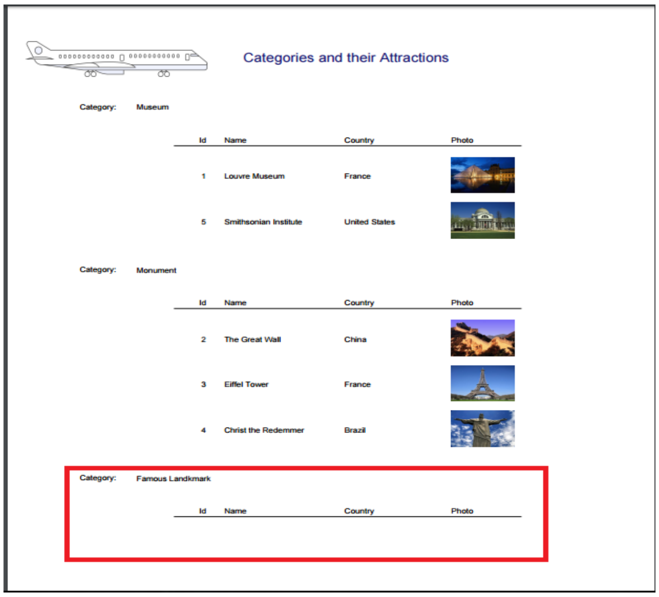
How to list grouped information

Nested For Eachs

Listing grouped information



Categories and their Attractions

Category: Museum

Id	Name	Country	Photo
1	Louvre Museum	France	
5	Smithsonian Institute	United States	

Category: Monument

Id	Name	Country	Photo
2	The Great Wall	China	
3	Eiffel Tower	France	
4	Christ the Redemmer	Brazil	

Category: Famous Landkmark

Id	Name	Country	Photo

- This was the listing of categories with its attractions.
- The Famous Landmark category does not have any attractions at the moment.
- What if now we don’t want to list ALL the categories, but rather only the ones that have attractions?

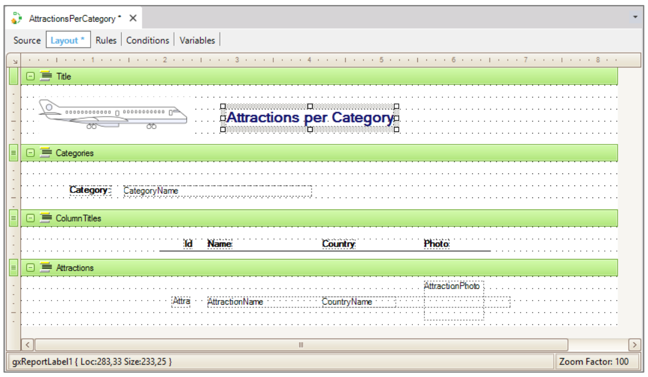
In the previous section we saw a list requested by the travel agency that showed all the tourist attraction Categories. For each category, it showed the list of attractions that had been entered.

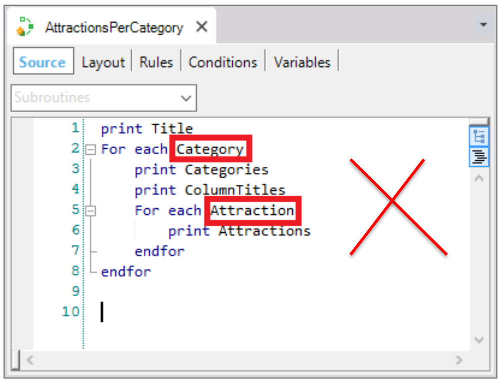
Let's change the category of “The Great Wall” attraction, so that “Famous Landmark” no longer has an associated attraction.

Now we execute the list again.
As we can see, this list shows ALL the categories that have been entered, even those that don't have associated attractions.

If this is not what we want, that is to say, if we want to show only the categories that have attractions. What do we do?

