Contents

[Scripts: 1](#_Toc145601004)

[Note: 6](#_Toc145601005)

[Structure: Flowchart of LINQ 6](#_Toc145601006)

[XML Practice: 8](#_Toc145601007)

[LINQ Functions 10](#_Toc145601008)

[Errors: 13](#_Toc145601009)

# Scripts:

1. **Compare the values of 2 columns using the static condition:**

dtInput.AsEnumerable().Where(Function(row) row("Col1").ToString="A" AndAlso CInt(row("Col2").ToString)=Math.Abs(3)).CopyToDataTable

1. **Group the values according to their keys:**

dtInput.AsEnumerable().GroupBy(Function (row) row("Col1").ToString)

1. **Group the values of a speific key & performs the given operation:**

(

From row In dtInput.AsEnumerable

Group row By k=row("Col1").toString.trim Into grp=Group

Let ra = New Object(){k,grp.Sum(Function (x) Convert.ToDouble(x("Col2").toString.Trim))}

Select dtInput.Rows.Add(ra)

).CopyToDataTable

1. **Get duplicate rows from data table:**

(

From row In dtInput

Group row By a=row("Col1").ToString.Trim, b=Math.Abs(CInt(row("Col2").ToString.Trim)) Into grp=Group

Where grp.Count>1

Select grp.ToList

).SelectMany(Function(x) x).CopyToDataTable

1. **Get unique values by ignoring the duplicates and getting only First unique row value where it appears more than once:**

Here we are not using “Where” keyword as we are not writing any condition. If you want to include one or more columns in the requirement, see point 4. At the end we want to get the output as data table so we write CopyToDataTable

(

From row In dtInput

Group row By a=row("Col1").ToString.Trim Into grp=Group

Select grp.First

).CopyToDataTable

1. **Get row values that are not duplicates (remove duplicate rows):**

Here we have used keyword “Where” inorder to put the condition saying group count should be only 1 which means there should be no duplicates of the group & then “Select” the First group value (by default it will be only 1 group as it is not duplicate and so First keyword). If you want to include one or more columns in the requirement, see point 4. At the end we want to get the output as data table so we write CopyToDataTable

(

From row In dtInput

Group row By a=row("Col1").ToString.Trim Into grp=Group

Where grp.Count=1

Select grp.First

).CopyToDataTable

1. **Find the duplicate rows and get only 1st row value out of those duplicates:**

Here we have used keyword “Where” in order to put the condition saying group count should be greater than 1 (which means there are duplicates of the group) & then “Select” the First group value (ignore rest of the duplicates). At the end we want to get the output as data table so we write CopyToDataTable

(

From row In dtInput

Group row By a=row("Col1").ToString.Trim Into grp=Group

Where grp.Count>1

Select grp.First

).CopyToDataTable

1. **Find duplicate rows from 2 columns (2nd column will have integers with both + and – sign which should be ignored) and get only those duplicate rows:**

Here we are using Math.Abs to make sure the sign of the integer will be ignored & only values will be compared. Count>1 says that the group should be repeated more than once. ToList says that we are converting the group to List. SelectMany says that from this List we want toselect many rows. To do this, we create a function (x) and from this function we want to get each row so we write x. At the end we want to get the output as data table so we write CopyToDataTable

(

From row In dtInput

Group row By a=row("Col1").ToString.Trim, b=Math.Abs(CInt(row("Col2").ToString.Trim)) Into grp=Group

Where grp.Count>1

Select grp.ToList

).SelectMany(Function(x) x).CopyToDataTable

1. **In an array, get the numbers mod 2=0 and store in array:**

Here we aren’t using Group function because it is not needed

(

From num In NumArray

Where num Mod 2=0

Select num

).ToArray

If we want to get the count of output array, then just change the last part from ToArray->Count

(

From num In NumArray

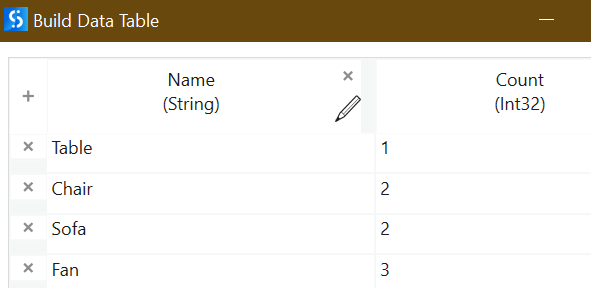
Where num Mod 2=0

Select num

).Count

Else without writing this 2nd query, we can just take the 1st script and assign it to a variable “output\_NumArray” and then use assign activity count=output\_NumArray.Count

1. **Input is data table, output should be array where it has only the names having count column as 2:**



(

From row In dtInput

Where row("Count").ToString="2" Let a=row("Name").ToString

Select a

).ToArray

**The same requirement but output should be datatable: without keyword “GROUP”**

**1st method: without AsEnumerable():**

(

From row In dtInput

Where row("Count").ToString="2"

Select row

).CopyToDataTable

**2nd method: with AsEnumerable():**

dtInput.AsEnumerable().Where (Function(row) row("Count").ToString="2").CopyToDataTable

1. **Sort the Column 1 values order:**

**Ascending 1 column:**

(

From row In dtInput

Order By row("Column1") Ascending

Select row

).CopyToDataTable

**Descending 2 columns:**

(

From row In dtInput

Order By row("Column1") Descending, row("Column2") Descending

Select row

).CopyToDataTable

1. **Join 2 data tables by getting only the common values in both, include rest of the columns as well which are corresponding to this common column value:**

(

From a In dtInput1

**Join** b In dtInput2

**On** a("Col1").ToString Equals b("Col1").ToString

Select dtOutput.Rows.Add({a("Col1"),a("Col2"),b("Col3")})

).CopyToDataTable

1. **Get a row which doesn’t meet the criteria:**

(

From row In dtInput

**Where Not** row("Column2").ToString.**Contains**("USA")

Select row

).CopyToDataTable

Instead of hardcoding USA, we can pass variable as well. Variable Country=”USA”

