

Building the Super Tables Behind Data-Driven Operations

By Richard G. Lamb, PE, CPA, ICBB; Analytics4Strategy.com

“Data-driven” is defined as processes, decisions and activities spurred by data rather than only experience and intuition. How strongly a firm is data-driven rests upon the fit, breadth and depth of its insight deliverables—system standard reports, task spreadsheets, business intelligence and [data analytics](#)—it weaves into the function and control of its operations. They, in turn, rest upon a firm’s ability to capture and store all of its operational data and then join them into the super tables.

Operational data occurs and are stored naturally in the process of doing the enterprise’s work. Many thousands of tables exist inside and outside (see article, “[Purge the Fused Spreadsheets That Undermine Data-Drivenness](#)”) our operating systems. However, unless we know how to join single tables from many sources, the insights with which our organizations function are limited to what our systems give us—not much in the grand scheme of things.

This article will explain how to slip the surly bonds of our systems and sources. The explanation will be made in the context of the most available software for the purpose—MS Access. The depth of explanation will enable its readers to begin building super tables.

The Grassroot Software—MS Access

There are more than one available software with which to build super tables. The explanation of the article will be demonstrated with a software in the MS Office suite—Access. There are practical reasons for the choice.

First, Access is a “grassroot” software to any firm for which Excel is grassroot. This means that Access is a software embedded in a firm’s technological culture of systems, software and skills. This is reflected in the next several reasons to initially build a firm’s capability to be data-driven

with the software (see the article, [“First Things First; Become Capable of Data-Driven Operations”](#)).

Second, process operatives and analysts are already significantly prepared to absorb the software. This is because the skills that have become ubiquitous through the firm’s history with Excel transfer directly to Access.

Third, most organizations already have license rights to the software through their MS Office license. Individuals only need to request that the software be downloaded to their computer. At the same time, just as for Excel, there are no limitations on allowed seats.

Fourth, there are of course other query-type offerings. However, Access can do almost all of what the most powerful can. It is easy to work around anything it cannot do—enough that the previous three virtues make Access the choice.

Fifth, one’s career may take them to other systems with query functionality. The experience and skills honed with Access will prepare them to easily step into them.

The Process to Build Super Tables

Figure 1 charts the process of building super tables with the individual tables within and across systems and tables to Excel-based process tasks.

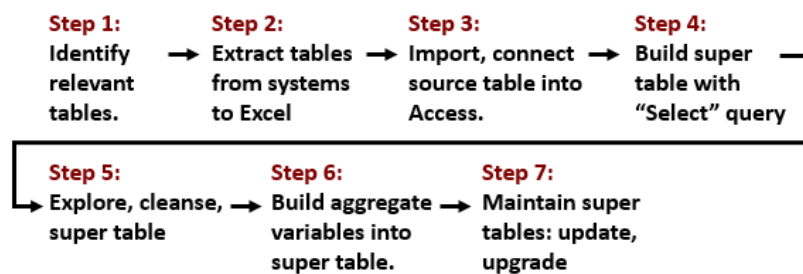


Figure 1: The process to build and maintain super tables.

Step 1 is obviously to identify the individual sources tables. They are selected with respect to each variable we will need for one or more envisioned insight deliverables.

Typically, many of the required source tables are standard reports in the firm’s operating systems. **Step 2** is to extract them by exporting them to an Excel file. Each becomes a worksheet tab in the destination Excel file.

Step 3 takes place in Access. From Access we either “import,” or alternately, “connect” to the source tables. The sources can be Excel, Access or operating systems. The point is that we have no limits as to source and two mainstream ways to bring them into a super table.

Step 4 is the integration of data in Access by building what is called a “Select” query. The data of the imported and connected tables are integrated by joining them with the variables that are common between tables.

Now that we have all data in one super table, **Step 5** is to “know thy data.” This means to explore it every which way while concurrently cleansing data. The same step could be conducted upstream for the individual tables to the Select query.

Step 6 builds upon the Select query by building summary- or aggregate-type variables into the super table. That is, if such variables are envisioned for the super table.

Step 7 is to maintain the super table. This will entail updating subtables to the super table for periodic insight deliverables. Occasionally, it will be to upgrade the super table to serve up newly envisioned insights.

Steps 1 and 2 are self-explanatory. They will not be further explained in the article. Each of the remaining steps will be explained by the sections to come.

Step 3: Import Source Tables to MS Access

Steps 1 and 2 identified the tables we want to join and placed them where they must to be available to build the super table we have in mind. Now we pull them into Access as depicted by Figure 2.

