Data Export User Guide

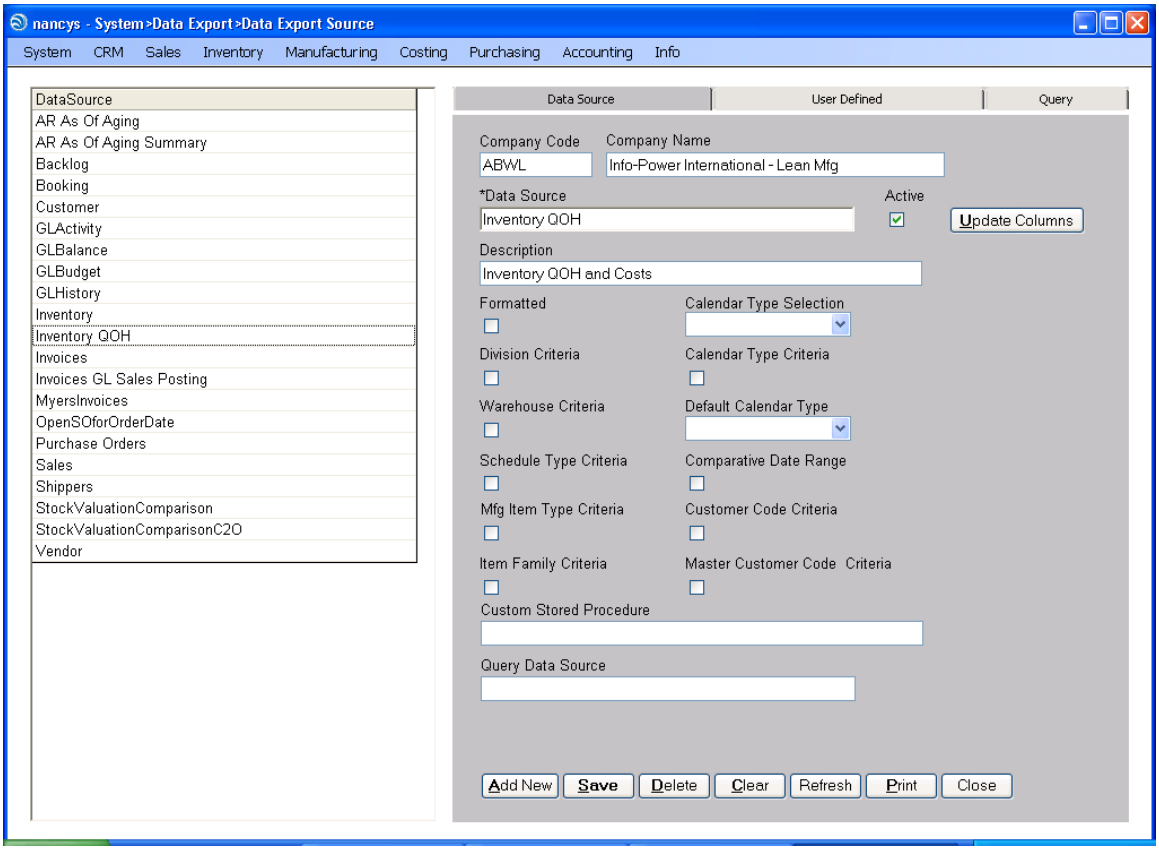
Info-Power International, Inc.
3345 Silverstone • Plano, Texas 75023
(972) 424-4447

Table of Contents

Data Export 3
Creating a Simple Data Export 3
Creating a Data Export with Totals 13
Data Export Modification 19
Creating a Query with User Input for Selection 22
Using Sub-Queries and Joins within a Data Export 29
Example Using Group By and Order By 31
Using SQL Management Studio or Query Analyzer to Create Data Exports 33
Using a Stored Procedure to Produce a Data Export 34
Using an Existing ABW Report to Create a Data Export 40
Additional Examples 46

Data Export

Creating a Simple Data Export



To begin creating a Data Export, open the Data Export Source maintenance function, System > Data Export > Data Export Source.

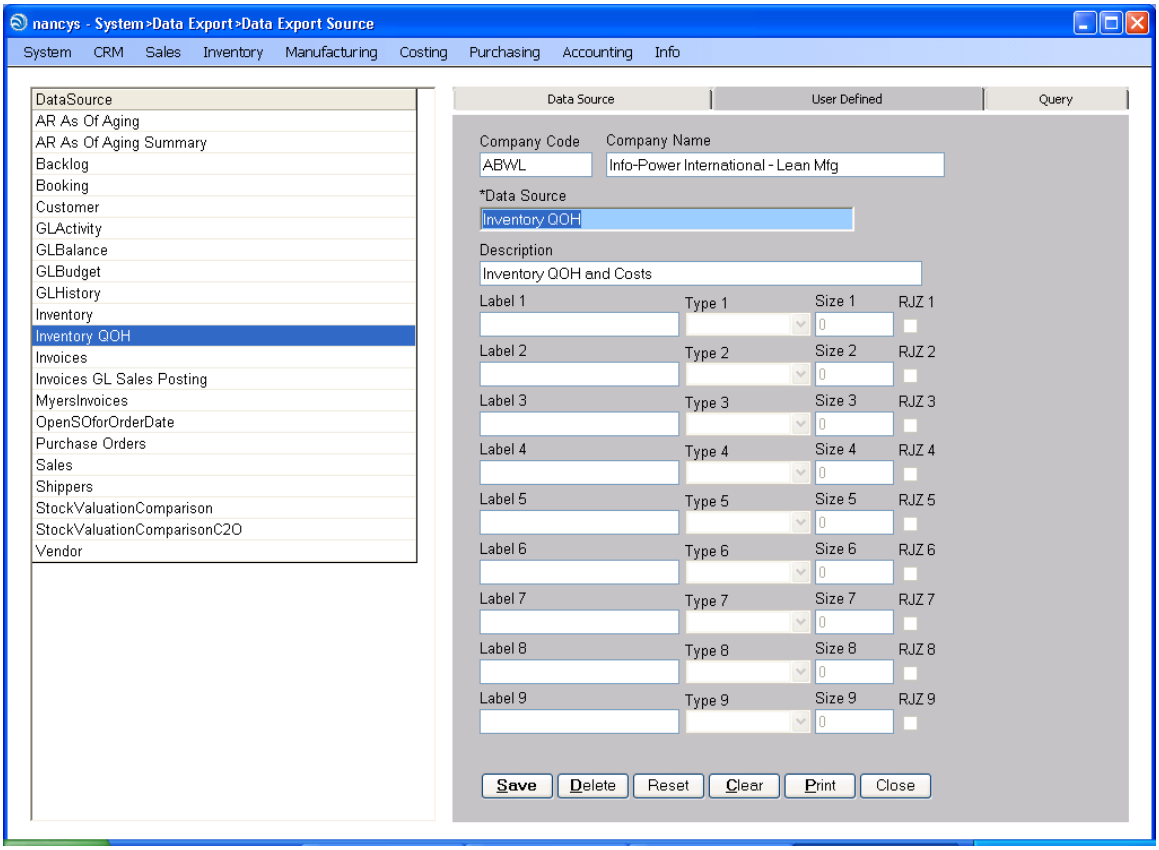
Select an existing Data Export by clicking on the desired export displayed in the grid.

To create a new Data Export, click the **Add New** button.

Enter a unique code for identifying the Data Export in the Data Source field.

Enter a more detailed description of the Data Export in the Description field.

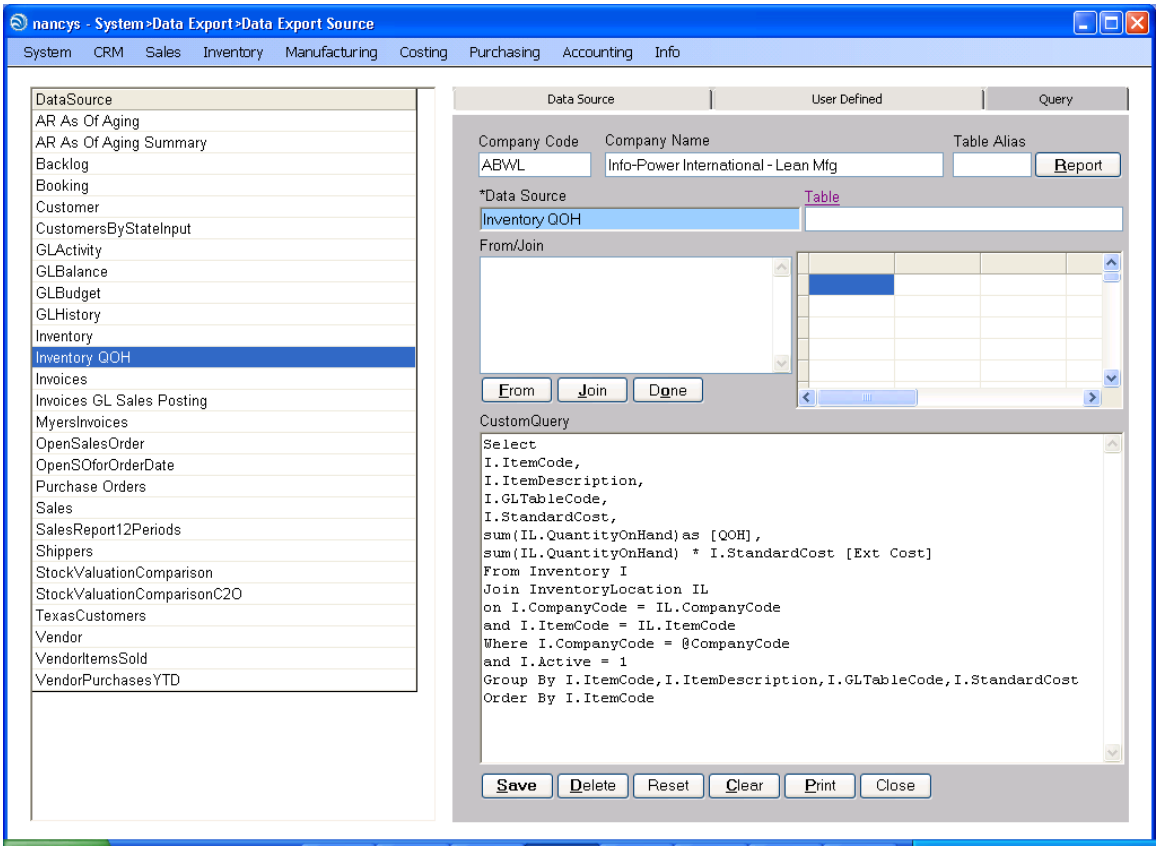
Click the Save button and select the User Defined Tab.



The User Defined tab, allows you to define 9 user inputs that can be used for selection and retrieval logic in your query.

In addition to the 9 user inputs, there are standard variables that can be used for selection:

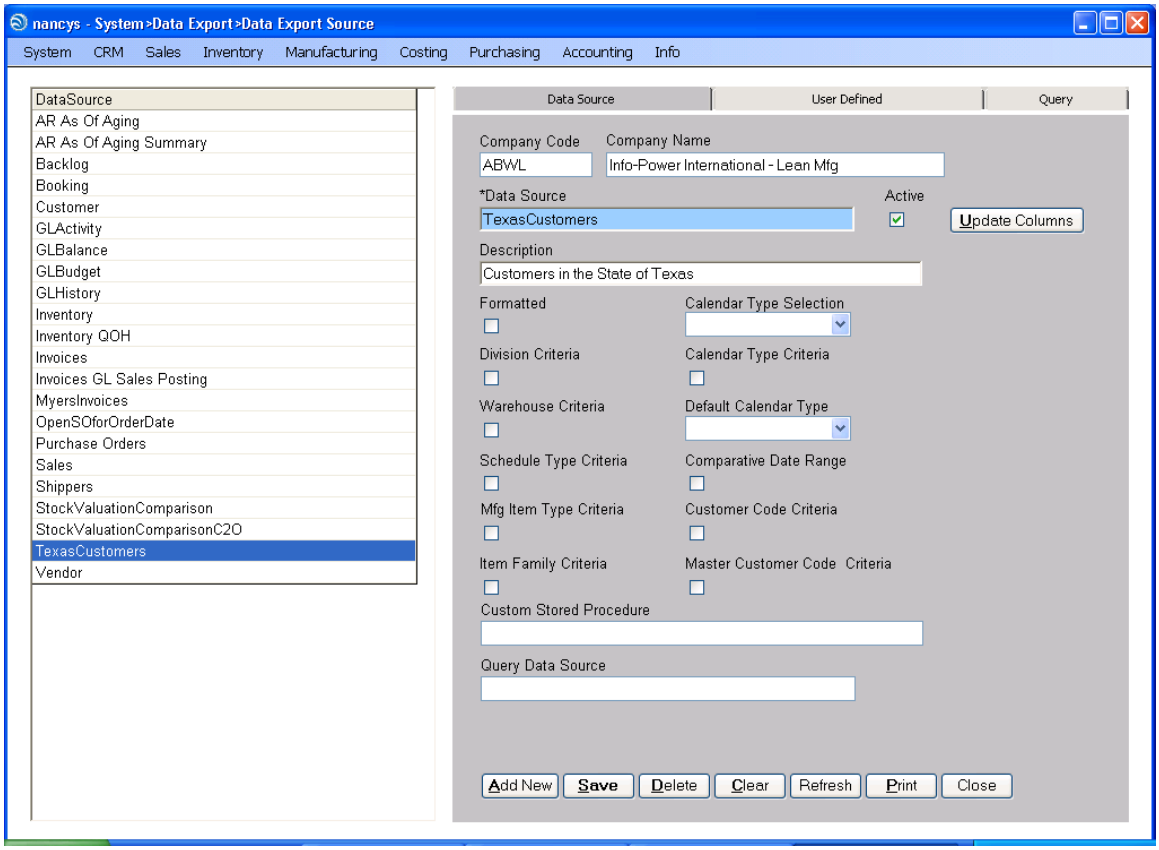
@CompanyCode	char(4)
@UserCode	char(11)



The Query tab allows you to define your query statement.

Now that you have seen an overview of the Data Export source function let's create a simple data export.

Task: To identify all customers located in the state of Texas.



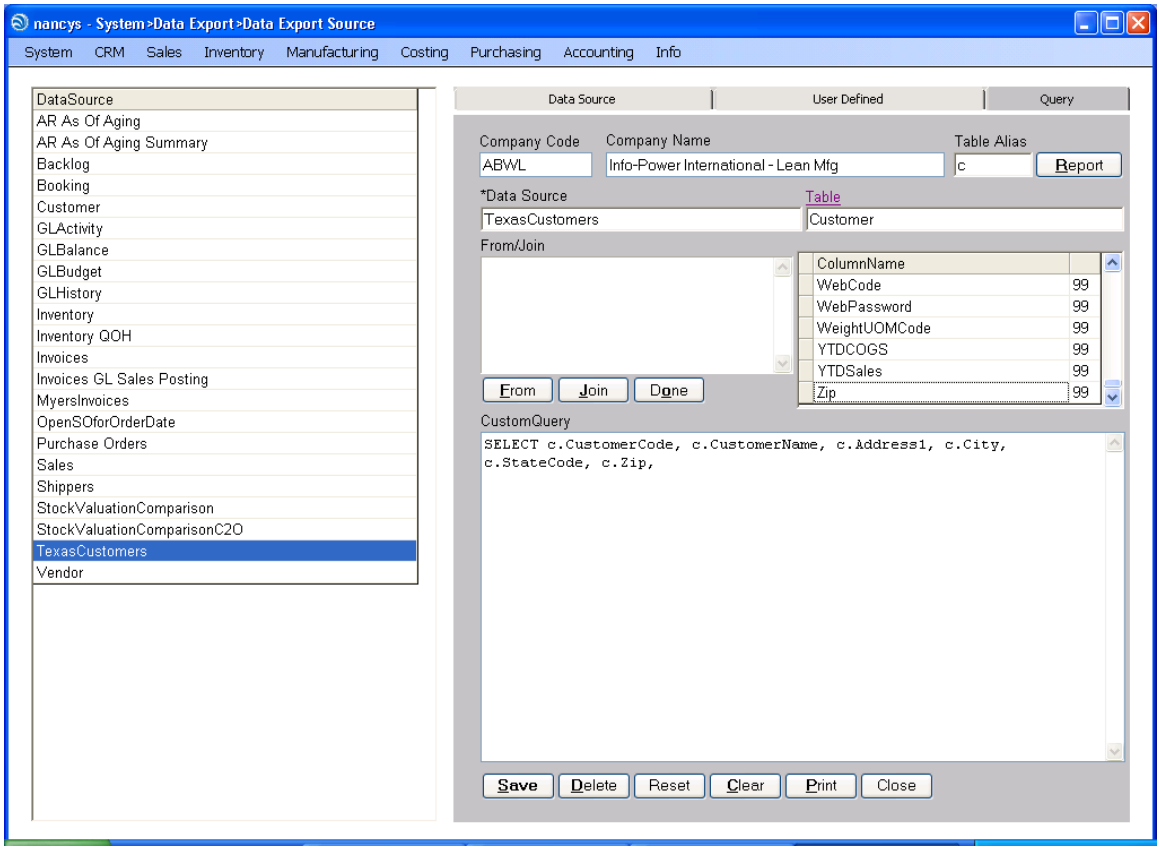
1) Click the **Add New** button.

2) *Data Source: TexasCustomers.

3) Description: Customers in the State of Texas.

