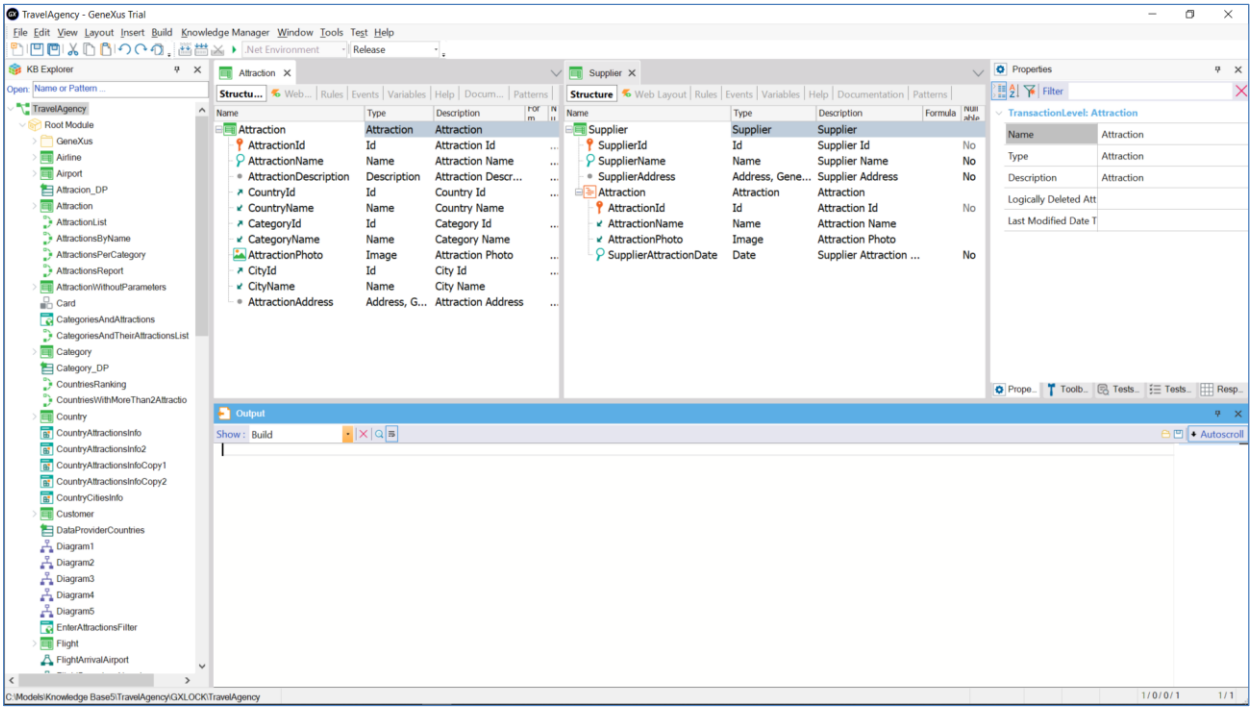

GeneXus™

Some Considerations on Loading Grids Without a Base Table

Diego Marranghello | GeneXus Training

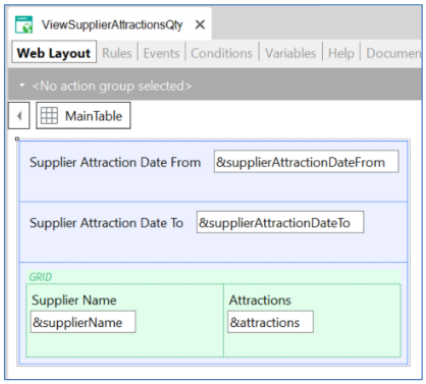
GeneXus™

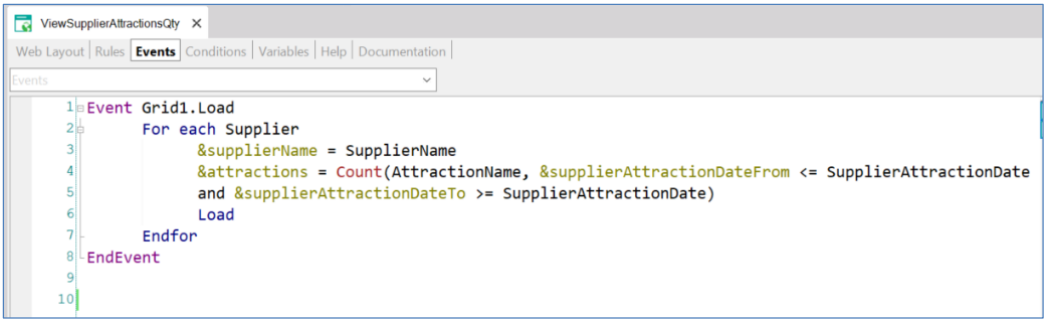


To implement the different examples that we will see in this video, we will use the application for a travel agency that we have used in previous videos.

Note that in our application we have, among others, the Attraction transaction and the Supplier transaction. And the latter has as a second level these attributes of the Attraction transaction, plus a new attribute, SupplierAttractionDate, which will be used to record the date on which the supplier offers the attraction.

Therefore, each supplier will have N attractions.





Suppose we have the following Web Panel.

In the fixed part, it has two variables of the Date type, SupplierAttractionDateFrom and SupplierAttractionDateTo.