**Where Not** row("Column2").ToString.**Contains**(**Country**)

1. **Remove empty rows**

**from data table column A & keep the rest:**

(

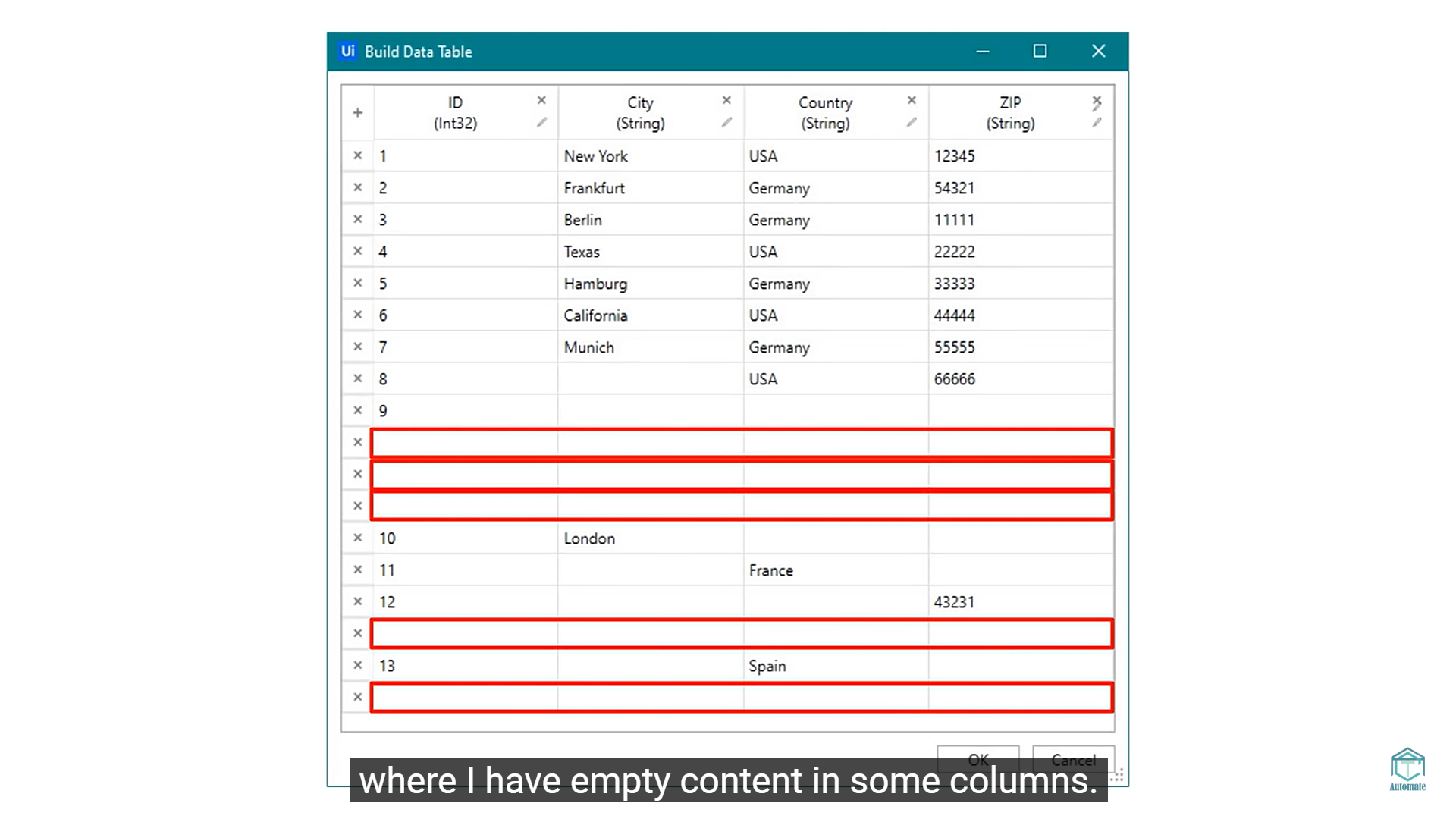
From row In dtInput

Where Not row("A").ToString=Nothing

Select row

).CopyToDataTable

**If data table is having 100+ columns and we need to remove the rows that has all columns in it empty**



**The below query gives output more than once as there are 2 loops we have introduced here. That’s why we use the word “Distinct”. “Cast” is used because we are looping through the columns as well**

(

From row In dtInput

From col In dtInput.Columns.**Cast**(Of DataColumn)

Where Not row(col).ToString.Trim=Nothing

Select row

).**Distinct**.CopyToDataTable

1. **Get the row having number in any of the columns:**

(

From row In dtInput

From **col** In dtInput.Columns.**Cast**(Of DataColumn)