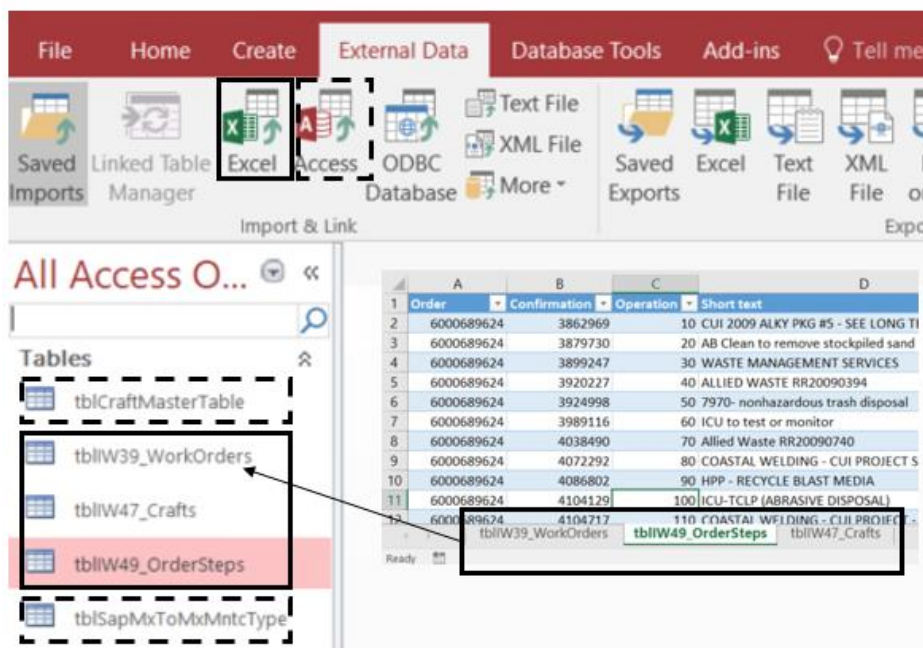


Figure 2: The tables from different origins are made accessible in MS Access.

There are three ways to get at external data from Access—click either one and follow instructions. As shown in the figure, the first and second reach directly out to Excel and Access files. The third, allows us to reach out to other database management systems—something most of us will never be allowed to do.

In the figure, three tables (tblIW39_WorkOrders, tblIW47_Crafts and tblIW49_OrderSteps) were imported from an Excel file. They were originally the outcome of Step 2 to export standard reports to an Excel file. The insert to the figure shows them in the origin Excel file.

The other two are not imported. Instead, they are connected to tables maintained in another Access file. Whenever the query, or any other query connected to the maintained table, is run, each will go to the source file for the latest version.

The figure shows two Access-sourced tables that were built to classify or clarify the data. The table named tblCraftMasterTable brings craft variables from another operating system that are not included in the core three tables. The other, tblSapMxToMxMntcType, was designed to translate a coded variable of SAP to plain English.

Step 4: Build Super Table with “Select” Query

Now all source tables to the super table are reflected in Access. It is time to join them as a single table. This is done by clicking “Select” and pulling the tables into what is called the “design area” of Access. Thence, they are joined by a common variable. The resulting query is given a name. Figure 3 shows what that looks like.

