NestedFlow Automation – Test Data Options

Creation Date: 10/13/2023

Table Of Contents

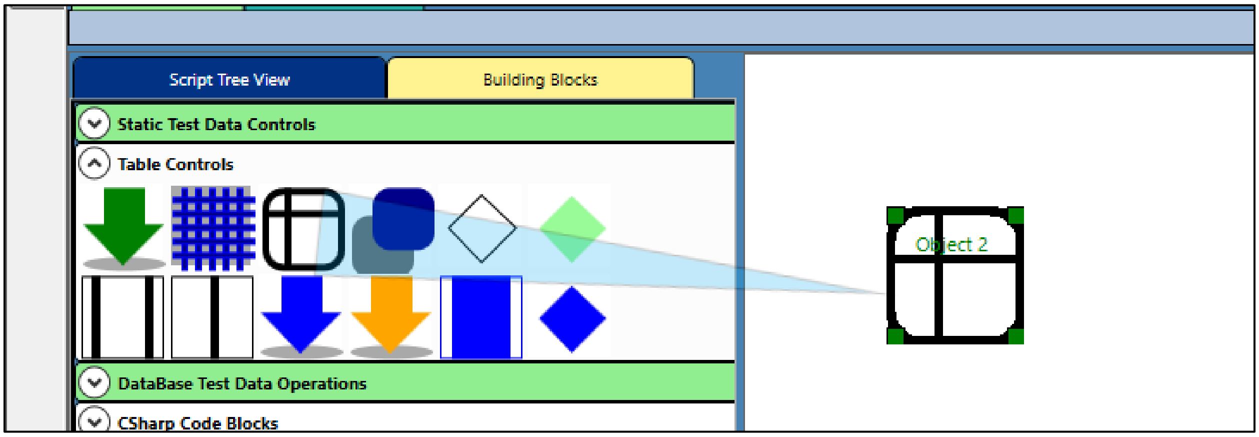
Introduction

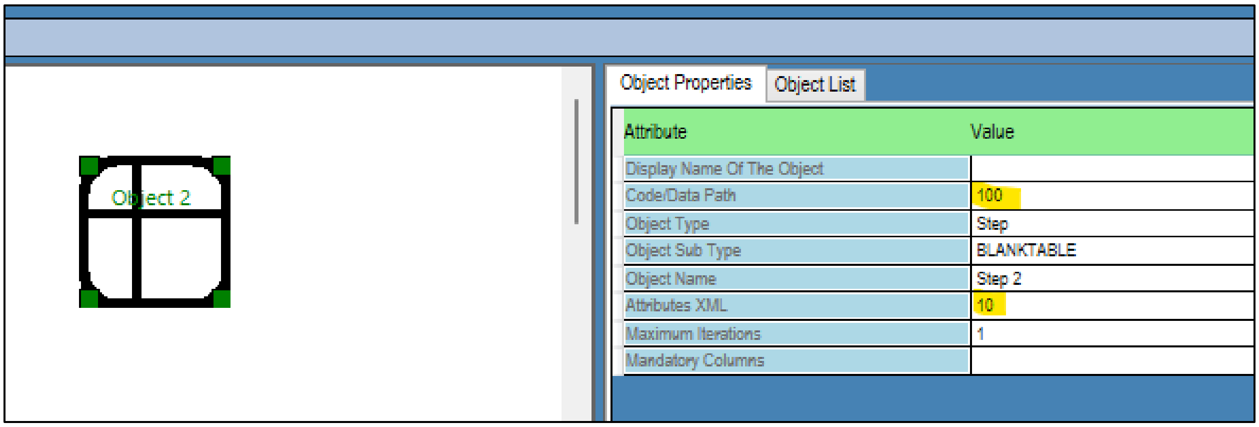
In this document we will look in to the test data generation options provided in the NestedFlow Automation tool which can be used to quickly generated test data needed to feed automated test scripts. Please refer to the below document to see how the tool uses external files to be used as test data input:

NestedFlowRepo/Documents/File Based Test Data Options and controls.pdf at main · nestedflow-Automation/NestedFlowRepo (github.com)

Components To Create Test Data

Create Empty Table With Default Column Names



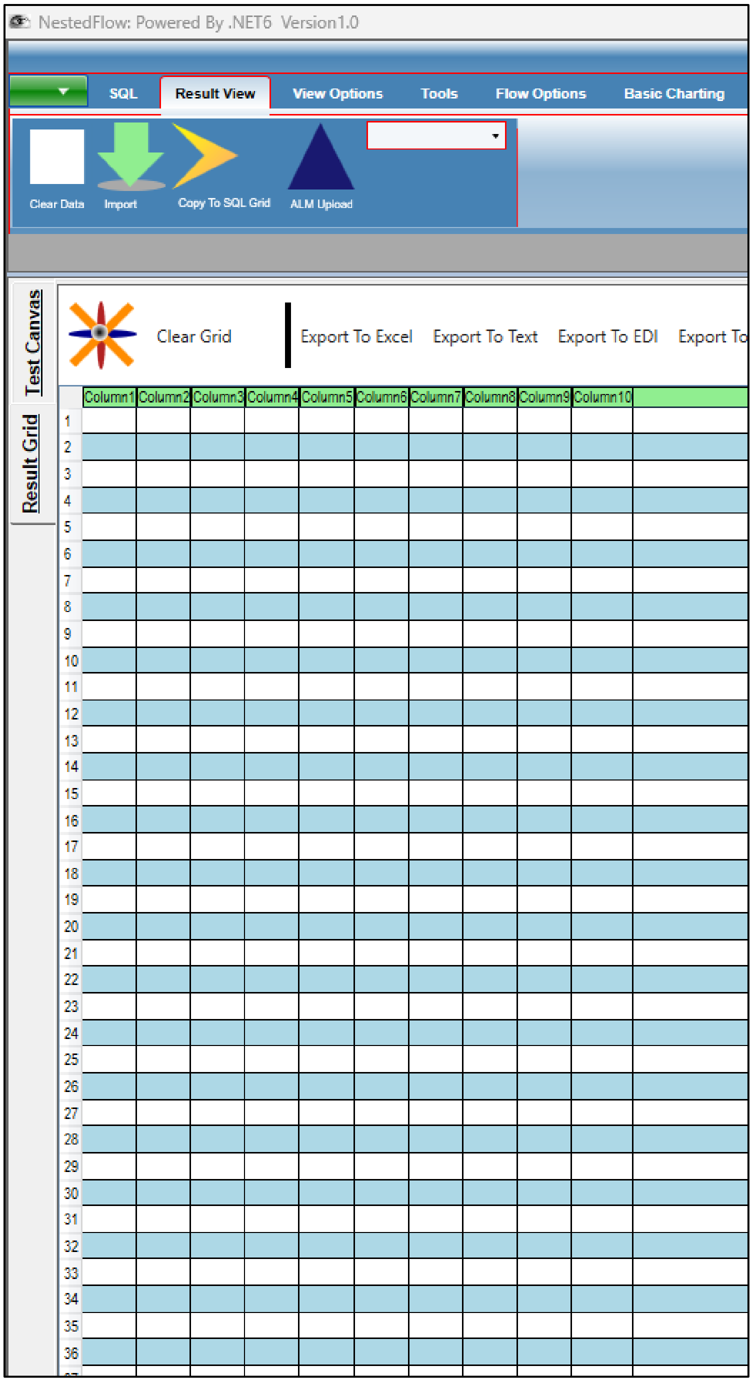


Attribute	Value
Display Name Of The Object	
Code/Data Path	100
Object Type	Step
Object Sub Type	BLANKTABLE
Object Name	Step 2
Attributes XML	10
Maximum Iterations	1
Mandatory Columns	

Double click on Value filed in Code/Data Path and enter number of rows needed.

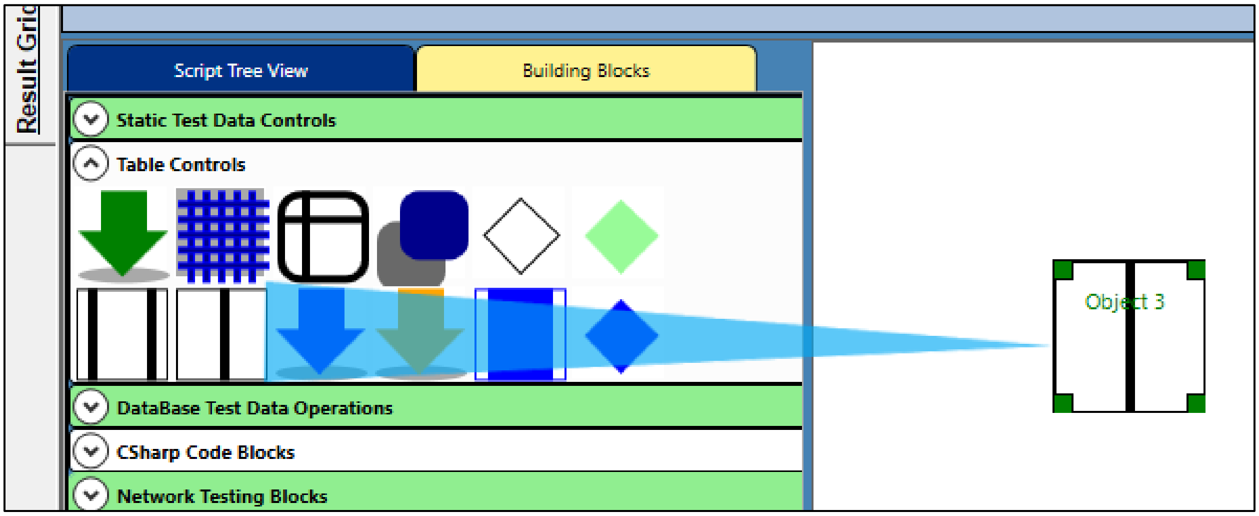
Double click on Value field in Attributes XML and enter number of columns needed.