Where (row(col).ToString.Trim).IsNumeric

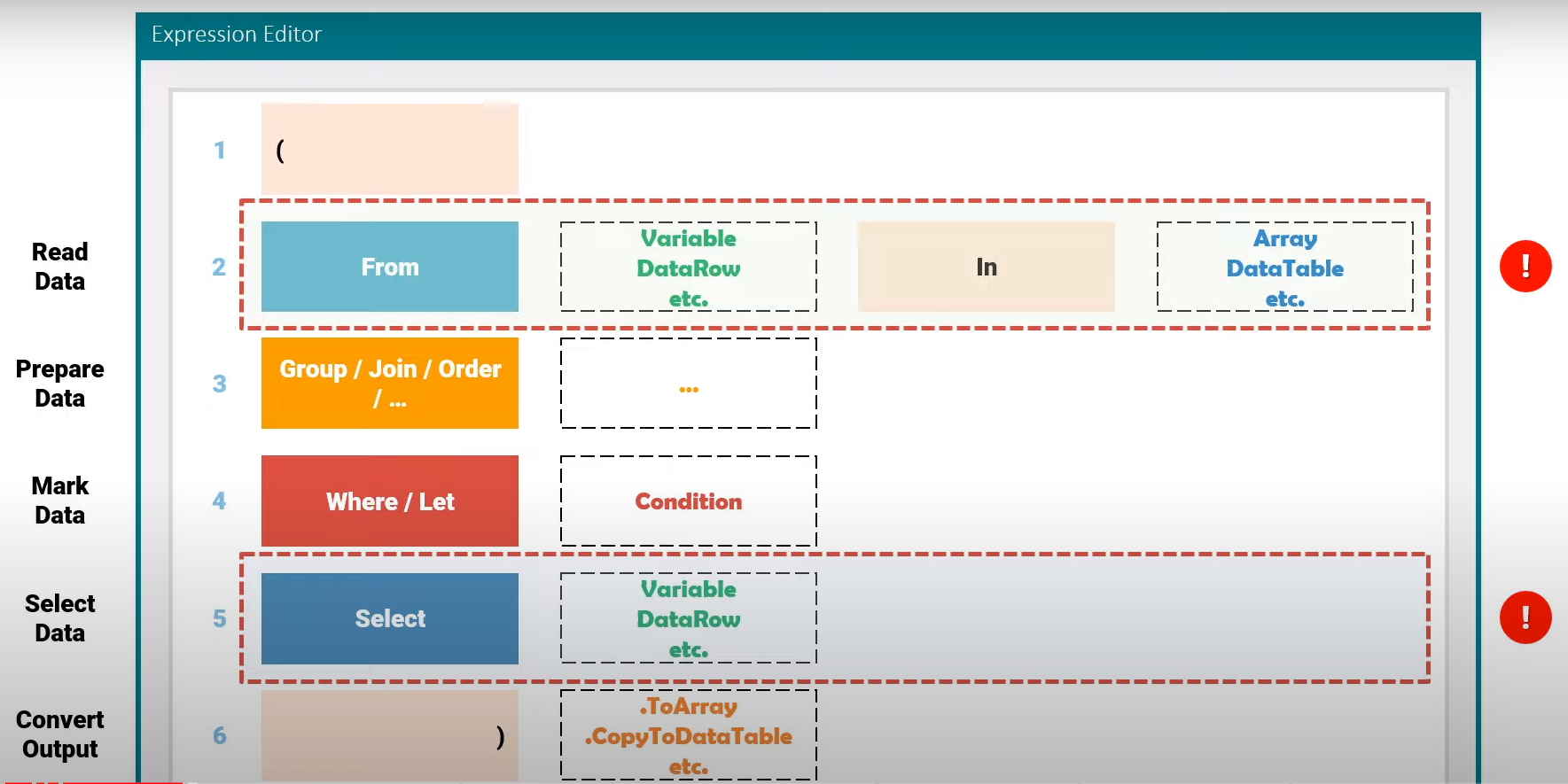
Select row

).Distinct.CopyToDataTable

# Note:

1. Instead of writing Column names (like row(“Col1”).ToString), we can even write column index (like row(0).ToString)
2. While using “JOIN” function, it’s not mandatory to use same column name in outputDT while building it but it’s good to use as it will be clearer to us
3. Instead of writing all the parent elements in XML case, we can just write “Descendants”
4. We can add 2 conditions in “Where” function by using “And” or “Or” keywords

# Structure: Flowchart of LINQ



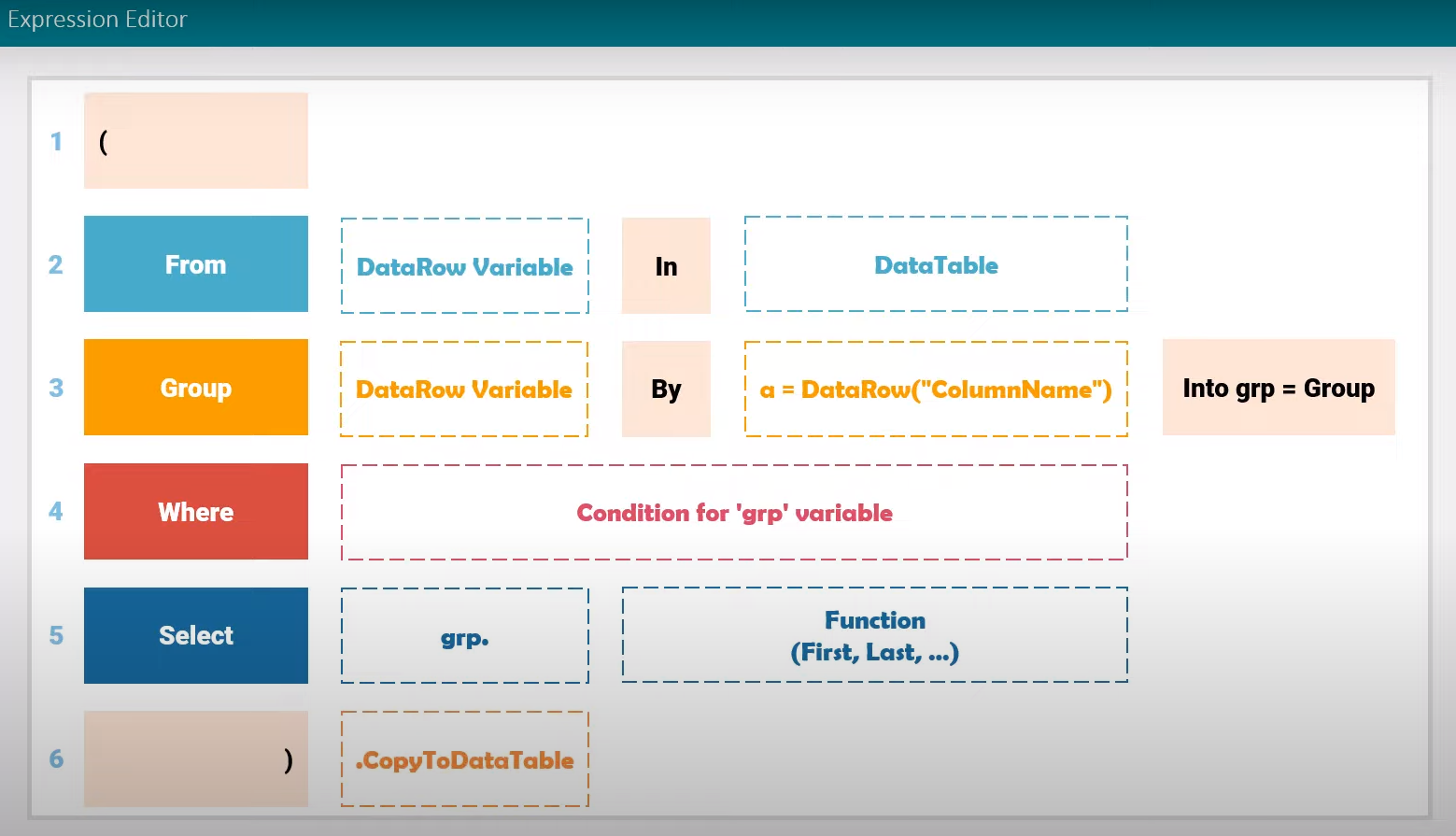
1. Bracket at the beginning and at the end Within these brackets we have the other four parts
2. The first part is FROM statement. With this statement we can get the data from the data source.