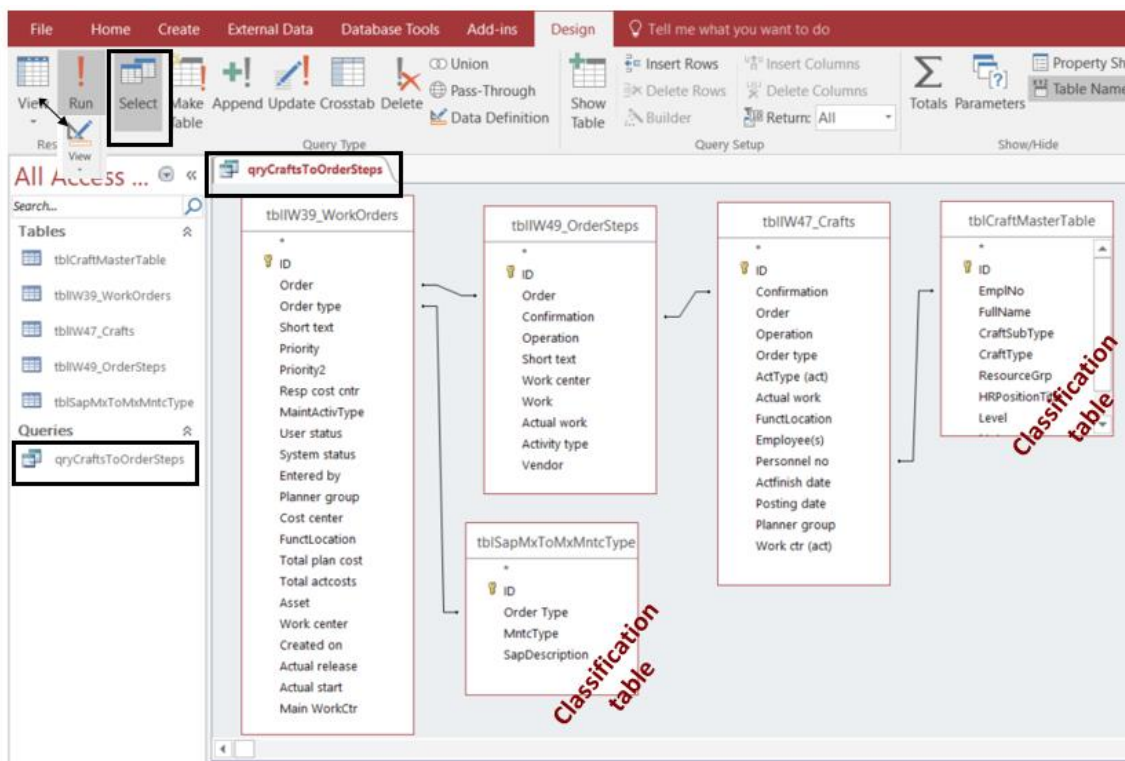


Figure 3: The tables are pulled into the design area and joined.

The joining process—line shown between the tables—is simple. Click on a variable in one table and drag to the equivalent variable in the other table. Each line in the figure is a “join.”

The variables need not have the same name. They are only required to have a common format with respect to text or numeric.

The joins are readily apparent in the figure. The left-most table is joined with the two tables to its immediate right. One is a data table, the other a classification table. However, tables need not be placed in any order.

Note that we are not limited by a single join between two tables. Although not shown, it is possible that a single join does not create “uniqueness.” In such a case, multiple joins between two tables can step over the problem. An alternative is to concatenate two or more variables (to be explained later) to form a common unique variable in both tables and, thence, join on them.

However, there are more than a one type of join. Instead, Access offers three options as shown in Figure 4. We can click on the join line to cause the Join Properties options to appear from which we can select one of the three types of joins.

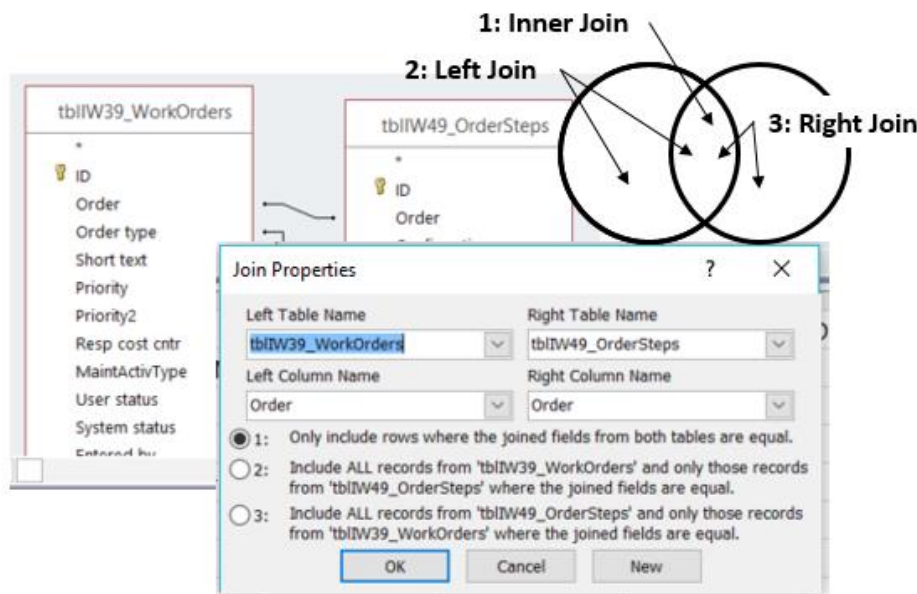


Figure 4: There are three types of joins.

