

# Joining Tables Using SAS® Enterprise Guide®

Anita Measey, Bank of Montreal, Canada

## ABSTRACT

SAS® Enterprise Guide® provides a point and click interface that allows users to manipulate and analyse data without having to write code. But what do you do when your data is not “perfect”? The following examples will show you how to combine data that is “perfect”; where your data types and lengths match and your variable names are the same; where your data types don’t match; include a “having” clause to your grouped data; and adding a subquery to your results. All examples presented were created in SAS® Enterprise Guide® 5.1 and are intended for a beginning level.

## INTRODUCTION

“Perfect” data rarely exists; most people have to deal with some kind of data cleaning. We often get data from different sources that we need to combine together. If the variables you need to join on need to be manipulated, then a point and click solution isn’t always obvious. The following examples show how you can still point and click your way to more complex joins by using the Query Builder, with helpful tools such as the WHERE clause, the HAVING clause, and subqueries.

The Query Builder is a tool that allows you to manipulate data by building a set of instructions that create a query. You can access it on the Workspace Toolbar when you have a data table open; when you right-click on the data icon in the Project Tree or Process Flow; through selecting Tasks > Data > Query Builder. You can create new columns based on values of existing columns, summarize data, sort data, create subsets of your data and join data tables.

## THE PERFECT JOIN

Two data sources:-

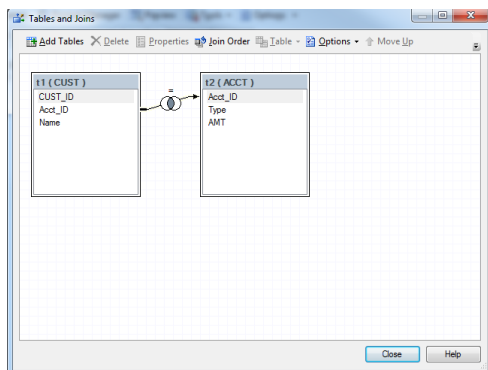
Account

	Acct_ID	Type	AMT
1	1	Sav	100
2	2	Sav	100
3	3	Chq	50
4	4	Sav	25
5	5	Chq	30
6	6	Sav	75
7	7	Chq	10

Customer

	CUST_ID	Acct_ID	Name
1	C_001	1	Tom
2	C_002	2	David
3	C_002	3	David
4	C_003	4	Peter
5	C_004	5	Jane
6	C_005	6	Sarah
7	C_005	7	Sarah

When joining these two tables using the Query Builder, the join is automatically found. The variable name is the same and the data type is the same.



Result:

	CUST_ID	Acct_ID	Name	Type	AMT
1	C_001	1	Tom	Sav	100
2	C_002	2	David	Sav	100
3	C_002	3	David	Chq	50
4	C_003	4	Peter	Sav	25
5	C_004	5	Jane	Chq	30
6	C_005	6	Sarah	Sav	75
7	C_005	7	Sarah	Chq	10

Code created by the Query Builder:

```
%_eg_conditional_dropds (WORK.QUERY_FOR_CUST_0000) ;
```

```
PROC SQL;
```

```
CREATE TABLE WORK.QUERY_FOR_CUST_0000 (label="QUERY_FOR_CUST") AS  
SELECT t1.CUST_ID,  
       t1.Acct_ID,  
       t1.Name,  
       t2.Type,  
       t2.AMT  
FROM WORK.CUST t1  
     INNER JOIN WORK.ACCT t2 ON (t1.Acct_ID = t2.Acct_ID);
```

```
QUIT;
```