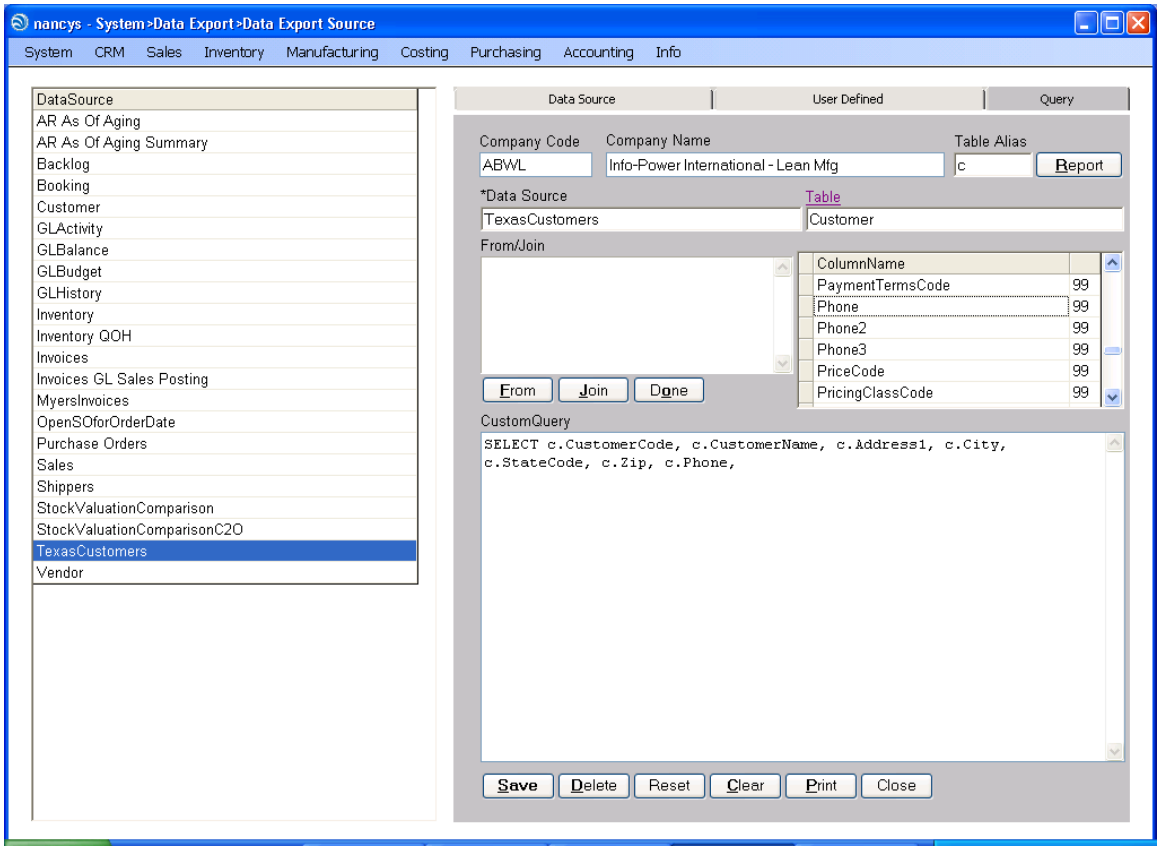
4) Click the **Save** button.

5) Click on the Query tab.

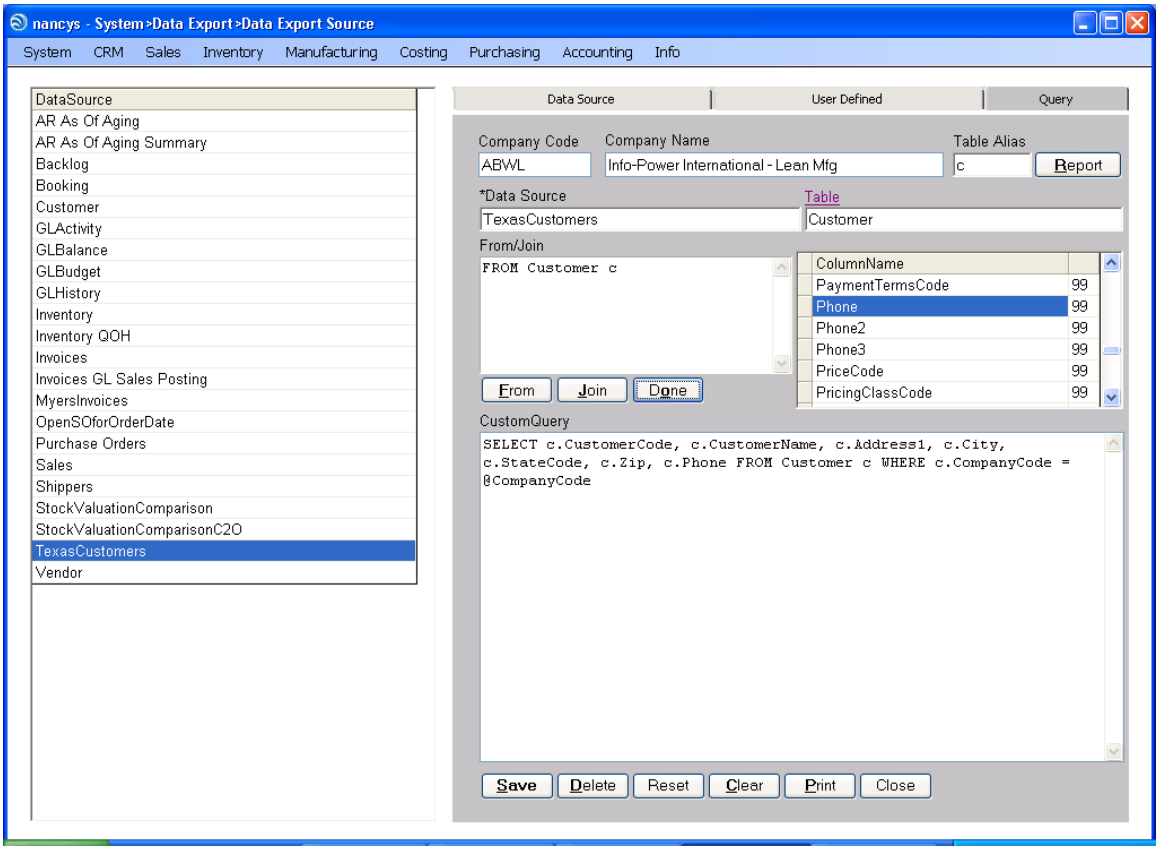


Enter the table name from which you wish to retrieve data or click on the field heading Table to display a listing of tables. Find the Customer table and double click on it to retrieve the columns (fields).

Enter a Table Alias. The table alias is a shortcut to identify the table you are referencing. For example, the company code field is referenced in many tables you must explicitly identify which table to pull company code from. In our example, you would specify Customer.CompanyCode, Customer.Name, etc. The Table Alias lets us shortcut that to c.CompanyCode, c.CustomerName, etc. – Less typing.



Select the following fields from the list: Customer Code, Customer Name, Address 1, City, State Code, ZIP and Phone; by scrolling and finding each field and double clicking on the name or by entering them as discussed on the previous page. Notice as you click on each field it is added to the SELECT statement in the CustomQuery window with the Table Alias appended to the front.



Once you have completed the field selection:

1) Click the **From** button.
2) Click the **Done** button.

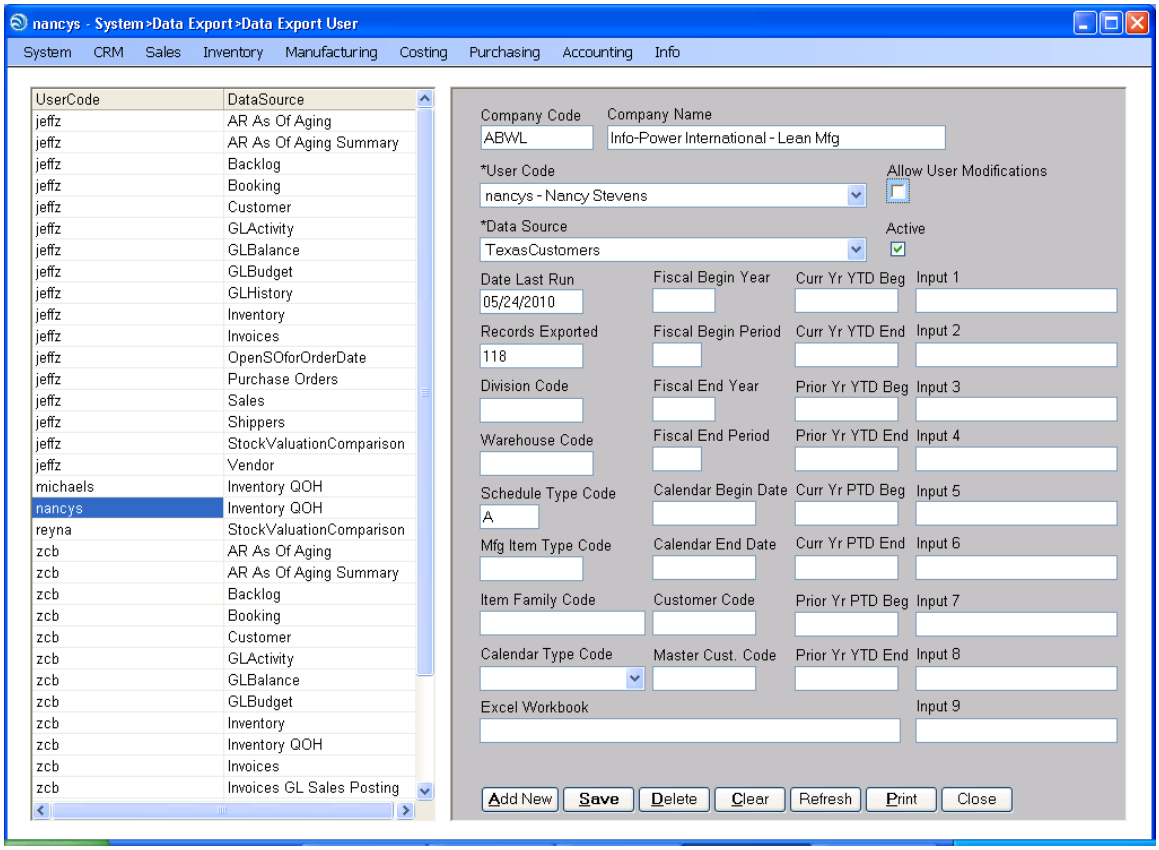
This will add the FROM and WHERE clauses to the query. Notice that the WHERE clause added logical selection on CompanyCode. The @CompanyCode variable is always set to the current company into which you are logged in.

Now, we need to add the selection logic to find all customers within Texas.
Add the following after @CompanyCode.

AND c.StateCode = 'TX'

Click the **Save** button.

The next step is to identify the users that should be able to run your query.



Open the Data Export User maintenance window, System > Data Export > Data Export User.

Click the **Add New** button to create a new record.

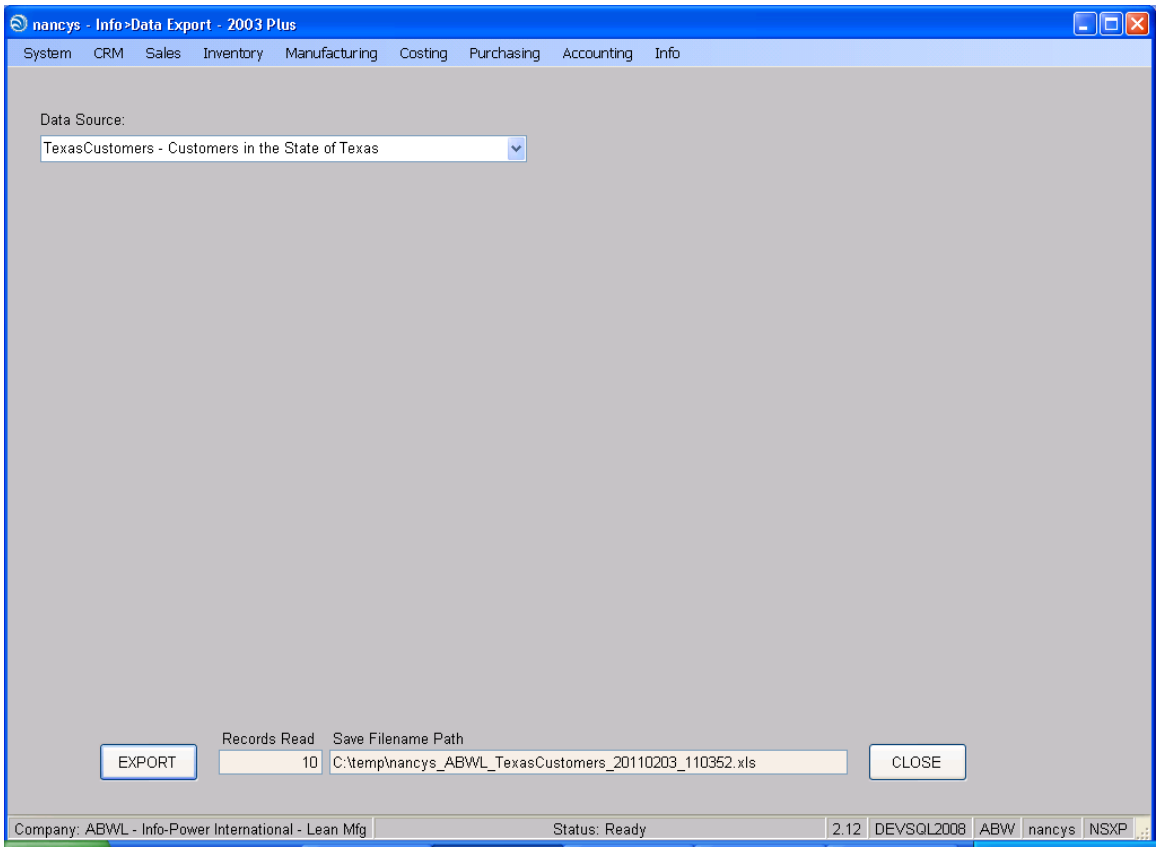
Click on the **Drop Down** for User Code and select the user, for today's purposes select yourself, to which you wish to grant access. To a drop down listing is available for every field that has a down arrow at the end of the field. To access the drop down, click on the down arrow.

Click on the **Drop Down** for Data Source and select the 'TexasCustomers' query.

Click the **Save** button to save your information.

For this example, don't worry about the remaining fields.

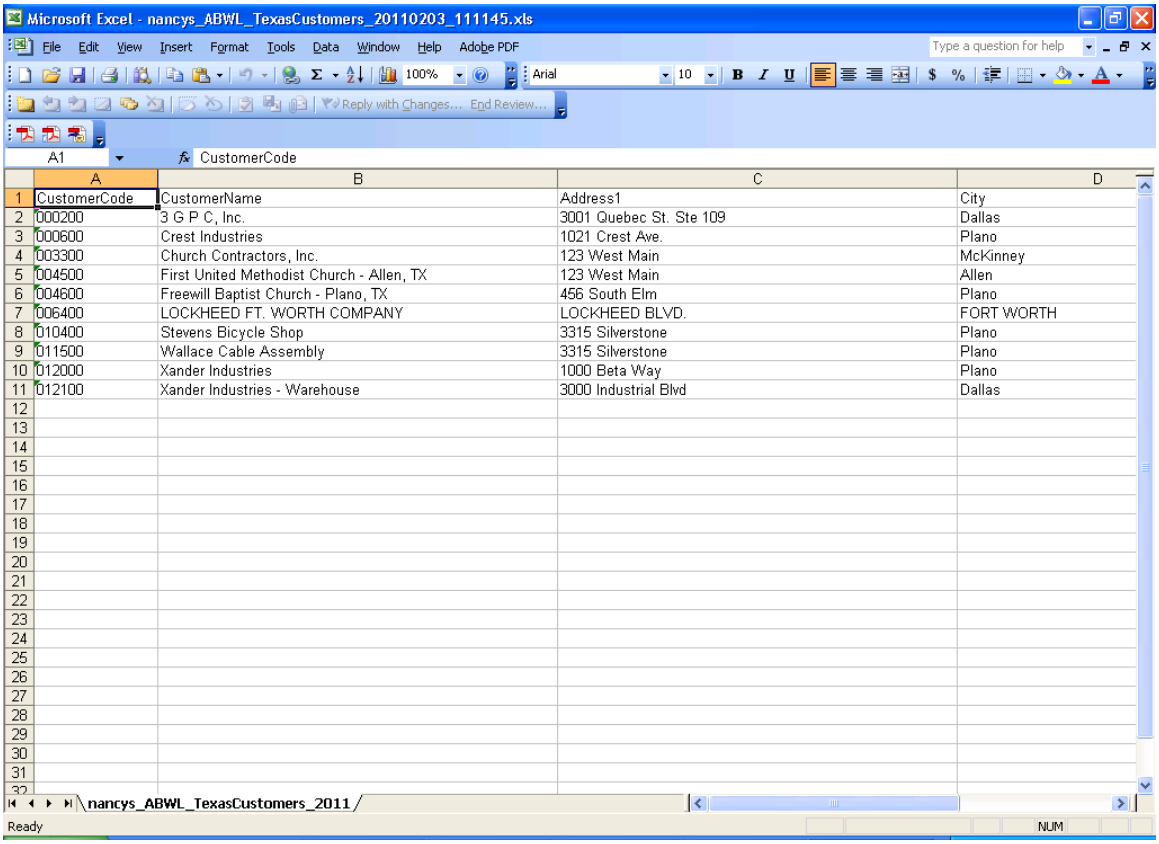
You are now ready to run your query. From the Info menu select Data Export 2003 Plus, Info > Data Export – 2003 Plus.



Using the Drop Down selection box, select your query.

Click the **Export** button.

The query will run and moments later Excel will start and display your results.