It will create table accordingly

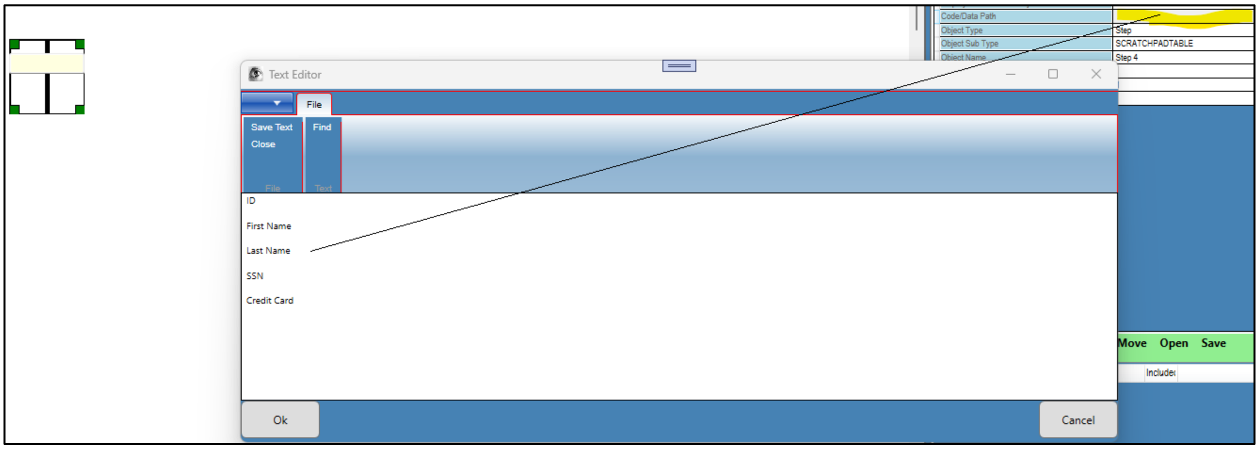


	Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9	Column10	
1											
2											
3											
4											
5											
6											
7											
8											
9											
10											
11											
12											
13											
14											
15											
16											
17											
18											
19											
20											
21											
22											
23											
24											
25											
26											
27											
28											
29											
30											
31											
32											
33											
34											
35											
36											

Create Empty Table With Pre-Determined Column Names



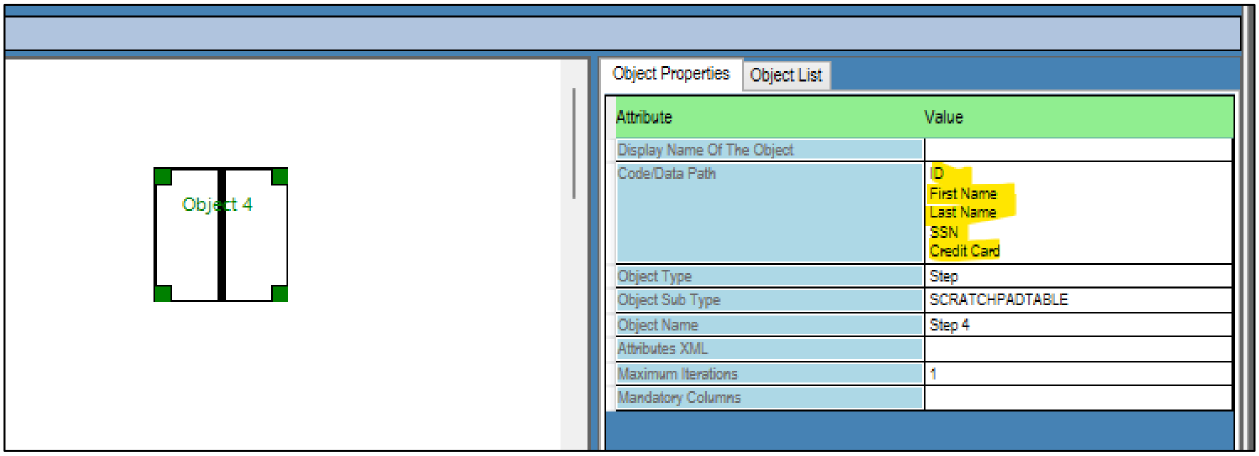
Click on the item and property of the item will be displayed. Double click on the Code/Data Path



Code/Data Path	
Object Type	Step
Object Sub Type	SCRATCHPADTABLE
Object Name	Step 4

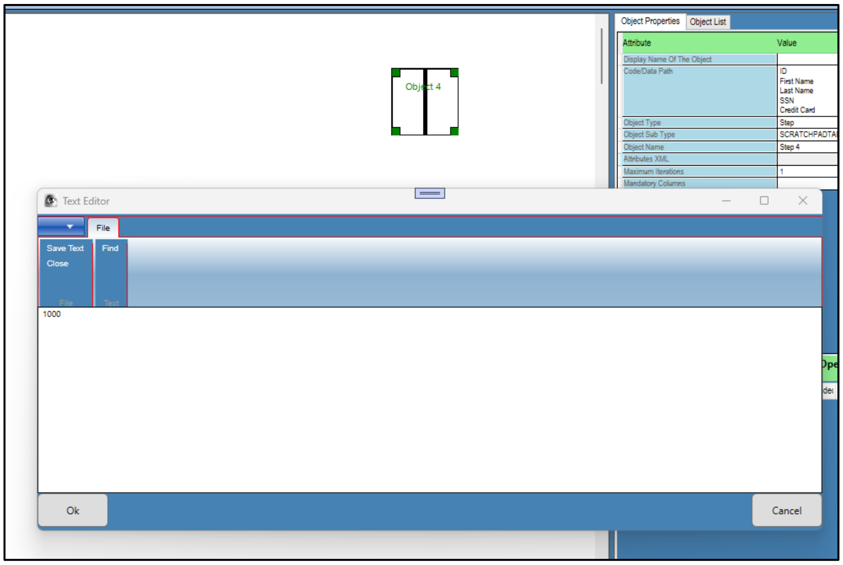
```
ID
First Name
Last Name
SSN
Credit Card
```

Enter the Expected column names (each row a single column name). Click on OK



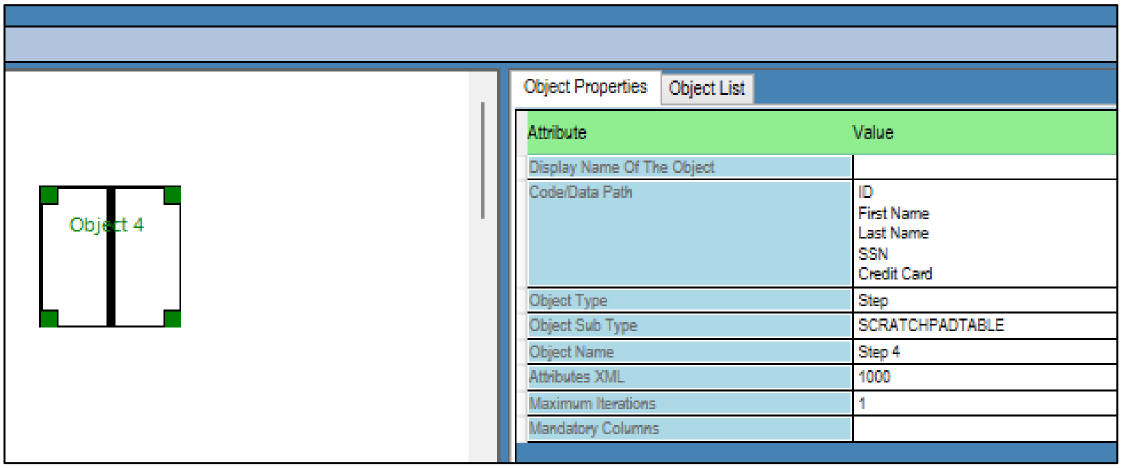
Attribute	Value
Display Name Of The Object	
Code/Data Path	ID First Name Last Name SSN Credit Card
Object Type	Step
Object Sub Type	SCRATCHPADTABLE
Object Name	Step 4
Attributes XML	
Maximum Iterations	1
Mandatory Columns	

Double click on the Attribute XML value field and enter number of rows you expect to be created



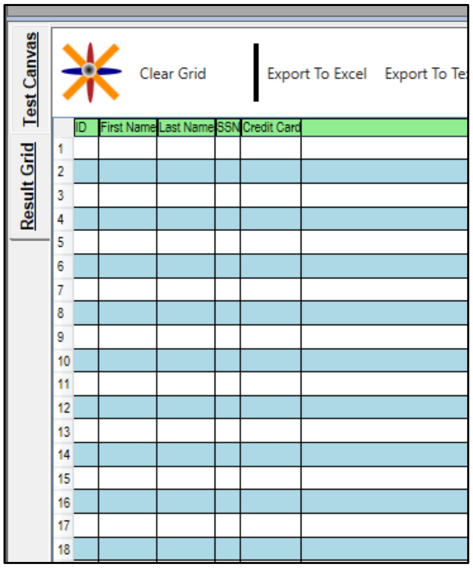
Attribute	Value
Display Name Of The Object	
Code/Data Path	ID First Name Last Name SSN Credit Card
Object Type	Step
Object Sub Type	SCRATCHPADTA
Object Name	Step 4
Attributes XML	
Maximum Iterations	1
Mandatory Columns	

```
1000
```



Attribute	Value
Display Name Of The Object	
Code/Data Path	ID First Name Last Name SSN Credit Card
Object Type	Step
Object Sub Type	SCRATCHPADTABLE
Object Name	Step 4
Attributes XML	1000
Maximum Iterations	1
Mandatory Columns	

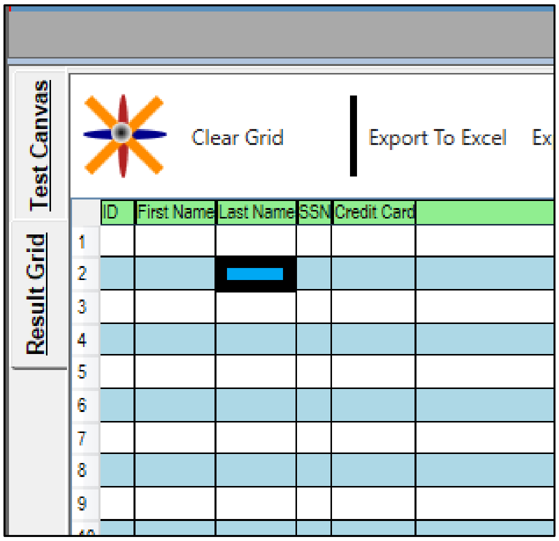
When executed it will generate a table as expected



	ID	First Name	Last Name	SSN	Credit Card
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					

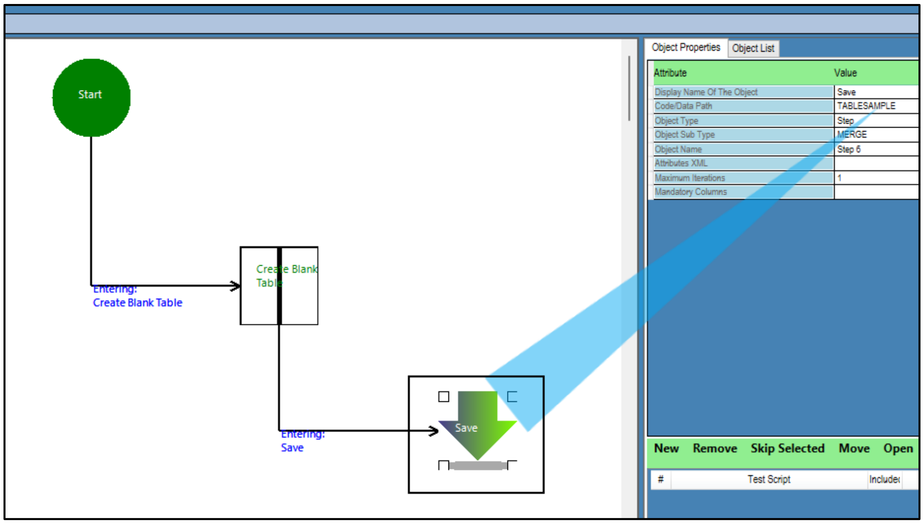
Update a Column

Let us say we want to update 2nd row Last Name field



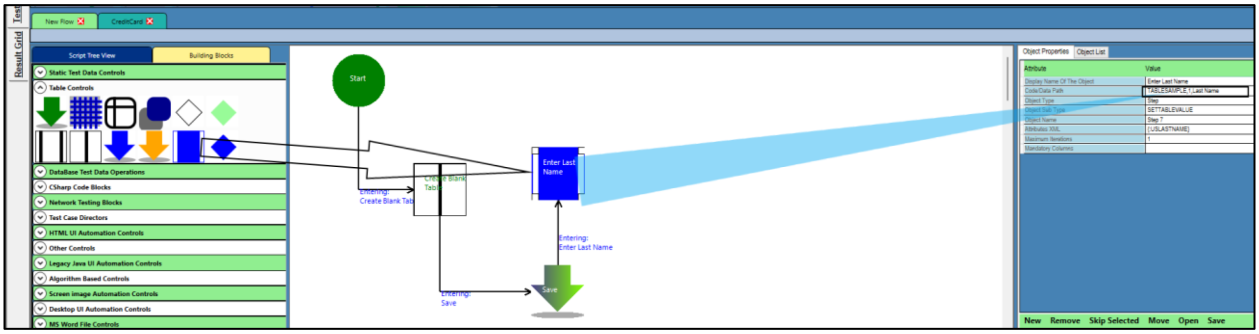
	ID	First Name	Last Name	SSN	Credit Card
1					
2					
3					
4					
5					
6					
7					
8					
9					