## ADDING THE JOIN TO THE WHERE CLAUSE

But what if your data looks like this?-

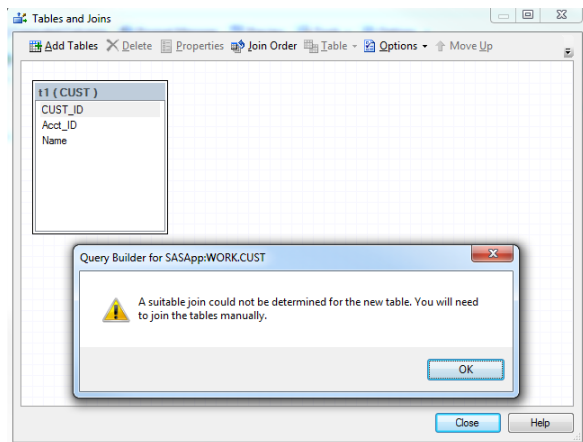
Account\_2

	Acct_IND	AMT
1	S_1	100
2	S_2	100
3	C_3	50
4	S_4	25
5	C_5	30
6	S_6	75
7	C_7	10

Customer

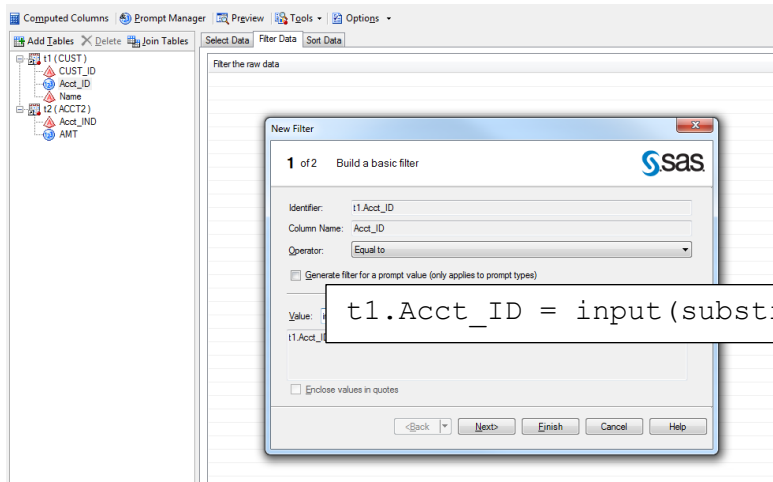
	CUST_ID	Acct_ID	Name
1	C_001	1	Tom
2	C_002	2	David
3	C_002	3	David
4	C_003	4	Peter
5	C_004	5	Jane
6	C_005	6	Sarah
7	C_005	7	Sarah

When you try to join you get a message warning you that no suitable join could be found.

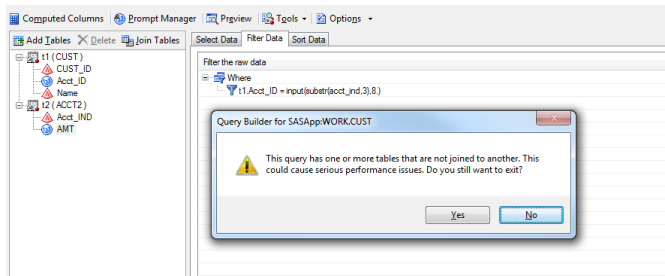


You know your data well enough to know that the person providing the Account\_2 file adds the account type to the start of each account number. If you take off the first two characters from the Acct\_IND and convert it to numeric then you can join on the result.

Create the join in the Filter Data tab as a Where Clause.



Accept the warning and run.



Result:

	CUST_ID	Acct_ID	Name	Acct_IND	AMT
1	C_001	1	Tom	S_1	100
2	C_002	2	David	S_2	100
3	C_002	3	David	C_3	50
4	C_003	4	Peter	S_4	25
5	C_004	5	Jane	C_5	30
6	C_005	6	Sarah	S_6	75
7	C_005	7	Sarah	C_7	10

Code created by the Query Builder:

```
%_eg_conditional_dropds (WORK.QUERY_FOR_CUST);
```

```
PROC SQL;
```

```
CREATE TABLE WORK.QUERY_FOR_CUST AS
SELECT t1.CUST_ID,
       t1.Acct_ID,
       t1.Name,
       t2.Acct_IND,
       t2.AMT
```

```
FROM WORK.CUST t1,WORK.ACCT2 t2
```

```
WHERE t1.Acct_ID = input(substr(acct_ind,3),8.);
```