The default join is called an inner join—option 1. In the figure, two tables are inner-joined on “Order.” As the Venn diagram shows, the resulting table will return the cases where both tables have populated the Order variable. A left join—option 2—commands that all cases of the Order variable of tblIW39_WorkOrders be returned along with the populated cases in tbl49-OrderSteps. The right-join is the reverse of the left-join.

There is a fourth join that Access does not make available. It is called an “outer-join.” It is the entirety of the Venn diagram. Imagine the combined result of the left- and right-joins. The work-around is to do a right- or left-join and then append to it the empty (null) variable rows to the opposite join.

From the joined tables, the concept is to pull the variables we want into the grid shown in Figure 5 and shape it with filters and expressions. Now we form the envisioned table as shown in the figure.

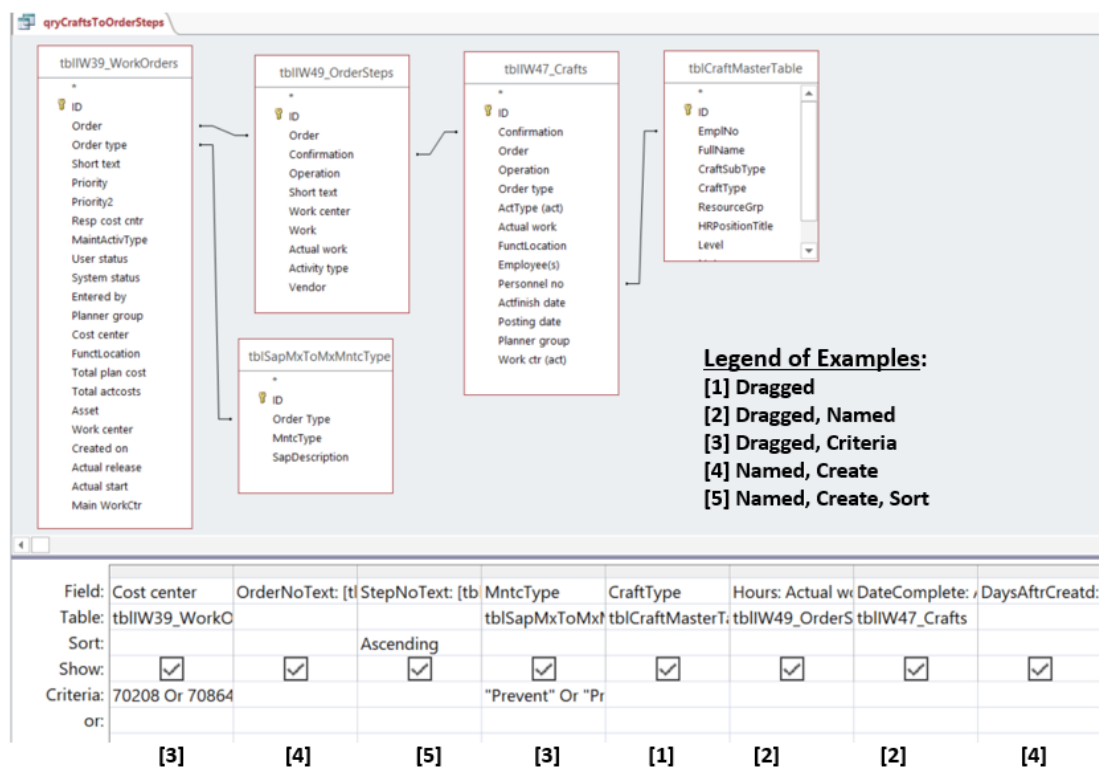


Figure 5: Design is done by dragging variable into the design grid, forming new variables and filtering.

The available operators and expressions we use to shape the table are vast. We all know most of them because of our histories with Excel spreadsheets and Pivots. The article will focus on those that readers will find to be predominate and with which their instincts for Excel will drive them to extend upon. Otherwise, it would be good to spend a little (literally) time with the reference literature.

The most basic action is to simply to drag variables from the tables into the “Field” row of a grid column. When we do, the second row, “Table,” shows the source table to the variable. The “Sort” row allows us to specify whether we want the column in ascending or descending order. The “Show” row, when checked, determines if the variable actually appears in the generated table. The “Criteria” row specifies how the participating columns are to be filtered. The “Or” row allows multiple criteria scenarios.

Another common case is to drag a variable into the design grid, but the name the variable to our desire, rather than be controlled by the source table. An example in the figure is the variable named “Actual work.” However, the designer’s preference is for the table to show the variable to be named as “Hours.” This is done with the expression for the variable in the field row—Hours: Actual work.