For achieving it we must save the table first



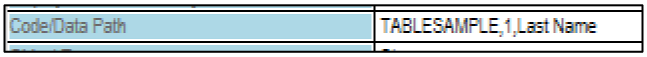
Attribute	Value
Display Name Of The Object	Save
Code/Data Path	TABLESAMPLE
Object Type	Step
Object Sub Type	MERGE
Object Name	Step 6
Attributes XML	
Maximum Iterations	1
Mandatory Columns	

Use Column update feature to achieve 2nd row Last Name field



Attribute	Value
Display Name Of The Object	Enter Last Name
Code/Data Path	TABLESAMPLE,1,Last Name
Object Type	Step
Object Sub Type	SETTABLEVALUE
Object Name	Step 7
Attributes XML	{:USLASTNAME}
Maximum Iterations	1
Mandatory Columns	

The **Code/Data Path** is 3 part string separated by comma



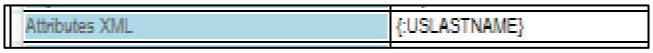
Code/Data Path	TABLESAMPLE,1,Last Name

First component is Table name

2nd component is the 0 based row index (2nd row will be 1, 3rd row will be 2 and so on)

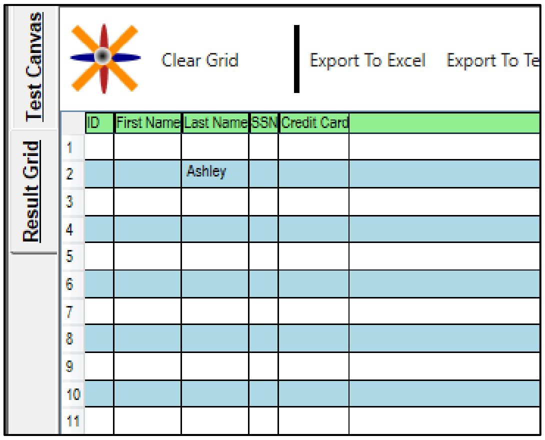
3rd component is the name of the column to be updated

Attribute XML is the value to be updated with in this case it is system parameter {:USLASTNAME}



Attributes XML	{:USLASTNAME}

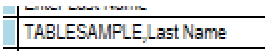
When executed the result will be as shown below



	ID	First Name	Last Name	SSN	Credit Card	
1						
2			Ashley			
3						
4						
5						
6						
7						
8						
9						
10						
11						

Update Entire Column

It can be achieved by altering **Code/Data Path**


TABLESAMPLE,Last Name

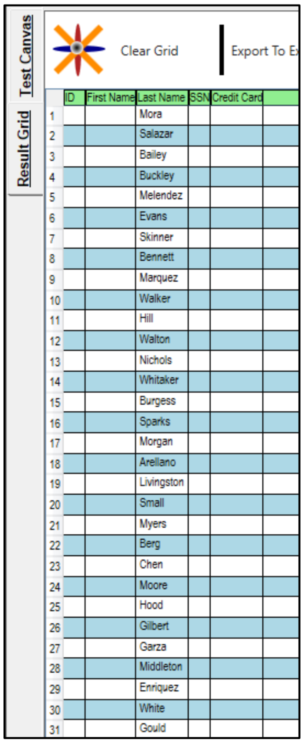
If the string entered is only 2 components

First component is Table name

2nd component is the name of the column to be updated

(No row number to be provided)

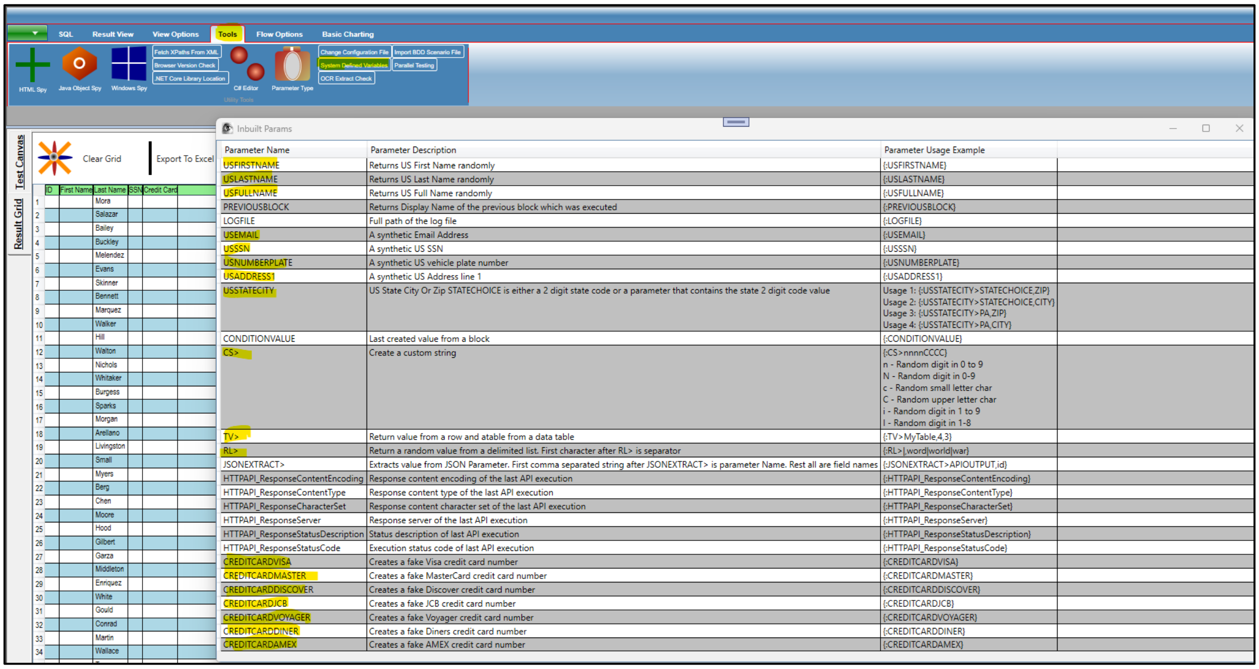
When executed entire column will be updated



	ID	First Name	Last Name	SSN	Credit Card	
1			Mora			
2			Salazar			
3			Bailey			
4			Buckley			
5			Melendez			
6			Evans			
7			Skinner			
8			Bennett			
9			Marquez			
10			Walker			
11			Hill			
12			Walton			
13			Nichols			
14			Whitaker			
15			Burgess			
16			Sparks			
17			Morgan			
18			Arellano			
19			Livingston			
20			Small			
21			Myers			
22			Berg			
23			Chen			
24			Moore			
25			Hood			
26			Gilbert			
27			Garza			
28			Middleton			
29			Enriquez			
30			White			
31			Gould			

Available System Variables For Data Generation

Navigate to **Tools** ribbon Menu → **System Defined Variables**



	ID	First Name	Last Name	SSN	Credit Card
1			Mora		
2			Salazar		
3			Bailey		
4			Buckley		
5			Melendez		
6			Evans		
7			Skinner		
8			Bennett		
9			Marquez		
10			Walker		
11			Hill		
12			Walton		
13			Nichols		
14			Whitaker		
15			Burgess		
16			Sparks		
17			Morgan		
18			Arellano		
19			Livingston		
20			Small		
21			Myers		
22			Berg		
23			Chen		
24			Moore		
25			Hood		
26			Gilbert		
27			Garza		
28			Middleton		
29			Enriquez		
30			White		
31			Gould		
32			Conrad		
33			Martin		
34			Wallace		

Parameter Name	Parameter Description	Parameter Usage Example
USFIRSTNAME	Returns US First Name randomly	{:USFIRSTNAME}
USLASTNAME	Returns US Last Name randomly	{:USLASTNAME}
USFULLNAME	Returns US Full Name randomly	{:USFULLNAME}
PREVIOUSBLOCK	Returns Display Name of the previous block which was executed	{:PREVIOUSBLOCK}
LOGFILE	Full path of the log file	{:LOGFILE}
USEMAIL	A synthetic Email Address	{:USEMAIL}
USSSN	A synthetic US SSN	{:USSSN}
USNUMBERPLATE	A synthetic US vehicle plate number	{:USNUMBERPLATE}
USADDRESS1	A synthetic US Address line 1	{:USADDRESS1}

USSTATECITY	US State City Or Zip STATECHOICE is either a 2 digit state code or a parameter that contains the state 2 digit code value	Usage 1: {:USSTATECITY>STATECHOICE,ZIP}
		Usage 2: {:USSTATECITY>STATECHOICE,CITY}
		Usage 3: {:USSTATECITY>PA,ZIP}
		Usage 4: {:USSTATECITY>PA,CITY}

CONDITIONVALUE	Last created value from a block	{:CONDITIONVALUE}
CS>	Create a custom string	{:CS>nnnnCCCC} n - Random digit in 0 to 9 N - Random digit in 0-9 c - Random small letter char C - Random upper letter char i - Random digit in 1 to 9 I - Random digit in 1-8
TV>	Return value from a row and atable from a data table	{:TV>MyTable,4,3}
RL>	Return a random value from a delimited list. First character after RL> is separator	{:RL>|,word|world|war}
JSONEXTRACT>	Extracts value from JSON Parameter. First comma separated string after JSONEXTRACT> is parameter Name. Rest all are field names	{:JSONEXTRACT>APIOUTPUT,id}
HTTPAPI_ResponseContentEncoding	Response content encoding of the last API execution	{:HTTPAPI_ResponseContentEncoding}
HTTPAPI_ResponseContentType	Response content type of the last API execution	{:HTTPAPI_ResponseContentType}
HTTPAPI_ResponseCharacterSet	Response content character set of the last API execution	{:HTTPAPI_ResponseCharacterSet}
HTTPAPI_ResponseServer	Response server of the last API execution	{:HTTPAPI_ResponseServer}
HTTPAPI_ResponseStatusDescription	Status description of last API execution	{:HTTPAPI_ResponseStatusDescription}
HTTPAPI_ResponseStatusCode	Execution status code of last API execution	{:HTTPAPI_ResponseStatusCode}
CREDITCARDVISA	Creates a fake Visa credit card number	{:CREDITCARDVISA}
CREDITCARDMASTER	Creates a fake MasterCard credit card number	{:CREDITCARDMASTER}
CREDITCARDDISCOVER	Creates a fake Discover credit card number	{:CREDITCARDDISCOVER}
CREDITCARDJCB	Creates a fake JCB credit card number	{:CREDITCARDJCB}
CREDITCARDVOYAGER	Creates a fake Voyager credit card number	{:CREDITCARDVOYAGER}
CREDITCARDDINER	Creates a fake Diners credit card number	{:CREDITCARDDINER}
CREDITCARDAMEX	Creates a fake AMEX credit card number	{:CREDITCARDAMEX}

This will give available variables for data generation.

USFIRSTNAME → Random first name

USLASTNAME → Random last name

USEMAIL → Random email id

USSSN → Randomvalid SSN number

CS> → Custom random string

TV> → Table values from other data tables created during execution.

