# Logic for querying the database with GeneXus

For each command: Unique clause

GeneXus



We will discuss in some detail the Unique clause applied to the For each command, knowing that it can also be used in groups of data providers and grids with a base table.

# For each *BaseTrn*<sub>1</sub>, ..., *BaseTrn*<sub>n</sub>

skip expression1 count expression2
order att1, att2, ..., attn [when condition]
order att1, att2, ..., attn [when condition]
order none [when condition]
unique att1, att2, ..., attn
using DataSelector ( parm1, parm2, ..., parmn)
where condition [when condition]
where att IN DataSelector ( parm1, parm2, ..., parmn)
blocking n
main\_code

#### when duplicate

when\_duplicate\_code

#### when none

when\_none\_code

# endfor

Here we see it among the other clauses of the For each.



unique D, E —

$\sim$	>



When the value of a set of attributes is repeated for many records—in this example, D and E—we can use the unique clause to work with one of all the records whose value is repeated (as if it represented the group); for example, printing the values of those attributes (since they will be the same for all the records of the group), and then moving on to the next group, to do the same. And so on until using all of them.



For this to make sense, only attributes whose value is unique for each and every record in the group can appear in the code to be executed for each group.

The attributes included there do not have to be part of the same table. They can be in the extended one.



Let's suppose that this is a graphical representation of the extended table and not a physical table.

For example, suppose that E is a foreign key that determines H and I. That is, this physical table exists.

This means that for all the records in the group, since the values of E are the same, the values of H and I will also be the same.



This means that we can also use the H and I attributes within the code that will be executed for the group, since they will also be unique to that set of records.





for each Trip.Attraction
 order AttractionId
 for each Trip.Attraction
 endfor
 print AttractionInfo //AttractionName
endfor

for each Trip.Attraction
 unique AttractionId
 print AttractionInfo //AttractionName
endfor

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
3	1	200
1	3	120
3	6	180
4	6	240
2	10	120
2	12	90

AttractionId*	AttractionName	
1	Louvre Museum	
2	The Great Wall	
3	Eiffel Tower	
4	Christ the Redeemer	
5	Smithsonian Institute	
6	Matisse Museum	
7	Forbidden city	
10	Rifugio Nuvolau	
12	Cinque Terre	

Let's see examples.

We have the trips that can be made to different tourist attractions with an assigned amount of time for each visit.

We want to obtain a list of the tourist attractions included in these trips. Let's suppose that this is the current data from the tables. We will want to list only these attractions.

A first option could be to implement a control break that navigates TripAttraction and groups by AttractionId.

In this way, we can be sure that we are only listing attractions actually included in trips. Also, by placing the print command after the nested For each (although placing it before would have been the same), we know that for each group we will be printing only the AttractionName that is repeated.

This can be solved in a much simpler way using the Unique clause. It's the most obvious use case.

TRACTION1
FirstRecord NotEndOfTable

AttractionId\*

TripId\*

EEAttraction ( AttractionId ) INTO AttractionName

TripAttractionVisitTime

for each Trip.Attraction
 order AttractionId
 for each Trip.Attraction
 endfor
 print AttractionInfo //AttractionName
endfor

for each Trip.Attraction
 unique AttractionId
 print AttractionInfo //AttractionName
endfor

AttractionId*	AttractionName	
1	Louvre Museum	
2	The Great Wall	
3	Eiffel Tower	
4	Christ the Redeemer	
5	Smithsonian Institute	
6	Matisse Museum	
7	Forbidden city	
10	Rifugio Nuvolau	
12	Cinque Terre	

In the navigation list of the For each with Unique, we see that since it has an index by AttractionId in TripAttraction (because it is a foreign key) it will choose to sort by that attribute.

Therefore, it starts in the first group where the value of AttractionId is repeated, and its AttractionName (which is unique for all the records of the group) is printed in the output: Louvre Museum.

				Attraction
For Each TripAttraction (Line: 23)			ctionId	
			Attraction	Trip.Attraction
Order:	AttractionId Index: ITRIPA	TTRACTION1	Louvre Museum	tractionInfo //AttractionName
Unique: Navigation filters:	AttractionId Start from:	FirstRecor	Eiffel Tower	
Join location:	Server	Notendon	Matisse Museum	Concerning Address the second
		Rifugio Nuvolau	for each Trip.Attraction unique AttractionId print AttractionInfo //AttractionName	
			Cinque Terre	endfor

Tripld*	AttractionId*	TripAttractionVisitTime
1	1	180
3	1	200
1	3	120
3	6	180
4	6	240
2	10	120
2	12	90

AttractionId*	AttractionName	
1	Louvre Museum	
2	The Great Wall	
3	Eiffel Tower	
4	Christ the Redeemer	
5	Smithsonian Institute	
6	Matisse Museum	
7	Forbidden city	
10	Rifugio Nuvolau	
12	Cinque Terre	

The next group is a single record: Eiffel Tower is printed.

Then the attraction with ID 6, which is Matisse Museum. Then 10, and finally 12.



TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
3	1	200
1	3	120
3	6	180
4	6	240
2	10	120
2	12	90

AttractionId*	AttractionName	
1	Louvre Museum	
2	The Great Wall	
3	Eiffel Tower	
4	Christ the Redeemer	
5	Smithsonian Institute	
6	Matisse Museum	
7	Forbidden city	
10	Rifugio Nuvolau	
12	Cinque Terre	

If we wanted these attractions to be sorted by attraction name...

Trip		Attraction					
For Each TripAttrac	ction (Line: 23)		Attract	ion			
Order:	AttractionName		Cinque	Terre for	each Trip.Attrac	tion	
Unique: Navigation filters:	AttractionId Start from:	FirstRecord	Eiffel To	wer	order AttractionN	ame Td	
Join location:	Loop while: Server	NotEndOfTable	Louvre	Museum	print Attracti	onInfo //A	ttraction
= <u>TripA</u>	ttraction ( <u>TripId</u> , <u>Att</u>	tractionId ) INTO AttractionId	Matisse	Museum			
		tionio ) INTO <u>Attractioniname</u>	Rifugio	Nuvolau			
				AttractionId*	AttractionNa	me	
Tripld*	AttractionId*	TripAttractionVisitTime		1	Louvre Museu	um	
1	1	180		2	The Great Wa	all	
3	1	200					

3	1	200
1	3	120
3	6	180
4	6	240
2	10	120
2	12	90

AttractionId*	AttractionName	
1	Louvre Museum	
2	The Great Wall	
3	Eiffel Tower	
4	Christ the Redeemer	
5	Smithsonian Institute	
6	Matisse Museum	
7	Forbidden city	
10	Rifugio Nuvolau	
12	Cinque Terre	

...we could add the order clause.