Let's look at a case solving several issues. First, we want a variable to be a combination of two variables—Order and Short text. We want to name the combination OrderNoText. The code—`OrderNoText: [tblIW39_WorkOrders].[Order]&"`:
`"&[tblIW39_WorkOrders].[Short text]`—in the Field row makes it work.

Let's break it down. There are several key tricks in the expression that will arise frequently. If you know them, you are in good shape. First, we can see the code that gives the variable a name—`OrderNoText:`.

Next we have to deal with a small problem. Notice that the variable, "Order," exists in two of the tables. When we drag the variable from one of the tables into the grid, a pop-up tells us it cannot make the distinction.

A practical solution is to change the variable name in the source table before or after importing. The other is in the code of the overall expression—`[tblIW39_WorkOrders].[Order]`. Notice that first part is the table in brackets and the second part is the field in brackets. Between them is a dot. This very exactly identifies the variable. Problem solved.

The next piece of the expression is extremely valuable to know. Very often in a table we want to combine two or more variables to be a single variable. The code—`&"`: `"&`—does the trick. The first `&` follows the Order variable with a colon and space. The second `&` follows the space with the Short text variable.

Finally, there is an example of a computed variable in the grid—"DaysAfterCreated." It is created from a computation—`DaysAfterCreated: [ActFinish date]-[Created on]`. Once again the variable is named as we chose. Thence, a formula is defined upon two date variables.

Not shown as a case, but important to make note of are the conditional functions IFF and Switch. The first we all know in Excel as the IF function. The second is a simplifying alternative for nesting IFF cases. It is explained in the referenced self-directed learning text or Google.

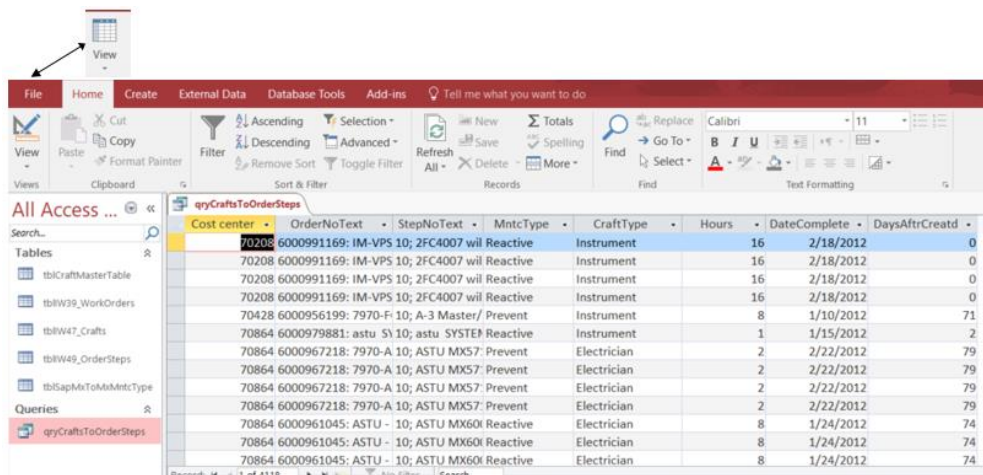
The previous methods are directed to the columns to be the table. The "Criteria" and "Or" rows of the design grid determine the records (rows) to be returned out of all records.

Let's define "Or/And." For example, the CostCenter variable uses "Or" to select particular centers from all centers. The opposite is that "And" places constraints. For example, a greater-less-than case.

Most typically, the "Or" case will play in categorical variables and the "And" case may play in numerical variables. "And" also plays between variables. Because of the "And" between variables, if we want multiple scenarios in the super table, we would place each scenario in an "Or" row.

Once all of the different design choices are made, a table is generated from the choices. This done by clicking the Table view icon on the ribbon to generate the table as shown in Figure 6.

The table can be exported to an Excel file or to another Access file. Alternately, the table can be accessed via a connect at the insight deliverable back to the query.