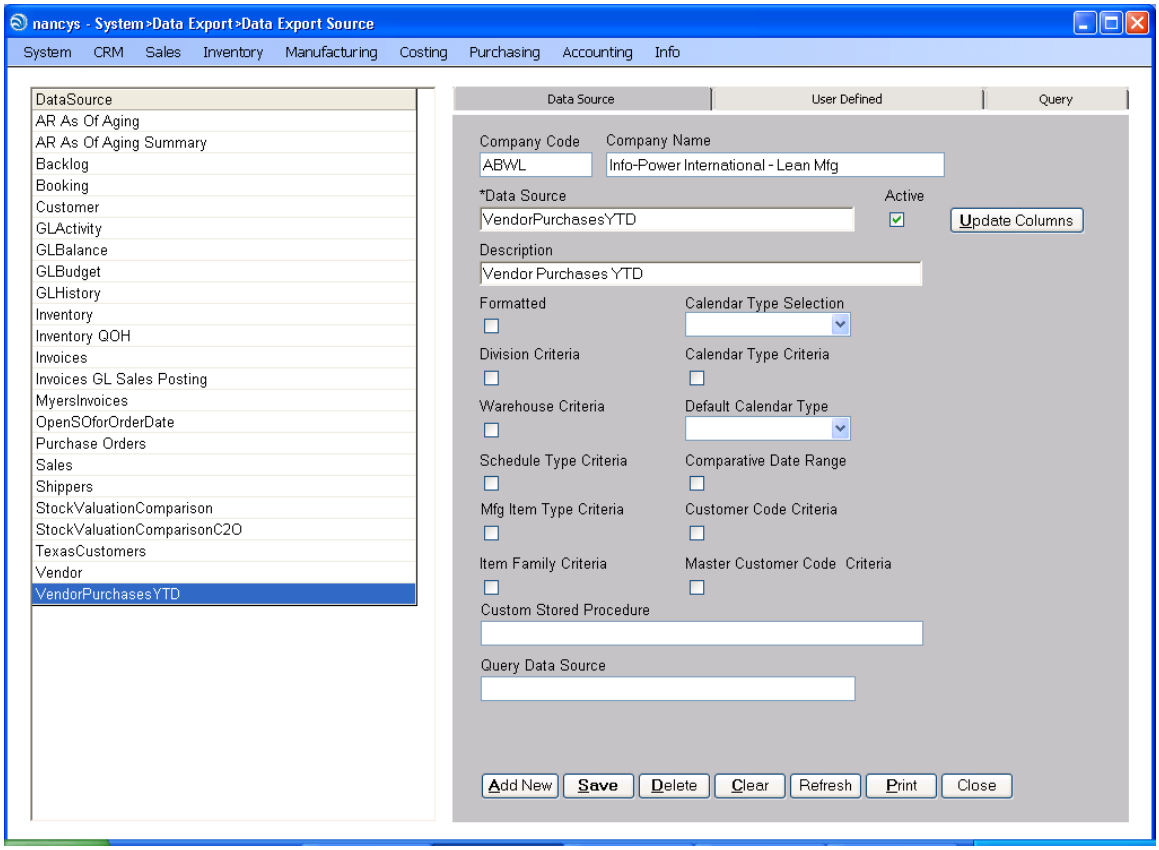


	A	B	C	D
1	CustomerCode	CustomerName	Address1	City
2	000200	3 G P C, Inc.	3001 Quebec St. Ste 109	Dallas
3	000600	Crest Industries	1021 Crest Ave.	Plano
4	003300	Church Contractors, Inc.	123 West Main	McKinney
5	004500	First United Methodist Church - Allen, TX	123 West Main	Allen
6	004600	Freewill Baptist Church - Plano, TX	456 South Elm	Plano
7	006400	LOCKHEED FT. WORTH COMPANY	LOCKHEED BLVD.	FORT WORTH
8	010400	Stevens Bicycle Shop	3315 Silverstone	Plano
9	011500	Wallace Cable Assembly	3315 Silverstone	Plano
10	012000	Xander Industries	1000 Beta Way	Plano
11	012100	Xander Industries - Warehouse	3000 Industrial Blvd	Dallas

Your results should look similar to the display shown above.

You have just created a simple Data Export!

Creating a Data Export with Totals



For this exercise we are going to create a listing by vendor that will show total purchases year to date for each vendor and total purchases for the query.

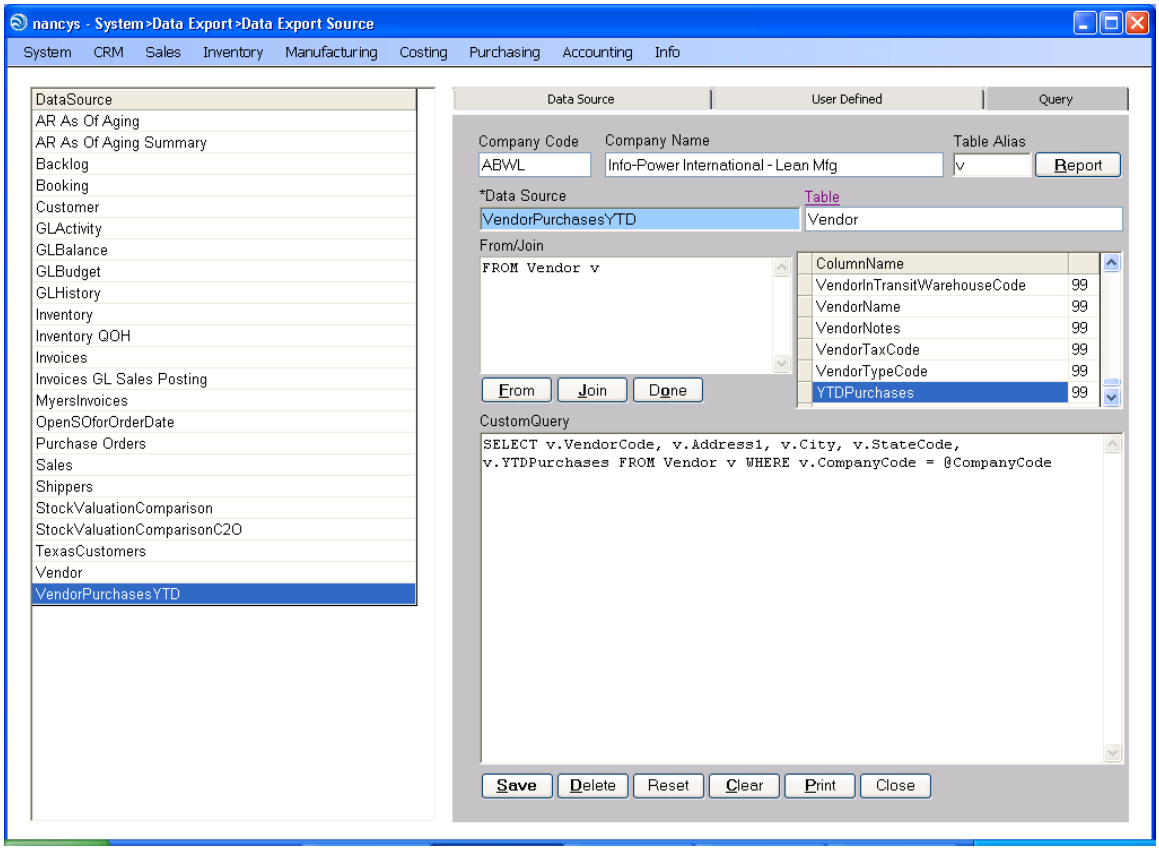
Click the **Add New** button.

Enter VendorPurchasesYTD as the Data Source.

Enter Vendor Purchases YTD as the Description.

Click the **Save** button.

Move to the Query tab.



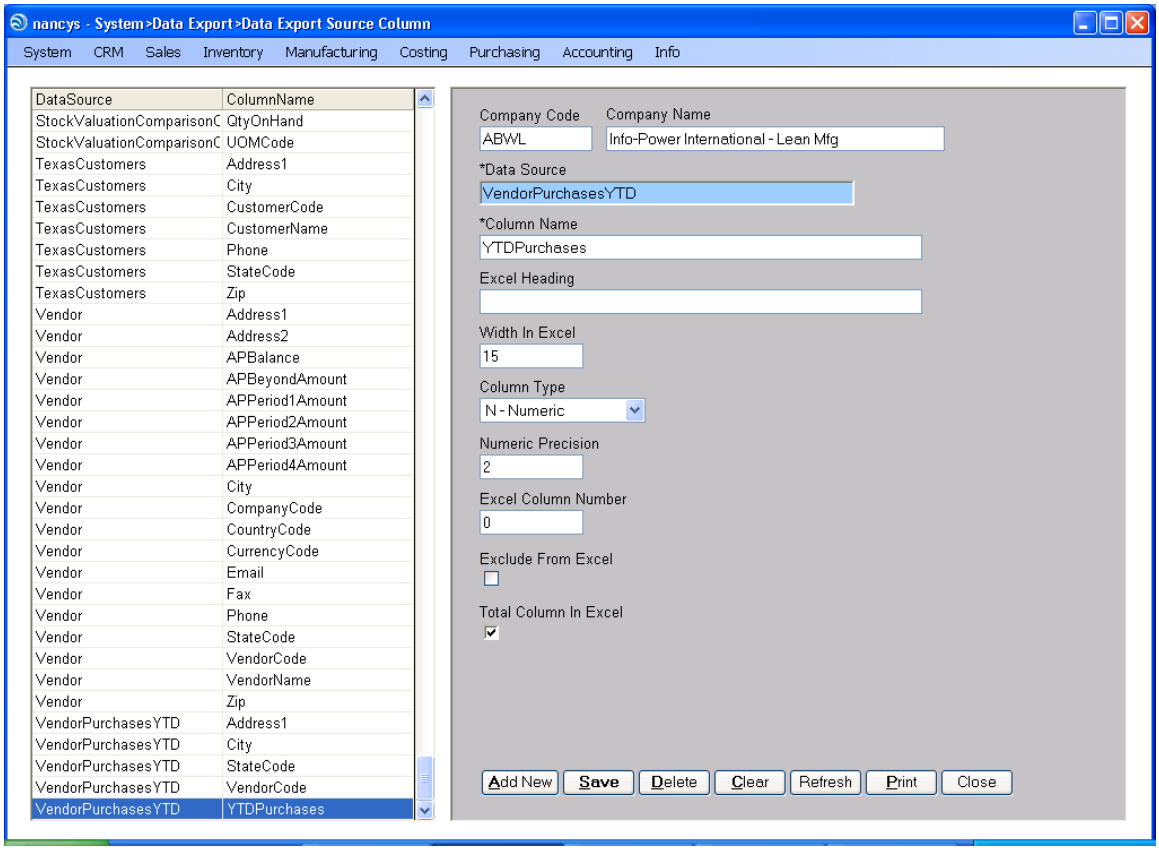
Enter v for Table Alias.

Find the Vendor table by clicking on the Table field heading and scrolling down until you find it. Then select it by double clicking on it to retrieve it back to the form.

Next, select the following fields: VendorCode, Address 1, City, StateCode and YTDPurchases.

Finally, click the **From** button, the **Done** button and the **Save** button.

Next, move back to the Data Source tab and click the Update Columns button. This process will attempt to execute your query and if successful will update the Data Export Source Column table with the columns from your Data Export. If your query cannot be executed you will receive an error message and you must correct the query statement.



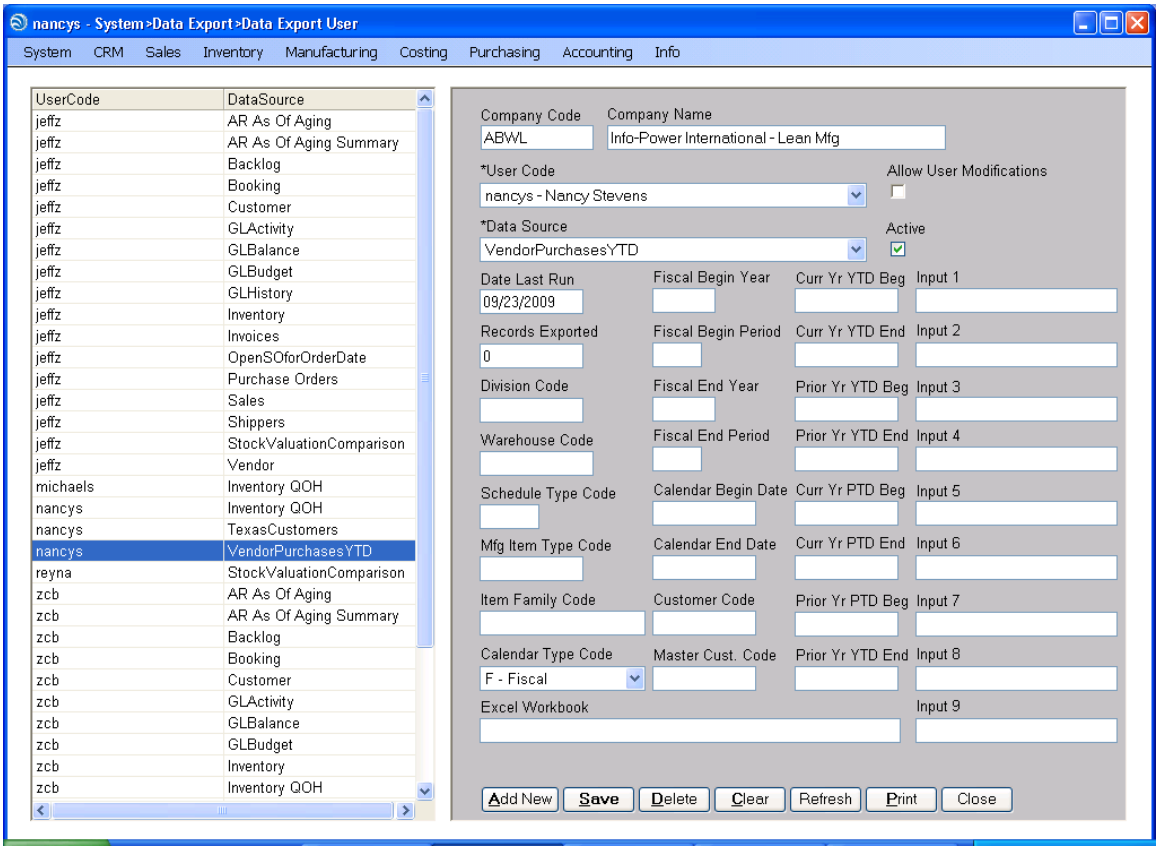
To flag a column to be totaled open the Data Export Source Column maintenance screen, System > Data Export > Data Export Source Column.

Scroll down until your Data Export is displayed, or you can quickly position to your Data Export by clicking in the grid and typing the name, or DataSource, of your Data Export.

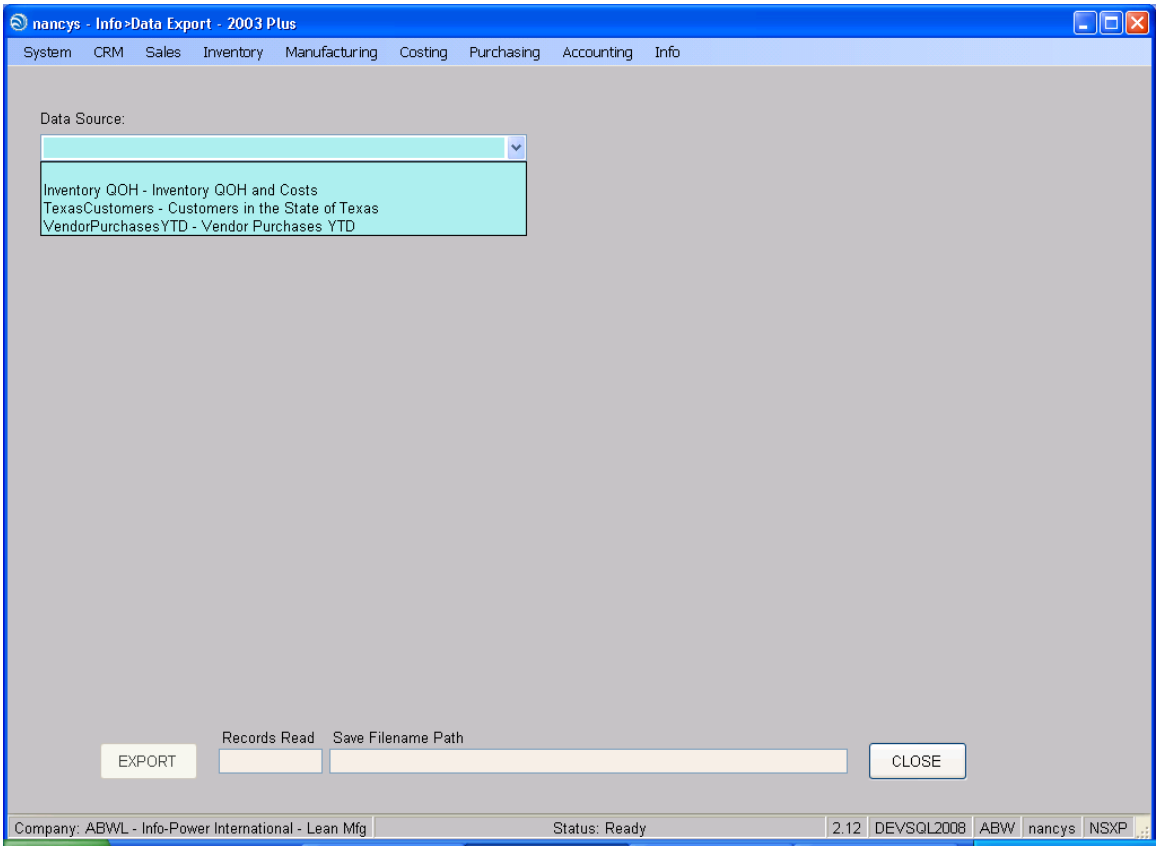
Find and click on the column to be totaled. Place a check in the Total Column in Excel check box.

The remaining fields on the screen are determined by the **Update Columns** process.

Click the **Save** button.