And so we see the navigation list. The single value being sought is still AttractionId but the result of the query will be ordered by AttractionName.



Could we instead use the AttractionName attribute directly in the unique clause?

Yes, but note two things. <u>On one hand</u>, doing this does not ensure that the list is also sorted by AttractionName. Note that the navigation list shows Order NONE. GeneXus doesn't know about the existence of an index by AttractionName. So we should still order by AttractionName if that is what we want.

<u>On the other hand</u>, let's think about what would happen if we had two attractions with the same name in the database.



for each Trip.Attraction
order AttractionName
unique AttractionId
<pre>print AttractionInfo //AttractionName</pre>
endfor
for each Trip.Attraction
order AttractionName
unique AttractionName
<pre>print AttractionInfo //AttractionName</pre>

endfor

d*	AttractionId*	TripAttractionVisitTime
	1	180
	1	200
	3	120
	6	180
	6	240
	8	60
	10	120
	12	90

AttractionId*	AttractionName	
1	Louvre Museum	
2	The Great Wall	
3	Eiffel Tower	
4	Christ the Redeemer	
5	Smithsonian Institute	
6	Matisse Museum	
7	Forbidden city	
8	Louvre Museum	
10	Rifugio Nuvolau	
12	Cinque Terre	

For example, 1 and 8. If we don't have a unique index by AttractionName this will be allowed. And note that we have attraction 1 in 2 trips, and 8 in 1.

The only difference between this For each and this other one is the unique clause.

In the first case, these two records will be handled together, and Louvre Museum will be listed; and this one will be handled separately, in another group, and Louvre Museum will be listed again.

In the second case, however, the three records will be in the same group, and Louvre Museum will be listed only once, even if it corresponds to two different attractions.



#### endfor

				AttractionId*	AttractionName	
TripId*	AttractionId*	TripAttractionVisitTime	AttractionName	1	Louvre Museum	
1	1	180	Louvre Museum	2	The Great Wall	
3	1	200	Louvre Museum	3	Eiffel Tower	
1	3	120	Eiffel Tower	4	Christ the Redeemer	
3	6	180	Matisse Museum	5	Smithsonian Institute	
4	6	240	Matisse Museum	6	Matisse Museum	
5	8	60	Louvre Museum	7	Forbidden city	
2	10	120	Rifugio Nuvolau	8	Louvre Museum	
2	12	90	Cinque Terre	10	Rifugio Nuvolau	
				12	Cinque Terre	

We can see it very clearly if we imagine the data like this, with the extended table as a super table.



#### endfor

				AttractionId*	AttractionName	
TripId*	AttractionId*	TripAttractionVisitTime	AttractionName	1	Louvre Museum	
2	12	90	Cinque Terre	2	The Great Wall	
1	3	120	Eiffel Tower	3	Eiffel Tower	
1	1	180	Louvre Museum	4	Christ the Redeemer	
3	1	200	Louvre Museum	5	Smithsonian Institute	
5	8	60	Louvre Museum	6	Matisse Museum	
3	6	180	Matisse Museum	7	Forbidden city	
4	6	240	Matisse Museum	8	Louvre Museum	
2	10	120	Rifugio Nuvolau	10	Rifugio Nuvolau	
				12	Cinque Terre	

If we order it by AttractionName we see it more clearly... Note that there is no problem with the attribute of the unique clause being in the extended table and not in the base table.

In short, it will not be the same to ask for unique values for AttractionId and for AttractionName.

If the unique index exists, then the result of both For each commands will be exactly the same, because this record 8 cannot possibly exist.



In addition, we can not only keep one of the repeated records to do something with the information that doesn't vary (like printing it), but we can also run aggregation formulas on the repeated ones; for example, to count them. Of course, the formula must navigate the same table.

So, we take the first group and run the count on its records (it will give 3 in this case). And it prints D and E—information that is unique to that group—and 3. Then the next group, for which the count will give 2.

Then the third one, for which the count will give 1. And finally the fourth one, for which the count will give 3.



Attraction AttractionId 9 AttractionName 者 CountryId ✔ CountryName CategoryId ✔ CategoryName

AttractionPhoto

CityId 🖌 CityName for each Trip.Attraction unique AttractionId &qty = Count(TripAttractionVisitTime) print AttractionInfo //AttractionName, &qty endfor

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
3	1	200
1	3	120
3	6	180
4	6	240
5	8	60
2	10	120
2	12	90

AttractionId*	AttractionName	
1	Louvre Museum	
2	The Great Wall	
3	Eiffel Tower	
4	Christ the Redeemer	
5	Smithsonian Institute	
6	Matisse Museum	
7	Forbidden city	
8	Louvre Museum	
10	Rifugio Nuvolau	
12	Cinque Terre	

In this example, besides keeping the non-repeated AttractionId to list its name, we want to count how many times it is repeated.

In short, the number of trips where it is present.



The Count formula is using the secondary attribute of the table you want to navigate, so when this is determined, the Count formula will have a special behavior: it will group by the unique attribute, as we see in the navigation list.

TripDescription     TripDescription     Attraction     Y AttractionId     AttractionNan	- ŷ Attr - ¾ Cou - ¥ Cou - ¾ Cat	actionName intryId intryName egoryId e egoryName	unique AttractionId &qty = Count(TripAttractionVisitTime) print AttractionInfo //AttractionName, &qty endfor
	VisitTime Attr	actionPhoto	For Each TripAttraction (Line: 23)
Build Comparer     Editors     Help     KB Explorer     Knowledge Base     Output     Patterns     Search     Table & Transaction Diagrams	Image: Second Specification	False False True	Order:     AttractionId Index: ITRIPATTRACTION1       Unique:     AttractionId Navigation filters:       Start from:     FirstRecord Loop while:       Join location:     Server
> Team Development Themes Trace Wiki	Build With This Only Call tree for Build Concurrent Generation Concurrent Generation Instances Concurrent Specification Instances Detailed Navigation	Check Full True 4 5 True	
< >>	Detailed Navigation Show detailed navigation	OK Cancel	Given: <u>AttractionId</u> Index: ITRIPATTRACTION Group by: <u>AttractionId</u>

Remember that for the list to show the navigation of the formula we must activate the detailed navigation... through Tools/Options....