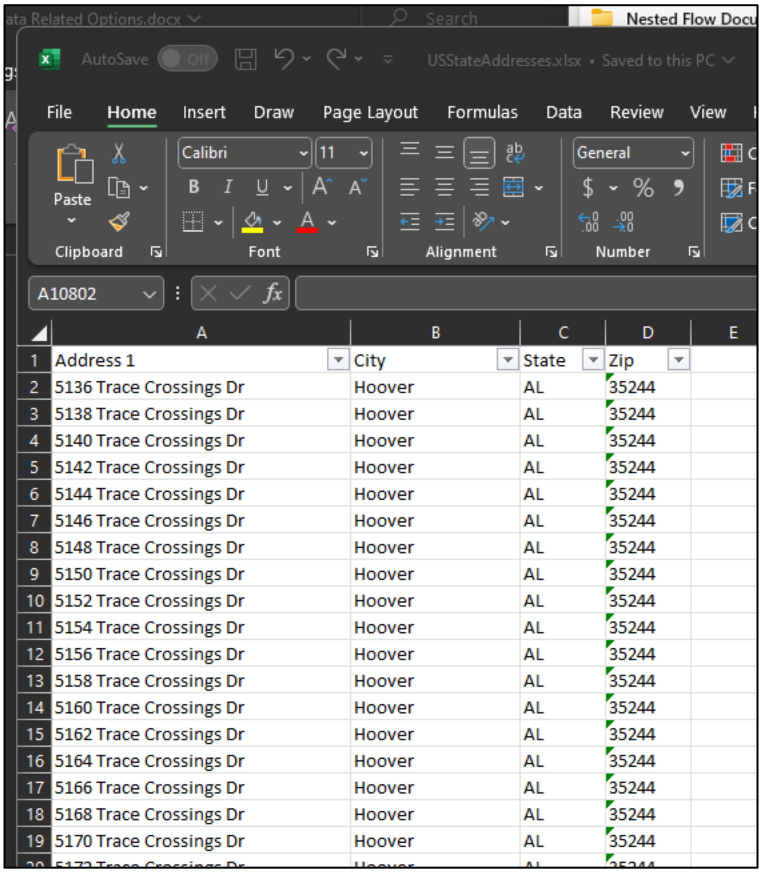
All other test data variables are credit card numbers.

Database Hosted DataSets

This feature requires the tool to be connected to SQL Server database.

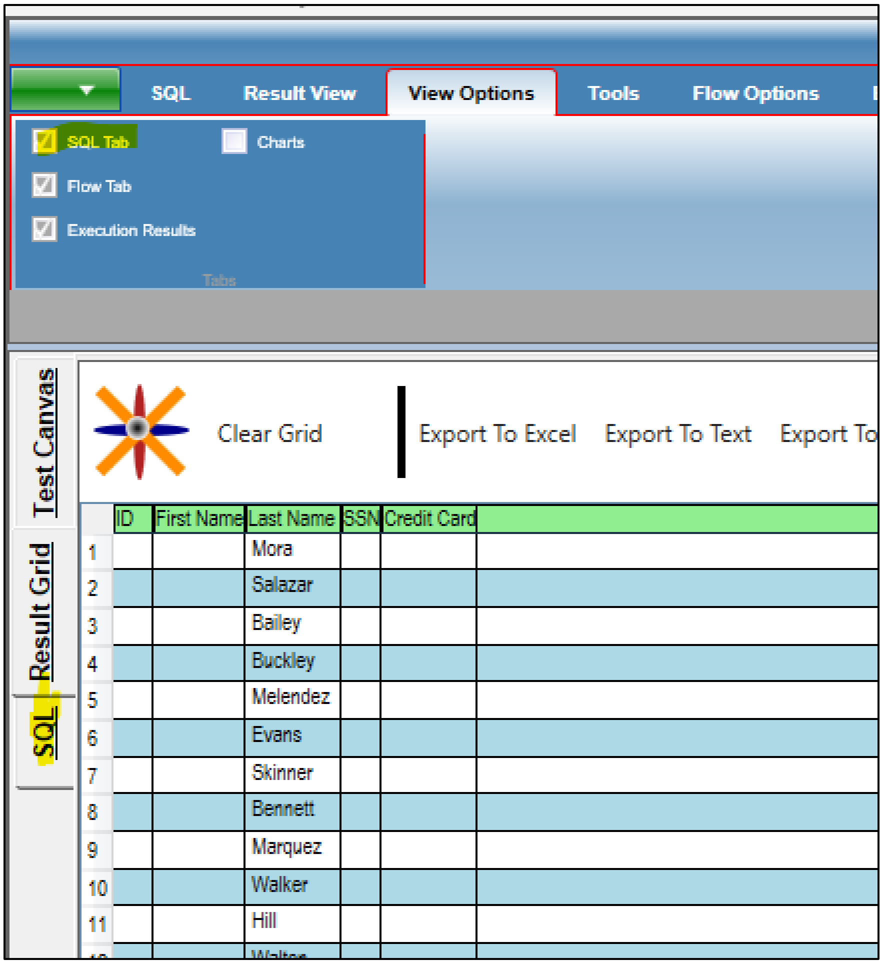
Example of Data Set:

Download the file NestedFlowRepo/Data/USStateAddresses.xlsx at main · nestedflow-Automation/NestedFlowRepo (github.com). This file has 200 valid addresses for each state and most of the territories.



	Address 1	City	State	Zip
1	Address 1	City	State	Zip
2	5136 Trace Crossings Dr	Hoover	AL	35244
3	5138 Trace Crossings Dr	Hoover	AL	35244
4	5140 Trace Crossings Dr	Hoover	AL	35244
5	5142 Trace Crossings Dr	Hoover	AL	35244
6	5144 Trace Crossings Dr	Hoover	AL	35244
7	5146 Trace Crossings Dr	Hoover	AL	35244
8	5148 Trace Crossings Dr	Hoover	AL	35244
9	5150 Trace Crossings Dr	Hoover	AL	35244
10	5152 Trace Crossings Dr	Hoover	AL	35244
11	5154 Trace Crossings Dr	Hoover	AL	35244
12	5156 Trace Crossings Dr	Hoover	AL	35244
13	5158 Trace Crossings Dr	Hoover	AL	35244
14	5160 Trace Crossings Dr	Hoover	AL	35244
15	5162 Trace Crossings Dr	Hoover	AL	35244
16	5164 Trace Crossings Dr	Hoover	AL	35244
17	5166 Trace Crossings Dr	Hoover	AL	35244
18	5168 Trace Crossings Dr	Hoover	AL	35244
19	5170 Trace Crossings Dr	Hoover	AL	35244
20	5172 Trace Crossings Dr	Hoover	AL	35244

Navigate to View Options and select **SQL Tab** option. SQL Tab will appear



☑ SQL Tab

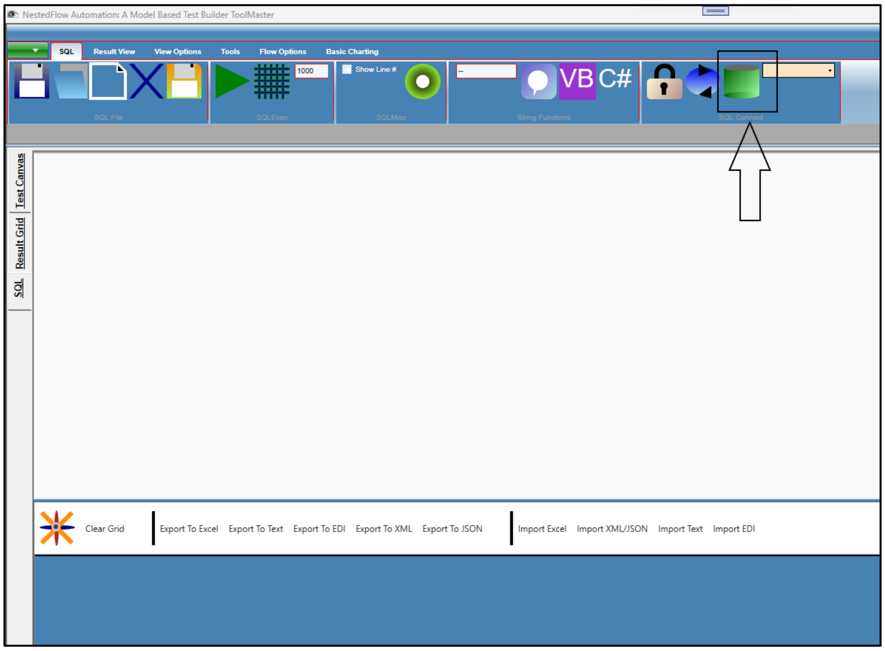
☑ Flow Tab

☑ Execution Results

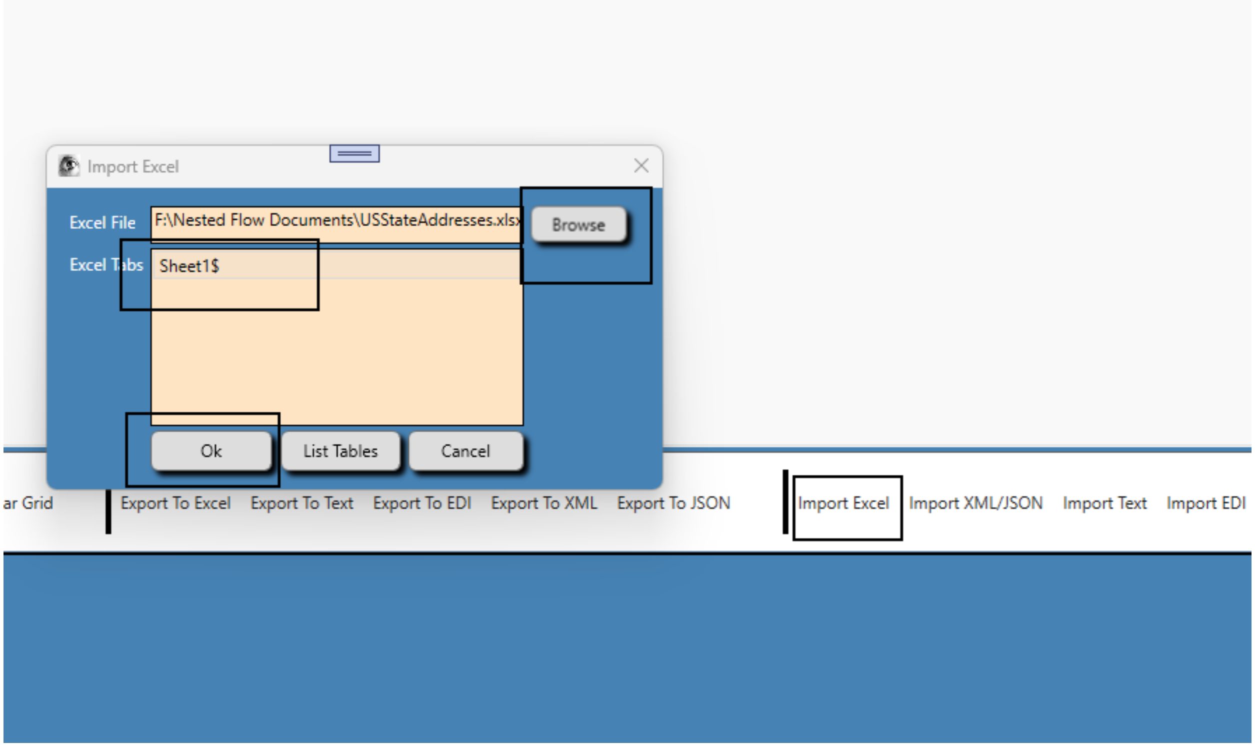
□ Charts

	ID	First Name	Last Name	SSN	Credit Card
1			Mora		
2			Salazar		
3			Bailey		
4			Buckley		
5			Melendez		
6			Evans		
7			Skinner		
8			Bennett		
9			Marquez		
10			Walker		
11			Hill		