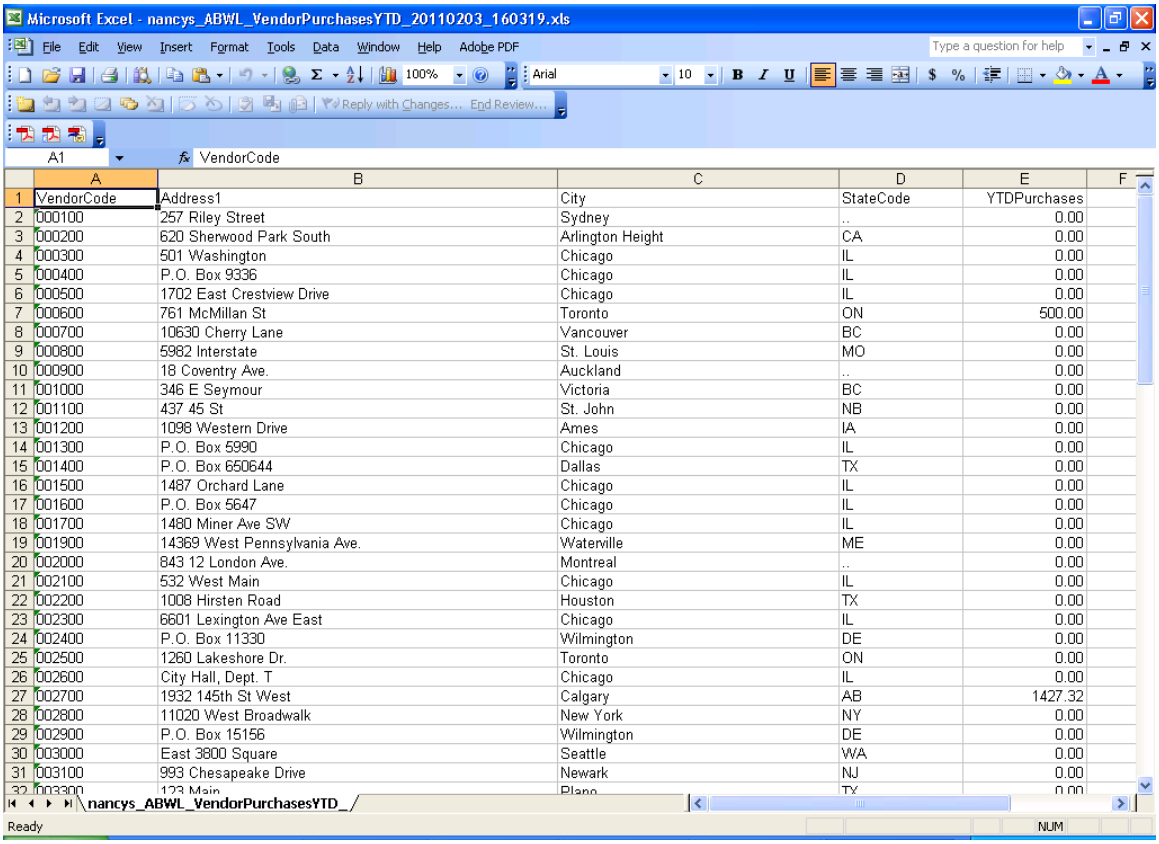


Grant access to the users that will be able to execute this query.



Execute your Data Export by selecting the Data Source and clicking the **Export** button.

Only those Data Exports that you have permission to execute will be displayed in the Data Source drop down selection.



Microsoft Excel - nancys_ABWL_VendorPurchasesYTD_20110203_160319.xls

	A	B	C	D	E
1	VendorCode	Address1	City	StateCode	YTDPurchases
2	000100	257 Riley Street	Sydney	..	0.00
3	000200	620 Sherwood Park South	Arlington Height	CA	0.00
4	000300	501 Washington	Chicago	IL	0.00
5	000400	P.O. Box 9336	Chicago	IL	0.00
6	000500	1702 East Crestview Drive	Chicago	IL	0.00
7	000600	761 McMillan St	Toronto	ON	500.00
8	000700	10630 Cherry Lane	Vancouver	BC	0.00
9	000800	5982 Interstate	St. Louis	MO	0.00
10	000900	18 Coventry Ave.	Auckland	..	0.00
11	001000	346 E Seymour	Victoria	BC	0.00
12	001100	437 45 St	St. John	NB	0.00
13	001200	1098 Western Drive	Ames	IA	0.00
14	001300	P.O. Box 5990	Chicago	IL	0.00
15	001400	P.O. Box 650644	Dallas	TX	0.00
16	001500	1487 Orchard Lane	Chicago	IL	0.00
17	001600	P.O. Box 5647	Chicago	IL	0.00
18	001700	1480 Miner Ave SW	Chicago	IL	0.00
19	001900	14369 West Pennsylvania Ave.	Waterville	ME	0.00
20	002000	843 12 London Ave.	Montreal	..	0.00
21	002100	532 West Main	Chicago	IL	0.00
22	002200	1008 Hirsten Road	Houston	TX	0.00
23	002300	6601 Lexington Ave East	Chicago	IL	0.00
24	002400	P.O. Box 11330	Wilmington	DE	0.00
25	002500	1260 Lakeshore Dr.	Toronto	ON	0.00
26	002600	City Hall, Dept. T	Chicago	IL	0.00
27	002700	1932 145th St West	Calgary	AB	1427.32
28	002800	11020 West Broadwalk	New York	NY	0.00
29	002900	P.O. Box 15156	Wilmington	DE	0.00
30	003000	East 3800 Square	Seattle	WA	0.00
31	003100	993 Chesapeake Drive	Newark	NJ	0.00
32	003300	123 Main	Plano	TX	0.00

nancys_ABWL_VendorPurchasesYTD_

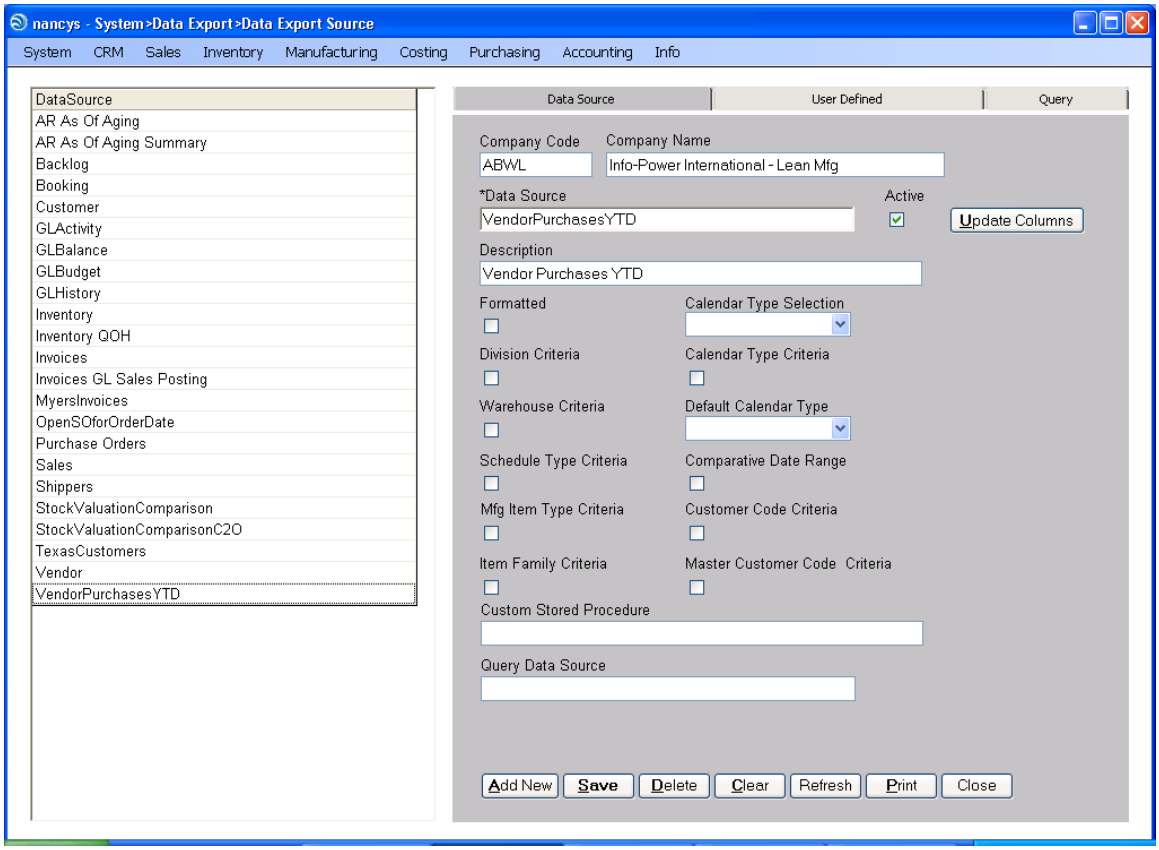
Ready

The results of your Data Export should be similar to the above.

Notice in the above example, the Vendor's Name was not included and the query is showing several vendors where the YTDPurchases column is zero.

This brings us to modifying a Data Export.

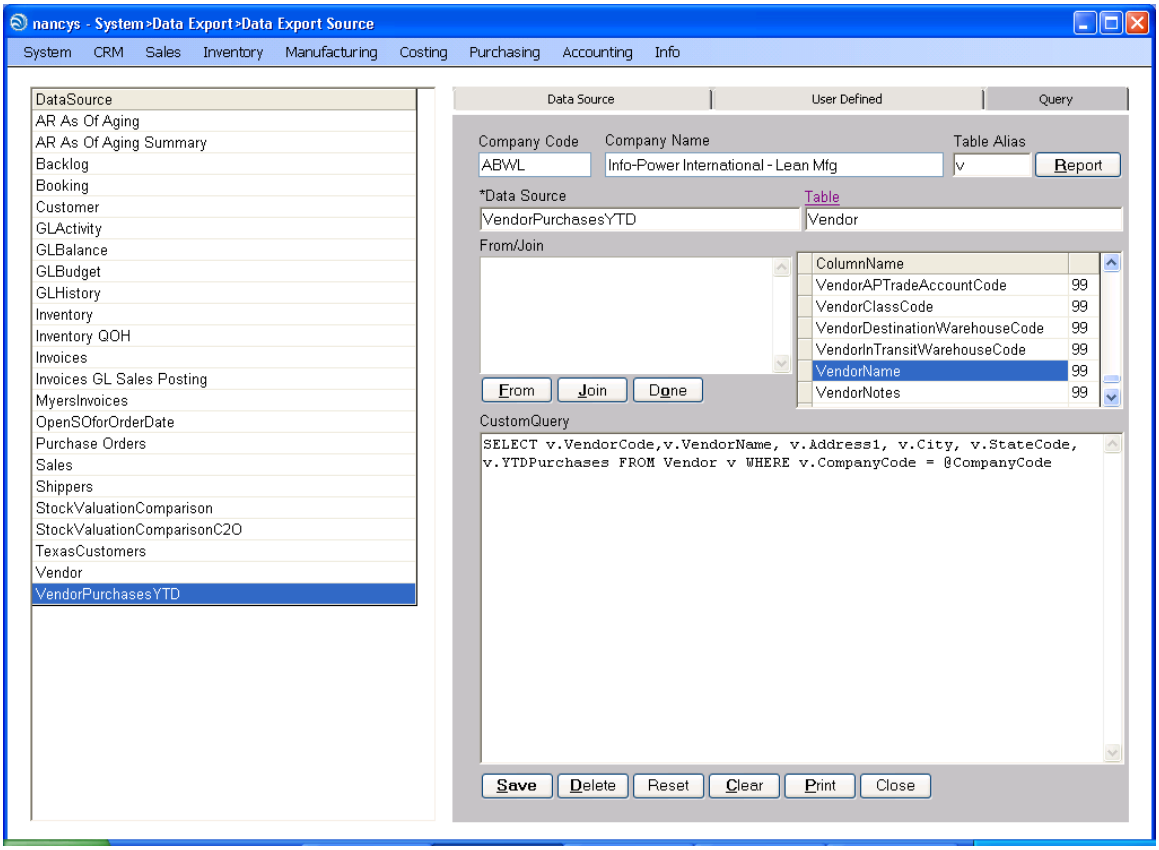
Data Export Modification



Open the Data Export Source maintenance screen, System > Data Export > Data Export Source.

We want to add the vendor's name and exclude vendors with an YTD Purchase amount not equal to zero.

Select your Data Export and move to the Query tab.



Position your cursor at the point where you would like to insert the vendor's name.

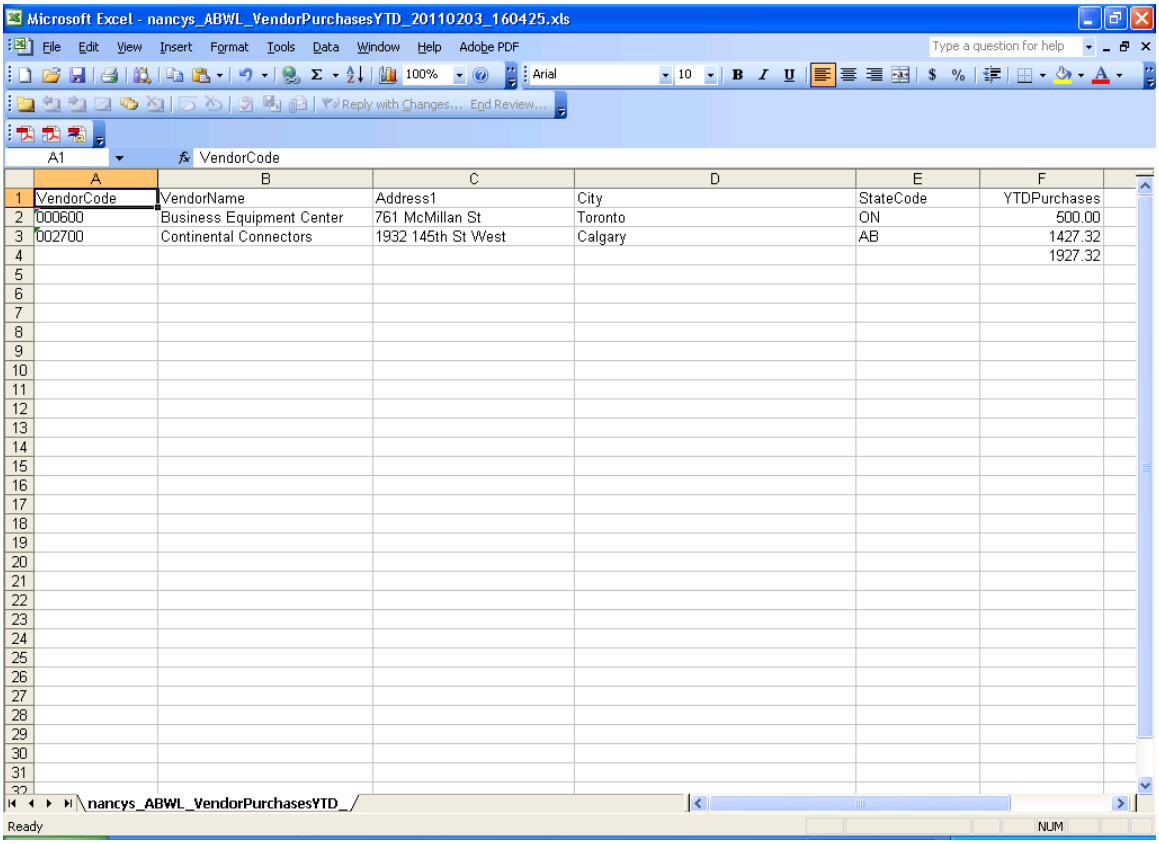
Caution: When you first click in the CustomQuery section you are in replacement mode. You must double click to be in insert mode. If you happen to make a mistake just close the maintenance screen without saving.

Enter **v.VendorName,** to insert the vendor's name.

To add the logical check for YTDPurchases <> 0, position the cursor to the end of the query statement and enter **and v.YTDPurchases <> 0** into the WHERE clause.

Click the **Save** button to save your changes.

Move back to the Data Source tab and click the **Update Columns** button. Remember this will test your query statement and you will receive a warning message if your query cannot be executed.



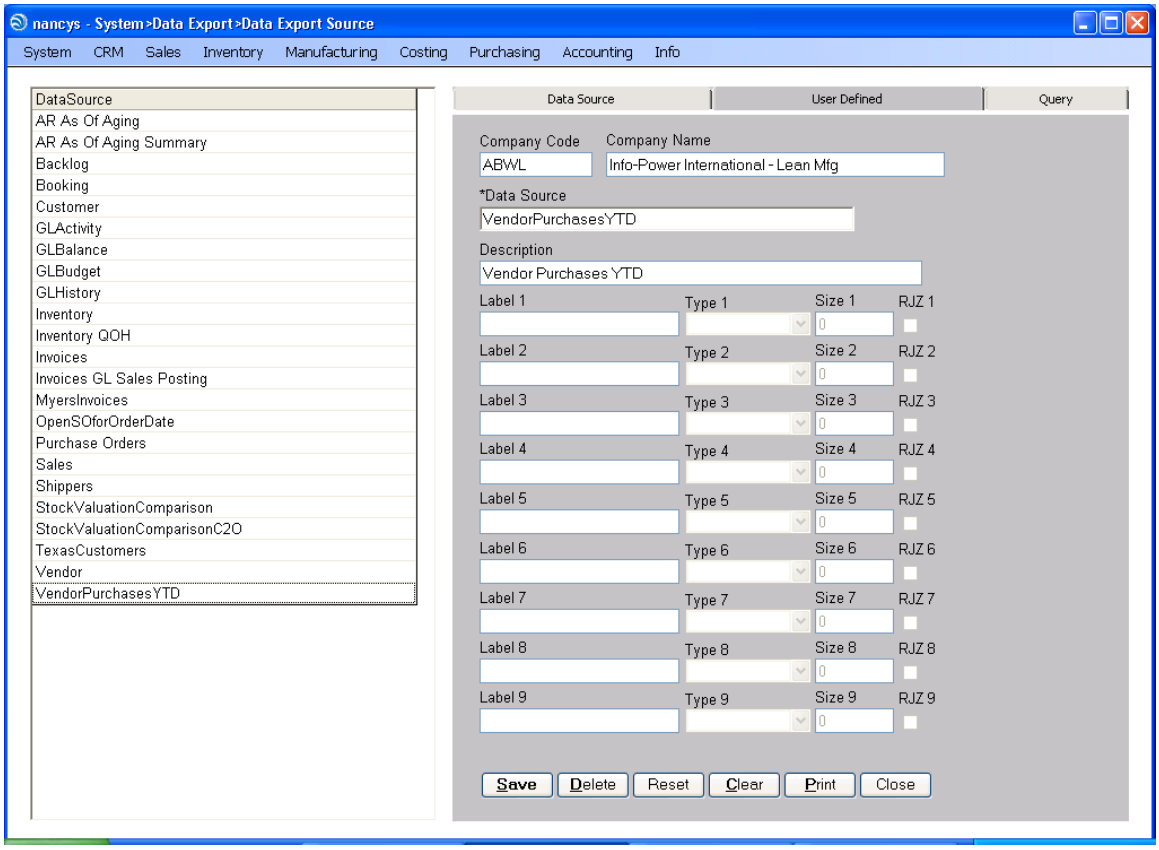
	A	B	C	D	E	F
1	VendorCode	VendorName	Address1	City	StateCode	YTDPurchases
2	000600	Business Equipment Center	761 McMillan St	Toronto	ON	500.00
3	002700	Continental Connectors	1932 145th St West	Calgary	AB	1427.32
4						1927.32

When executed your query should now have the vendor's name and only show vendors where the YTD Purchases amount is not equal to zero.