<u>ا</u>	Attrac	tion	Trips	for each Trip.Attraction	
	Louvre	Louvre Museum 2		<pre>unique AttractionId    &amp;qty = Count(TripAttractionVisitTime)</pre>	
	Eiffel T	ower	1	<pre>print AttractionInfo //AttractionName, &amp;qty endfor</pre>	
	Matisse	Museum	2		
	Louvre	Museum	1	For Each TripAttraction (Line: 23)	
	Rifugio	Nuvolau	1	Order: <u>AttractionId</u>	
	Cinque	Terre	1	Unique: <u>AttractionId</u> Navigation filters: Start from: FirstRecord Loop while: NotEndOfTable	
	TripId*	AttractionId*	TripAttractionVisitTime	Join location: Server	
	Tripld* 1	AttractionId* 1	TripAttractionVisitTime 180	Join location: Server = <u>TripAttraction</u> ( <i>TripId, AttractionId</i> ) INTO <u>AttractionId</u>	
	Tripld* 1 3	AttractionId* 1 1	TripAttractionVisitTime 180 200	Join location: Server Server Join location: <u>Server</u> <u>Server</u> <u>Server</u> <u>INTO AttractionId</u> <u>INTO AttractionId</u> <u>INTO AttractionId</u>	
	Tripld* 1 3 1	AttractionId* 1 1 3	TripAttractionVisitTime 180 200 120	Join location: Server ITIPAttraction ( <i>TripId, AttractionId</i> ) INTO <u>AttractionId</u> Attraction ( <i>AttractionId</i> ) INTO <u>AttractionName</u> ITIPAttraction( <i>TripAttractionVisitTime</i> ) navigation ( <i>AttractionId</i> )	
	TripId* 1 3 1 3 3 3	AttractionId* 1 1 3 6	TripAttractionVisitTime           180           200           120           180	Join location: Server           Image: Server           Image: IripAttraction ( TripId, AttractionId ) INTO AttractionId           Image: IripAttraction ( AttractionId ) INTO AttractionName           Image: IripAttraction ( AttractionId ) INTO AttractionName           Image: IripAttraction ( TripAttractionVisitTime ) navigation ( AttractionId )           Formulas	
	Tripld* 1 3 1 3 4	AttractionId* 1 1 3 6 6 6	TripAttractionVisitTime           180           200           120           180           240	Join location: Server ITipAttraction ( <i>TripId, AttractionId</i> ) INTO AttractionId H=Attraction ( <i>AttractionId</i> ) INTO AttractionName H=count( <i>TripAttractionVisitTime</i> ) navigation ( <i>AttractionId</i> ) Formulas	
	Tripld* 1 3 1 3 4 5	AttractionId* 1 1 3 6 6 8	TripAttractionVisitTime           180           200           120           180           240           60	Join location: Server Impediate Server Impedia	
	Tripld* 1 3 1 3 4 5 2	AttractionId* 1 1 3 6 6 6 8 10	TripAttractionVisitTime           180           200           120           180           240           60           120	Join location: Server  Image: Server Image: Server Image: Server  Image: Server Image: Server  Image: Server  Image: Server Image: Server  Image: Server Ima	
	Tripld*           1           3           1           3           4           5           2           2           2	AttractionId* 1 1 3 6 6 6 8 10 12	TripAttractionVisitTime           180           200           120           180           240           60           120           90	Join location: Server  Image: Server Image: Server Image: Server Image: Server  Image: Server	

E.

Then, for each group of repeated ones, it will count the records for that given AttractionId, that of each group. Thus, the first group with repeated AttractionId is obtained, its records are counted—those with the same AttractionId—and the attraction name and that number are printed in the output.

Attraction	Trips
Louvre Museum	2
Eiffel Tower	1
Matisse Museum	2
Louvre Museum	1
Rifugio Nuvolau	1
Cinque Terre	1

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
3	1	200
1	3	120
3	6	180
4	6	240
5	8	60
2	10	120
2	12	90



endfor

AttractionId*	AttractionName	
1	Louvre Museum	
2	The Great Wall	
3	Eiffel Tower	
4	Christ the Redeemer	
5	Smithsonian Institute	
6	Matisse Museum	
7	Forbidden city	
8	Louvre Museum	
10	Rifugio Nuvolau	
12	Cinque Terre	

Then the next group, for which the count is 1.

Attraction	Trips
Louvre Museum	2
Eiffel Tower	1
Matisse Museum	2
Louvre Museum	1
Rifugio Nuvolau	1
Cinque Terre	1

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
3	1	200
1	3	120
3	6	180
4	6	240
5	8	60
2	10	120
2	12	90

# for each Trip.Attraction unique AttractionId &qty = Count(TripAttractionVisitTime) print AttractionInfo //AttractionName, &qty

endfor

AttractionId*	AttractionName	
1	Louvre Museum	
2	The Great Wall	
3	Eiffel Tower	
4	Christ the Redeemer	
5	Smithsonian Institute	
6	Matisse Museum	
7	Forbidden city	
8	Louvre Museum	
10	Rifugio Nuvolau	
12	Cinque Terre	

Then the next one, which gives 2.

Attraction	Trips	
Louvre Museum	2	
Eiffel Tower	1	
Matisse Museum	2	
Louvre Museum	1	
Rifugio Nuvolau	1	
Cinque Terre	1	

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
3	1	200
1	3	120
3	6	180
4	6	240
5	8	60
2	10	120
2	12	90

# for each Trip.Attraction unique AttractionId &qty = Count(TripAttractionVisitTime) print AttractionInfo //AttractionName, &qty

endfor

AttractionId*	AttractionName	
1	Louvre Museum	
2	The Great Wall	
3	Eiffel Tower	
4	Christ the Redeemer	
5	Smithsonian Institute	
6	Matisse Museum	
7	Forbidden city	
8	Louvre Museum	
10	Rifugio Nuvolau	
12	Cinque Terre	

The next one, with the same name as the first one, gives 1.

Attraction	Trips
Louvre Museum	2
Eiffel Tower	1
Matisse Museum	2
Louvre Museum	1
Rifugio Nuvolau	1
Cinque Terre	1

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
3	1	200
1	3	120
3	6	180
4	6	240
5	8	60
2	10	120
2	12	90



AttractionId*	AttractionName	
1	Louvre Museum	
2	The Great Wall	
3	Eiffel Tower	
4	Christ the Redeemer	
5	Smithsonian Institute	
6	Matisse Museum	
7	Forbidden city	
8	Louvre Museum	
10	Rifugio Nuvolau	
12	Cinque Terre	

The next one also gives 1, and the last one too.

On the other hand, if instead of AttractionId...