To do that we need to define a variable Then we set the preposition IN And then we write the name of the data source, from which we will read the data. For example, we have a data table, and we want to get the rows from this data table. So, we define row variable then we write the preposition in and then we write the name of this data table. So, the first part is to get the entire data.

1. In this part we can use a function like, group, join, order etc. With this part we can prepare the data, which we got from the data source, before we continue with the next parts. For example: we want to group the rows of a data table using the city column. We can do that in this part. So in this line we can prepare the data that we have fetched from the data source, so that we only look at this data and not the entire data source. For example, we group the rows based on a column in a data table, then we can later get the data from this group data. So this part is to prepare the data
2. With this statement we can set a condition to mark the data (WHERE), which we need to select later or with LET statement we can also define a new variable to get the data from the data source with some calculations. For example, I need to mark all rows of the data table, where the name of an employee is Max. So, this part is to mark the data which we want to get as output later. With the Where statement we mark the desired data then with the Select statement (next point 5) we get the data that we have selected.
3. The last part in the brackets is Select statement. After we defined the variable and the data source and set the condition, we can here select the data, which we need to get as output. For example, I need to get all rows of the data table where the name is Max. So, this part is to select the data to get them as output.
4. After brackets, we can set a function to convert the output to a certain data type. For example, I want to get the output as a data table, so I can add the function copy to data table. Simply means to change the data we got into a specific data type. So, this part is to convert the output

**NOTE**: ***From these six parts, only two are necessary (From and Select statements). All others can be used but not mandatory.***

**Structure: Data Table LINQ**

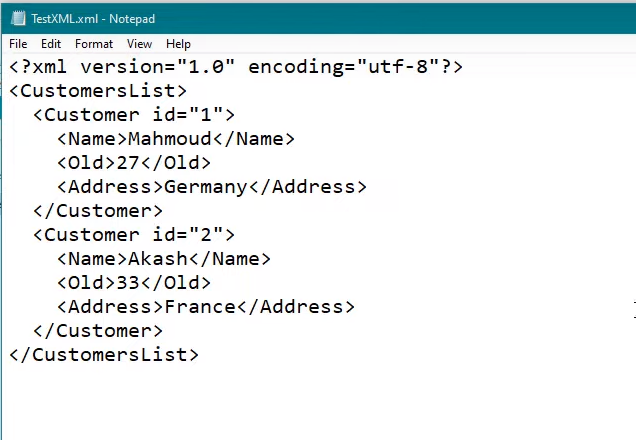


# XML Practice:

1. In an xml containing Customers List, get the name of the customer whose ID=1



Created Input file of type xml



***<?xml version=”1.0” encoding=”utf-8”?>***

***<CustomersList>***

***<Customer id=”1”>***

***<Name>Achu</Name>***

***<Old>27</Old>***

***<Address>Germany</Address>***

***</Customer>***

***<Customer id=”2”>***

***<Name>Ammu</Name>***

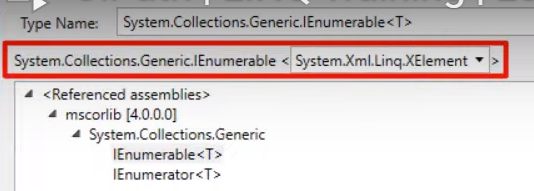
***<Old>23</Old>***

***<Address>France</Address>***

***</Customer>***

***</CustomersList>***

Create a variable xDoc of type System.Xml.Linq.XDocument and another output variable xNames of type System.Collections.Generic.IEnumerable<System.Xml.Linq.XDocument>



Assigned: **xDoc=XDocument.Load("C:\Users\anl001\Documents\UiPath\LINQ\SameValueFilter\Data\xmlDocInput.xml")**

Assigned:

**xNames=**

**(**

**From el In xDoc.Elements("CustomersList").Elements("Customer")**

**Where el.Attribute("id").ToString="1"**

**Select el.Element("Name")**

**)**

Question: Is there a way to get the element without calling their parent elements as there might be 1+ parent elements in few cases?

Yes, there is a way called **Descendants**. This will make sure we no need to call parent elements

(

From el In xDoc.**Descendants**("Customer")

Where CStr(el.Attribute("id"))="1"

Select el.Element("Name")