We also have a Grid, with two variables inside it: SupplierName and attractions. The first one is based on the SupplierName attribute belonging to the Supplier table, and the other is of the numeric type.

We want to implement the following in this Web Panel:
That the user can select two dates, from and to. And that all the names of the suppliers (Supplier) are shown on the screen, with their corresponding number of attractions. Note that for each supplier we are only interested in counting its attractions whose date is between those entered by the user.

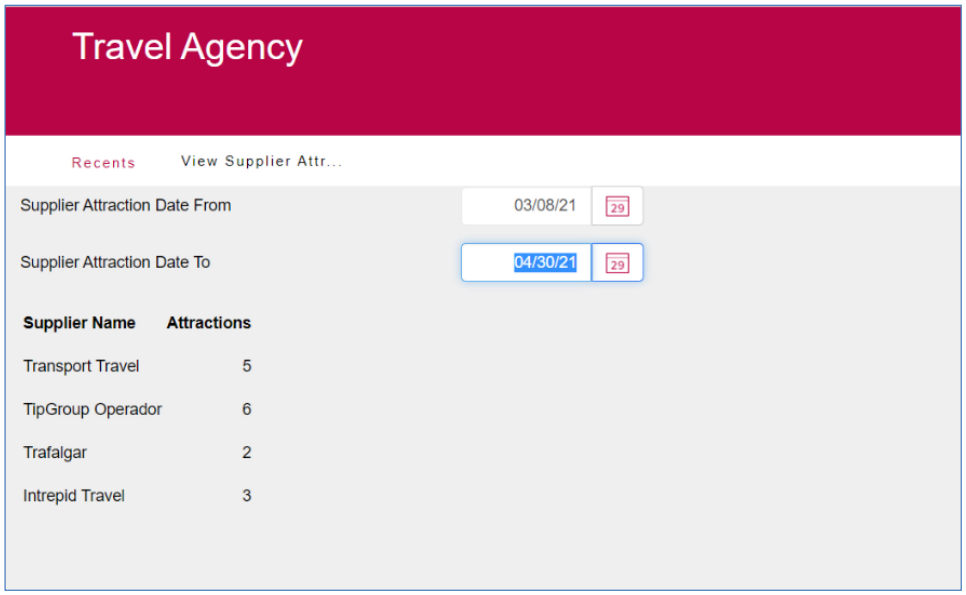
Let's see how we implement it in the Events section.
In the Load event of the grid, we have a For each that will run through the Supplier table.
For each record found, the &SupplierName variable will be assigned the value of the SupplierName attribute.
And for the attractions variable, an inline formula has been defined, which will count all the attractions of the supplier, with the condition that the date of the supplier attraction (SupplierAttractionDate) in which we are located must be within the date range entered by the user.