Creating a Query with User Input for Selection

In the previous two examples, we have developed queries where the selection criteria was hard-coded, only show customers from Texas or only show vendors with a non-zero YTD Purchases. To allow more flexibility you would like to create user inputs to be used as selection criteria, i.e., allow the user to enter the State Code.

You will accomplish this using the User Defined tab.



The User Defined tab has 9 user identified inputs that are available to use for selection in a Data Export.

Using one of the earlier examples, you may want to create a customer listing and enter the StateCode to be used for selection.

The Label 1 through Label 9 fields will contain the prompt to be displayed to the user when the Data Export is executed. For example, you may choose one of the following to ask for state: Enter State Code, State Code, or State.

The Type 1 through Type 9 fields indicates the nature of the data to be input it can contain one of the following values:

C – Char or Character
N – Numeric
I – Integer
D – Date
B – Boolean (True or False)

The fields Size 1 through Size 9 indicate the number of characters to be allowed for input.

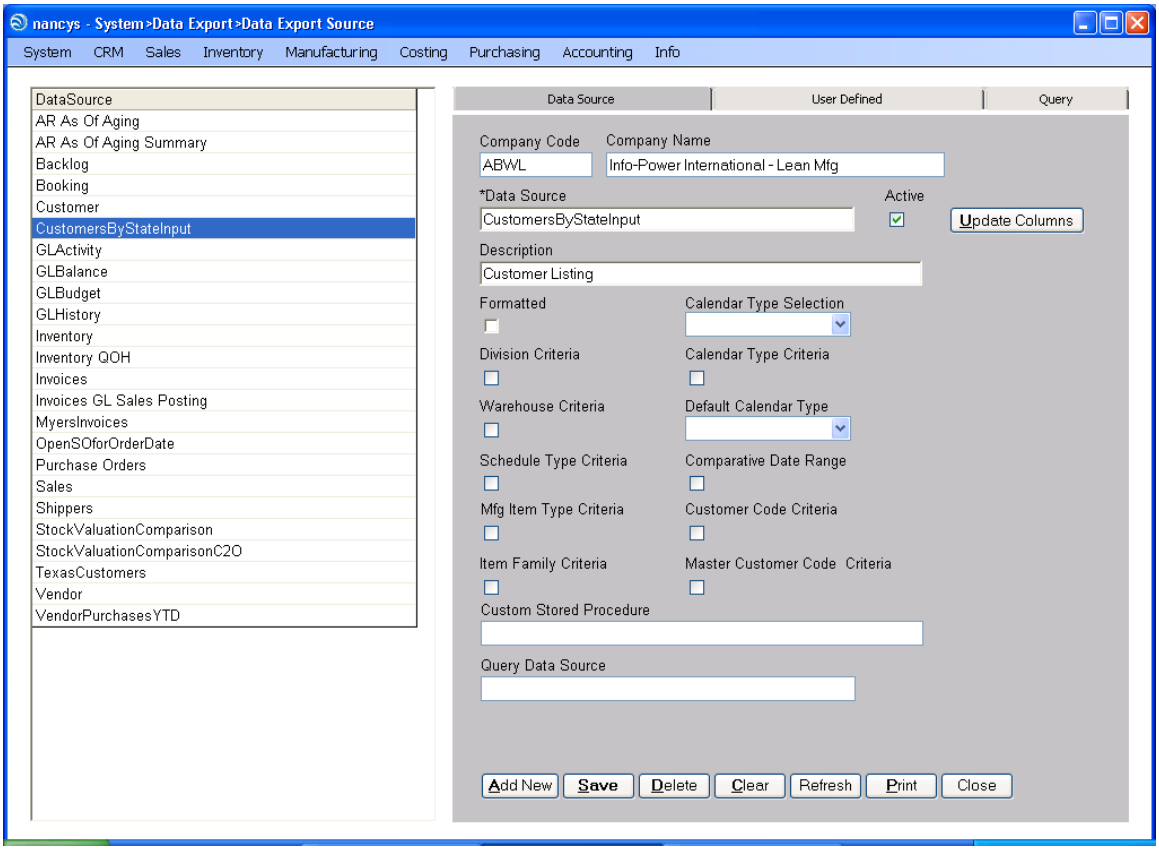
The fields RJZ 1 through RJZ 9 indicate that the data input is to be right justified and zero filled.

Continuing with the StateCode example, the fields would be set as follows:

Label 1 = Enter State Code
Type 1 = C
Size 1 = 2
RJZ 1 = remain unchecked

To reference an input value in the selection clause or WHERE statement, you will use the following:

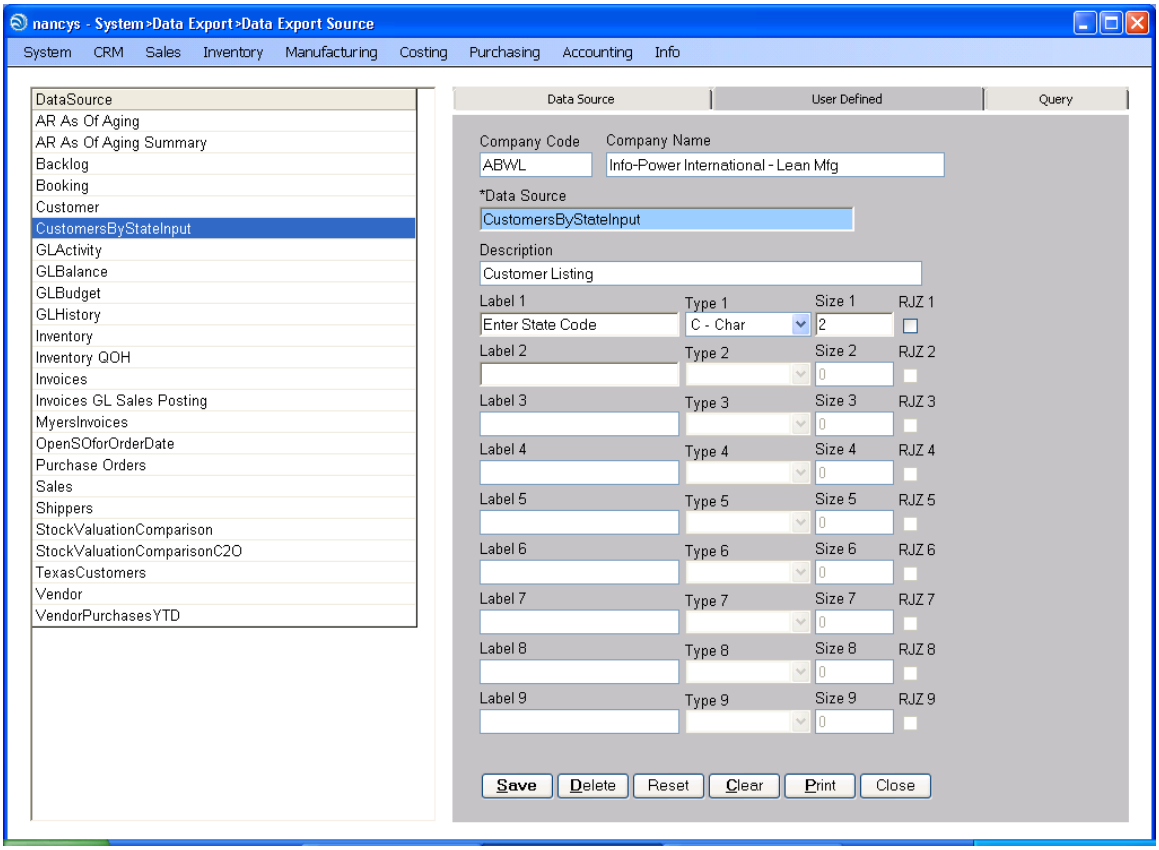
@Input1
@Input2
@Input3
@Input4
@Input5
@Input6
@Input7
@Input8
@Input9



Create a new Data Export, CustomersByStateInput.

Enter **Customer Listing** for the Data Export description.

Click the **Save** button and move to the User Defined tab.

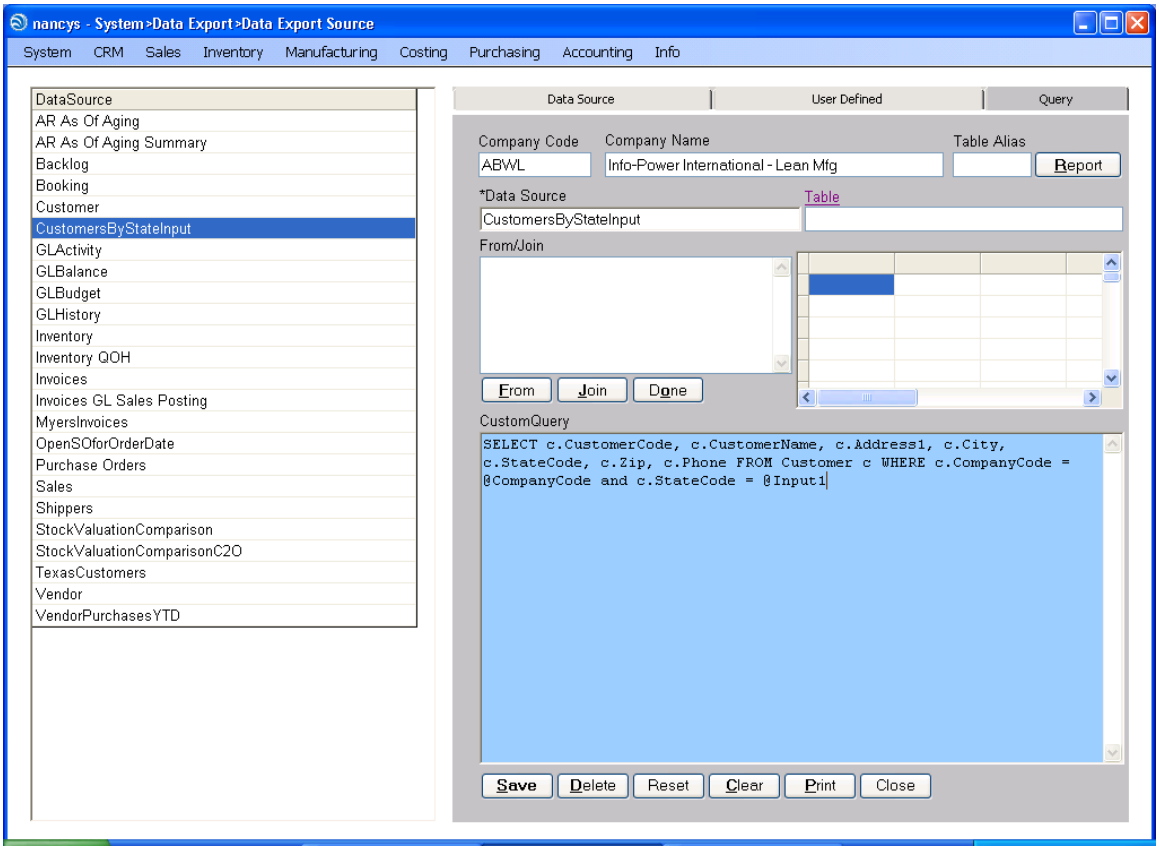


Enter **Enter State Code** for Label 1.

Enter **C** for Type 1.

Enter **2** for Size 1.

Click the **Save** button and move to the Query tab.



Select the same fields as we did in the first exercise, CustomerCode, CustomerName, Address1, City, StateCode, Zip and Phone.

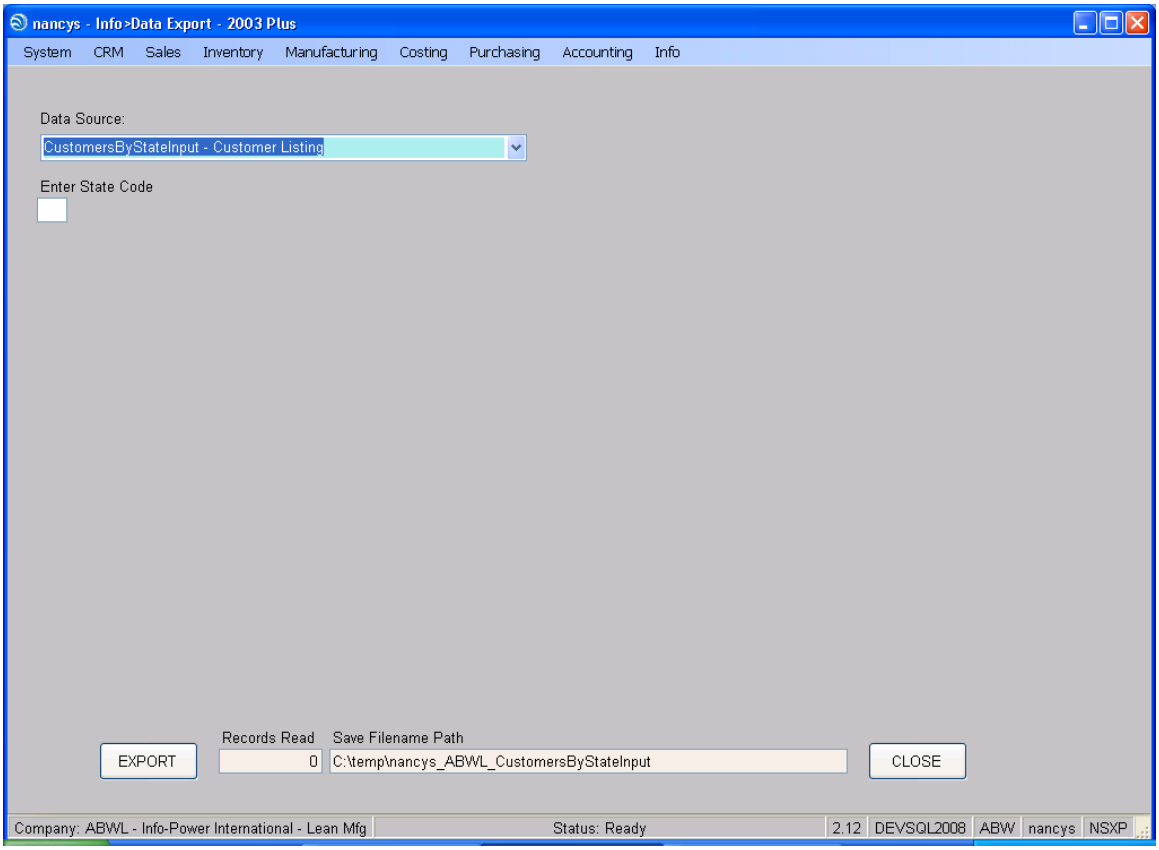
In the first example, we selected only customers that were in Texas. In this example, we want to change the selection criteria to be the value input by the user. To do this replace the = **'TX'** with = **@Input1**.

Click the **Save** button and move to the Data Source tab.

Click the **Update Columns** button to check your query statement for errors.