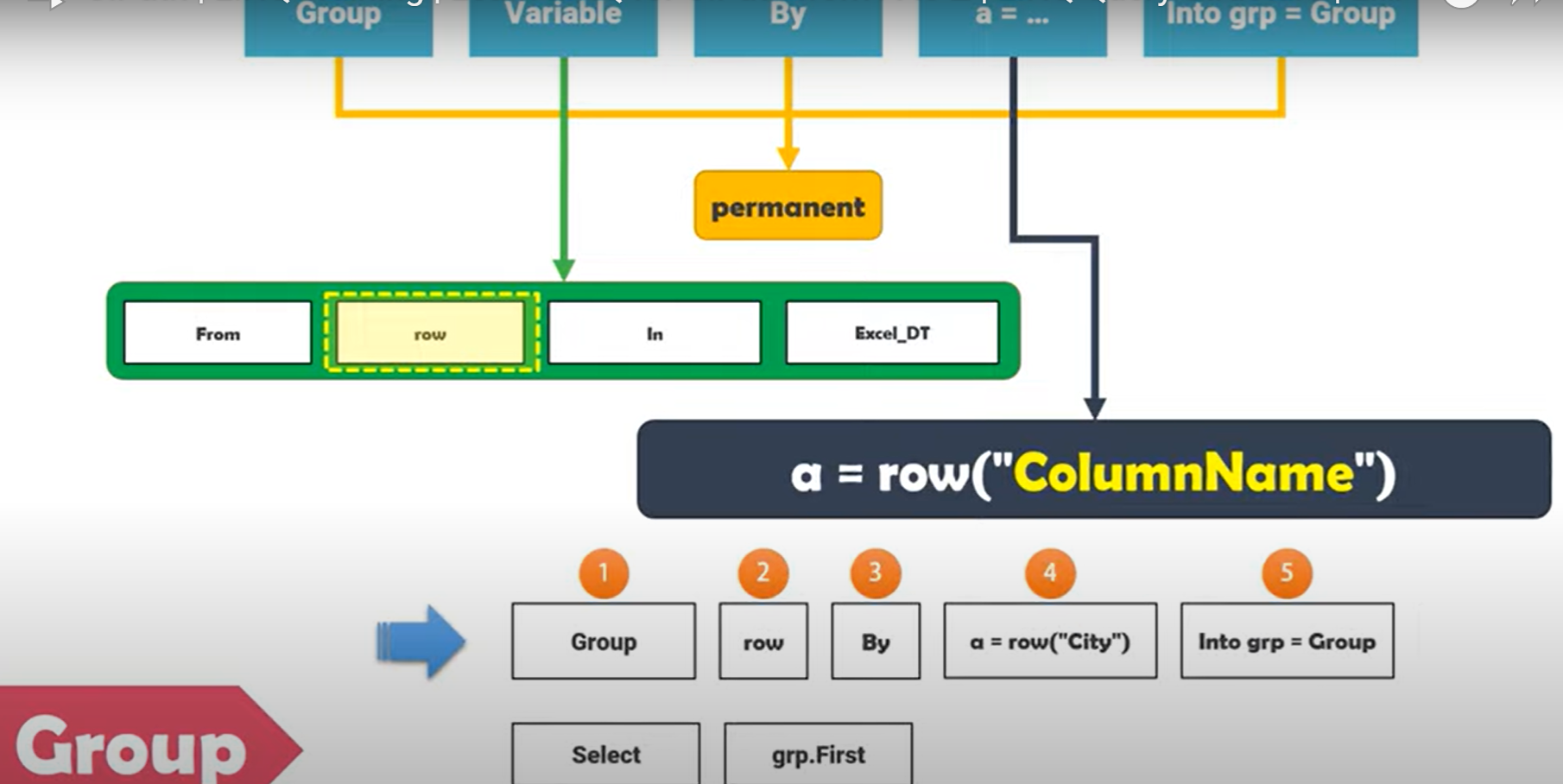
)

# LINQ Functions

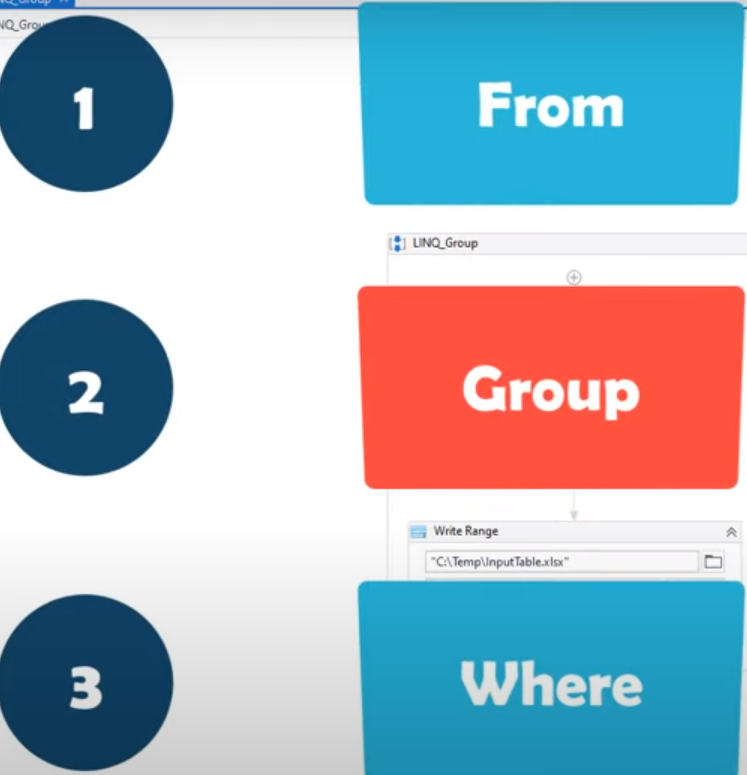
There are 3 functions: Group, Order & Join

1. **GROUP**:

It has 5 parts in it – Group, variable, By, condition, assignment



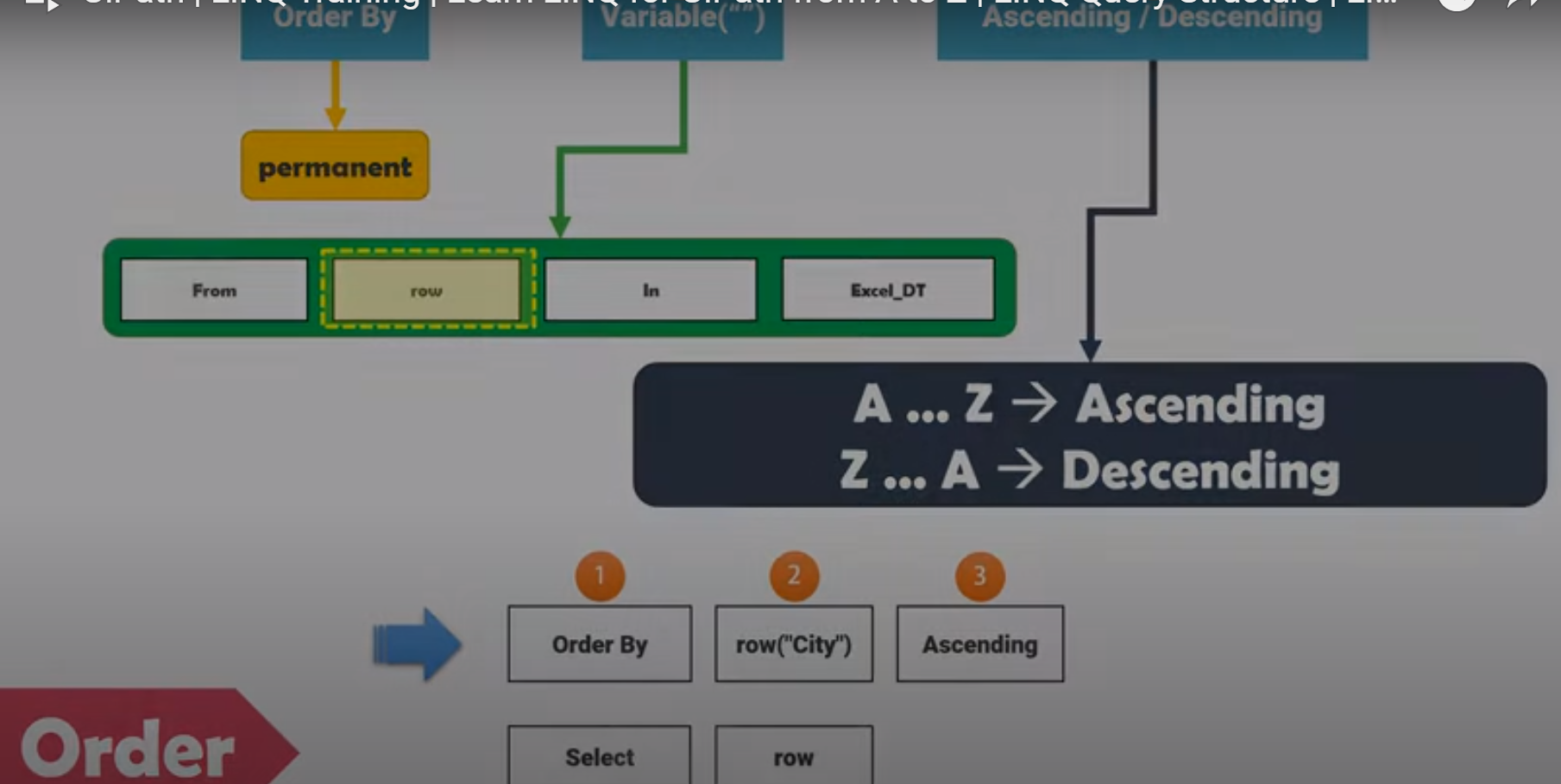
Order hierarchy to place this function “GROUP”



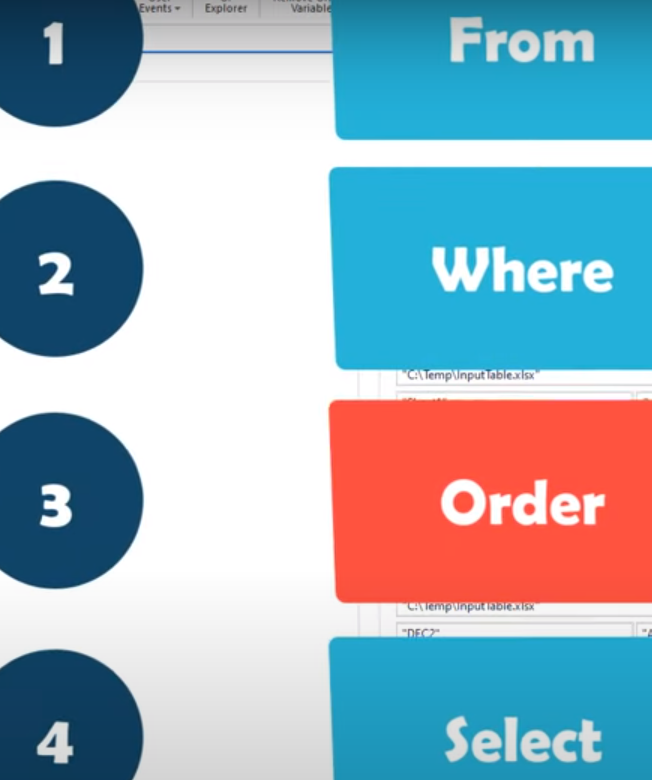
1. **ORDER**:

It has 3 parts in it – Order By, Variable, Ascending/Descending

Example: Script 11



Order hierarchy to place this function “GROUP”

