us to execute the RunQueries macro when our new button is clicked. Enter the following code:

```vba
Sub RunQueries()
    Call RunQSheet (Sheets("_1010q Sheet"))
    Call RunQSheet (Sheets("_1010q Sheet (2)"))
End Sub
```

The 1010data Excel Add-in VBA function **RunQSheet** validates and runs the two specified q-sheets, Sheets("_1010q Sheet") and Sheets("_1010q Sheet (2)").
Note: If you don't want the status bar to appear while queries are being run, or if you are running several q-sheets and don't want to click OK after each one completes, you can add the parameter Quiet:=True to the RunQSheet function calls. For example, Call RunQSheet (Sheets("_1010q Sheet"), Quiet:=True)

Let's make sure our VBA project has a reference to the 1010data Excel Add-in VBA library. In the Microsoft Visual Basic for Applications window, click Tools > References and select A1010data from the list of Available References, if it is not already selected.

Click OK, then save the macro by pressing Ctrl+S.

After you close the Microsoft Visual Basic for Applications window, you should see the new button on the dashboard:
Change the text of the button to "RUN" and change the font size and style as you like. When you’re done, right-click the new button and click Exit Edit Text from the context menu.

Now we’re ready to test out our new button. Let’s modify the date range once again so that we can see if the sales totals update. Let’s change the start date to 20111001 and the end date to 20111231. Then click the RUN button.

**Note:** After each query completes, you will be presented with a dialog that you must dismiss before the next query runs.

When the queries complete, we can see the results on the dashboard.
You will notice that the chart has also been updated. However, it only shows the top seven items returned from our first query. Let's modify the chart so that it will dynamically include all of the items returned from the first query. We'll also add a second chart that shows the results of the query that aggregates total sales by date.
Utilize more advanced charting techniques in Excel to show the details in the results returned by our 1010data queries.

Here’s what our dashboard looks like up to this point:

Our chart only shows the top seven items returned from our first query. Let’s modify the chart so that it will dynamically include all of the items returned from the first query. We’ll also add a line chart for the results of the second query to show the total sales by date for the same department.

Making the bar chart more dynamic

Our first query returns the total sales for each group or brand in a specific department. The bar chart we created earlier only shows the top seven items returned by the query, because we hard-coded a range of values for the x-axis and y-axis variables. Let’s modify those variables, so that our chart will dynamically display all of the items returned by the query.

On the Formulas tab, click the Name Manager button. In the Name Manager dialog, click the total_sales_category item in the list, and then click the Edit... button. For the value of this variable, we’re going to use the Excel OFFSET function. In the Refers to field in the Edit Name dialog, we’ll enter the following:

=OFFSET(query_results,1,0,COUNTA(Dashboard!$D:$D),1)

The first parameter specifies the cell from which you want to base the offset, so we’ll specify query_results for that parameter. The list of groups or brands returned from our first query begins at one row below in the same column as the cell named query_results, so we’ll specify 1 for the second parameter to indicate one row below, and we’ll specify 0 for the third parameter to indicate the same column. The fourth parameter uses the Excel COUNTA function to calculate the number of non-empty cells in column D, because all information charted has to correspond to a value along the x-axis, and those values are in column D. The last parameter tells the OFFSET function that we just want one column of data from the results.
Click OK to modify the variable.

Now let's modify the y-axis variable. In the Name Manager dialog, click the total_sales_sum item in the list, and then click the Edit... button. In the Refers to field in the Edit Name dialog, we'll enter the following:

```
=OFFSET(query_results,1,1,COUNTA(Dashboard!$D:$D),1)
```

For the first parameter, we specify query_results like we did for the x-axis variable. The list of total sales returned from our first query begins at one row below and one column to the right of the cell named query_results, so we'll specify 1 for the second parameter to indicate one row below, and we'll specify 1 for the third parameter to indicate one column to the right. For the fourth parameter, we'll use the Excel COUNTA function again to calculate the number of non-empty cells in column D, because all information charted has to correspond to a value along the x-axis, and those values are in column D. The last parameter tells the OFFSET function that we just want one column of data from the results.

Click OK to modify the variable.

In the Name Manager dialog, click Close.

The chart will automatically update to show all of the items returned from our query:
Creating a line chart

We can also create a line chart that shows the results from our second query, so that we can see total sales by date for the department we have selected.

Once again, we need to create variables that will represent the x-axis and y-axis values for the chart. For the x-axis, we will specify the list of dates returned by our query. The determination of their values follows the same logic as the variables we created for the bar chart in the previous section, so we won't go into as detailed an explanation here.