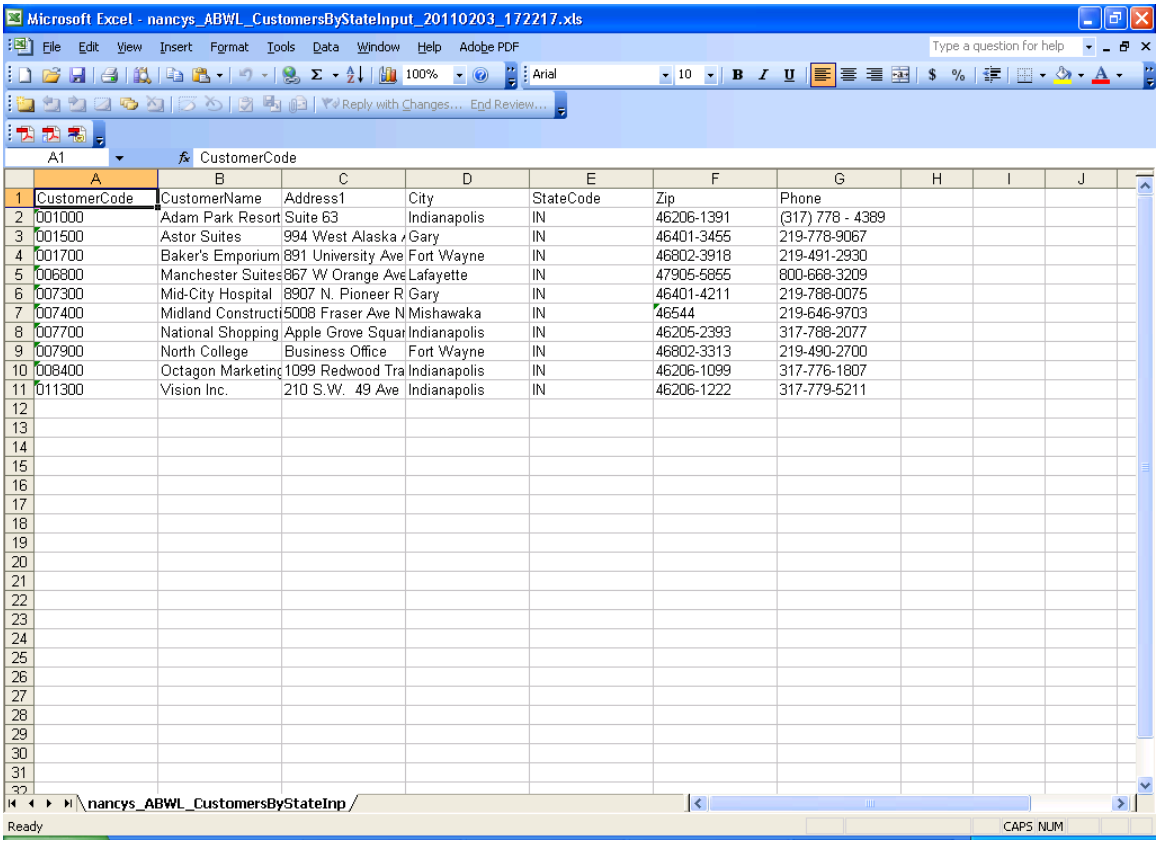
Assign the new Data Export to the users that will be running the query.

Execute the Data Export, Info > Data Export – 2003 Plus.



Notice the screen changes after selecting the Data Export.

Enter a valid 2 digit state code and click the **Export** button.



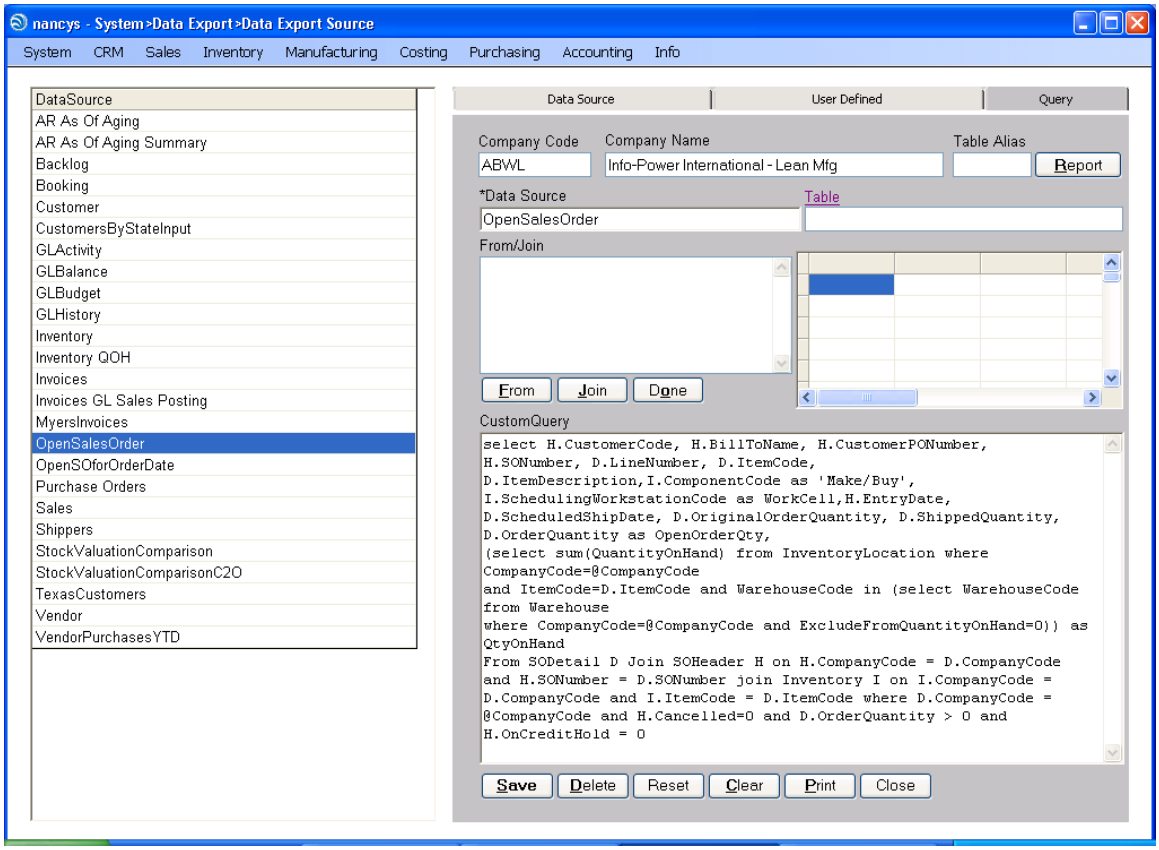
	A	B	C	D	E	F	G
1	CustomerCode	CustomerName	Address1	City	StateCode	Zip	Phone
2	001000	Adam Park Resort	Suite 63	Indianapolis	IN	46206-1391	(317) 778 - 4389
3	001500	Astor Suites	994 West Alaska A	Gary	IN	46401-3455	219-778-9067
4	001700	Baker's Emporium	891 University Ave	Fort Wayne	IN	46802-3918	219-491-2930
5	006800	Manchester Suites	867 W Orange Ave	Lafayette	IN	47905-5855	800-668-3209
6	007300	Mid-City Hospital	8907 N. Pioneer R	Gary	IN	46401-4211	219-788-0075
7	007400	Midland Constructi	5008 Fraser Ave N	Mishawaka	IN	46544	219-646-9703
8	007700	National Shopping	Apple Grove Squar	Indianapolis	IN	46205-2393	317-788-2077
9	007900	North College	Business Office	Fort Wayne	IN	46802-3313	219-490-2700
10	008400	Octagon Marketing	1099 Redwood Tra	Indianapolis	IN	46206-1099	317-776-1807
11	011300	Vision Inc.	210 S.W. 49 Ave	Indianapolis	IN	46206-1222	317-779-5211

Your Data Export should return something similar to the above.

If you query was unable to return any data, you will see a box displayed stating that No data was returned by the Query!

Using Sub-Queries and Joins within a Data Export

There will be times when you need to pull data from multiple tables. Maybe you need to create an open sales order query that shows available inventory, you would need to link the SODetail table and the InventoryLocations table and show the total quantity on-hand from all warehouse locations.



select H.CustomerCode, H.BillToName, H.CustomerPONumber, H.SONumber, D.LineNumber, D.ItemCode,
D.ItemDescription,I.ComponentCode as 'Make/Buy', I.SchedulingWorkstationCode as WorkCell,H.EntryDate, D.ScheduledShipDate, D.OriginalOrderQuantity, D.ShippedQuantity,
D.OrderQuantity as OpenOrderQty,
(select sum(QuantityOnHand) from InventoryLocation where CompanyCode=@CompanyCode
and ItemCode=D.ItemCode and WarehouseCode in (select WarehouseCode from Warehouse
where CompanyCode=@CompanyCode and ExcludeFromQuantityOnHand=0)) as QtyOnHand
From SODetail D Join SOHeader H on H.CompanyCode = D.CompanyCode and H.SONumber = D.SONumber join Inventory I on I.CompanyCode = D.CompanyCode

and I.ItemCode = D.ItemCode where D.CompanyCode = @CompanyCode and H.Cancelled=0 and D.OrderQuantity > 0 and H.OnCreditHold = 0

The above example is a little more complicated then the earlier examples.

There two points of interest in this example:

Use of a Sub-query:

(select sum(QuantityOnHand) from InventoryLocation where CompanyCode=@CompanyCode
and ItemCode=D.ItemCode and WarehouseCode in (select WarehouseCode from Warehouse
where CompanyCode=@CompanyCode and ExcludeFromQuantityOnHand=0)) as QtyOnHand

Use of a Join:

Join SOHeader H on H.CompanyCode = D.CompanyCode and H.SONumber = D.SONumber

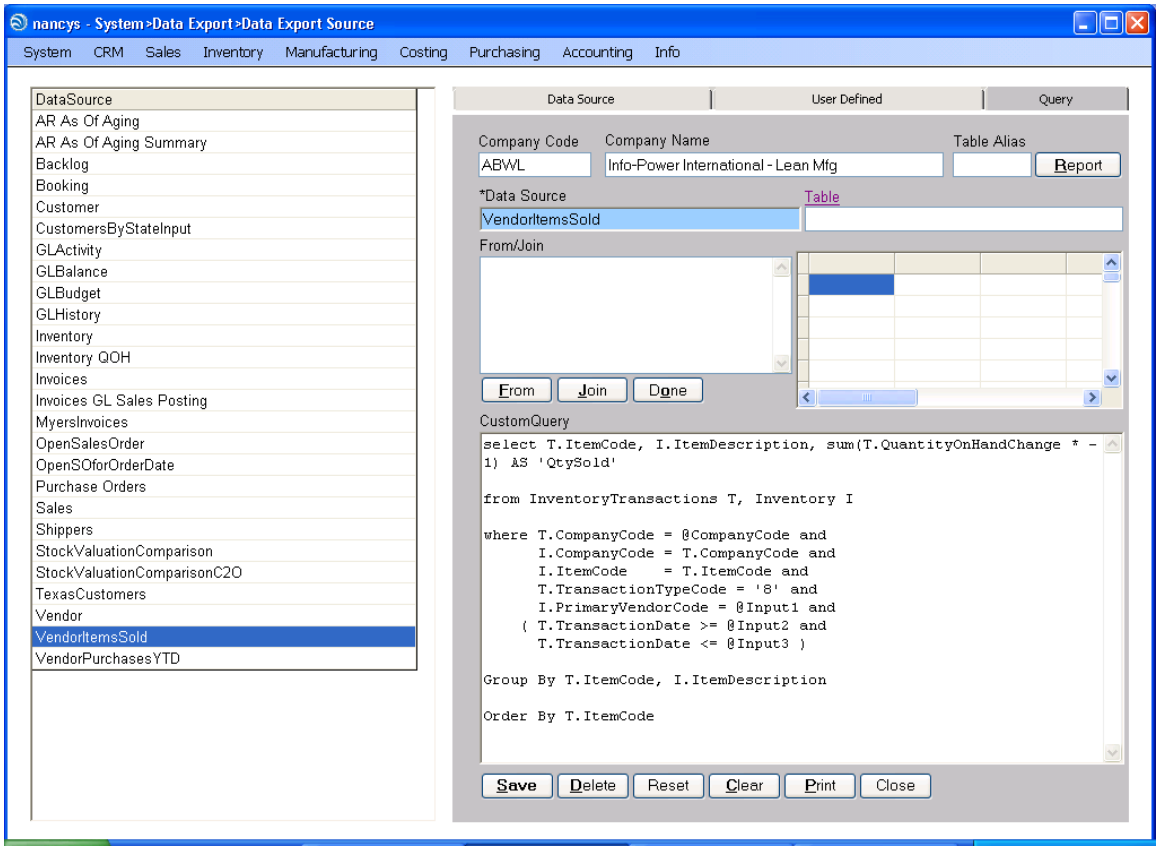
Sub-Queries:

Sub-queries let you take the result of one query and use it as information in the second query.

Joins:

Joins let you retrieve information from multiple tables to use in your query. As there are several types of joins you may want to study them, as this document will not specifically discuss each type.

Example Using Group By and Order By




```
select T.ItemCode, I.ItemDescription, sum(T.QuantityOnHandChange * -1) AS 'QtySold'

from InventoryTransactions T, Inventory I

where T.CompanyCode = @CompanyCode and
      I.CompanyCode = T.CompanyCode and
      I.ItemCode    = T.ItemCode and
      T.TransactionTypeCode = '8' and
      I.PrimaryVendorCode = @Input1 and
    ( T.TransactionDate >= @Input2 and
      T.TransactionDate <= @Input3 )

Group By T.ItemCode, I.ItemDescription

Order By T.ItemCode
```

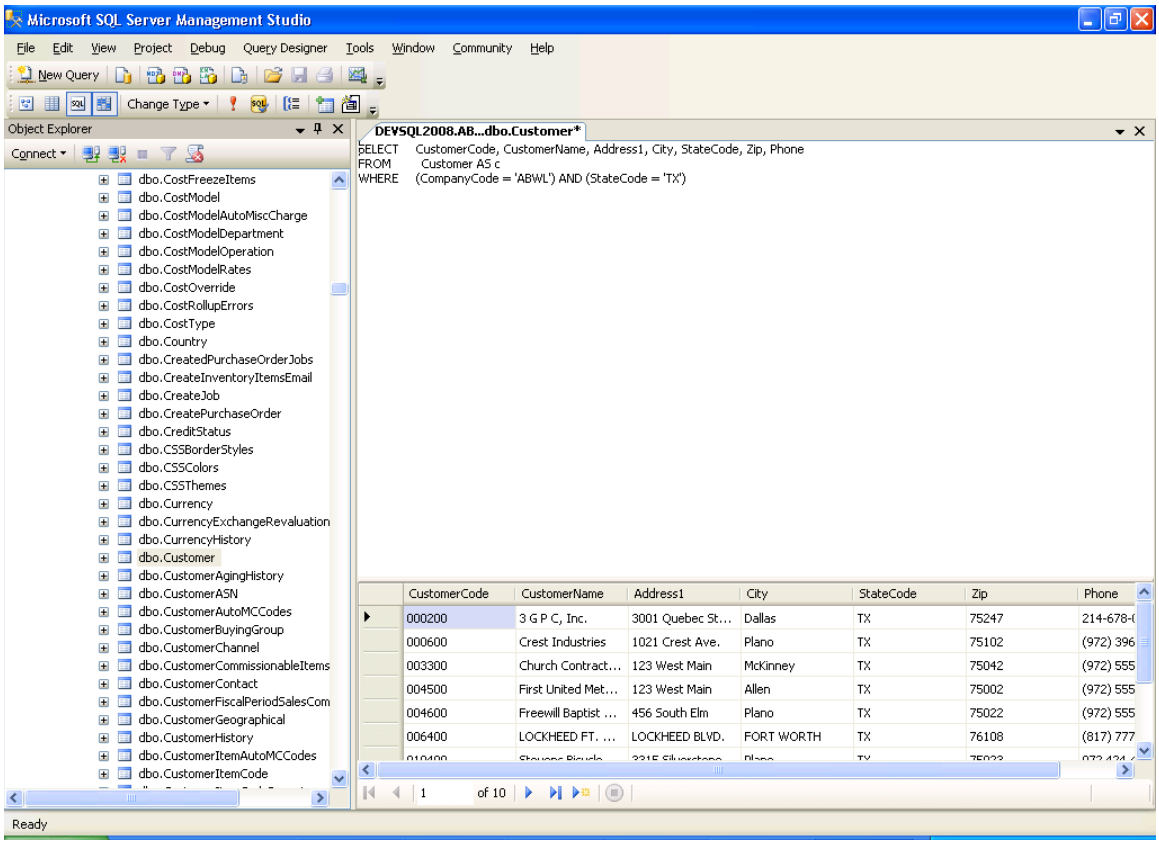
The above example uses the SQL query commands Group By and Order By.

The Group By command indicates logical breaks in a query where subtotals can be generated.

The Order By command is used to indicate how your data should be sorted in your query. In the above example your data will be printed in ItemCode sequence.

Also of importance, notice the use of the @Input2 and @Input3 values. These input variables contain dates that are used to select a data range of inventory items.

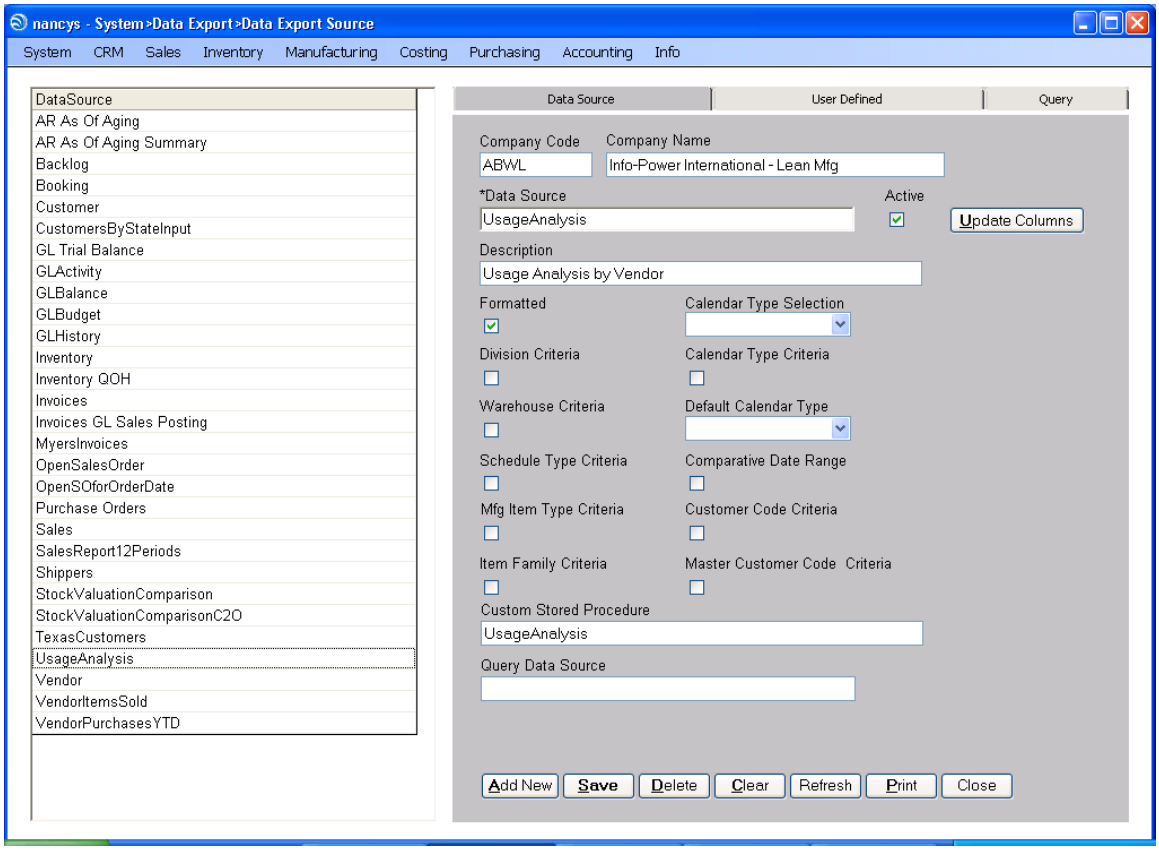
Using SQL Management Studio or Query Analyzer to Create Data Exports



If you are comfortable using the Microsoft tools SQL Server Management Studio and Query Analyzer, you can use those tools to develop you query. Then using Copy/Paste you can copy your query into the Query field in Data Export.