Click on SQL Tab and click on Connect to TDM DB button



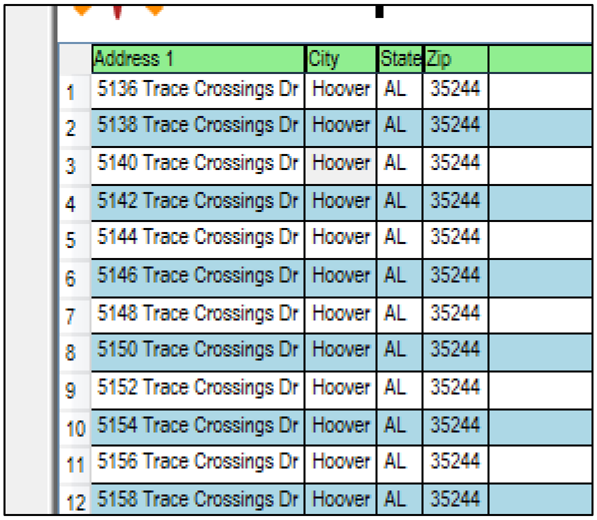
Click on **Import Excel** button



Excel File: F:\Nested Flow Documents\USStateAddresses.xlsx

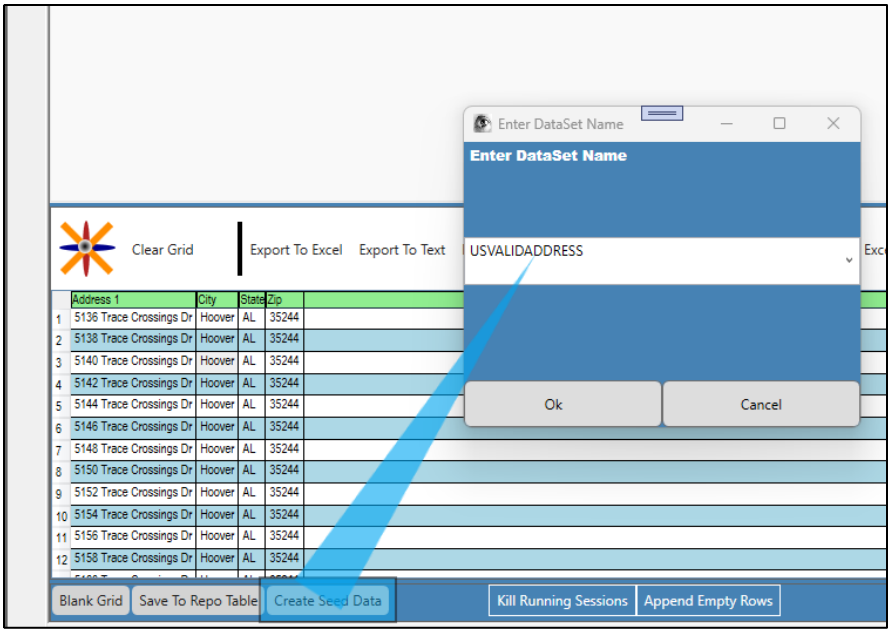
Excel Tabs: Sheet1$

In the **Import Excel** form, click on **Browse** button to choose the excel and click on the sheet you want to import and click on **Ok** button (Or double click on the sheet name). The Excel data will be shown on the grid.



	Address 1	City	State	Zip	
1	5136 Trace Crossings Dr	Hoover	AL	35244	
2	5138 Trace Crossings Dr	Hoover	AL	35244	
3	5140 Trace Crossings Dr	Hoover	AL	35244	
4	5142 Trace Crossings Dr	Hoover	AL	35244	
5	5144 Trace Crossings Dr	Hoover	AL	35244	
6	5146 Trace Crossings Dr	Hoover	AL	35244	
7	5148 Trace Crossings Dr	Hoover	AL	35244	
8	5150 Trace Crossings Dr	Hoover	AL	35244	
9	5152 Trace Crossings Dr	Hoover	AL	35244	
10	5154 Trace Crossings Dr	Hoover	AL	35244	
11	5156 Trace Crossings Dr	Hoover	AL	35244	
12	5158 Trace Crossings Dr	Hoover	AL	35244	

Click on **Create Seed Data** button



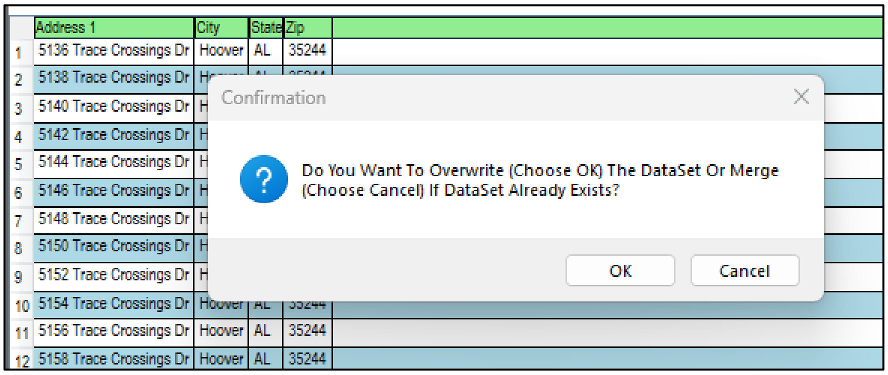
Enter DataSet Name

USVALIDADDRESS

	Address 1	City	State	Zip
1	5136 Trace Crossings Dr	Hoover	AL	35244
2	5138 Trace Crossings Dr	Hoover	AL	35244
3	5140 Trace Crossings Dr	Hoover	AL	35244
4	5142 Trace Crossings Dr	Hoover	AL	35244
5	5144 Trace Crossings Dr	Hoover	AL	35244
6	5146 Trace Crossings Dr	Hoover	AL	35244
7	5148 Trace Crossings Dr	Hoover	AL	35244
8	5150 Trace Crossings Dr	Hoover	AL	35244
9	5152 Trace Crossings Dr	Hoover	AL	35244
10	5154 Trace Crossings Dr	Hoover	AL	35244
11	5156 Trace Crossings Dr	Hoover	AL	35244
12	5158 Trace Crossings Dr	Hoover	AL	35244

Enter seed data name to be created and Click on **Ok** button

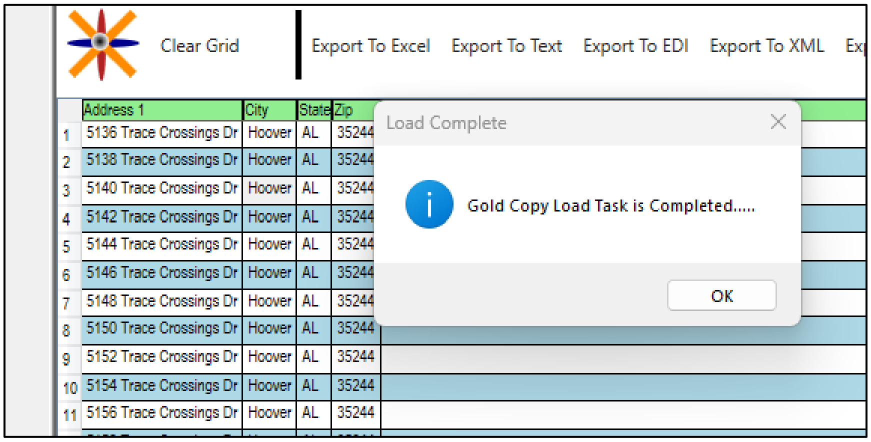
Click on **Ok** button to overwrite if dataset already exists or **Cancel** button if you want to merge with existing data in the seed data



Do You Want To Overwrite (Choose OK) The DataSet Or Merge (Choose Cancel) If DataSet Already Exists?

	Address 1	City	State	Zip
1	5136 Trace Crossings Dr	Hoover	AL	35244
2	5138 Trace Crossings Dr	H		
3	5140 Trace Crossings Dr	H		
4	5142 Trace Crossings Dr	H		
5	5144 Trace Crossings Dr	H		
6	5146 Trace Crossings Dr	H		
7	5148 Trace Crossings Dr	H		
8	5150 Trace Crossings Dr	H		
9	5152 Trace Crossings Dr	H		
10	5154 Trace Crossings Dr	Hoover	AL	35244
11	5156 Trace Crossings Dr	Hoover	AL	35244
12	5158 Trace Crossings Dr	Hoover	AL	35244

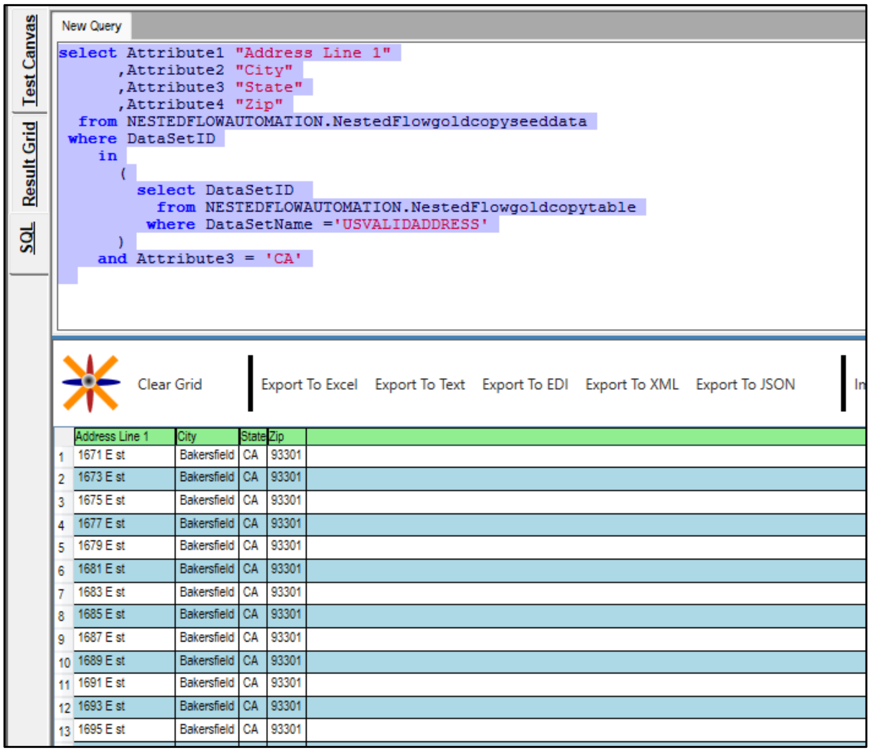
Click on **Ok** button on the success message



Gold Copy Load Task is Completed.....