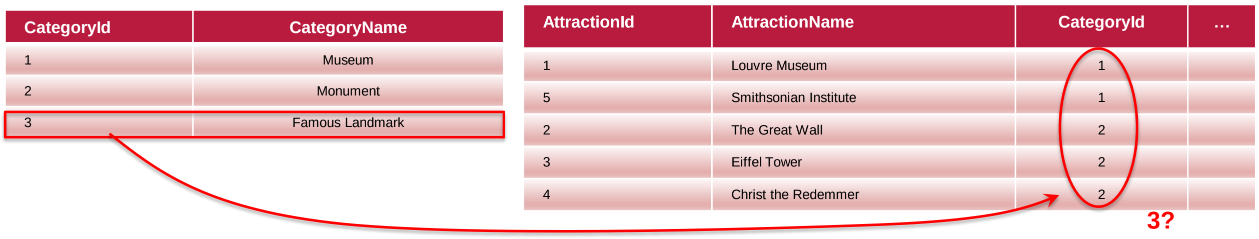
¿Why can’t we use the previous solution?





```
print Title
For each Category
    print Categories
    print ColumnTitles
    For each Attraction
        print Attractions
    endfor
endfor

```



CategoryId	CategoryName
1	Museum
2	Monument
3	Famous Landmark

AttractionId	AttractionName	CategoryId	...
1	Louvre Museum	1	
5	Smithsonian Institute	1	
2	The Great Wall	2	
3	Eiffel Tower	2	
4	Christ the Redemmer	2	

We will implement this in another procedure. To do so, we save the one we had with another name. And let’s change the text block’s title.

If we examine the For Each commands that we had implemented, we can see that the base table of the external For Each command is Category, and the base table of the nested For Each command is Attraction.

But in this way, we first access the Category table; next, the record data is printed and finally the nested For Each command is executed. As a result, the category will be printed before we know if it has related attractions or not.

This is not what we need. We need to access the **attractions' categories** because it is the only way to make sure that the category to be printed has at least one attraction.

GeneXus

Solution

Foreign Key

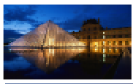
AttractionId	AttractionName	CountryId	CategoryId	…
1	Louvre Museum	2	1	
5	Smithsonian Institute	4	1	
2	The Great Wall	3	2	
3	Eiffel Tower	2	2	
4	Christ the Redemmer	1	3	

Primary Key

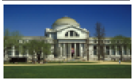
CategoryId	CategoryName
1	Museum
2	Monument
3	Famous Landmark

Category: Museum

1 Louvre Museum France

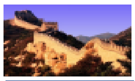


5 Smithsonian Institute United States

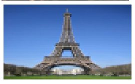


Category: Monument

2 The Great Wall China



3 Eiffel Tower France



```
For each Attraction                    order CategoryId
    print Categories
    For each Attraction
        print Attractions
    endfor
endfor
```

Grouping or 'break' criterion

Break control

The idea is to group the attractions in the Attraction table by category, and then go over those groups, printing the category for each of them, for which we will have to access the Category table in order to recover its name. In addition to printing each attraction in the group.

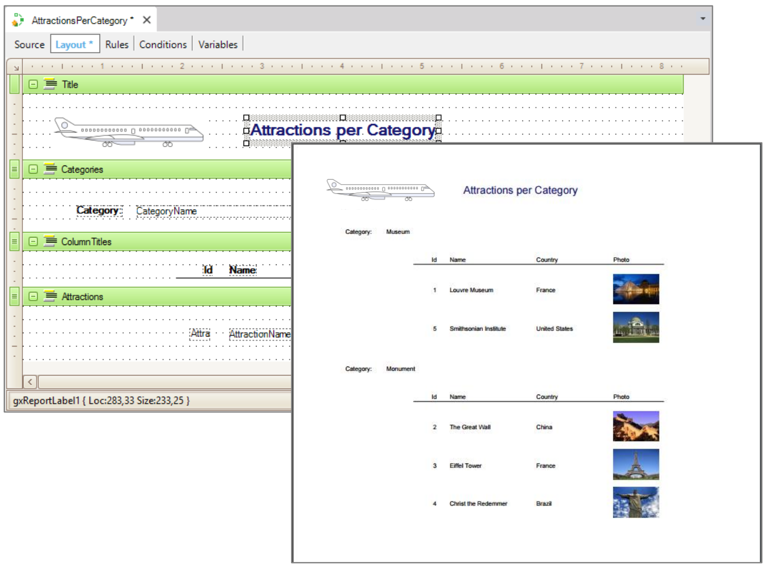
To then move on to the next group … and so on.