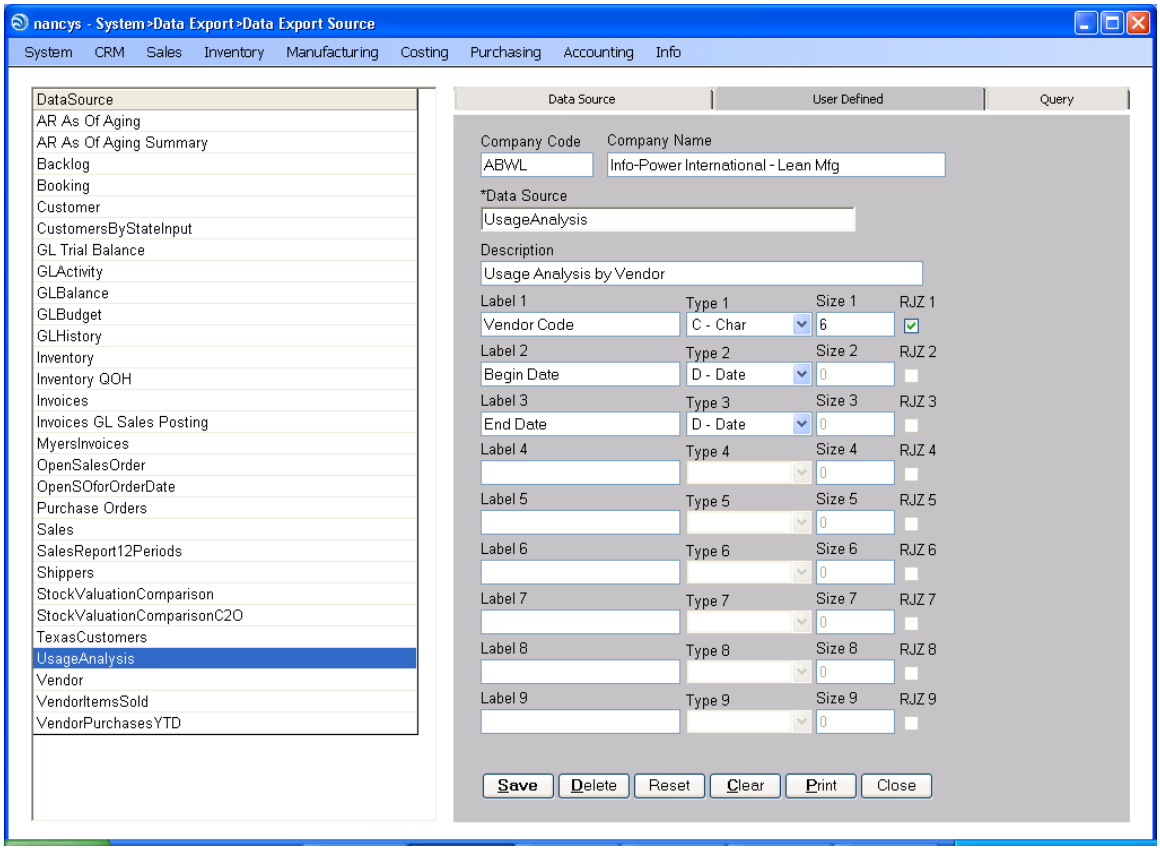
Remember you will then need to modify the query using inputs to allow user to dynamically run your query.

Using a Stored Procedure to Produce a Data Export



On occasion, you might have a very complex report, similar to the UsageAnalysis shown below. This example uses temp tables to process the data.

To use a stored procedure, enter the name of the stored procedure in the Custom Stored Procedure field.



Using the User Defined tab, define the input variables to be used during the processing of the query for data retrieval.

Custom Query for UsageAnalysis data export.

```
PRINT 'UsageAnalysis'
GO
if exists (select * from sysobjects where id =
object_id('dbo.UsageAnalysis') and sysstat & 0xf = 4) drop procedure
dbo.UsageAnalysis
GO
CREATE PROCEDURE dbo.UsageAnalysis
(
@CompanyCode                    char(4),
@UserCode                       char(11),
@DataSource                     char(50),
@DivisionCode                   char(10),
@WarehouseCode                  char(10),
@ScheduleTypeCode       char(1),
@MfgItemTypeCode        char(10),
@ItemFamilyCode                 char(20),
@CalendarTypeCode       char(1),
@DateRangeTypeCode              char(1),
@FiscalBeginYear        char(4),
```

```
@FiscalBeginPeriod            char(2),
@FiscalEndYear                char(4),
@FiscalEndPeriod        char(2),
@CalendarBeginDate            datetime,
@CalendarEndDate        datetime,
@CustomerCode                 char(10),
@MasterCustomerCode           char(10),
@Input1                             varchar(50),
@Input2                             varchar(50),
@Input3                             varchar(50),
@Input4                             varchar(50),
@Input5                             varchar(50),
@Input6                             varchar(50),
@Input7                             varchar(50),
@Input8                             varchar(50),
@Input9                             varchar(50),
@Debug                              int = 0
) AS
declare @Count             int,
        @errno             int,
        @errmsg            varchar(255),
        @err1              varchar(255),
        @err2              varchar(255),
        @err3              varchar(255),
        @err4              varchar(255),
        @err5              varchar(255),
        @err6              varchar(255),
        @err7              varchar(255),
        @err8              varchar(255),
        @err9              varchar(255),
        @Gosub             varchar(20)
select @Count = 0
/*
      Copyright (C) 2001 Info-Power International, Inc.  All rights
reserved.
      Proprietary Information not for Disclosure.

      $Header: /ABW/SPSql/UsageAnalysis.sql 1     2/10/11 11:33a Jeffz
$
*/
if @Debug = 1
begin
      select      @DataSource
end

set nocount on
/*
declare     @CompanyCode            char(4)
,           @Input1                       char(10)
,           @Input2                       datetime
,           @Input3                       datetime
*/
declare     @fetch_Inventory  int
,           @fetch_SystemCalendar int
,           @ItemCode               char(24)
```

```
,           @ItemSalesClassCode     char(10)
,           @SafetyStock            decimal(12,2)
,           @LeadTime               int
,           @MeanDays               int
,           @Mean                   decimal(12,2)
,           @Quantity               decimal(12,2)
,           @ScheduleDate           date
,           @UsageQty               decimal(12,2)
,           @StandardCost           decimal(16,6)
,           @QuantityOnHandChange decimal(12,2)
,           @QuantityOnHand         decimal(12,2)
,           @VendorCode             char(10)
/*
select      @CompanyCode      = 'STRK'
,           @Input1                 = '00050480'
,           @Input2                 = '2010-04-01'
,           @Input3                 = '2010-06-30'
*/

select @MeanDays  =     Count(*)
from SystemCalendar
where CompanyCode = @CompanyCode
and ScheduleDate between @Input2 and @Input3
and WorkDay = 1

delete UsageAnalysis_1
where UserCode = @UserCode

delete UsageAnalysis_2
where UserCode = @UserCode

declare csr_Inventory cursor local for select ItemCode,
ItemSalesClassCode,LeadTime, StandardCost,
      (select sum(QuantityOnHand) from InventoryLocation where
CompanyCode = im.CompanyCode and ItemCode = im.ItemCode) as OnHandQty,
      (select max(SafetyStock) from InventoryLocation where CompanyCode
= im.CompanyCode and ItemCode = im.ItemCode) as SafetyStock
      from Inventory im
      where CompanyCode = @CompanyCode
      and PrimaryVendorCode = @Input1
      and Active = 1
      and (select sum(QuantityOnHand) from InventoryLocation where
CompanyCode = im.CompanyCode and ItemCode = im.ItemCode) <> 0


open csr_Inventory
FETCH NEXT FROM csr_Inventory INTO
      @ItemCode
,     @ItemSalesClassCode
,     @LeadTime
,     @StandardCost
,     @QuantityOnHand
,     @SafetyStock

select @fetch_Inventory = @@fetch_status
```

```
while @fetch_Inventory = 0
begin

      --select @Mean = (sum(QuantityOnHandChange) * (-1) ) / @MeanDays
      select @Quantity = sum(QuantityOnHandChange) * (-1)
      from InventoryTransactions
      where CompanyCode = @CompanyCode
      and ItemCode = @ItemCode
      and TransactionTypeCode = 6
      and TransactionDate between @Input2 and @Input3

      select @Mean = @Quantity / @MeanDays
      --print @ItemCode + 'Mean Days '+str(@MeanDays) +' Qty
'+str(@Quantity)+' Mean '+cast(@Mean as varchar(20))
      insert into UsageAnalysis_1 (UserCode,CompanyCode,
ItemCode,ItemSalesClassCode,
PeriodUsage,OnHandQty,LeadTime,StandardCost,Mean,SafetyStock)
      select
@UserCode,@CompanyCode,@ItemCode,@ItemSalesClassCode,ISNULL(@Quantity,0
),@QuantityOnHand,@LeadTime,@StandardCost,ISNULL(@Mean,0),@SafetyStock


      declare csr_SystemCalendar cursor local for select ScheduleDate
      from SystemCalendar
            where CompanyCode = @CompanyCode
            and WorkDay = 1
            and ScheduleDate between @Input2 and @Input3
      open csr_SystemCalendar
      FETCH NEXT FROM csr_SystemCalendar INTO
            @ScheduleDate

      select @fetch_SystemCalendar = @@fetch_status
      while @fetch_SystemCalendar = 0
      begin

            select @QuantityOnHandChange = (select
sum(QuantityOnHandChange) *(-1))
            from InventoryTransactions
            where CompanyCode = @CompanyCode
            and ItemCode = @ItemCode
            and TransactionTypeCode = 6
            and TransactionDate = @ScheduleDate

            insert into UsageAnalysis_2(UserCode,CompanyCode, ItemCode,
TransactionDate, TransactionQty, Mean, TransMinusMean,TransMinusMeanSQ)
            select
@UserCode,@CompanyCode,@ItemCode,@ScheduleDate,ISNULL(@QuantityOnHandCh
ange,0),ISNULL(@Mean,0),
            ISNULL(@QuantityOnHandChange,0)-
ISNULL(@Mean,0),POWER((ISNULL(@QuantityOnHandChange,0)-
ISNULL(@Mean,0)),2)--(@QuantityOnHandChange-@Mean)^2

            FETCH NEXT FROM csr_SystemCalendar INTO
                  @ScheduleDate
```

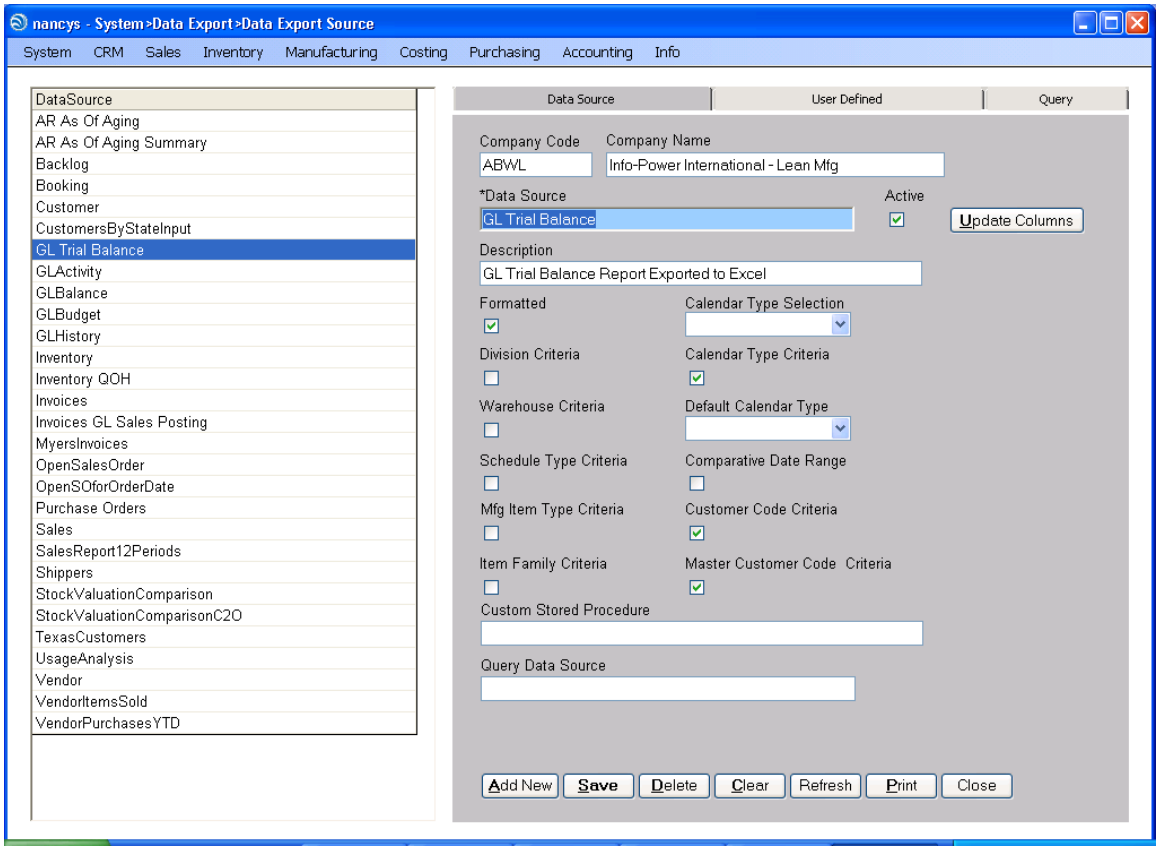
```
            select @fetch_SystemCalendar = @@fetch_status
      end
      close csr_SystemCalendar
      deallocate csr_SystemCalendar

      FETCH NEXT FROM csr_Inventory INTO
      @ItemCode
,     @ItemSalesClassCode
,     @LeadTime
,     @StandardCost
,     @QuantityOnHand
,     @SafetyStock

      select @fetch_Inventory = @@fetch_status
end
close csr_Inventory
deallocate csr_Inventory

select
@Input1 as VendorCode,@Input2 as BeginDate,@Input3 as EndDate,
t1.ItemCode,ItemDescription,t1.ItemSalesClassCode,t1.StandardCost,Perio
dUsage,Mean,
(select sqrt(sum(TransMinusMeanSQ)) from UsageAnalysis_2 where
CompanyCode=t1.CompanyCode and ItemCode=t1.ItemCode) / sqrt(@MeanDays-
1) as StdDeviation,
SafetyStock,OnHandQty,t1.LeadTime
from UsageAnalysis_1 t1
join Inventory i on i.CompanyCode=t1.CompanyCode and
i.ItemCode=t1.ItemCode
where UserCode = @UserCode

/*
select *
from UsageAnalysis_2

select *
from UsageAnalysis_1

*/
```

Using an Existing ABW Report to Create a Data Export

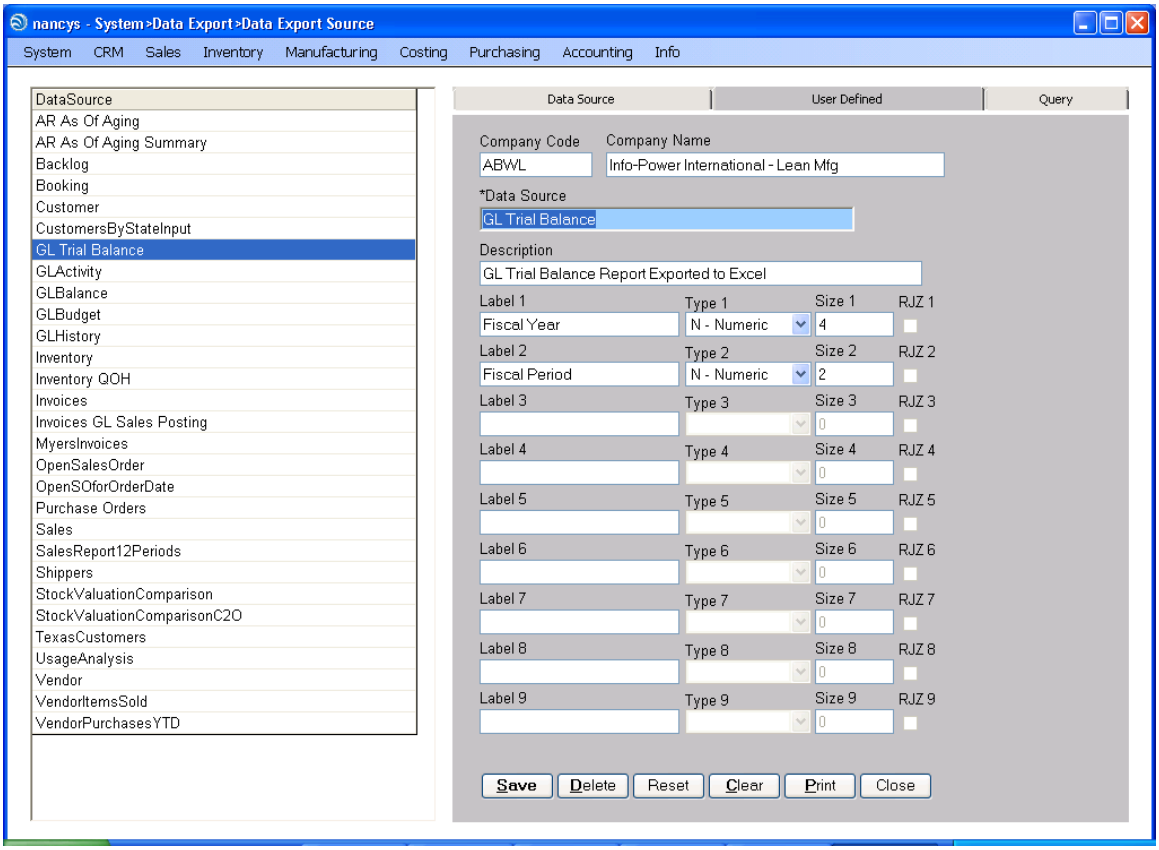
Sometimes you might have the need to expand on an existing ABW report or even just take an existing report and export that information to an Excel spreadsheet.