On the **Formulas** tab, click **Define Name**. In the **New Name** dialog, enter `daily_sales_date` for the **Name**. In the **Refers to** field, we'll enter the following:

\[=\text{OFFSET}(\text{query}_2_{\text{results}},1,0,\text{COUNTA(Dashboard!G:G)},1)\]

Click **OK** to create the variable.

Now let's create a variable for the y-axis. Click **Define Name** and in the **New Name** dialog, enter `daily_sales_sum` for the **Name**. In the **Refers to** field, enter the following:

\[=\text{OFFSET}(\text{query}_2_{\text{results}},1,1,\text{COUNTA(Dashboard!G:G)},1)\]
Click OK to create the variable.

Now that we've created our variables, we can create our chart. Click in any empty cell in the dashboard. Then, on the Insert tab, click Line and select the first 2-D Line chart (Line):

Position the chart where you want it to appear in the dashboard and resize it as desired:
Now let's incorporate the x-axis and y-axis variables we created earlier.

Let's start with the y-axis:

1. Right-click anywhere on the new line chart and click Select Data... from the context menu.
2. In the Select Data Source dialog, click the Add button under Legend Entries (Series).
3. In the Edit Series dialog, enter Daily Sales for the Series name.
4. In the Series values field, enter Dashboard!daily_sales_sum, which is the name of the y-axis variable we just created.
5. Click OK.

And now, for the x-axis:

1. In the Select Data Source dialog, click the Edit button under Horizontal (Category) Axis Labels.
2. In the Axis Labels dialog, enter Dashboard!daily_sales_date, which is the name of the x-axis variable we just created.
3. Click OK.

The Select Data Source dialog should look similar to the following:
Click OK.

We now have a line chart showing the total sales by date for the department we've selected:

which appears on our dashboard:
The chart will dynamically update when the query is run with different parameters. For instance if we enter 20110401 for the start date and 20110630 for the end date, and we select Group from the Aggregate by drop-down and PRODUCE from the Department drop-down, and then click RUN, the dashboard will update with the results, including the chart we just created:
Using 1010data libraries and blocks

You can make the Excel Add-in even more dynamic by accessing query code in libraries and blocks on 1010data.

One of the most powerful features of 1010data is the use of libraries and blocks to encapsulate Macro Language code. Libraries consist of one or more blocks, each of which contains Macro Language code that can be inserted within other queries. This allows you to keep your query code in a central location and give multiple users access to it. If you need to change the code for any reason (e.g., one of the calculations needs to change), you can simply change the block. Whenever anybody inserts the block into their query, they will get the most up-to-date code.

To see how this works, we'll log into the 1010data web interface to create a library. The library will consist of one block that contains the code for the second query on our dashboard. As you'll recall, this query calculates the total sales by date for a given department.

Note: If you are not already logged into the 1010data web interface, when you attempt to log in, you will see a message prompting you to either re-enter or end your existing session. You must select End existing session, which will log you out of your Excel Add-in session. Consequently, you will need to log in via the Excel Add-in again before you can run your q-sheets.

Our <library>, which consists of a <block> named sales_by_date, would look something like the following:

```xml
<library>
  <block name="sales_by_date" start="20110101" end="20110331" department="19">
    <sel value="between(date;{@start};{@end})"/>
    <note type="link">The following link is to table: All Databases/Retail Demo Data/Product Master</note>
    <link table2="retaildemo.products" col="sku" col2="sku" type="select" suffix="_prod">
      <sel value="(dept={@department})"/>
    </link>
    <willbe name="date_formatted" value="date" format="type:date" label="Date"/>
    <tabu label="Tabulation on Sales Detail" breaks="date_formatted">
      <tcol source="xsales" fun="sum" name="tot_sales" label="Sum of `Extended Sales"/>
    </tabu>
  </block>
</library>
```

You'll notice that the code within the <block> is almost identical to the code that we used in Adding another query to the dashboard on page 41.

You'll also notice that the <block> takes three parameters: start, end, and department. The parameters are then referenced in the query using the {@PARAM_NAME} syntax (e.g., {@start}).

Note: We specify default values for these parameters in the <block> definition (e.g., start="20110101").

The library is then saved as a Quick Query in a folder on 1010data.

Note: Users who want to use any of the blocks in the library must be given the appropriate permissions to the requisite folders and tables. See How to Share Quick Queries and Folders in the Quick Start Guide for more information.