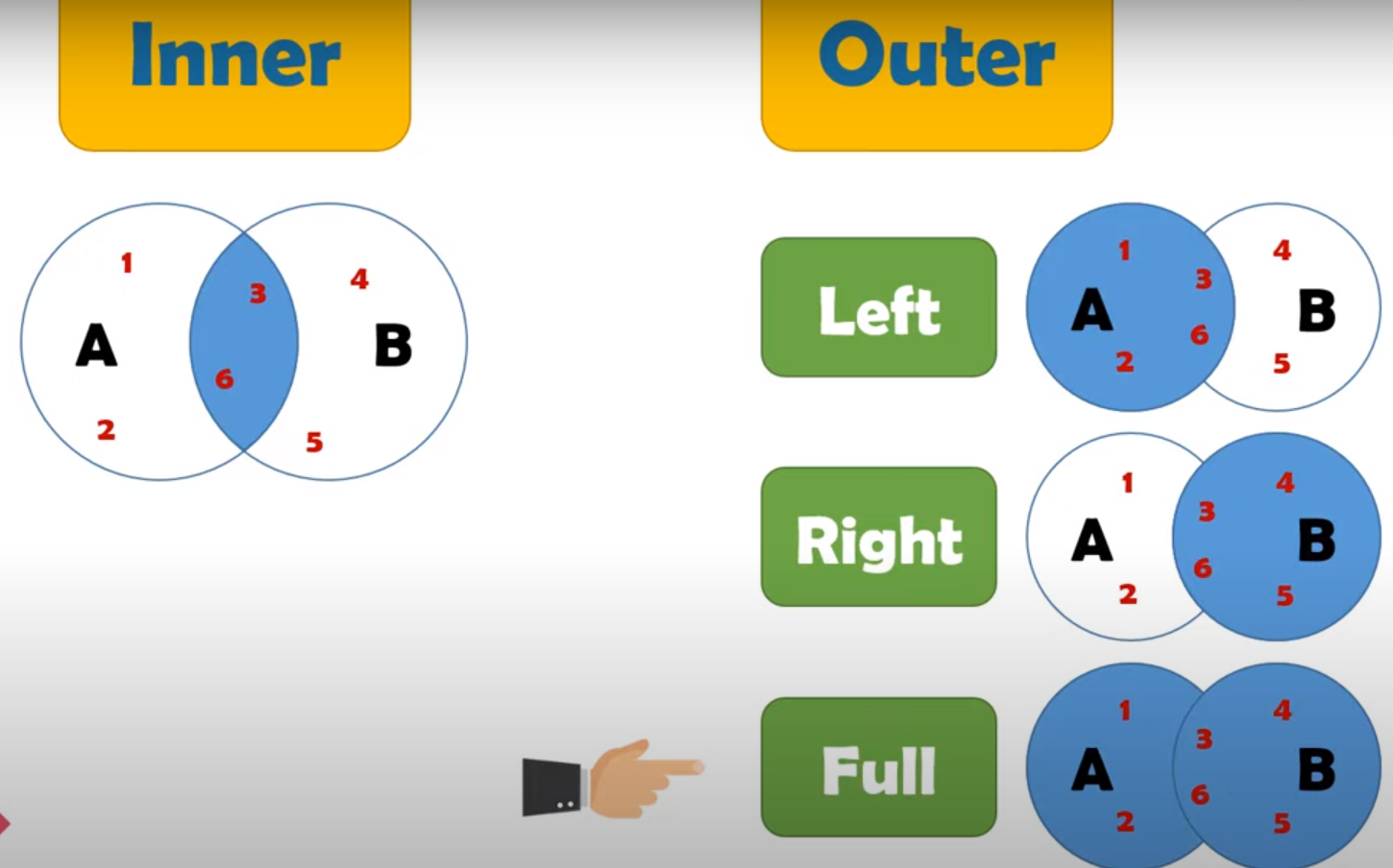


1. **JOIN**

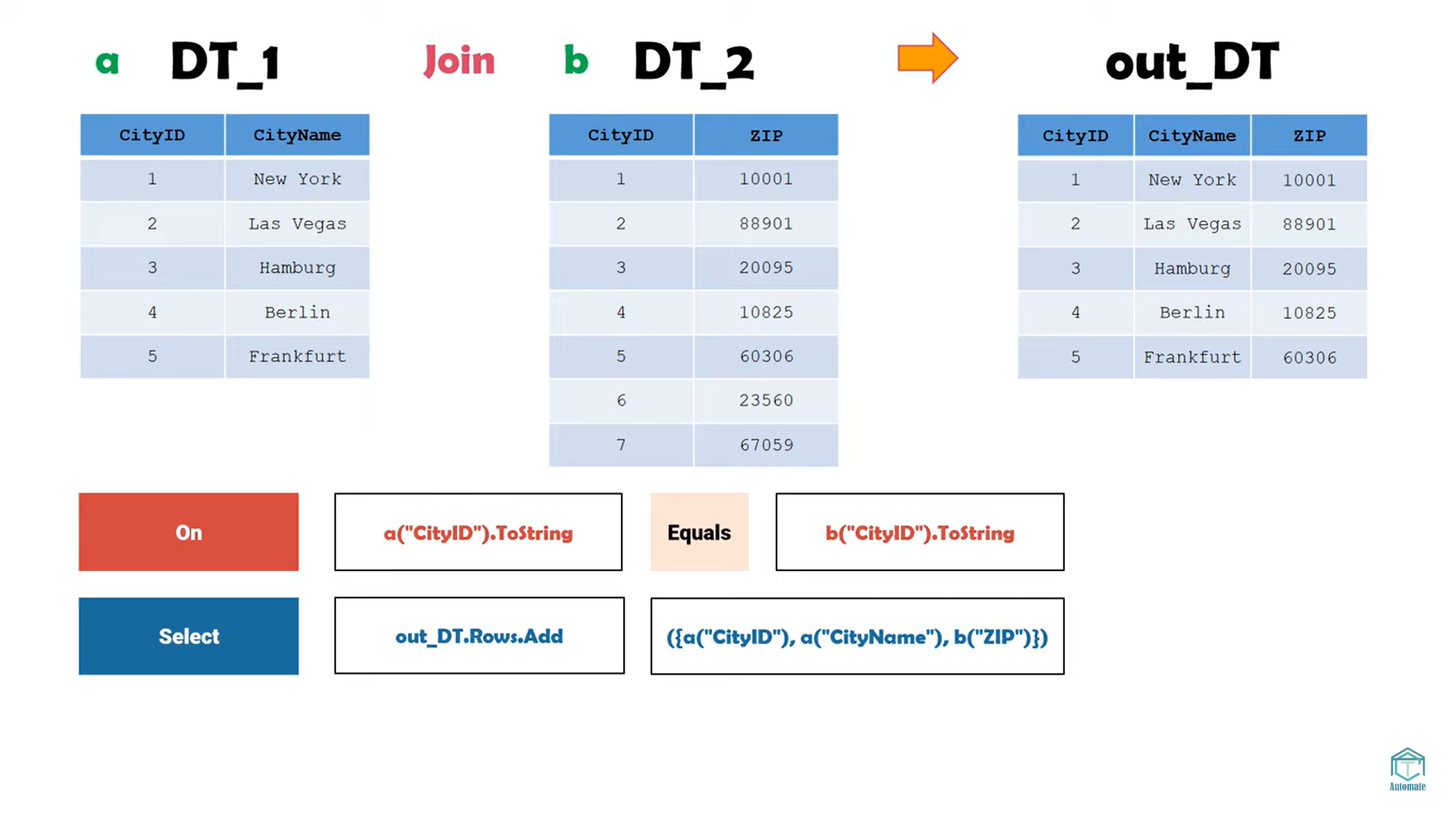
It helps in merging 2 data tables together.