This is a web panel without a base table because there aren't any attributes declared in the Form nor in the grid properties. Also, in the

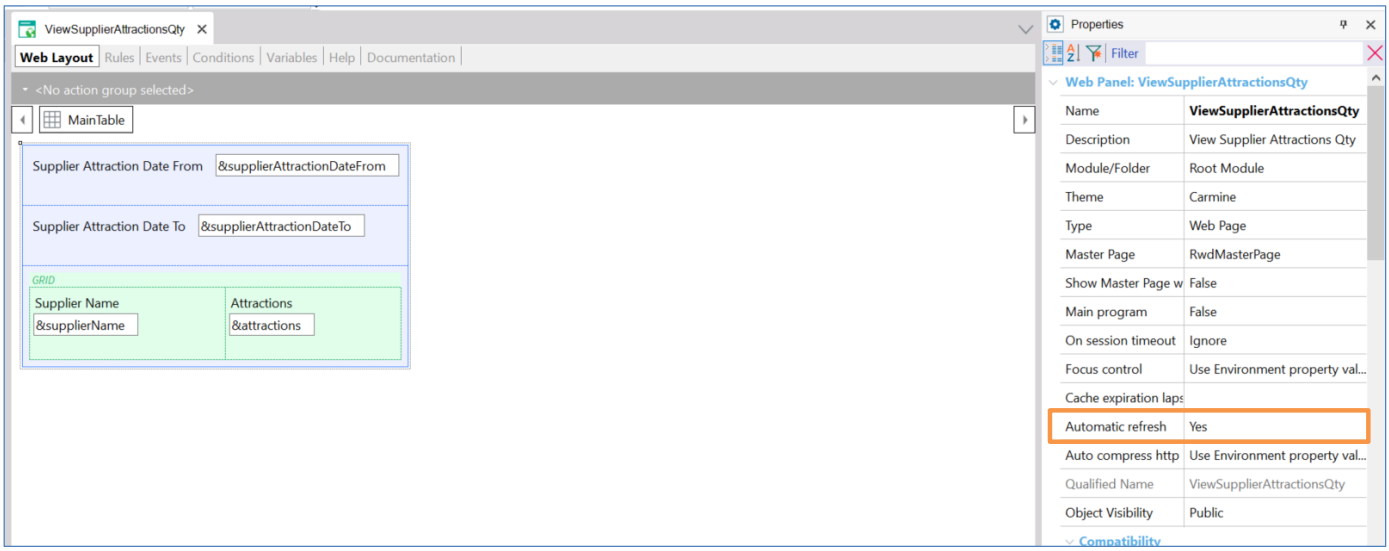
events there are no attributes outside of a context. This one we see here is inside the For each, so, in this case, if we didn't have this transaction entered, it would help to determine the table that the For each will run through, but it would have no influence on the determination of the grid's base table. And these other attributes are also within a context, that of the Count formula. They will only help determine the table to be navigated by this formula.

This is why we must enter the Load command to load each record found in the grid.
As it is programmed, we could think that it is necessary to add a button in our panel, so that once the dates are entered, we click on it to update our grid.