Attraction     Trips       Cinque Terre     1       Eiffel Tower     1       Louvre Museum     3       Matisse Museum     2       Rifugio Nuvolau     1       TripId*     AttractionId*       1     180
Cinque Terre       1         Eiffel Tower       1         Louvre Museum       3         Matisse Museum       2         Rifugio Nuvolau       1         TripId*       AttractionId*       TripAttractionVisitTime         1       1       180
Eiffel Tower     1       Louvre Museum     3       Matisse Museum     2       Rifugio Nuvolau     1       TripId*     AttractionId*       1     1       1     180
Louvre Museum     3       Matisse Museum     2       Rifugio Nuvolau     1       TripId*     AttractionId*     TripAttractionVisitTime       1     1     180
Matisse Museum     2       Rifugio Nuvolau     1       TripId*     AttractionId*     TripAttractionVisitTime       1     1     180
Rifugio Nuvolau     1       TripId*     AttractionId*     TripAttractionVisitTime       1     1     180
TripId*         AttractionId*         TripAttractionVisitTime           1         1         180
1 1 180
3 1 200
1 3 120
3 6 180
4 6 240
5 8 60
2 10 120



AttractionId*	AttractionName	
1	Louvre Museum	
2	The Great Wall	
3	Eiffel Tower	
4	Christ the Redeemer	
5	Smithsonian Institute	
6	Matisse Museum	
7	Forbidden city	
8	Louvre Museum	
10	Rifugio Nuvolau	
12	Cinque Terre	

...we use AttractionName in the unique clause, the group corresponding to Louvre Museum will count 3 records.

Attraction			Trips	
	Attraction			
	Cinque	e Terre	1	
Eiffel Tower		lower	1	
	Louvre	Museum	3	
	Matisse Museum		2	
	Rifugio Nuvolau		1	
Trip	ld*	AttractionId*	TripAttractionVisitTime	
1	L	1	180	
3				
	3	1	200	
1	s L	1 3	200 120	
1	3 L 3	1 3 6	200 120 180	
1	s L S	1 3 6 6	200 120 180 240	
1 3 2 5	3 L 3 L	1 3 6 6 8	200 120 180 240 60	
	3 1 3 4 5 2	1 3 6 8 8 10	200 120 180 240 60 120	

<pre>for each Trip.Attraction     unique AttractionName     &amp;qty = Count(TripAttractionVisitTime)     print AttractionInfo //AttractionName, &amp;qty endfor</pre>				
For Each TripAttraction (Line: 23)				
Order:       NONE         Unique:       AttractionName         Navigation filters:       Start from:       FirstRecord         Loop while:       NotEndOfTable         Join location:       Server         Image: Introduction (Interface)       INTO AttractionId         Image: Introduction (Interface)       INTO AttractionId         Image: Interface       Image: Interface)         Image: Interface       Image: Interface)         Image: Interface       Image: Interface)         Image: Interface       Image: Interface)         Image: Interface       Image: Image				
Formulas				
Navigation to evaluate: count(         TripAttractionVisitTime         )           Given:         AttractionName         )           Index:         ITRIPATTRACTION         )           Group by:         AttractionName         )				
$\blacksquare = \underline{\text{TripAttraction}} \\ \blacksquare = \underline{\text{Attraction}} ( \underline{\text{AttractionId}} )$				

In the navigation, we see Given and Group by.



Attraction AttractionId AttractionName CountryId

- CountryName
- CategoryName
- AttractionPhoto AttractionPhoto
- CityName

for each Trip.Attraction
 unique AttractionName
 &qty = Count(TripAttractionVisitTime)

 $\label{eq:print_attraction_on_state} print \mbox{ AttractionInfo } //\mbox{ AttractionName, &qty endfor}$ 

Let's look at this particular case. If there is no secondary attribute in the table we want to navigate, then we may have to do something so that GeneXus understands that we want to navigate that table for the formula.

Trip TripId TripDate TripDescription Attraction AttractionId AttractionName TripAttractionVisitTime	Attraction AttractionId AttractionName CountryId CountryId CountryName CategoryId CategoryId AttractionPhoto CityId CityId CityName	<pre>for each Trip.Attraction     unique AttractionName     &amp;qty = Count(TripId), AttractionId.IsEmpty() or</pre>
For Each TripAttraction (Line: 23) Order: NONE Unique: AttractionName Navigation filters: Start from: Loop while: Join location: Server Martinet Construction ( <i>Tripld</i> , A Hartaction ( <i>Attrac</i>	FirstRecord NotEndOfTable <u>ttractionId</u> ) INTO <u>AttractionId</u> <u>stionId</u> ) INTO <u>AttractionName</u>	Order: NONE Unique: <u>AttractionName</u> Navigation filters: Start from: FirstRecord Loop while: NotEndOfTable Join location: Server  Impediatraction ( <i>Tripld, AttractionId</i> ) INTO AttractionId Impediatraction ( <i>AttractionId</i> ) INTO AttractionName Impediatraction ( <i>AttractionId</i> ) INTO AttractionName Formulas
HHH=count(_Inpld ) navigation         Formulas         Navigation to evaluate: count( I         Index: ITRIPATTRACTION         ITTIP	ripld )	Navigation to evaluate:       count( <u>TripId</u> )         Where: <u>AttractionId</u> .isempty() or not <u>AttractionId</u> .isempty()         Given: <u>AttractionName</u> Index:       ITRIPATTRACTION         Group by: <u>AttractionName</u> Image:       ITRIPATTRACTION         Group by: <u>AttractionName</u> Image:       ITRIPATTRACTION         Group by: <u>AttractionName</u> Image: <u>Image:</u> Attraction ( <u>AttractionId</u> )

That is to say, if, for example, we place the TripId attribute for the Count and in unique we leave AttractionName, GeneXus may not choose the TripAttraction table to solve the Count formula, but Trip, and the result will not be the desired one.

It is reported that it will navigate the Trip table and count all the trips then, because there is no condition reported for the formula.

It should choose to navigate TripAttraction for it to do what we want. Since we don't have a base transaction for the formulas, we can use a trick: add a condition that is always true and contains an attribute that causes it to determine the base table we want.

For example, this condition that uses AttractionId and that will always be true. Note the navigation list indicating what we want.

Now it is navigating TripAttraction and also grouping by the AttractionName given in the For each; therefore, only counting the tripattractions of the same AttractionName.

Trip TripId TripDat TripDes	e scription on	Attraction  AttractionId  AttractionName  CountryId  CountryName	for each Trip.Attraction unique TripDate, Attra &qty = Count(TripA print AttractionIn
∲ Attra ₽ Attra ₽ Trip.	actionId actionName AttractionVisitTime	CategoryId CategoryName CategoryName AttractionPhoto CityId CityName	endfor 01/01/2023 Eiffel Tower 1
TripId*	TripDate	TripDescription	
$\begin{pmatrix} 1 \end{pmatrix}$	1/1/2023		
2	4/4/2023		
3	1/1/2023		
4	5/5/2023		

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
$\begin{pmatrix} 1 \end{pmatrix}$	3	120
3	6	180
4	6	240
3	8	60

AttractionId*	AttractionName	
3	Eiffel Tower	
1	Louvre Museum	
8	Louvre Museum	
6	Matisse Museum	

unique TripDate, AttractionName

&qty = Count(TripAttractionVisitTime)

print AttractionInfo //TripDate, AttractionName, &qty

Let's go a little further. We know that we can specify several attributes in the unique clause, and that they do not have to belong to the base table of the For each, as in this example.