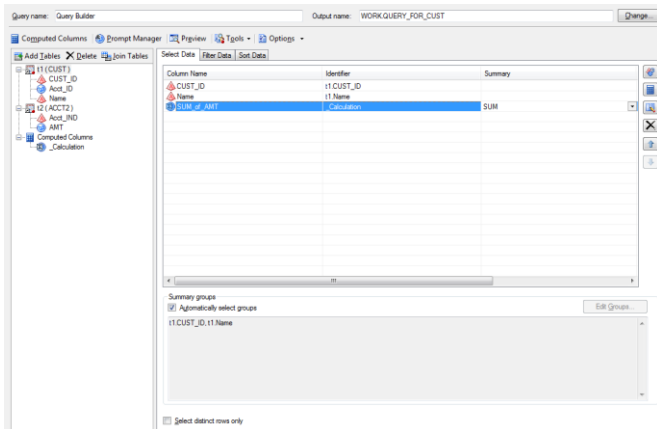
```
QUIT;
```

## USING THE HAVING CLAUSE

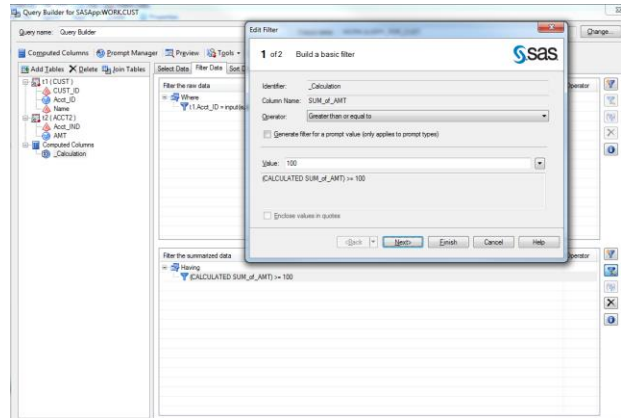
What about when you want to subset your data based on a group variable? You have joined your data but now you only want to select records where a grouped summary meets a certain condition.

If we look at the results from the last example and want to only pick customers who have a total amount of 100 or more then we can add a having clause.

First create the summary data



In the Filter Tab add the summary filter



Code created by the Query Builder:

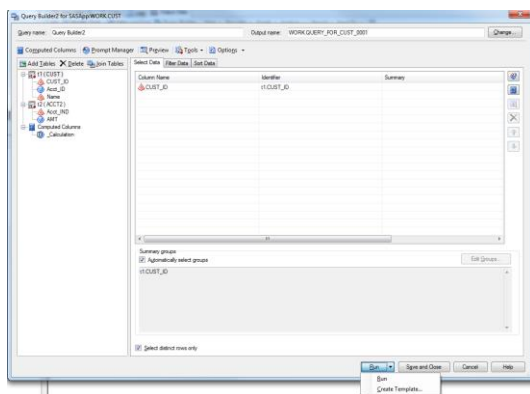
```
%_eg_conditional_dropds(WORK.QUERY_FOR_CUST);

PROC SQL;
  CREATE TABLE WORK.QUERY_FOR_CUST AS
  SELECT t1.CUST_ID,
         t1.Name,
         /* SUM_of_AMT */
         (SUM(t2.AMT)) FORMAT=BEST12. AS SUM_of_AMT
  FROM WORK.CUST t1,WORK.ACCT2 t2
  WHERE t1.Acct_ID = input(substr(acct_ind,3),8.)
  GROUP BY t1.CUST_ID,
           t1.Name
  HAVING (CALCULATED SUM_of_AMT) >= 100;
QUIT;
```

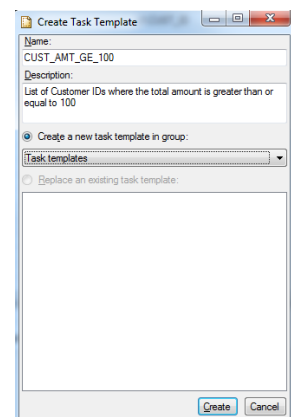
## CREATING AND USING A SUBQUERY

Now we want to use this result and use it to “subset” our original data. We want to pick the details out of the data for any customer who had a total amt greater or equal to 100.