You can execute the report's stored procedure and pass the required parameters to export the report.

The following example uses the standard ABW General Ledger Trial Balance Report.



Create the Data Source in the same manor as all previous examples.



Add the User Inputs required for processing the query; these will be passed as parameters to the stored procedure.

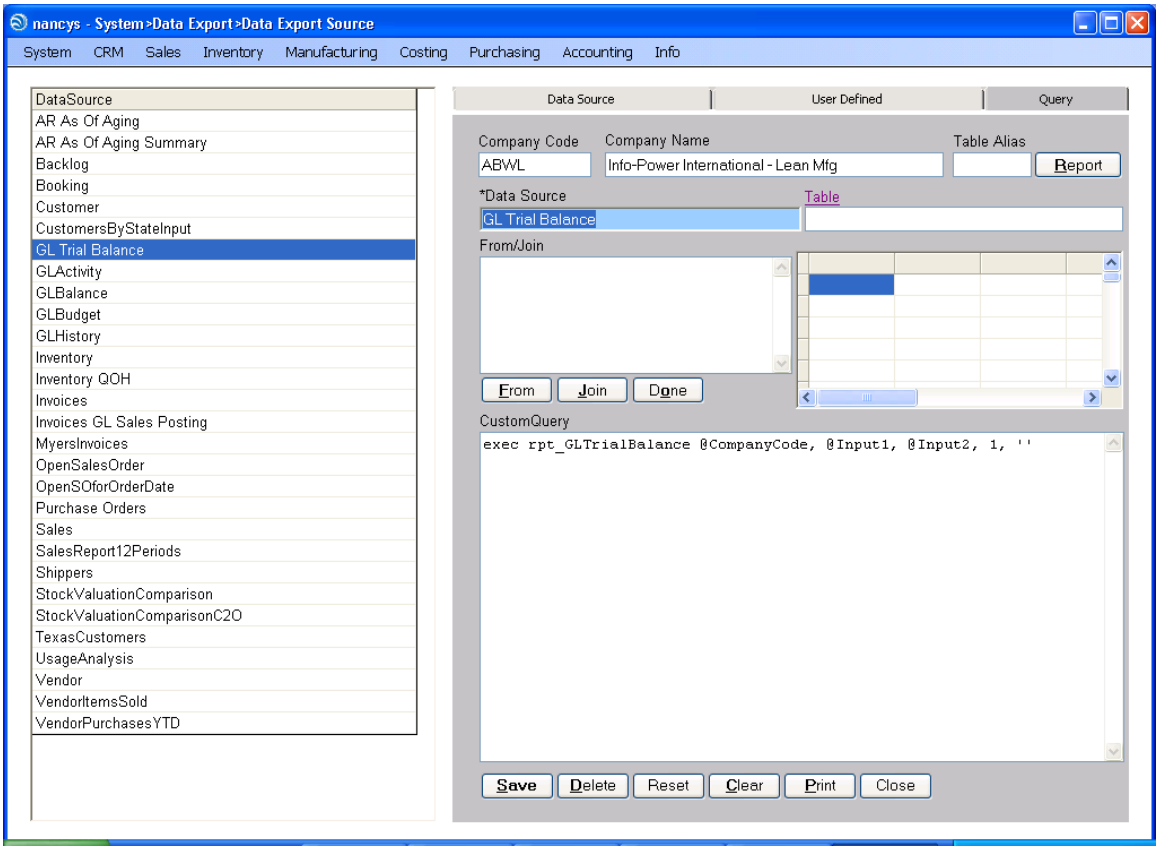
Below is the stored procedure ‘rpt_GLTrialBalance’. To determine the parameters that will need to be gathered from the user and passed to the stored procedure look immediately after the CREATE PROCEUDRE section. You will see the variables declared.

```
PRINT 'rpt_GLTrialBalance'
GO
if exists (select * from sysobjects where id =
object_id('dbo.rpt_GLTrialBalance') and sysstat & 0xf = 4) drop
procedure dbo.rpt_GLTrialBalance
GO
CREATE PROCEDURE dbo.rpt_GLTrialBalance
(
@CompanyCode            char(4),
@FiscalYear             char(4),
@GLPeriod               char(2),
@Exclude                bit,
@DivisionCode           char(10)
) AS
```

```
declare @Count             int,
        @errno             int,
        @errmsg            varchar(255),
        @ReportDivisionCode   char,
        @SiteStart                 int,
        @SiteLength                int

select      @Count = 0,
            @SiteStart = 0,
            @SiteLength = 0

/*
      Copyright (C) 2001 Info-Power International, Inc.  All rights
reserved.
      Proprietary Information not for Disclosure.

      $Header: /ABW/SPReports/rpt_GLTrialBalance.SQL 3     1/07/09
9:45a Mac $
*/

Declare @Period As Char(2)
Set @Period = @GLPeriod

If @DivisionCode = 'ALL'
begin
      select @ReportDivisionCode = '%'
end
else
begin
      select @ReportDivisionCode = @DivisionCode
end

Set NoCount On

select      @SiteStart = NumericParameter
from  ApplicationControls
where CompanyCode = @CompanyCode
and         ModuleCode = 'GL'
and         ApplicationControlCode = 'AccountSiteStart'

select      @SiteLength = NumericParameter
from  ApplicationControls
where CompanyCode = @CompanyCode
and         ModuleCode = 'GL'
and         ApplicationControlCode = 'AccountSiteLength'

Select
      COA.AccountCode,
      COA.Description,
      CODA.FiscalYear,
      Case
            When @Period = '01' Then Sum(CODA.BalanceForward + 0)
            When @Period = '02' Then Sum(CODA.BalanceForward +
                  CODA.Amount01)
            When @Period = '03' Then Sum(CODA.BalanceForward +
```

```
            CODA.Amount01 + CODA.Amount02)
      When @Period = '04' Then Sum(CODA.BalanceForward +
            CODA.Amount01 + CODA.Amount02 + CODA.Amount03)
      When @Period = '05' Then Sum(CODA.BalanceForward +
            CODA.Amount01 + CODA.Amount02 + CODA.Amount03 +
            CODA.Amount04)
      When @Period = '06' Then Sum(CODA.BalanceForward +
            CODA.Amount01 + CODA.Amount02 + CODA.Amount03 +
            CODA.Amount04 + CODA.Amount05)
      When @Period = '07' Then Sum(CODA.BalanceForward +
            CODA.Amount01 + CODA.Amount02 + CODA.Amount03 +
            CODA.Amount04 + CODA.Amount05 + CODA.Amount06)
      When @Period = '08' Then Sum(CODA.BalanceForward +
            CODA.Amount01 + CODA.Amount02 + CODA.Amount03 +
            CODA.Amount04 + CODA.Amount05 + CODA.Amount06 +
            CODA.Amount07)
      When @Period = '09' Then Sum(CODA.BalanceForward +
            CODA.Amount01 + CODA.Amount02 + CODA.Amount03 +
            CODA.Amount04 + CODA.Amount05 + CODA.Amount06 +
            CODA.Amount07 + CODA.Amount08)
      When @Period = '10' Then Sum(CODA.BalanceForward +
            CODA.Amount01 + CODA.Amount02 + CODA.Amount03 +
            CODA.Amount04 + CODA.Amount05 + CODA.Amount06 +
            CODA.Amount07 + CODA.Amount08 + CODA.Amount09)
      When @Period = '11' Then Sum(CODA.BalanceForward +
            CODA.Amount01 + CODA.Amount02 + CODA.Amount03 +
            CODA.Amount04 + CODA.Amount05 + CODA.Amount06 +
            CODA.Amount07 + CODA.Amount08 + CODA.Amount09 +
            CODA.Amount10)
      When @Period = '12' Then Sum(CODA.BalanceForward +
            CODA.Amount01 + CODA.Amount02 + CODA.Amount03 +
            CODA.Amount04 + CODA.Amount05 + CODA.Amount06 +
            CODA.Amount07 + CODA.Amount08 + CODA.Amount09 +
            CODA.Amount10 + CODA.Amount11)
      When @Period = '13' Then Sum(CODA.BalanceForward +
            CODA.Amount01 + CODA.Amount02 + CODA.Amount03 +
            CODA.Amount04 + CODA.Amount05 + CODA.Amount06 +
            CODA.Amount07 + CODA.Amount08 + CODA.Amount09 +
            CODA.Amount10 + CODA.Amount11 + CODA.Amount12)
      Else 0.00
End As BeginningBalance,
Case
      When @Period = '01' Then CODA.Amount01
      When @Period = '02' Then CODA.Amount02
      When @Period = '03' Then CODA.Amount03
      When @Period = '04' Then CODA.Amount04
      When @Period = '05' Then CODA.Amount05
      When @Period = '06' Then CODA.Amount06
      When @Period = '07' Then CODA.Amount07
      When @Period = '08' Then CODA.Amount08
      When @Period = '09' Then CODA.Amount09
      When @Period = '10' Then CODA.Amount10
      When @Period = '11' Then CODA.Amount11
      When @Period = '12' Then CODA.Amount12
      When @Period = '13' Then CODA.Amount13
```

```
            Else 0.00
      End As Amount,
      @Period As GLPeriod,
      @Exclude As Exclude,
      @DivisionCode as DivisionCode,
      case when @DivisionCode = 'ALL' then 'ALL'
            else (select Description from Division D
                  where @CompanyCode = D.CompanyCode and
                        @DivisionCode = D.DivisionCode)end
      as DivDescription
From
      ChartOfAccounts COA
Inner Join
      ChartOfAccountsData CODA
On
      COA.CompanyCode = CODA.CompanyCode And
      COA.AccountCode = CODA.AccountCode
Where
      COA.CompanyCode = @CompanyCode And
      CODA.FiscalYear = @FiscalYear And
      CODA.GLDataTypeCode = '1' And
      substring(COA.AccountCode, @SiteStart, @SiteLength) LIKE
@ReportDivisionCode and
      @Period = @GLPeriod
Group By
      COA.AccountCode,
      COA.Description,
      CODA.FiscalYear,
      CODA.BalanceForward,
      CODA.Amount01,
      CODA.Amount02,
      CODA.Amount03,
      CODA.Amount04,
      CODA.Amount05,
      CODA.Amount06,
      CODA.Amount07,
      CODA.Amount08,
      CODA.Amount08,
      CODA.Amount09,
      CODA.Amount10,
      CODA.Amount11,
      CODA.Amount12,
      CODA.AMount13

CLEANUP:
return

GO
return
GO
```



The Custom Query will contain a SQL statement to execute the report's stored procedure along with the parameters required.

Additional Examples

1) Using a UNION

```
Select S.CustomerCode, C.CustomerName,
substring (A.SalesAccountCode, 6,4)as Account, S.ItemCode as ItemCode,
I.ItemDescription as ItemDescription,
sum(case when S.SalesPeriod = 1 then (S.QuantityInvoiced * S.UnitPrice ) else 0 end) as
'JanSales',
sum(case when S.SalesPeriod = 2 then (S.QuantityInvoiced * S.UnitPrice ) else 0 end) as
'FebSales',
sum(case when S.SalesPeriod = 3 then (S.QuantityInvoiced * S.UnitPrice ) else 0 end) as
'MarSales',
sum(case when S.SalesPeriod = 4 then (S.QuantityInvoiced * S.UnitPrice ) else 0 end) as
'AprSales',
sum(case when S.SalesPeriod = 5 then (S.QuantityInvoiced * S.UnitPrice ) else 0 end) as
'MaySales',
sum(case when S.SalesPeriod = 6 then (S.QuantityInvoiced * S.UnitPrice ) else 0 end) as
'JunSales',
sum(case when S.SalesPeriod = 7 then (S.QuantityInvoiced * S.UnitPrice ) else 0 end) as
'JulySales',
sum(case when S.SalesPeriod = 8 then (S.QuantityInvoiced * S.UnitPrice ) else 0 end) as
'AugSales',
sum(case when S.SalesPeriod = 9 then (S.QuantityInvoiced * S.UnitPrice ) else 0 end) as
'SepSales',
sum(case when S.SalesPeriod = 10 then (S.QuantityInvoiced * S.UnitPrice ) else 0 end)
as 'OctSales',
sum(case when S.SalesPeriod = 11 then (S.QuantityInvoiced * S.UnitPrice ) else 0 end)
as 'NovSales',
sum(case when S.SalesPeriod = 12 then (S.QuantityInvoiced * S.UnitPrice ) else 0 end)
as 'DecSales'

From SalesAnalysisHistory S, Inventory I, Customer C, SalesCOGSControl A
where S.CompanyCode = @CompanyCode and
S.SalesYear = @Input1 and

I.CompanyCode = S.CompanyCode and
I.ItemCode = S.ItemCode and
C.CompanyCode = S.CompanyCode and
C.CustomerCode = S.CustomerCode and
A.CompanyCode = S.CompanyCode and
A.CustomerSalesClassCode = C.CustomerSalesClassCode and
A.ItemSalesClassCode = I.ItemSalesClassCode and
substring (A.SalesAccountCode, 6,4) in ('3002','3004')
```

```
group by S.CustomerCode, C.CustomerName, A.SalesAccountCode, S.ItemCode,
I.ItemDescription


UNION

Select S.CustomerCode, C.CustomerName,
substring (S.AccountCode, 6,4)as Account, S.MiscChargeCode as ItemCode,
S.Description as ItemDescription,
sum(case when S.SalesPeriod = 1 then (S.LineItemAmount ) else 0 end) as 'JanSales',
sum(case when S.SalesPeriod = 2 then (S.LineItemAmount ) else 0 end) as 'FebSales',
sum(case when S.SalesPeriod = 3 then (S.LineItemAmount ) else 0 end) as 'MarSales',
sum(case when S.SalesPeriod = 4 then (S.LineItemAmount ) else 0 end) as 'AprSales',
sum(case when S.SalesPeriod = 5 then (S.LineItemAmount ) else 0 end) as 'MaySales',
sum(case when S.SalesPeriod = 6 then (S.LineItemAmount) else 0 end) as 'JunSales',
sum(case when S.SalesPeriod = 7 then (S.LineItemAmount ) else 0 end) as 'JulySales',
sum(case when S.SalesPeriod = 8 then (S.LineItemAmount) else 0 end) as 'AugSales',
sum(case when S.SalesPeriod = 9 then (S.LineItemAmount ) else 0 end) as 'SepSales',
sum(case when S.SalesPeriod = 10 then (S.LineItemAmount ) else 0 end) as 'OctSales',
sum(case when S.SalesPeriod = 11 then (S.LineItemAmount) else 0 end) as 'NovSales',
sum(case when S.SalesPeriod = 12 then (S.LineItemAmount ) else 0 end) as 'DecSales'

From SalesAnalysisMiscChargesHistory S, Customer C
where S.CompanyCode = @CompanyCode and
S.SalesYear = @Input1 and
C.CompanyCode = S.CompanyCode and
C.CustomerCode = S.CustomerCode and

substring (S.AccountCode, 6,4) in ('3002','3004')

group by S.CustomerCode, C.CustomerName, s.AccountCode, S.MiscChargeCode,
S.Description

Order By S.CustomerCode, ItemCode, Account
```

Example 2) Using SQL commands to trim data

```
SELECT IM.GLTableCode, IT.TransactionDate, IT.ItemCode, IM.ItemDescription,
IM.ItemSalesClassCode, IT.DocumentNumber, IT.TransactionTypeCode,
IT.QuantityOnHandChange, IT.TransactionStandardCost, IT.TransactionStandardCost *
IT.QuantityOnHandChange as ExtendedCost, IT.UserCode

FROM InventoryTransactions IT, Inventory IM

WHERE IT.CompanyCode = @CompanyCode AND
     IM.CompanyCode = IT.CompanyCode AND
     IM.ItemCode = IT.ItemCode AND
    ((LEN(LTRIM(RTRIM(@Input1)))<> 0 AND
              IT.ItemCode = @Input1) OR
     (LEN(LTRIM(RTRIM(@Input1))) = 0 )) AND
    ((LEN(LTRIM(RTRIM(@Input4)))<> 0 AND
      IT.TransactionTypeCode = @Input4) OR
     (LEN(LTRIM(RTRIM(@Input4))) = 0 ))AND
  IT.TransactionDate>= @Input2 AND IT.TransactionDate <= @Input3
```