We want to count the number of trips that include a visit to the same attraction name on the same date. That is, for the same TripDate and AttractionName, how many records there are in TripAttraction.

If this is the data from the tables (we only show the relevant records), we see that for Eiffel Tower there will only be one record in TripAttraction: the one for trip 1, which is on this date.

Trip TripId TripDate TripDesc Attraction	iption i tionId tionName tractionVisitTime	Attraction AttractionId AttractionName CountryId CountryId CategoryId CategoryName AttractionPhoto CityId CityId CityName	for end	each Trip.Att unique TripDat &qty = Coum print Attra for 01/01/2023 Eiffel	raction e, AttractionName t(TripAttractionVisitTim ctionInfo //TripDate, At Tower1	e) tractionName	≥, &qty
Tripld*	TripDate	TripDescription		01/01/2023 Louvi	re Museum 2		
	1/1/2023						
2	4/4/2023						
(3)	1/1/2023						
4	5/5/2023						
TripId*	AttractionId*	TripAttractionVisitTime					
	1	180		AttractionId*	AttractionName		
2	1	200		3	Eiffel Tower		
1	3	120		1	Louvre Museum		
3	6	180		8	Louvre Museum		
4	6	240		6	Matisse Museum		
$\left(3\right)$	8	60					

We have the Louvre Museum in these 3 records. When we look at the dates, for trip 1 and for trip 3 they are the same, so in the output we will have...



Attraction			
9	AttractionId		
9	AttractionName		
7	CountryId		
¥	CountryName		
8	CategoryId		
Ł	CategoryName		

		🤄 🖌 CityName
ipld*	TripDate	TripDescription
1)	1/1/2023	
2	4/4/2023	
3)	1/1/2023	
4	5/5/2023	

TripId*	AttractionId*	TripAttractionVisitTime
(1)	1	180
2	1	200
1	3	120
3	6	180
4	6	240
(3)	8	60

01/01/2023 Eiffel Tower 1		
01/01/2023 Louvre Museum	2	

&qty = Count(TripAttractionVisitTime)

print AttractionInfo //TripDate, AttractionName, &qty

1

for each Trip.Attraction

4/4/2023 Louvre Museum

endfor

unique TripDate, AttractionName

AttractionId*	AttractionName	
3	Eiffel Tower	
1	Louvre Museum	
8	Louvre Museum	
6	Matisse Museum	

And for 2 it's this other one, so this will be shown in the output.



 ĸ	CountryName
 8	CategoryId
 ĸ	CategoryName
 -	AttractionPhoto
_	

TripId*	TripDate	TripDescription
1	1/1/2023	
2	4/4/2023	
(3)	1/1/2023	
(4)	5/5/2023	

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
(3)	6	180
4	6	240
3	8	60

for each Trip.Attraction	
unique TripDate, AttractionName	
<pre>&amp;qty = Count(TripAttractionVisitTime)</pre>	
<pre>print AttractionInfo //TripDate, AttractionName,</pre>	&qty

endfor

01/01/2023 Eiffel Tower 1			
1/1/2023 Louvre Museum	2		
4/4/2023 Louvre Museum	1		
01/01/2023 Matisse Museum	1		
05/05/2023 Matisse Museum	1		

AttractionId*	AttractionName	
3	Eiffel Tower	
1	Louvre Museum	
8	Louvre Museum	
6	Matisse Museum	

Lastly, for Matisse Museum: we have trip 3 and 4, which, since they have different dates, will lead to two prints in the output.



At	traction
9	AttractionId
9	AttractionName
٨	CountryId
R	CountryName

CategoryIdCategoryName

- 🚵 AttractionPhoto
- CityId
   CityName

TripId*	TripDate	TripDescription
1	1/1/2023	
2	4/4/2023	
3	1/1/2023	
4	5/5/2023	

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	6	180
4	6	240
3	8	60

for each Trip.Attraction	
unique TripDate, AttractionName	
<pre>&amp;qty = Count(TripAttractionVisitTime)</pre>	
<pre>print AttractionInfo //TripDate, AttractionName,</pre>	&qt

endfor

01/01/2023 Eiffel Tower 1	
01/01/2023 Louvre Museum	2
4/4/2023 Louvre Museum	1
01/01/2023 Matisse Museum	1
05/05/2023 Matisse Museum	1

AttractionId*	AttractionName	
3	Eiffel Tower	
1	Louvre Museum	
8	Louvre Museum	
6	Matisse Museum	

In this example, neither of the two attributes of the unique clause belongs to the base table of the For each.

We could also use formulas in the unique clause, as long as they are global.

Trip TripDate TripDate TripDes Attractic Attractic TripA	e cription n ctionId ctionName sttractionVisitTime	Attraction  AttractionId  AttractionName  CountryId  CountryId  CategoryId  CategoryIame  AttractionPhoto  CityId  CityId  CityName	fc	or each Trip.Att unique TripDato &tripYear = &qty = Coun print Attra ddfor 2023 Eiffel Tower	raction e.Year(), AttractionNam TripDate.Year() t(TripAttractionVisitTi ctionInfo //&tripYear, 1	ne me) AttractionNam	e, &qty
TripId*	TripDate	TripDescription					
$\begin{pmatrix} 1 \end{pmatrix}$	1/1/2023						
2	4/4/2023 202						
3	1/1/2023						
4	5/5/2023						
TripId*	AttractionId*	TripAttractionVisitTime					
1	1	180		AttractionId*	AttractionName		
2	1	200		3	Eiffel Tower		
$\begin{pmatrix} 1 \end{pmatrix}$	3	120		1	Louvre Museum		
3	6	180		8	Louvre Museum		

For example, let's suppose we want to count the trips by attraction name, according to the year of the trip. That is, for each attraction name, by year, how many trips it is in.

Matisse Museum

We will be tempted to write this unique clause, so that if we change the year of trip 2, for this other one, then the output would have to be as follows:

AttractionName Eiffel Tower is only on one trip, so its year is listed and the count will give 1.