Cost center	OrderNoText	StepNoText	MntType	CraftType	Hours	DateComplete	DaysAfterCreatd
70208	6000991169	IM-VPS 10; 2FC4007 wil Reactive	Instrument		16	2/18/2012	0
70208	6000991169	IM-VPS 10; 2FC4007 wil Reactive	Instrument		16	2/18/2012	0
70208	6000991169	IM-VPS 10; 2FC4007 wil Reactive	Instrument		16	2/18/2012	0
70208	6000991169	IM-VPS 10; 2FC4007 wil Reactive	Instrument		16	2/18/2012	0
70428	6000956199	7970-F 10; A-3 Master/ Prevent	Instrument		8	1/10/2012	71
70864	6000979881	astu S1 10; astu SYSTEM Reactive	Instrument		1	1/15/2012	2
70864	600097218	7970-A 10; ASTU MX57: Prevent	Electrician		2	2/22/2012	79
70864	600097218	7970-A 10; ASTU MX57: Prevent	Electrician		2	2/22/2012	79
70864	600097218	7970-A 10; ASTU MX57: Prevent	Electrician		2	2/22/2012	79
70864	6000961045	ASTU - 10; ASTU MX60i Reactive	Electrician		8	1/24/2012	74
70864	6000961045	ASTU - 10; ASTU MX60i Reactive	Electrician		8	1/24/2012	74
70864	6000961045	ASTU - 10; ASTU MX60i Reactive	Electrician		8	1/24/2012	74

Figure 6: The table formed from the query to be available to insight deliverables.

The reality is that a query is iteratively built and viewed. The table view will drive us back to the query view for adjustments and finally to the next step—exploring and cleansing.

Step 5: Explore and Cleanse the Super Table

Now to concurrently search for operational insights and bad data. As already mentioned, we will find ourselves exploring the queried data by iterations between table and query views. The iterations will entail interactively changing query criteria and viewing the returned table.

Changing the type of join between tables is also a means to explore. A left- or right-join in conjunction with the “Is Null” criteria is a powerful trick. Empty cells on the opposite side of the join case often reveal cases of inaction, non-compliance to operational rules and new questions.

Other methods allow us to look for data that is bad because they were entered without complying with format rules. A powerful trick is to build a Select query on a single variable, turn on the aggregate functionality (explained in the next section) and select “Group by” from the pull down menu for the “Total” row.

The method returns a list of all variations—some being bad data. Rather than directly cleanse the data, a classification table (see Figures 2 and 3) can be created and kept up to date to cleanse other queries. The table is joined to the source tables and the variable of the classification table is pulled into the design grid.

Updating the classifier table occurs whenever new anomalies are discovered at the time of updating a query or building a new one. Using “Is Null” as a criterion to the “should-be” of the

classifier variable will reveal new cases to be added to the associated cleanse-classification table.

Cleansing may entail transforming variables to a desired state. There are many functions in Access for doing so. They are very similar, if not the same, to those available in Excel.

The possibilities span issues of text case, removing leading and trailing spaces, finding and replacing specific parts of text, parsing text with characters and concatenation. Other than concatenation (explained in Step 4), the article will not explain them. However, readers can find a full explanation in the referenced instructional text or online under the topic of “transformation.”

There is a type of query that looms large in the cleansing process. It is the “Update” query and is visible as a choice among query types in Figure 3.

The process is to first run the Select query to assure we have correctly set the criteria for the variables of interest. Thence, we select the Update query, causing the design grid to take the form shown in Figure 7.

The transformation expression is inserted in the “Update to” row of the query. Upon execution, the variable’s data is cleansed accordingly.

Field:	Cost center	OrderNoText: [tl	StepNoText: [tl
Table:	tblIW39_WorkO		
Update To:			
Criteria:	70208 Or 70864		
or:			

Figure 7: Insert the transforming expression in the Update row of the design grid.

There are two other tools for cleansing. One is to delete rows with a “Delete” query. The other is to test for and remove duplicate rows. This article will not explain them, but once again, readers can find the detail in the referenced instructional text or online.

Access, with its SQL (standard query language) engine, is not graphic. As will be seen in the next section, we can get some small degree of descriptive statistics. An important tool for cleansing is a range of graphic and report outputs to explore a table’s data from various perspectives. The practice will not be shown in this article. The point is that full descriptive statistics are beyond the functionality of Access.

However, the free top-tier, unrestricted, open system software, R, places descriptive statistics within our reach. We can pull the table of the Select query into the R functions for descriptive statistics. The action will reveal interesting patterns, correlations and outliers to explore via the query back to the associated actual records.

Step 6: Build Aggregate Variables into the Super Table

Select queries show every record. Aggregation enhances the query to include measures with respect to groups in the query. We unleash the ability by clicking the summation symbol on the ribbon. When we do, a new row—“Total”—appears in the design grid as shown in Figure 8.