We have 2 types:

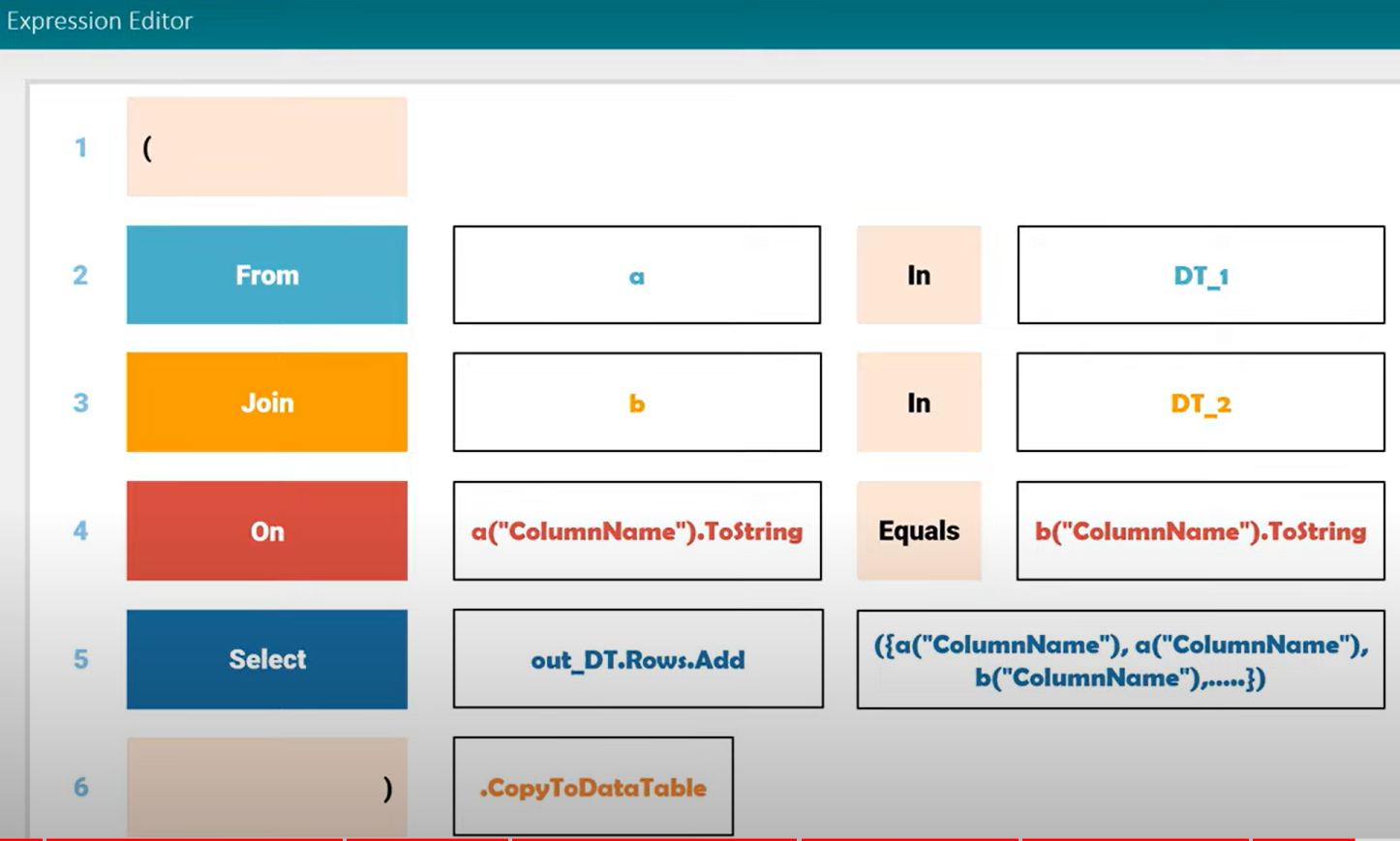
1. **Inner Join** (We get only those which are part of both sets)
2. **Outer Join**: It has 3 types: Left outer (full a), Right outer (full b) & Full outer (**a or b**)



Let’s write the script for JOIN Inner for the below requirement



**Script sample:**



(

From a In dtInput1

Join b In dtInput2

On a("Col1").ToString Equals b("Col1").ToString

Select dtOutput.Rows.Add({a("Col1"),a("Col2"),b("Col3")})

).CopyToDataTable

**The reason why we call it inner is, we are getting common col from both as well as merging other 2 cols each from 1 data table.**

**17:30 mins**

[**https://www.youtube.com/watch?v=QGjlSeEVmws&list=PL8Przw6Rdrj553wll-n9veqGS-Y9WtlaT&index=2**](https://www.youtube.com/watch?v=QGjlSeEVmws&list=PL8Przw6Rdrj553wll-n9veqGS-Y9WtlaT&index=2)

**Exercise: Compare 2 data tables having 3 columns. Output the rows having different values in both data tables**

# Errors:

(From p in DT.Select() where( From q in DT.Select() where q(“Column4”).Equals(p(“Column4”)) and q(“Column5”).Equals(p(“Column5”)) Select q).ToArray.Count>1 Select p).ToArray.CopyToDataTable()

(From d in dtData.AsEnumerable  
Let sk = String.Join(“\_”,{“UID”,“Name”,“Gender”,“Age”}.Select(Function (x) d(x).toString.Trim))  
Group d by k=sk into grp = Group  
Where grp.Count > 1  
From g in grp  
Order by dtData.Rows.IndexOf(g)  
Select r=g).CopyToDataTable

(From p in dtData.Select() where( From q in dtData.Select() where string.Join(",",q.ItemArray).Equals(string.Join(",",p.ItemArray)) Select q).ToArray.Count>1 Select p).ToArray.CopyToDataTable()

dtOutput.Rows.Add({row("col1"),row("col2"),row("col3")})