		🖃 🧮 Attraction	
P TripId		📍 AttractionId	
🖓 TripDat	e	- 🆓 AttractionNam	9
<ul> <li>TripDes</li> </ul>	cription	- 🔁 CountryId	
🖃 들 Attractio	on	CountryName	
- 📍 Attra	uctionId	CategoryId	
🖌 🖌 Attra	loctionName	CategoryName	
🗆 🖓 Trip/	AttractionVisitTime	AttractionPhoto	)
		者 CityId	
		🔤 🖌 CityName	
TripId*	TripDate	TripDescription	
(1)	1/1/2023		
	1/1/2023 4/4/2023 2024	 4	
$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$	1/1/2023 4/4/2023 2024 1/1/2023	 4 	
$\begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$	1/1/2023 4/4/2023 2024 1/1/2023 5/5/2023	 4  	
$\begin{pmatrix} 1 \\ 2 \\ \hline 3 \\ 4 \end{pmatrix}$	1/1/2023 4/4/2023 202 1/1/2023 5/5/2023	 4  	
1 2 3 4 Tripld*	1/1/2023 4/4/2023 202 1/1/2023 5/5/2023 AttractionId*	 4  TripAttractionVis	itTime

$\sim$		
(1)	1	180
2	1	200
1	3	120
3	6	180
4	6	240
(3)	8	60

for each Trip.Attraction	
<pre>unique TripDate.Year(), AttractionName</pre>	
<pre>&amp;tripYear = TripDate.Year()</pre>	
<pre>&amp;qty = Count(TripAttractionVisitTime)</pre>	
<pre>print AttractionInfo //&amp;tripYear, AttractionName, &amp;q</pre>	ty
endfor	

2023 Eiffel Tower	1
2023 Louvre Museum	2

AttractionId*	AttractionName	
3	Eiffel Tower	
1	Louvre Museum	
8	Louvre Museum	
6	Matisse Museum	

Then comes Louvre Museum which is on 3 trips. The year of 1 and 3 match, so the output will show this.

🖃 🗐 Trip 🔤	Attraction
P TripId	- 📍 AttractionId
P TripDate	- ዖ AttractionName
<ul> <li>TripDescription</li> </ul>	🛛 🔁 CountryId
Attraction	🖌 🖌 CountryName
AttractionId	🗧 🤻 CategoryId
AttractionName	🖌 🖌 CategoryName
	AttractionPhoto
/ mp/addedonvisienne	🛛 🔁 CityId
	CityName

TripId*	TripDate	TripDescription
$\begin{pmatrix} 1 \end{pmatrix}$	1/1/2023	
2	4/4/2023 202	4
$\left( \begin{array}{c} 3 \end{array} \right)$	1/1/2023	
4	5/5/2023	

AttractionId*	TripAttractionVisitTime
1	180
1	200
3	120
6	180
6	240
8	60
	AttractionId* 1 1 3 6 6 6 8

for each Trip.Attraction	
<pre>unique TripDate.Year(), AttractionName</pre>	
<pre>&amp;tripYear = TripDate.Year()</pre>	
<pre>&amp;qty = Count(TripAttractionVisitTime)</pre>	
<pre>print AttractionInfo //&amp;tripYear, AttractionName,</pre>	&qty
endfor	

endfor

2023 Eiffel Tov	ver	1
2023 Louvre N	luseum	2
2024 Louvre N	luseum	1

AttractionId*	AttractionName	
3	Eiffel Tower	
1	Louvre Museum	
8	Louvre Museum	
6	Matisse Museum	

Since the year for trip 2 is different, this will be shown in the output.

Trip TripData TripDes Attractic Attractic TripVes TripVes TripVes TripVes TripVes TripVes TripVes TripVes TripData	e cription on octionId octionName AttractionVisitTime	Attraction AttractionId AttractionName AttractionName CountryId CountryName CategoryId CategoryId CategoryId CategoryName AttractionPhoto CityId CityId CityName
Tripld*	TripDate	TripDescription

Tripld*	TripDate	TripDescription
1	1/1/2023	
2	4/4/2023 2024	4
3	1/1/2023	
4	5/5/2023	

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	6	180
4	6	240
3	8	60

for each Trip.Attraction	
unique TripDate.Year(), AttractionName	
<pre>&amp;tripYear = TripDate.Year()</pre>	
<pre>&amp;qty = Count(TripAttractionVisitTime)</pre>	
<pre>print AttractionInfo //&amp;tripYear, AttractionName,</pre>	&qty
endfor	

endfor

2023 Eiffel Tower		1
2023 Louvre Museum		2
2024 Louvre Museum		1
2023 Matisse Museum	2	

AttractionId*	AttractionName	
3	Eiffel Tower	
1	Louvre Museum	
8	Louvre Museum	
6	Matisse Museum	

Then we move on to the last AttractionName, which is on two trips, 3 and 4, which are from the same year. So, the output will look as follows.

Trip TripDat TripDat TripDat TripDat TripDet Attracti	e scription on actionId actionName AttractionVisitTime	Attraction  AttractionId  AttractionName  AttractionName  CountryId  CountryId  CategoryId  CategoryName  AttractionPhoto  CityId  CityName	<pre>for each Trip.Attraction     unique TripDate.Year(), AttractionName     &amp;tripYear = TripDate.Year()     &amp;qty = Count(TripAttractionVisitTime)     print AttractionInfo //&amp;tripYear, AttractionName, &amp;qty endfor For each BaseTrn,, BaseTrn,</pre>
Tuintal*	TripData	TrinDescription	skip expression, count expression <sub>2</sub>
πρια	Inpuate	Inplescription	order $att_1, att_2, \dots, att_n$ [when condition]
1	1/1/2023		order att <sub>1</sub> , att <sub>2</sub> ,, att <sub>n</sub> [when condition]
2	4/4/2023 202	4	
3	1/1/2023		using DataSelector ( $parm_{e}$ $parm_{e}$ )
4	5/5/2023		where condition [when condition]
			where condition [when condition]
TripId*	AttractionId*	TripAttractionVisitTime	where att IN DataSelector ( $parm_1, parm_2,, parm_n$ )
1	1	180	blocking n
-	-	200	main_code
2	1	200	when duplicate
1	3	120	when_duplicate_code
3	6	180	when none code
4	6	240	endfor
3	8	60	chulor

The problem is that we will not be allowed to use an expression in the unique clause. We can only place attributes, as is clear from the syntax.

But those attributes may well be formula attributes.

□       Trip         □       ↑         □       ↑         □       ↑         □       ↑         □       ↓         □       ↓         □       ↓         □       ↓         ↓       ↓         <	TripDate on Id Name ctionVisitTime	Attraction  AttractionId  AttractionName  AttractionVame  CountryId  CountryName  CategoryId  CategoryId  CategoryName  Category	<pre>for each Trip.Attraction     unique TripYear , AttractionName     &amp;tripYear = TripDate.Year()     &amp;qty = Count(TripAttractionVisitTime)     print AttractionInfo //&amp;tripYear, AttractionName, &amp;qty endfor</pre>
			For Each InpAttraction (Line: 23) Order: NONE Unique: AttractionName TrinYear
			Navigation filters: Start from: FirstRecord Loop while: NotEndOffable
			Join location: Server
			Implementation ( Implementation ) Implementation     Implementation ( Implementation ) Implementation ( Implementation )     Implementation ( Implementation ) Implementation ( Implementation )     Implementation ( Implementation ) Implementation ( Implementation )
			Formulas
			Navigation to evaluate: count( <u>TripAttractionVisitTime</u> )
			Given: <u>AttractionName</u> Index: ITRIPATTRACTION
			Group by: <u>TripDate</u> . year( ) <u>AttractionName</u>
			#=TripAttraction
			$= \underline{\operatorname{Trip}} (\operatorname{Tripld})$

Therefore, if the example had a TripYear attribute—formula—and we used it in the unique clause, everything would work as expected, as shown in the navigation list.