At runtime

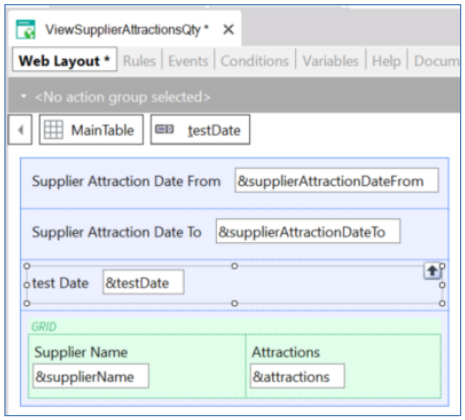


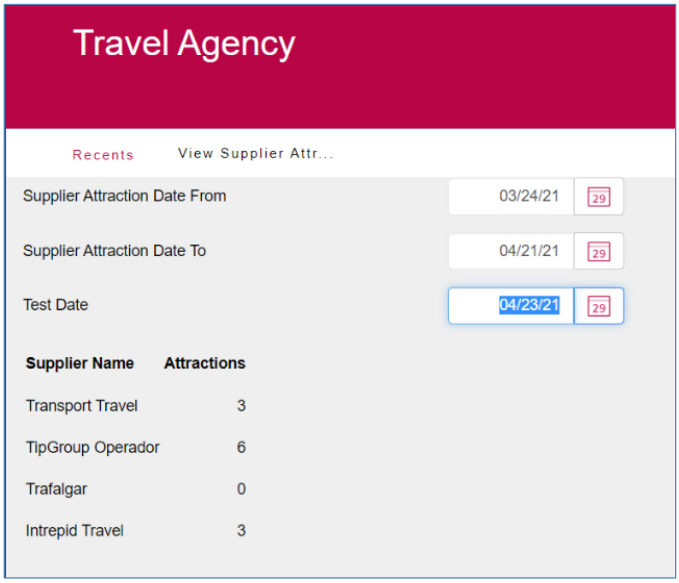
Let's try running it as it is now.
We see that the list of all suppliers is loaded, and with 0 number of attractions, because we still haven't entered the dates that condition the formula that will count them.
We enter two dates, from and to, which as we have just seen will condition the counting of attractions of each supplier. The number of attractions for each supplier conditioned by the dates entered will be automatically updated.
It was not necessary to tell it to update the grid when the dates changed, it was done automatically. Why did it have this behavior?



If we go to the properties of our web panel, we see one called Automatic refresh, which by default is set to Yes. What this property does is to update the grid when it detects changes in a variable defined in the web layout and that is then used: in the Refresh event, the Load event of the grid, or in the conditions of the grid or the web panel.

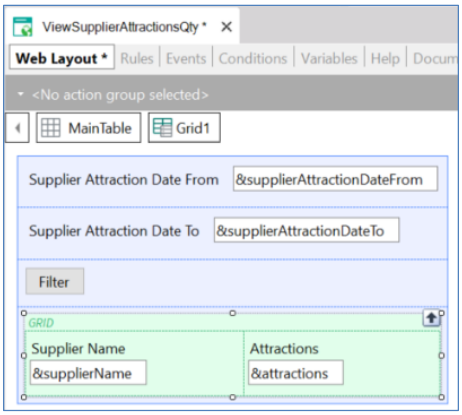
This is the case here, the variables SupplierAttractionDateFrom and SupplierAttractinDateTo, entered to filter the number of attractions to be shown. They are then defined inside the Load event of the grid, more precisely inside this Count formula. Since these variables are inside this event, it is understood that they have an impact on the data we want to show, so every time one of them is modified, our grid is updated; that is, the Refresh event of the grid is triggered.