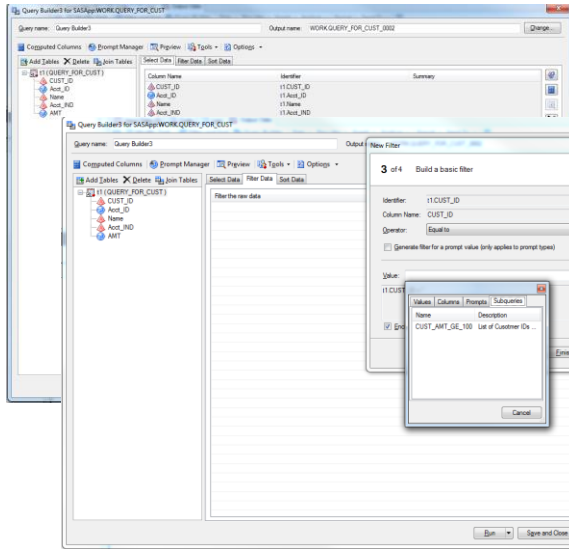
We can register the above query as a “template” then we can call this template into another query.



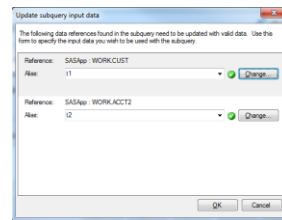
From the previous example, in the “Select” remove the variables that are not required in the subquery\*. Instead of clicking on run, select the dropdown on the run button you now see an option to “Create Template”. Selecting this will bring up the template and fill it in and select the Create button



If we start with the result from the join in the where clause then use the subquery we created to subset that data.



From the data that was created using the where clause join, select the query builder. Select all the columns from the table. Add a basic filter for CUST\_ID. Select the drop down from the Value Box and select the Subqueries tab. A list of valid subqueries that you have registered will be displayed. Select the subquery that you created and a box will pop up with the default table and Alias values assigned. Select the tables required for your subquery.



The subquery will appear in your filter

	CUST_ID	Acct_ID	Name	Acct_IND	AMT
1	C_001	1	Tom	S_1	100
2	C_002	2	David	S_2	100
3	C_002	3	David	C_3	50

Code created by the Query Builder:

```
%_eg_conditional_drops(WORK.QUERY_FOR_CUST_0002);
```

```
PROC SQL;
```

```
CREATE TABLE WORK.QUERY_FOR_CUST_0002(label="QUERY_FOR_CUST") AS
SELECT t1.CUST_ID,
       t1.Acct_ID,
       t1.Name,
       t1.Acct_IND,
       t1.AMT
FROM WORK.QUERY_FOR_CUST t1
WHERE t1.CUST_ID IN
      (SELECT DISTINCT t1.CUST_ID
       FROM WORK.CUST t1,WORK.ACCT2 t2
       WHERE t1.Acct_ID = input(substr(acct_ind,3),8.)
       GROUP BY t1.CUST ID
       HAVING (SUM(t2.AMT)) >= 100);
```

```
QUIT;
```

\*Subquery Rules:

Here are the types of subqueries that you can create using query-based templates:

- Subqueries that return a single value
- Subqueries that return multiple values (multiple rows of a single field)
- Subqueries that appear as part of a filter of the raw data (on the WHERE clause)
- Subqueries that appear as part of a filter on the grouped data (on the HAVING clause)
- Subqueries that appear as part of a recode condition as part of recoding a column in a computed column (on the SELECT clause)

Here are the types of subqueries that you cannot create using query-based templates:

- Subqueries that form a derived table (subqueries that appear on the FROM clause)
- Subqueries that refer to columns on the outer query (correlated subqueries)

## TASK TEMPLATE MANAGER.

The Task Template Manager from the tool bar (Tasks -> Task Templates -> Task Template Manager) allows you to manage your templates, you are able to create groups, edit, delete, add, import and export.

## CONCLUSION

Using SAS® Enterprise Guide® point and click interface to join “perfect” data is quick and easy. Doing more complex joins or adding criteria to those joins can be just as easy with a few simple steps using helpful tools such as the WHERE and HAVING clauses and subqueries.

## REFERENCES

Chris Hemedinger: “The SAS Dummy” June 28, 2013, “Building an SQL subquery in SAS Enterprise Guide”

<http://blogs.sas.com/content/sasdummy/2013/06/28/building-an-sql-subquery-in-sas-enterprise-guide/>

Michael Burke and I-kong Fu: SAS Global Forum 2012 Paper 292-2012 “Finding Your Inner Query with SAS Enterprise Guide”,

<http://support.sas.com/resources/papers/proceedings12/292-2012.pdf>

## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Anita Measey  
Bank of Montreal  
416-867-6728  
Anita.Measey@bmo.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.