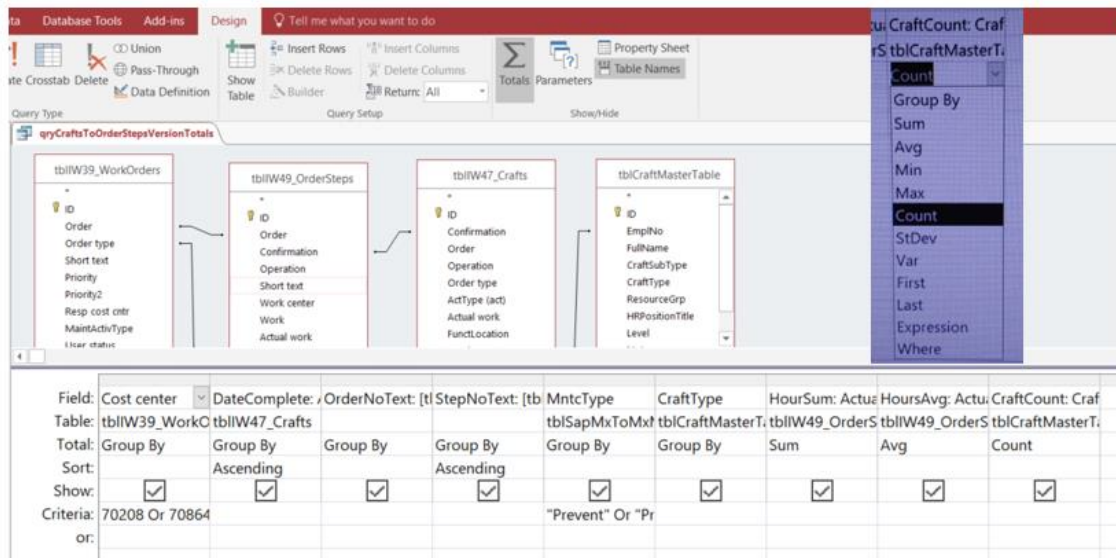


Figure 8: Add the Total row to the design grid by clicking the summation symbol.

The choices for aggregation are made from the shown pull-down menu at each variable. Because aggregation is based on groups, the default is “Group By.” All other choices, other than “Expression” and “Where” are types of aggregations.

Notice that median and mode are not on the list of options. Instead, the shown options are put into play to generate them. Although not explained in this article, the how-to can be found in the referenced literature for self-directed-learning or online.

In the figure, the groups are defined upon six variables. Thence, two variables are created with the “Actual Work” variable—HourSum (HoursSum: Actual work) and HoursAvg (HoursAvg: Actual work)—putting the “Sum” and “Avg” options into play. A third variable is created—CraftCount (CraftCount: CraftType)—pulling the “Count” option into play.

The “Expression” option (not shown in action) allows us to create a variable as a calculation of another. The reader is referred to the referenced instructional text or online for an explanation of the “Where” option.

A person skilled in Excel Pivot will immediately recognize that the same aggregations can easily be accomplished with an Excel Pivot upon the Select query. However, aggregation within Access has powerful ramifications. The most important is that we can build aggregation variables into the Select query.

For example, what if we wanted to compare the individual records to the aggregations of their respective groups? What if we wanted to make the same comparison, but between the records of different groups with very different magnitudes and ranges.

The process begins by setting groups with the Group-By option. Two new variables are created from the numeric variable of interest. For one we would apply the average (Avg) option and for the other the standard deviation (StdDev) option. Thence, we would give the query a new name and join it with its source Select query. Consequently, the select query will for every record have the variables of average and standard deviation for their respective group.

Now to create another insightful variable in the select query as a computation—record minus group average divided by group standard deviation. With the variable we can immediately spot which individual cost records should be subjected to deeper investigation because they exceed “3.” Furthermore, we will not be head-faked by the relative magnitude and range of records between groups—going after meaningless large numbers while stepping past what is truly meaningful.

The possibilities are limited by our creativeness with the functionalities of Access. Another example was to classify order types in a plant where the classifying variable was somewhat vague in the identification of work orders by craft type. In that case, the “Max” and “First” options were put in play with respect to craft type with greatest hours charged to each order—essentially a mode analysis.

Step 7: Maintain Super Tables

In the eyes of insight deliverables, queries are tables. Deliverables make no distinction between what Access calls a table or query. Furthermore, a query can be made into a table with the “Make Table” query. Just a note, sometimes we may want to convert a query to a table before joining it with another query if problems are arising in reaching the envisioned query.

Access tables and queries can be exported to Excel. That is, if the query does not exceed the capacity (1,048,576 rows) of an Excel worksheet. Otherwise, oversized queries must be grouped, by some criterion, in subtables that do not exceed the limit.