For this example, let's say that the path to the Quick Query containing this library is: uploads.t632564076_rd_jtaylor.
We can then go to the _1010q Sheet (2) tab, where the q-sheet containing our second query is located, and replace all of the query code in the 1010data Macro Code section with a call to our block:

```
<import path="uploads.t632564076_rd_jtaylor"/>
<insert block="sales_by_date" start="20110101" end="20110331" department="19"/>
```

We use the <import> operation to make the <library> available to our current query, and then we use the <insert> operation to insert the sales_by_date block code in our query.

Note that in the <insert> statement, we have provided hard-coded values for the start, end, and department parameters, but since we want our block to be called with the input values from the dashboard, we need to change the hard-coded values to references to the appropriate cells in the Excel workbook.

To reference input values from within the 1010data query, we will need to change the line in our macro code into a formula. We do this by:

- removing any blank spaces from the start of the line
- inserting an = at the beginning of the line
- enclosing the macro code in double quotes
- preceding any double quote within the macro code with another double quote so that Excel will not interpret any of them as the end of the formula
- replacing each hard-coded value with a reference to its corresponding input cell using the syntax: "& [CELL_REFERENCE]&" (e.g., "&B3&")

So, the line containing the <insert> command would change to:

```
="<insert block="sales_by_date" start=""&startdate&"" end=""&enddate&"" department=""&department_selected&""/>
```

Note: There are three sets of double quotes around each of the variables (e.g., ""&startdate&"). Although this may look odd, it is correct syntax for this Excel formula to work properly.

To see step-by-step examples of how to do this, see the sections Creating a static drop-down in the dashboard on page 25 and Incorporating a dynamic drop-down in the dashboard on page 31.

Our q-sheet would look similar to the following:
You can see in the **1010data Macro Code** section that the actual values corresponding to the `start`, `end`, and `department_selected` inputs on our dashboard appear in the code:

```
1010data Macro Code:
<note type="base">Applied to table: retaildemo.salesdetail</note>
<import path="uploads.t682664076_rd_jtaylor"/>
<insert block="sales_by_data" start="20110401" end="20111030" department="36"/>
```

If we change the values on the dashboard:

The values in our `<insert>` call in the q-sheet will change as well:
Excel Reporting with 1010data | Using 1010data libraries and blocks | 59

Note: Before you run the q-sheets, you may have to log in once again. From the Add-Ins tab, click 1010data > 1010data Login..., enter your credentials, and then click Secure Login.

Now, if we go to the dashboard and click the RUN button, the second query will run the code from our <block>, and the dashboard will update accordingly:

Libraries and blocks give you flexibility in providing your users with the most up-to-date query code. In this way, you can provide templates to your business users and change metric calculations without needing to update or send new Excel templates. If something within the block changes, the Excel Add-in will automatically leverage the new definition going forward.
Hiding worksheets and locking the workbook

If you are planning to share your workbook with other users, you may want to hide all q-sheets and lock the workbook to prevent accidental edits to your query code.

Hiding worksheets

Because our dashboard is self-contained, there is no need for users to see or access the q-sheets or, for that matter, the Dropdowns worksheet we used to hold the values for the drop-downs in our dashboard. It is also a good idea to hide q-sheets to prevent the user from changing the query code.

You can hide a worksheet by right-clicking its tab and then selecting Hide from the context menu:

For our example, we would hide the following worksheets:

• _1010q Sheet
• _1010q Sheet (2)
• _1010q Departments
• Dropdowns
• Sheet1

Note: In fact, you could delete Sheet1 since it is no longer needed.

Once those worksheets are hidden, our workbook looks like:
To unhide worksheets you have previously hidden, right-click anywhere along your worksheets pane at the bottom of your workbook and click **Unhide**. From the **Unhide** dialog, you can select which worksheets you would like to be visible again:

**Locking the workbook**

To prevent users from unhiding worksheets, you can lock the workbook. On the **Review** tab, click **Protect Workbook**: 

In the **Protect Structure and Windows** dialog, make sure that **Structure** is selected:
If you want to require the user to provide a password to unhide worksheets, enter a password in the **Password (optional)** field.

Click **OK** for the changes to take effect.

**Note:** Once the workbook has been locked, you need to click **Protect Workbook** on the **Review** tab to be able to make changes to the workbook. If a password is required, you must enter the password in the **Unprotect Workbook** dialog and click **OK**.

Now your dashboard is ready to send out to others!