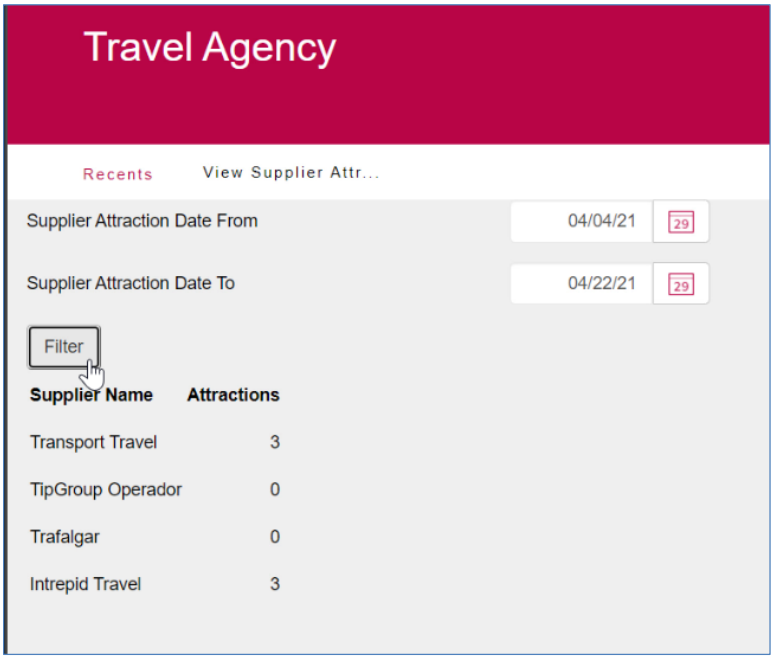


Let's see what happens if we add a new variable to the fixed part of our web panel, but then it is not defined in a Load event, nor in a Refresh event nor or in any conditions.

We can see that when modifying the value of this variable the refresh event of the grid is not triggered; there is no attempt to update the grid, since as we said, the Automatic refresh property doesn't apply to these cases.




```
Event 'Filter'
      Grid1.Refresh()
Endevent
```

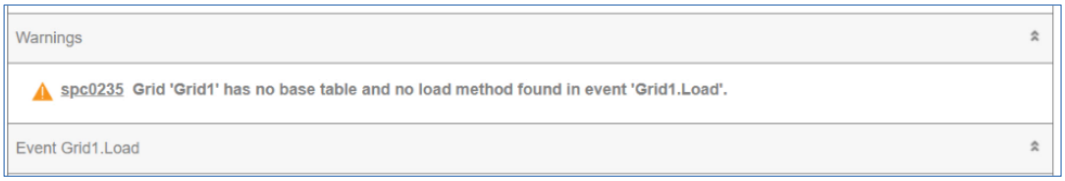


Let's remove this variable.

Now let's suppose that we don't want the behavior provided by the Automatic refresh property, because we want the grid to be updated whenever we want, and not every time one of these variables is modified.

The option would be to leave the Automatic refresh property set to No, and add a button in the fixed part of our panel to control when to trigger the grid refresh.

In the button event we simply define the refresh method of our grid.
We can see that now by changing the values of these variables the grid is no longer updated. Clicking on the button triggers the refresh event of the grid, and then the grid will be updated.



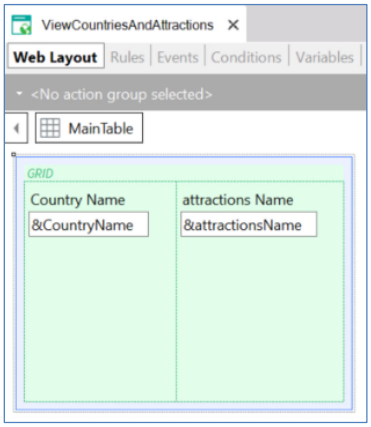
Let's remove the button alongside its event, and leave this property set to Yes again.

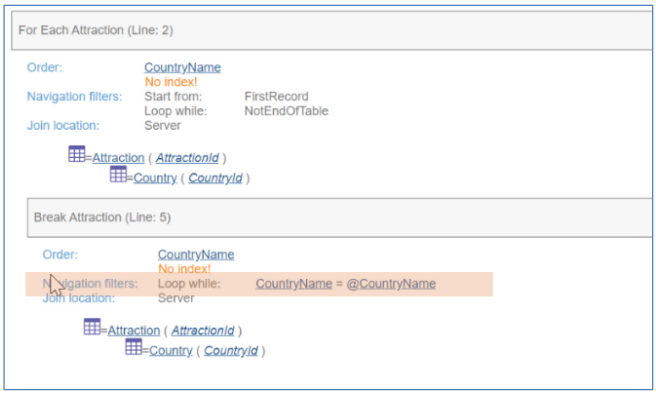
Now suppose we have forgotten to declare this Load command within the event. What will happen? How will it behave?
Let's save and implement this change.