	Address 1	City	State	Zip
1	5136 Trace Crossings Dr	Hoover	AL	35244
2	5138 Trace Crossings Dr	Hoover	AL	35244
3	5140 Trace Crossings Dr	Hoover	AL	35244
4	5142 Trace Crossings Dr	Hoover	AL	35244
5	5144 Trace Crossings Dr	Hoover	AL	35244
6	5146 Trace Crossings Dr	Hoover	AL	35244
7	5148 Trace Crossings Dr	Hoover	AL	35244
8	5150 Trace Crossings Dr	Hoover	AL	35244
9	5152 Trace Crossings Dr	Hoover	AL	35244
10	5154 Trace Crossings Dr	Hoover	AL	35244
11	5156 Trace Crossings Dr	Hoover	AL	35244

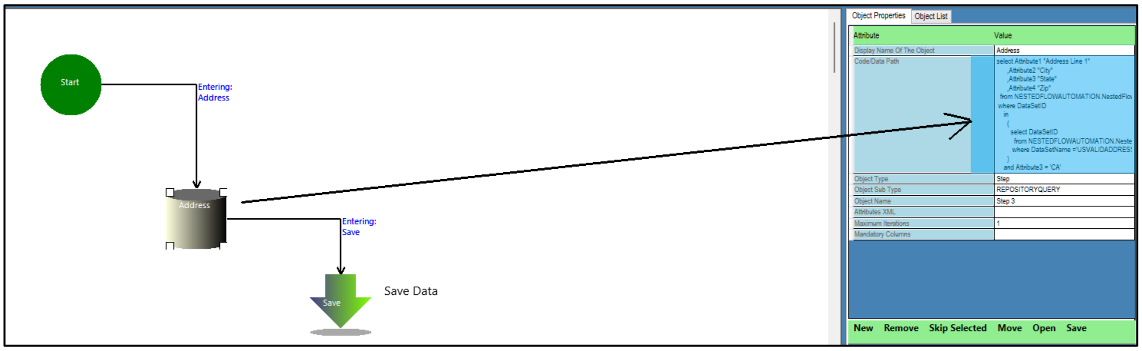
Now, gold copy seed data can be queries to generate the data we want (below example gives addresses for California)



```
select Attribute1 "Address Line 1"
      ,Attribute2 "City"
      ,Attribute3 "State"
      ,Attribute4 "Zip"
  from NESTEDFLOWAUTOMATION.NestedFlowgoldcopyseeddata
 where DataSetID
    in
      (
        select DataSetID
          from NESTEDFLOWAUTOMATION.NestedFlowgoldcopytable
         where DataSetName ='USVALIDADDRESS'
      )
    and Attribute3 = 'CA'
```

	Address Line 1	City	State	Zip
1	1671 E st	Bakersfield	CA	93301
2	1673 E st	Bakersfield	CA	93301
3	1675 E st	Bakersfield	CA	93301
4	1677 E st	Bakersfield	CA	93301
5	1679 E st	Bakersfield	CA	93301
6	1681 E st	Bakersfield	CA	93301
7	1683 E st	Bakersfield	CA	93301
8	1685 E st	Bakersfield	CA	93301
9	1687 E st	Bakersfield	CA	93301
10	1689 E st	Bakersfield	CA	93301
11	1691 E st	Bakersfield	CA	93301
12	1693 E st	Bakersfield	CA	93301
13	1695 E st	Bakersfield	CA	93301

Same can be derived in the flow and can be saved to be used as runtime datatable



Attribute	Value
Display Name Of The Object	Address
Code/Data Path	select Attribute1 "Address Line 1" ,Attribute2 "City" ,Attribute3 "State" ,Attribute4 "Zip" from NESTEDFLOWAUTOMATION.NestedFlo where DataSetID in (select DataSetID from NESTEDFLOWAUTOMATION.Neste where DataSetName ='USVALIDADDRES) and Attribute3 = 'CA'
Object Type	Step
Object Sub Type	REPOSITORYQUERY
Object Name	Step 3
Attributes XML	
Maximum Iterations	1
Mandatory Columns	

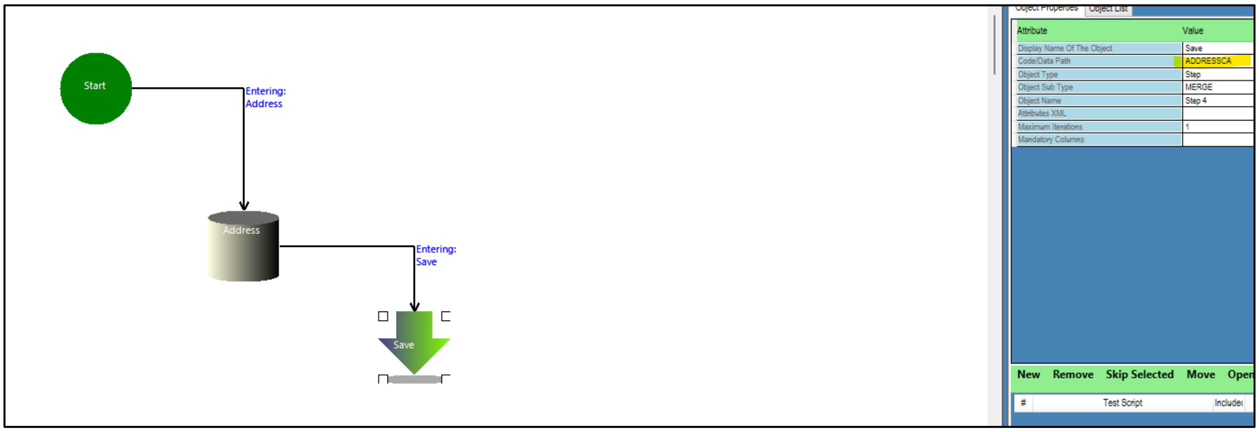
You can also choose a random address by below query

```
select TOP 1*
  from
  (
    select Attribute1 "Address Line 1"
          ,Attribute2 "City"
          ,Attribute3 "State"
          ,Attribute4 "Zip"
      from NESTEDFLOWAUTOMATION.NestedFlowgoldcopyseeddata
     where DataSetID
        in
```

```
       (
         select DataSetID
           from NESTEDFLOWAUTOMATION.NestedFlowgoldcopytable
          where DataSetName ='USVALIDADDRESS'
       )
     and Attribute3 = 'CA'
)A
order by Newid()
```

Code On Data Table

We can write custom C# code for modifying data. In this example we will add a 4th column and provide full address data by concatenating all other fields in it

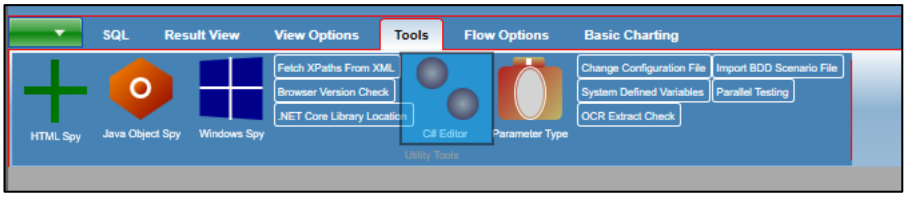


Attribute	Value
Display Name Of The Object	Save
Code/Data Path	ADDRESSCA
Object Type	Step
Object Sub Type	MERGE
Object Name	Step 4
Attributes XML	
Maximum Iterations	1
Mandatory Columns	

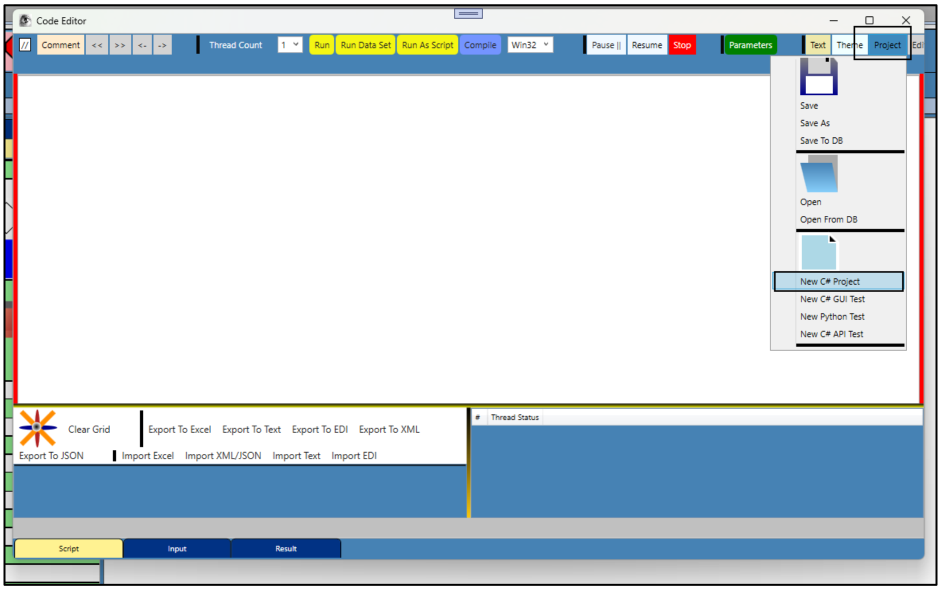
Let us save the Address data in a data table as shown above

Single Table Code

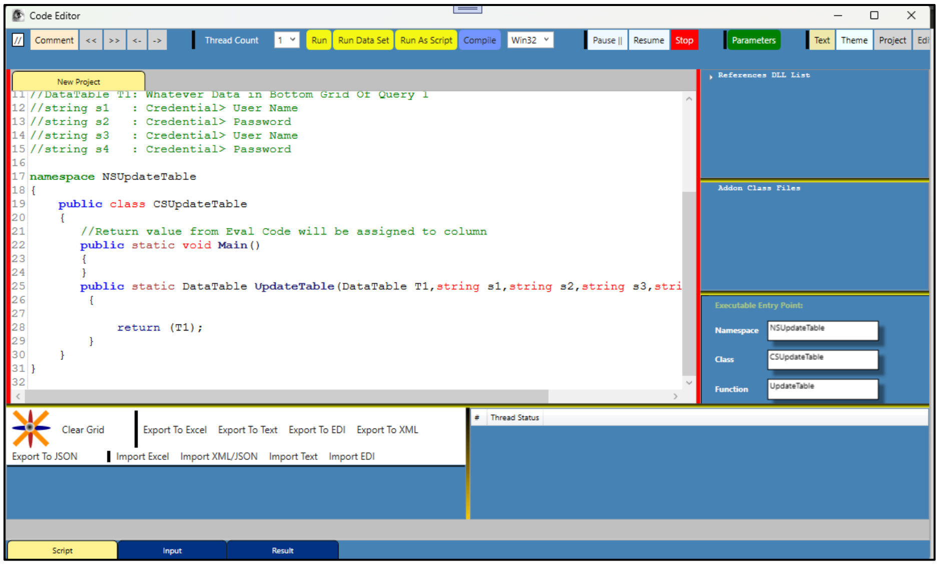
Navigate to **Tools → C# Editor**



Navigate to **Project → New C# Project**



Code structure will be auto created



```
//DataTable T1: Whatever Data in Bottom Grid Of Query 1
//string s1   : Credential> User Name
//string s2   : Credential> Password
//string s3   : Credential> User Name
//string s4   : Credential> Password

namespace NSUpdateTable
{
    public class CSUpdateTable
    {
       //Return value from Eval Code will be assigned to column
       public static void Main()
       {
       }
       public static DataTable UpdateTable(DataTable T1,string s1,string s2,string s3,stri
        {

            return (T1);
        }
    }
}
```