The concern for capacity points to one advantage of choosing Access to build and maintain super tables. All insight deliverables can connect to an Access query or table. Once the connection is made, reaching into the table is merely the action of clicking a button in the deliverable.

There are two reason for maintaining the queries. First is when the data is periodically updated. Second is when a table is upgraded.

An “Append” query is used to update a maintained table. It allows us to append each period’s data to the super table via its constituent subtables. A period’s data is imported into Access just as in Step 3 and appended to the subtables.

Figure 9 shows the append query in action. The period's data for the subject subtables are imported into Access. The data is pulled into a Select query and, thence, its variables are pulled into the design grid as shown in the upper section of the figure.

The Append query is clicked, pulling up the dialog box in which the subtable to be updated is selected for the appending action. Thence the design grid pops up as shown in the lower portion of the figure. In it, we confirm that we are correctly matching the variables between the period's and appended tables.

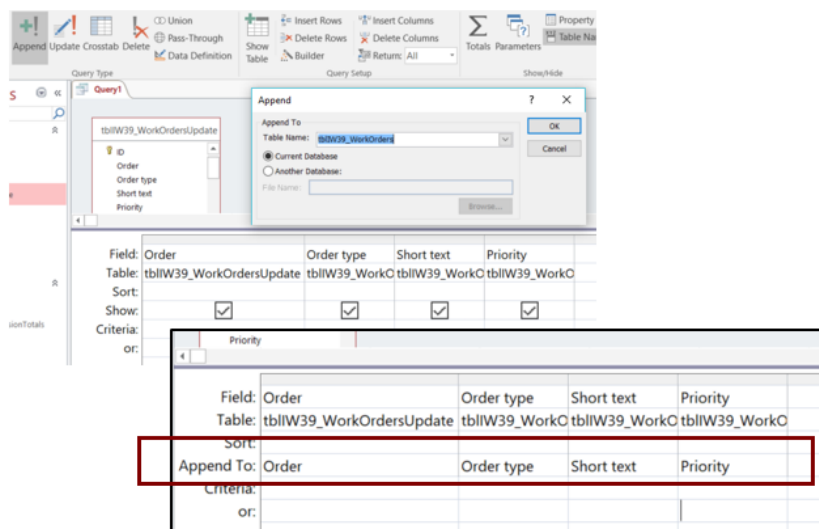


Figure 9: The Append query serves to update queries.

A second occasion of table maintenance is to upgrade queries. This may happen for a host of reasons, including new ideas for teasing data out of the organization's operations data. It may also be to serve a newly envisioned insight deliverable. In this case, we would essentially travel the steps 3 through 6 of Figure 1.

Getting Yourself in the Game

The collaboration of super tables and Excel Pivot is the new Excel spreadsheet—just as Excel worksheets were once the new 11-by-17 green ledger sheet. Happily, for anyone with Excel skills, Access is a very small, easy leap as compared to the leap from ledger sheets to Excel.

The purpose of this article was to explain how super tables are built with Access as the software. The article is enough to get started. However, the reader is well advised to read or watch self-directed learning materials (text or YouTube) that explain how to navigate the Software—open, ribbon, buttons, etc. Beyond that, a person can easily and quickly become a power-user with the text referenced self-directed-learning text; possibly with only YouTube.

In the resources for self-directed learning, a person can limit their focus to the chapters focused on building queries. Because of the article, a reader can identify which do. To give a sense of

effort, the number of text pages across the query chapters is typically less than 200, with many figures, all focused upon tutorials with provided data.

As an optional extra, if the reader is not familiar with the concept of relational databases, the chapters on the topic may be personally beneficial. This is optional because very few of us will ever find ourselves wanting to build a database—we only reach into them via operating systems for our tables. However, the perspective allows us to better understand what is going on behind the curtains of our operating systems and the internet of things that are increasingly appearing in their orbits.

Source for self-directed learning: Access 2016 Bible; The Comprehensive Tutorial Resource; Alexander and Kusleika, 2016, Chapters 8 through 13 | Pivot Tables in Depth for Microsoft Excel 2016, Oeska, 2017.

Related website articles: [First Things First: Become Capable of Data-Driven Operations](#) | [Purge the Fused Spreadsheets That undermine Data-Drivenness](#) | [Keep Your Old Career Hot in the New Age of Data Science](#)

Richard G. Lamb: [Professional Mission and Bio](#)

Educational website: analytics4strategy.com