Let's look at the navigation list.

A Warning is displayed, indicating that our grid doesn't have a base table and we don't have any Load command declared within the Load event. This Warning also appears in the output.

When testing it at runtime, we see that no records are loaded in the grid, because when the Load command is omitted, it is assumed that we are not interested in loading any records. Since this implementation doesn't make much sense, a warning is triggered.





Now suppose that, in this application, we are interested in having a Web Panel that shows all the countries that have at least one attraction, and then all the attractions, side by side, separated by a bar.

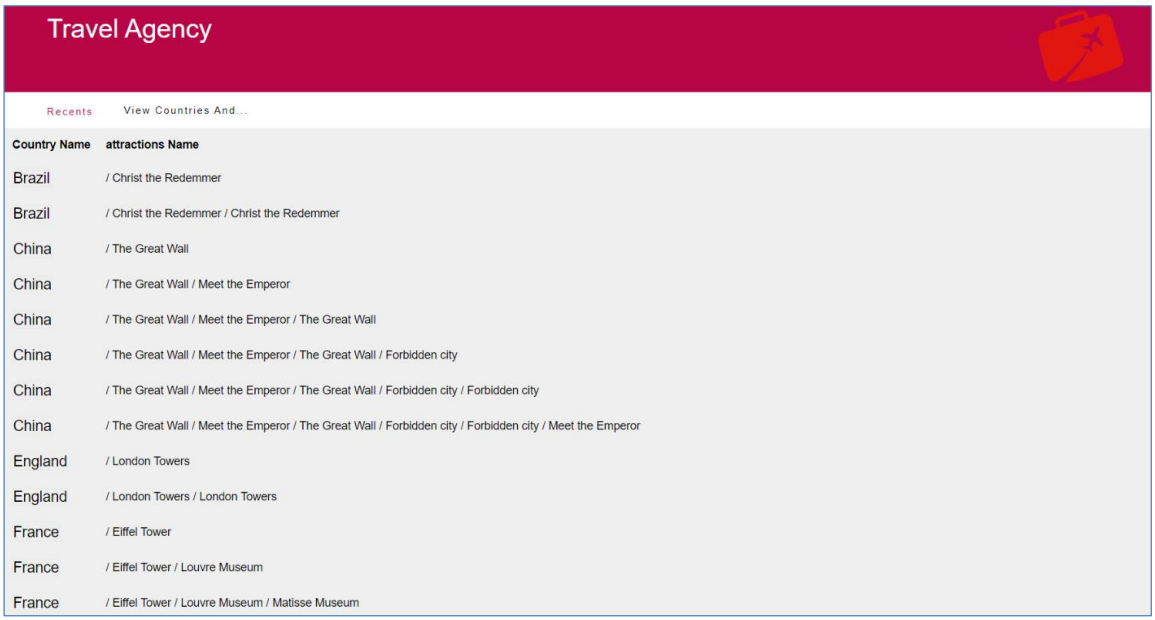
Remember that our Attraction transaction has the CountryId and CountryName attribute of the Country transaction. So each attraction will belong to a country.
For this in the Web Layout we have a grid with two variables, one based on the CountryName attribute that will show the name of the countries, and the other of the LongVarChar type that will show all the attractions.

In the events section we have defined the Load event of the grid, and inside two nested For each commands.
The external one will run through the attractions table sorting the records by country name. Also, it will assign the value of the CountryName attribute to the variable of that name.
Then we have a nested For each that will navigate the same table, Attraction. With this definition, all the attractions found will be concatenated in a single variable, attractionName.