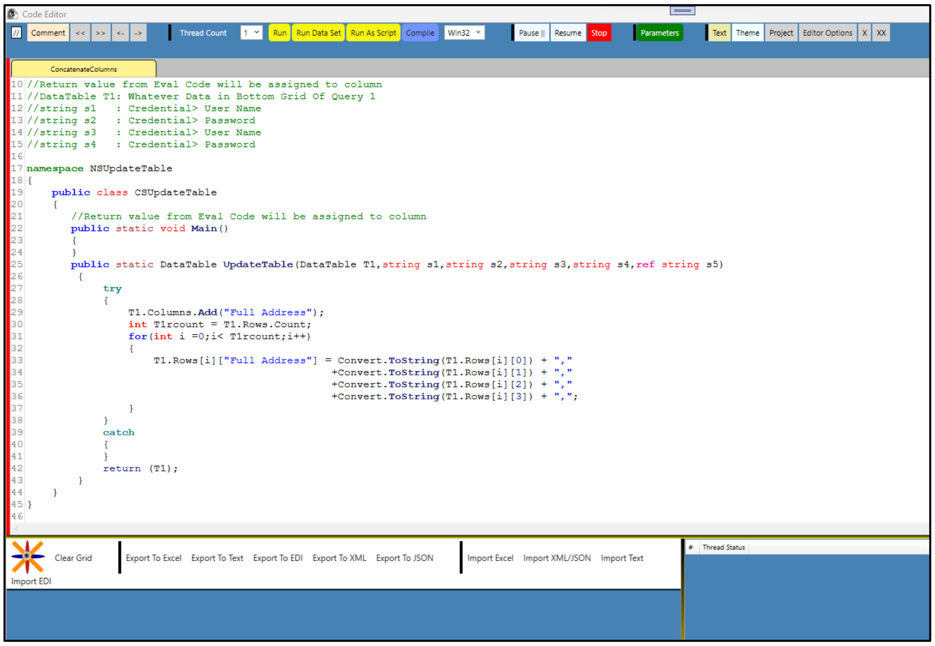
Executable Entry Point:

Namespace NSUpdateTable

Class CSUpdateTable

Function UpdateTable

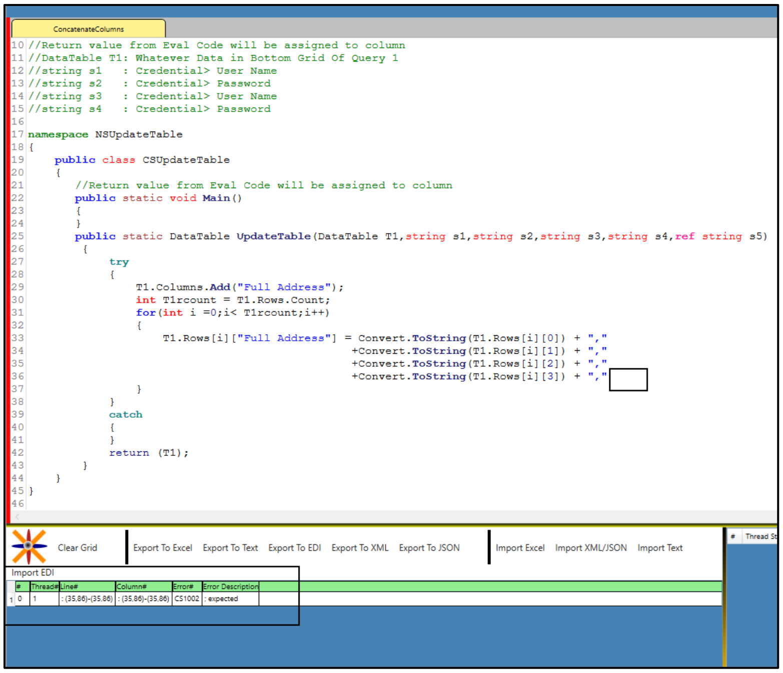
Either Save in file system or to database



```
//Return value from Eval Code will be assigned to column
//DataTable T1: Whatever Data in Bottom Grid Of Query 1
//string s1   : Credential> User Name
//string s2   : Credential> Password
//string s3   : Credential> User Name
//string s4   : Credential> Password

namespace NSUpdateTable
{
    public class CSUpdateTable
    {
       //Return value from Eval Code will be assigned to column
       public static void Main()
       {
       }
       public static DataTable UpdateTable(DataTable T1,string s1,string s2,string s3,string s4,ref string s5)
        {
            try
            {
                T1.Columns.Add("Full Address");
                int T1rcount = T1.Rows.Count;
                for(int i =0;i< T1rcount;i++)
                {
                    T1.Rows[i]["Full Address"] = Convert.ToString(T1.Rows[i][0]) + ","
                                                 +Convert.ToString(T1.Rows[i][1]) + ","
                                                 +Convert.ToString(T1.Rows[i][2]) + ","
                                                 +Convert.ToString(T1.Rows[i][3]) + ",";
                }
            }
            catch
            {
            }
            return (T1);
        }
    }
}
```

Click on **Compile** button

If errors exist, resolve them and recompile

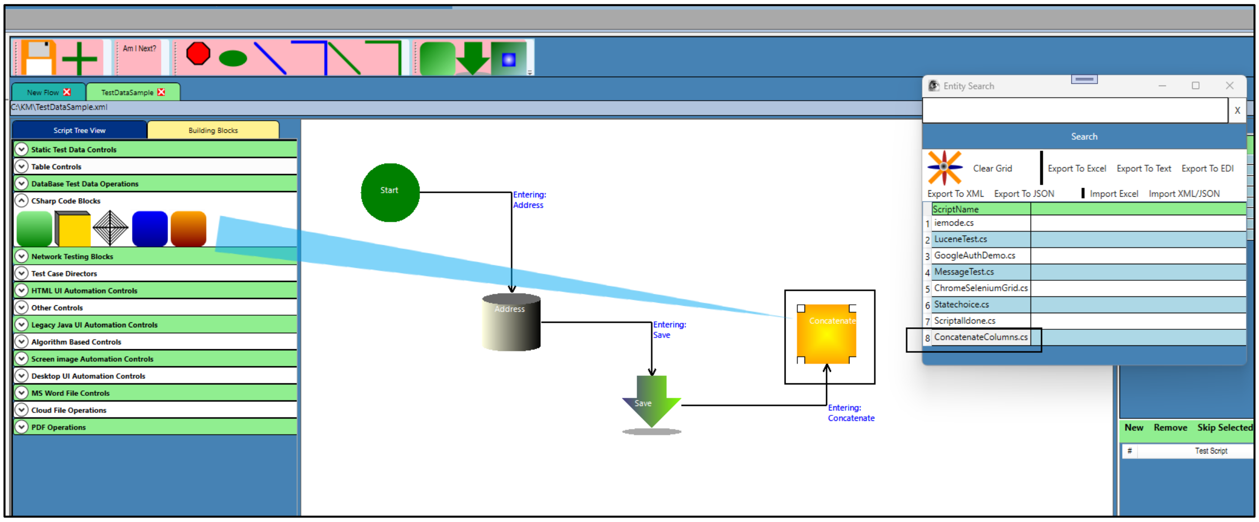


```
//Return value from Eval Code will be assigned to column
//DataTable T1: Whatever Data in Bottom Grid Of Query 1
//string s1   : Credential> User Name
//string s2   : Credential> Password
//string s3   : Credential> User Name
//string s4   : Credential> Password

namespace NSUpdateTable
{
    public class CSUpdateTable
    {
       //Return value from Eval Code will be assigned to column
       public static void Main()
       {
       }
       public static DataTable UpdateTable(DataTable T1,string s1,string s2,string s3,string s4,ref string s5)
        {
            try
            {
                T1.Columns.Add("Full Address");
                int T1rcount = T1.Rows.Count;
                for(int i =0;i< T1rcount;i++)
                {
                    T1.Rows[i]["Full Address"] = Convert.ToString(T1.Rows[i][0]) + ","
                                                +Convert.ToString(T1.Rows[i][1]) + ","
                                                +Convert.ToString(T1.Rows[i][2]) + ","
                                                +Convert.ToString(T1.Rows[i][3]) + ","
                }
            }
            catch
            {
            }
            return (T1);
        }
    }
}
```

	#	Thread#	Line#	Column#	Error#	Error Description
1	0	1	: (35,86)-(35,86)	: (35,86)-(35,86)	CS1002	; expected

Close the form.

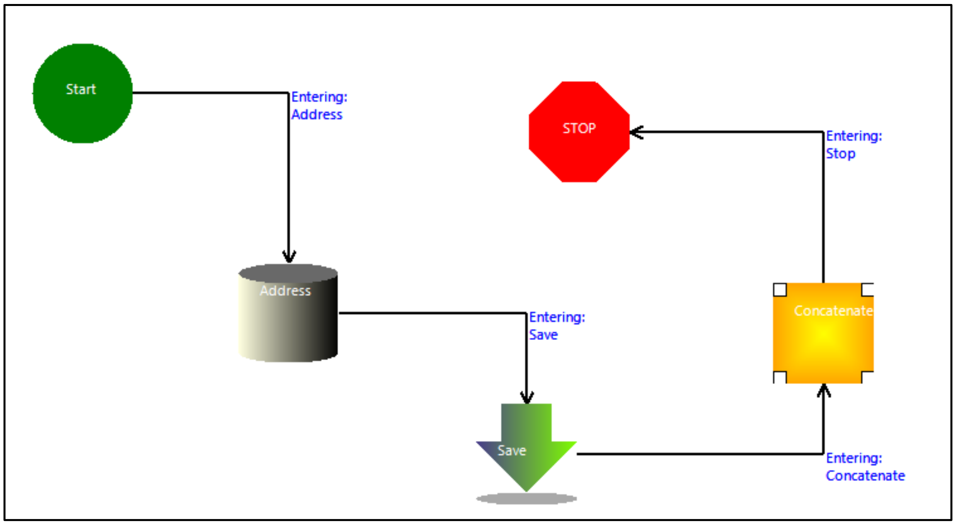
Add a Code block to the flow



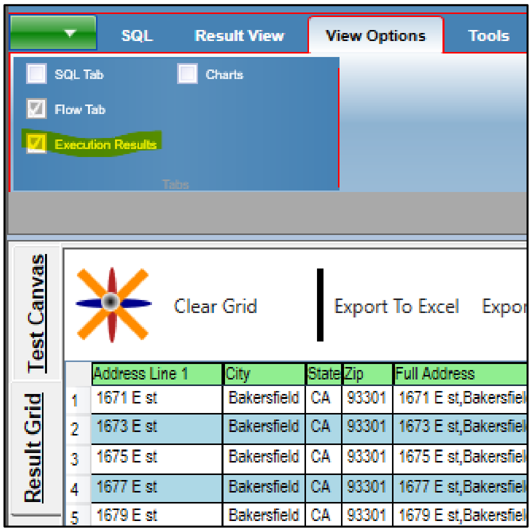
	ScriptName
1	iemode.cs
2	LuceneTest.cs
3	GoogleAuthDemo.cs
4	MessageTest.cs
5	ChromeSeleniumGrid.cs
6	Statechoice.cs
7	Scriptalldone.cs
8	ConcatenateColumns.cs

Right Click on the block and add the script either from file or from database (in this case used database)

Complete the flow by adding Stop block:



The result: (Result Grid can be enabled by ribbon menu **View Options → Execution Results**)