# For each *BaseTrn*<sub>1</sub>, ..., *BaseTrn*<sub>n</sub>

# Restrictions

skip expression<sub>1</sub> count expression<sub>2</sub>
order att<sub>1</sub>, att<sub>2</sub>, ..., att<sub>n</sub> [when condition]
order att<sub>1</sub>, att<sub>2</sub>, ..., att<sub>n</sub> [when condition]
order none [when condition]
unique att<sub>1</sub>, att<sub>2</sub>, ..., att<sub>n</sub>
using DataSelector ( parm<sub>1</sub>, parm<sub>2</sub>, ..., parm<sub>n</sub>)
where condition [when condition]
where att IN DataSelector ( parm<sub>1</sub>, parm<sub>2</sub>, ..., parm<sub>n</sub>)

### blocking n

main\_code

### when duplicate

when\_duplicate\_code

#### when none

when\_none\_code

# endfor

A restriction we saw is that only attributes (which can be formulas) but not expressions can be used.

Another restriction: only attributes that have unique values for those in the unique clause can be included in the main code or body of the For each.



Louvre Museum 1

AttractionId*	TripAttractionVisitTime
1	180
1	200
3	120
6	180
6	240
8	60
	AttractionId* 1 1 3 6 6 6 8

AttractionId*	AttractionName	
3	Eiffel Tower	
1	Louvre Museum	
8	Louvre Museum	
6	Matisse Museum	

In this example, where we are asking for unique tripattraction values according to AttractionId, we see that these two records will be processed once and Louvre Museum will be displayed together with 2 as the count result. On the other hand, this other record will be processed alone, showing Louvre Museum and 1 in the output.

In the printblock it was possible to put AttractionName because for each AttractionId of the unique clause its value is unique. We could also place CountryName, CityName, CategoryName; that is, unique attributes for AttractionId. But if we had placed TripAttractionVisitTime or TripDate, we would get an error.



Louvre Museum	3
---------------	---

TripId*	AttractionId*	TripAttractionVisitTime
1	1	180
2	1	200
1	3	120
3	6	180
4	6	240
3	8	60

AttractionId*	AttractionName	
3	Eiffel Tower	
1	Louvre Museum	
8	Louvre Museum	
6	Matisse Museum	

There are more subtle cases that can mislead us. For example, what would happen if instead of AttractionId we placed AttractionName in the unique clause?

With this code there will be no problem, but it will show something different than the previous one if there are attractions with repeated names, as in this case. The reason is that it will group these 3 records and will show 3 as the result of the count.

What would happen if out of distraction we put AttractionId in the printblock? We would get the same error as if we put TripId or TripDate. That is because from an AttractionName you don't get a single AttractionId.



unique att<sub>1</sub>, att<sub>2</sub>, ... , att<sub>n</sub>

main\_code

when\_duplicate\_code

when none code

blocking n

when none

endfor

when duplicate

using DataSelector (parm<sub>1</sub>, parm<sub>2</sub>, ..., parm<sub>n</sub>) where condition [when condition] where condition [when condition]

where att IN DataSelector (parm1, parm2, ..., parmn)

The restriction only applies to the body of the For each, not to the other clauses. In particular, it doesn't apply to the filters. That is, it applies to what happens after the records have been sorted and filtered.

So, for example, we might want to list every attraction name included in trips after today's date, together with the number of trips they are in. And with the query sorted by AttractionId.

🖃 🥅 Trip		Attraction								
📍 TripId 💡 TripDate		AttractionId	for each Trip.Attraction order AttractionId							
		AttractionName								
• Trip	Description		unique AttractionName							
Attr	action		Where IripDate > &today							
74	AttractionId		<pre>&amp;qty = Count(IripAttractionVisitiime, Iripuate&gt;&amp;today </pre>							
	AttractionName	AttractionPhoto	print Attractioninto //AttractionName, &qty							
Ϋ́	ripAttractionVisitTime		enuror							
		🖌 CityName								
			&today: 3/3/2023							
TripId*	TripDate	TripDescription								
1	1/1/2023		Louvre Museum	2	Louvre Mu	iseum	4			
2	4/4/2023									
3	1/1/2023									
4	5/5/2023									
TripId	* AttractionId*	TripAttractionVisitTime								
1	1	180								
2	1	200	AttractionId*	Attr	actionName					
1	3	120	1	Louv	vre Museum					
3	8	60	3	Ei	ffel Tower					
4	8	240	8	Louv	vre Museum					

If this is the data and the date of the &today variable, let's think about what should be the result of the query.

The TripAttraction table is run through by AttractionId, but also the records that share the same AttractionName are considered only once. In this case, for the first record there will be these four. But of those, how many will pass the filter? The first one does not, the second one does, the third one does not, and the fourth one does.

If the Count formula also includes the same condition, then the output will show Louvre Museum, and 2.

If we had not added the same filter condition for the formula, Louvre Museum and 4 would be displayed.

Next, we go to the next record to process in the For each. It is the only one left, and its date doesn't meet the filter, so nothing else is processed. The final result will be this (depending on whether or not the filter condition was added to the formula).