As we know, since the internal For each runs through the same table as the external one, and an order has been defined in this one, a control break will be made. The break criteria will be determined precisely by the attribute declared in the order clause of the external For each; that is to say, the grouping will be made by CountryName.

It can clearly be seen in the navigation list.

At runtime

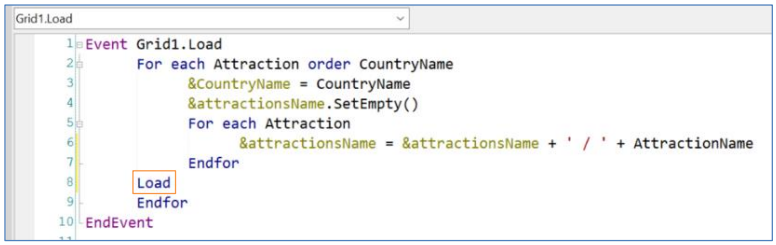


Let's try it.

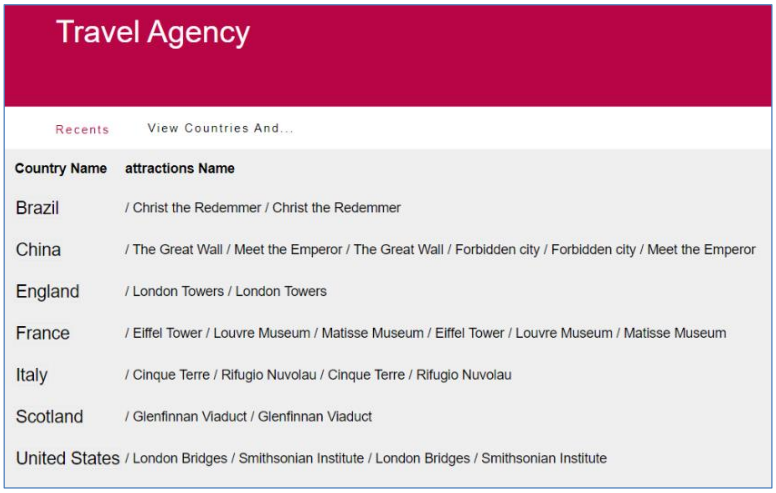
We are not getting the desired result.

For example, China is being printed once next to its first attraction. Then it is printed again with its first two concatenated attractions, and it is printed again with its first three concatenated attractions, and so on. The same happens with all the countries.

What's going on? What was the error in our definition?



```
Grid1.Load
Event Grid1.Load
    For each Attraction order CountryName
        &CountryName = CountryName
        &attractionsName.SetEmpty()
        For each Attraction
            &attractionsName = &attractionsName + ' / ' + AttractionName
        Endfor
    Load
    Endfor
EndEvent
```



Travel Agency

Recents View Countries And...

Country Name	attractions Name
Brazil	/ Christ the Redemmer / Christ the Redemmer
China	/ The Great Wall / Meet the Emperor / The Great Wall / Forbidden city / Forbidden city / Meet the Emperor
England	/ London Towers / London Towers
France	/ Eiffel Tower / Louvre Museum / Matisse Museum / Eiffel Tower / Louvre Museum / Matisse Museum
Italy	/ Cinque Terre / Rifugio Nuvolau / Cinque Terre / Rifugio Nuvolau
Scotland	/ Glenfinnan Viaduct / Glenfinnan Viaduct
United States	/ London Bridges / Smithsonian Institute / London Bridges / Smithsonian Institute

The error here is where we declared the Load command. We did it inside the nested For each command. So every time it finds an attraction and concatenates it with our variable, it prints it on the screen, but this is not what we want. We want this information to be displayed once all the attractions of that country are concatenated.
So we have to define the Load command outside of this For each command.

Let's do it.
We run it again.

We can see that now it is working as expected.
To learn more about the topics covered in this video, we encourage you to visit our Wiki.