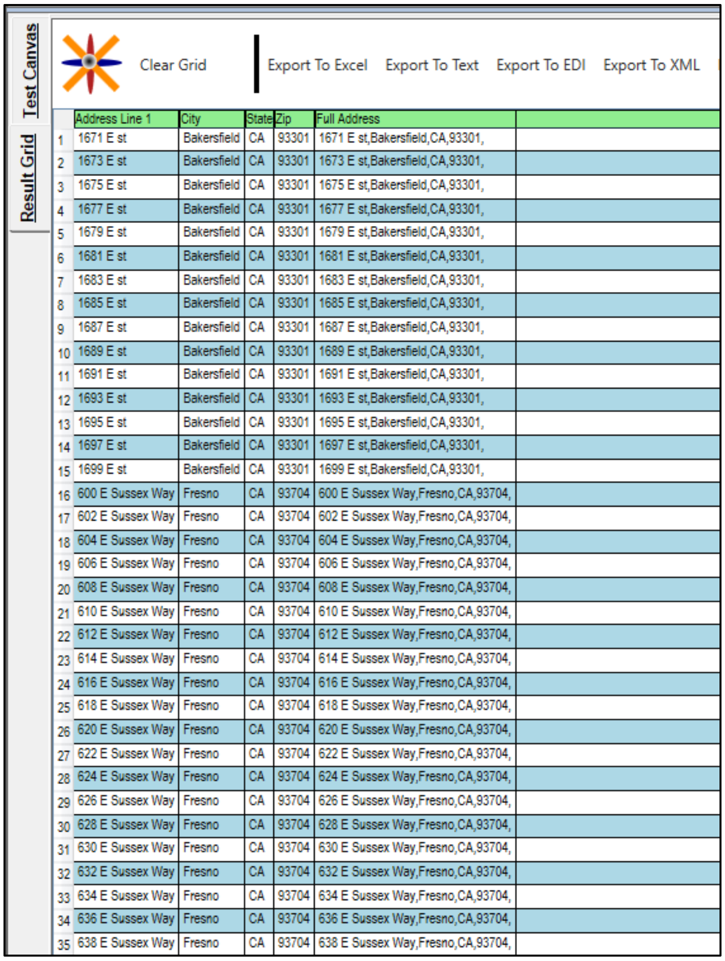


□ SQL Tab □ Charts

☑ Flow Tab

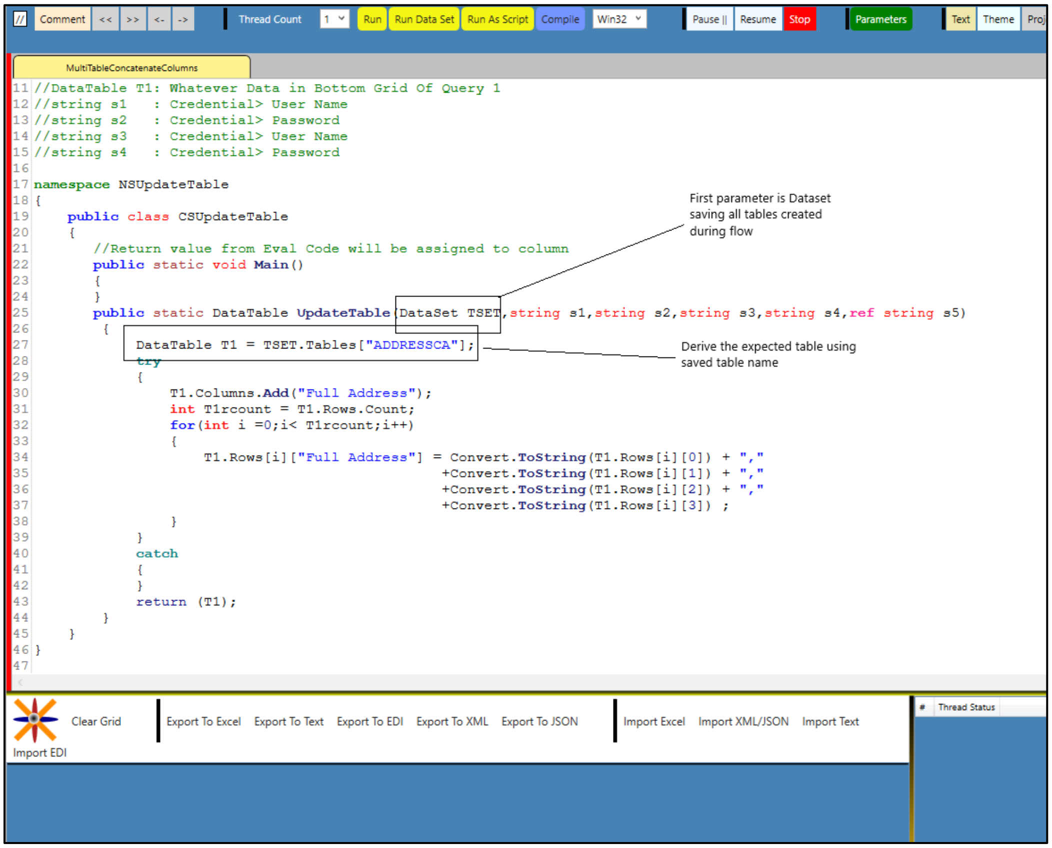
☑ Execution Results

	Address Line 1	City	State	Zip	Full Address
1	1671 E st	Bakersfield	CA	93301	1671 E st,Bakersfiel
2	1673 E st	Bakersfield	CA	93301	1673 E st,Bakersfiel
3	1675 E st	Bakersfield	CA	93301	1675 E st,Bakersfiel
4	1677 E st	Bakersfield	CA	93301	1677 E st,Bakersfiel
5	1679 E st	Bakersfield	CA	93301	1679 E st,Bakersfiel



	Address Line 1	City	State	Zip	Full Address	
1	1671 E st	Bakersfield	CA	93301	1671 E st,Bakersfield,CA,93301,	
2	1673 E st	Bakersfield	CA	93301	1673 E st,Bakersfield,CA,93301,	
3	1675 E st	Bakersfield	CA	93301	1675 E st,Bakersfield,CA,93301,	
4	1677 E st	Bakersfield	CA	93301	1677 E st,Bakersfield,CA,93301,	
5	1679 E st	Bakersfield	CA	93301	1679 E st,Bakersfield,CA,93301,	
6	1681 E st	Bakersfield	CA	93301	1681 E st,Bakersfield,CA,93301,	
7	1683 E st	Bakersfield	CA	93301	1683 E st,Bakersfield,CA,93301,	
8	1685 E st	Bakersfield	CA	93301	1685 E st,Bakersfield,CA,93301,	
9	1687 E st	Bakersfield	CA	93301	1687 E st,Bakersfield,CA,93301,	
10	1689 E st	Bakersfield	CA	93301	1689 E st,Bakersfield,CA,93301,	
11	1691 E st	Bakersfield	CA	93301	1691 E st,Bakersfield,CA,93301,	
12	1693 E st	Bakersfield	CA	93301	1693 E st,Bakersfield,CA,93301,	
13	1695 E st	Bakersfield	CA	93301	1695 E st,Bakersfield,CA,93301,	
14	1697 E st	Bakersfield	CA	93301	1697 E st,Bakersfield,CA,93301,	
15	1699 E st	Bakersfield	CA	93301	1699 E st,Bakersfield,CA,93301,	
16	600 E Sussex Way	Fresno	CA	93704	600 E Sussex Way,Fresno,CA,93704,	
17	602 E Sussex Way	Fresno	CA	93704	602 E Sussex Way,Fresno,CA,93704,	
18	604 E Sussex Way	Fresno	CA	93704	604 E Sussex Way,Fresno,CA,93704,	
19	606 E Sussex Way	Fresno	CA	93704	606 E Sussex Way,Fresno,CA,93704,	
20	608 E Sussex Way	Fresno	CA	93704	608 E Sussex Way,Fresno,CA,93704,	
21	610 E Sussex Way	Fresno	CA	93704	610 E Sussex Way,Fresno,CA,93704,	
22	612 E Sussex Way	Fresno	CA	93704	612 E Sussex Way,Fresno,CA,93704,	
23	614 E Sussex Way	Fresno	CA	93704	614 E Sussex Way,Fresno,CA,93704,	
24	616 E Sussex Way	Fresno	CA	93704	616 E Sussex Way,Fresno,CA,93704,	
25	618 E Sussex Way	Fresno	CA	93704	618 E Sussex Way,Fresno,CA,93704,	
26	620 E Sussex Way	Fresno	CA	93704	620 E Sussex Way,Fresno,CA,93704,	
27	622 E Sussex Way	Fresno	CA	93704	622 E Sussex Way,Fresno,CA,93704,	
28	624 E Sussex Way	Fresno	CA	93704	624 E Sussex Way,Fresno,CA,93704,	
29	626 E Sussex Way	Fresno	CA	93704	626 E Sussex Way,Fresno,CA,93704,	
30	628 E Sussex Way	Fresno	CA	93704	628 E Sussex Way,Fresno,CA,93704,	
31	630 E Sussex Way	Fresno	CA	93704	630 E Sussex Way,Fresno,CA,93704,	
32	632 E Sussex Way	Fresno	CA	93704	632 E Sussex Way,Fresno,CA,93704,	
33	634 E Sussex Way	Fresno	CA	93704	634 E Sussex Way,Fresno,CA,93704,	
34	636 E Sussex Way	Fresno	CA	93704	636 E Sussex Way,Fresno,CA,93704,	
35	638 E Sussex Way	Fresno	CA	93704	638 E Sussex Way,Fresno,CA,93704,	

Multi Table Code

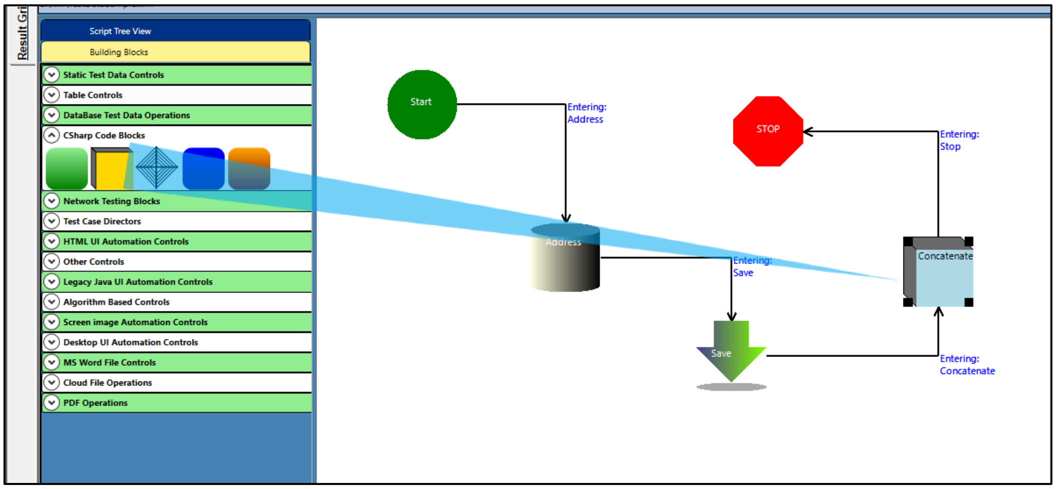


```
//DataTable T1: Whatever Data in Bottom Grid Of Query 1
//string s1   : Credential> User Name
//string s2   : Credential> Password
//string s3   : Credential> User Name
//string s4   : Credential> Password

namespace NSUpdateTable
{
    public class CSUpdateTable
    {
       //Return value from Eval Code will be assigned to column
       public static void Main()
       {
       }
       public static DataTable UpdateTable(DataSet TSET,string s1,string s2,string s3,string s4,ref string s5)
         {
            DataTable T1 = TSET.Tables["ADDRESSCA"];
            try
            {
                T1.Columns.Add("Full Address");
                int T1rcount = T1.Rows.Count;
                for(int i =0;i< T1rcount;i++)
                {
                    T1.Rows[i]["Full Address"] = Convert.ToString(T1.Rows[i][0]) + ","
                                                +Convert.ToString(T1.Rows[i][1]) + ","
                                                +Convert.ToString(T1.Rows[i][2]) + ","
                                                +Convert.ToString(T1.Rows[i][3]) ;

                }
            }
            catch
            {
            }
            return (T1);
         }
    }
}
```

First parameter is Dataset saving all tables created during flow

Derive the expected table using saved table name

And linking this table to DataSet code block



This kind of coding is useful when you need to access more than 1 data table at a time