🕒 🗐 Trip	Attraction		
For Each TripAttract	tion (Line: 16)	for ea ord	ch Trip.Attraction er AttractionId
Order: Unique: Navigation filters: Constraints: Join location:	AttractionId Index: ITRIPATTRACTION1 AttractionName Start from: FirstRecord Loop while: NotEndOfTable <u>TripDate</u> > &Today Server	uni Whe endfor	<pre>que AttractionName re TripDate &gt; &amp;today &amp;qty = Count(TripAttractionVisitTime, TripDate&gt;&amp;today) print AttractionInfo //AttractionName, &amp;qty</pre>
IIII= <u>TripAtt</u> IIII IIII	raction ( <i>Tripld</i> , <i>AttractionId</i> ) INTO <u>Tripld AttractionId</u> =Trip ( <i>Tripld</i> ) INTO <u>TripDate</u> =Attraction ( <i>AttractionId</i> ) INTO <u>AttractionName</u> = <u>count( TripAttractionVisitTime.</u> ) navigation ( <i>AttractionName</i> )		
Formulas			Formulas
Navigation to o Where: <u>Tr</u> Given: <u>At</u> Index: IT Group by: <u>At</u>	<mark>⊧valuate:</mark> count( <u>TripAttractionVisitTime</u> ) <u>ipDate</u> > &Today <u>tractionName</u> &Today RIPATTRACTION tractionName		Navigation to evaluate: count(         TripAttractionVisitTime         )           Given:         AttractionName         )           Index:         ITRIPATTRACTION         )           Group by:         AttractionName         )
anT==⊞	Attraction == <u>Trip ( <i>TripId</i> )</u> == <u>Attraction ( <i>AttractionId</i> )</u>		= <u>TripAttraction</u> = <u>Attraction</u> ( <u>AttractionId</u> )

In fact, if we look at the navigation list, we can clearly see that the formula is calculated as we want. Let's look at the Group by and how the Where shows the filter on the records to be counted.

If we remove the Count condition, then the navigation list will inform this for the formula.

or each	BaseTrn <sub>1</sub> , , BaseTrn <sub>r</sub>	1		Restrictions
skip	expression <sub>1</sub> count expres	ssion <sub>2</sub>		
orde	r att <sub>1</sub> , att <sub>2</sub> , , att <sub>n</sub> [whe	n condition]		
orde	r att <sub>1</sub> , att <sub>2</sub> , , att <sub>n</sub> [whe	n condition]		
orde	r none [when condition]			
uniq	ue att <sub>1</sub> , att <sub>2</sub> , , att <sub>n</sub>			
using	g DataSelector ( parm <sub>1</sub> , p	arm <sub>2</sub> , , parm <sub>n</sub> )		
whe	re condition [when cond	ition		
whe	re condition when cond	ition		
whe	re att IN DataSelector	parm <sub>1</sub> , parm <sub>2</sub> , , parm <sub>n</sub> )		
bloci	king n		For each	≠ Base table
	main_code —			
whe	n duplicate			
	when_duplicate_co	de	endfor	
whe	n none			
	when_none_code			

# endfor

Finally, let's mention the last restriction: if we use a unique clause, it only makes sense to nest another For each if it doesn't have the same base table. That is to say, we cannot use unique in control breaks, as one might think at first.

Trip     Ittraction       TripDate     AttractionName       TripDescription     CountryId       AttractionId     CountryId       AttractionName     CountryId       AttractionName     CategoryId       AttractionName     AttractionPhoto       TripAttractionVisitTime     AttractionPhoto			<pre>For each Trip.Attraction     order AttractionName     print AttractionInfo //AttractionName     For each Trip.Attraction         print TripAttractionInfo //TripDate TripAttractionVisitTime     endfor endfor</pre>					
			🔤 🖌 CityName	Eiffel To	ower			
					01/01/2023 120			
	TripId*	TripDate	TripDescription					
	1	1/1/2023						
	2	4/4/2023						
	3	1/1/2023						
	4	5/5/2023						
	TripId*	AttractionId*	TripAttractionVisitTime					
	1	1	180			Ļ		
	2	1	200		AttractionId*	AttractionName		
	1	3	120		3	Eiffel Tower		
	3	8	60		1	Louvre Museum		
	4	8	240		8	Louvre Museum		

So, for example, if we want a list of the attractions included in trips, with the duration of the visit to the attraction in each of those trips, we will have no alternative but to implement it as the typical control break.

We run through TripAttraction sorted by AttractionName and, for the first TripAttraction record with the first AttractionName, we print the attraction name. Then we iterate through the records with the same AttractionName, here only this one. We print the date of the trip and the length of the visit, and move on to the next group...

Image: TripId       Image: TripId<	\ttractionVisitTi
TripId* TripDate TripDescription 01/01/2023 120	
1 1/1/2023 Louvre Museum	
2 4/4/2023 01/01/2023 180	
3 <u>1/1/2023</u> <u>4/4/2023</u> 200	
4 <u>5/5/2023</u> <u>1/1/2023</u> 60	
5/5/2023 240	
TripId* AttractionId* TripAttractionVisitTime	
2         1         200         AttractionId*         AttractionName	
1 3 120 3 Eiffel Tower	
3         8         60         1         Louvre Museum	
4         8         240         8         Louvre Museum	

...which is the one corresponding to these records with the same name in Attraction. We print the name, and again, iterate with the nested For each as long as the AttractionName doesn't change. And so the output is...



If we wanted to implement this with the unique clause, GeneXus would not allow it. The navigation list would show this error.

Attraction AttractionId AttractionName CountryId CountryId CategoryId CategoryId AttractionPhoto AttractionPhoto CityId CityName Hotel	For each Att order Cou unique Co &qty = print for ea pri endfor	raction intryName count(Attract: MainInfo //Coun icch Hotel .nt HotelInfo /,	ionName) ntryName, & /HotelName	qty	Brazil Franc Cinqu	e Dh la la iberte ie Terre mperio	1 3 1
HotelName	AttractionId*	AttractionName	Countryld			Countryld*	CountryName
- 🔁 CountryId	2	Christ	5			5	Brazil
CountryName CityId CityName	3	Eiffel Tower	2			2	France
	1	Louvre Museum	2			15	Italy
-	8	Matisse Museum	2				
	4	Cinque Terre	15				
CountryId CountryName							
⊟ 📑 City	Hotelld*	HotelName	Countryld				
CityId	1	Oh la la	2				
Cityname	3	Imperio	15				
	4	Liberte	2				

But we can nest a For each that navigates another table.

For example, let's see this case in which we are navigating the attractions table sorting by CountryName, from the extended one, and asking to process only once all the records that repeat the value of CountryName.

For them, we want to count the attractions of the same country, print that country with the number of attractions, and navigate the Hotel table printing the names of hotels in the same country.

So, with this data, the output will look like this. Here we have the Attraction table sorted by CountryName.

There is only one record for the country of the first record: its country is then listed and 1 for the count. Since there is no hotel in Brazil, it will go on to the next record in Attraction, which will be 3, corresponding to France. There are 3 records with the country France, so in the output you will see... Then the nested For each is run, and it will print the hotels in France. Therefore, the output will show...

And finally we will have ...



For each Attraction
 order CountryName
 unique CountryName
 &qty = count(AttractionName)
 print MainInfo //CountryName, &qty
 for each Hotel
 print HotelInfo //HotelName
 endfor
endfor

The navigation list will clearly show that the implementation is as we wanted. Note the Constraint of the nested For each by CountryName.



Finally, let's mention again that although we focused on the unique clause for the For each command, its logic applies to all other forms of queries.



training.genexus.com wiki.genexus.com training.genexus.com/certifications