In sum, what we must do is go over the Attraction table only, first grouping it by category and printing the category, and then printing, from each category group (obviously by navigating the same Attractions table) the attractions in it.

Note that the way in which we instruct GeneXus that the grouping is to be done by CategoryId is by specifying the Order clause.

This case of nested For eachs that go over the same table is known as **control break.**

Listing grouped information: control break



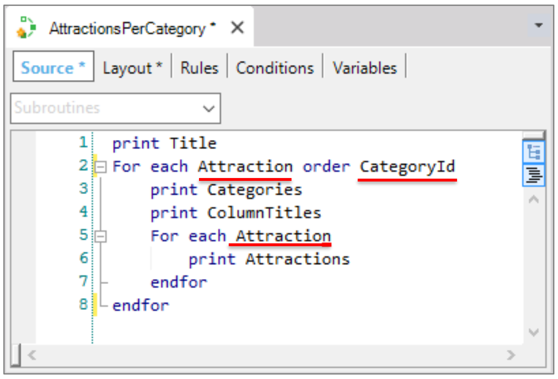
Attractions per Category

Category: Museum

Id	Name	Country	Photo
1	Louvre Museum	France	
5	Smithsonian Institute	United States	

Category: Monument

Id	Name	Country	Photo
2	The Great Wall	China	
3	Eiffel Tower	France	
4	Christ the Redemmer	Brazil	



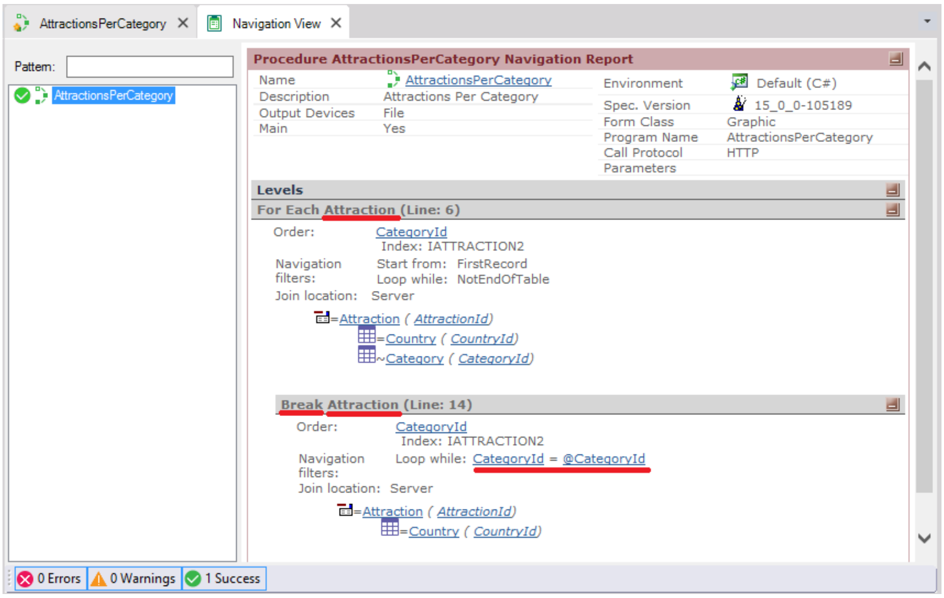
```
print Title
For each Attraction order CategoryId
    print Categories
    print ColumnTitles
    For each Attraction
        print Attractions
    endfor
endfor
```

Let us now do the changes in our procedure…

First we change the transaction of the external For each, and use Attraction…
Then we add the Order clause, to sort by CategoryId, which in the case of the control break will also apply to something more significant: for **grouping** by that attribute.

We run it. Note that the "Famous Landmark" category, which doesn't have any attractions, is not being listed.

Listing grouped information: control break



AttractionsPerCategory | Navigation View

Pattern:

AttractionsPerCategory

```
Procedure AttractionsPerCategory Navigation Report
Name             AttractionsPerCategory        Environment     Default (C#)
Description      Attractions Per Category      Spec. Version   15_0_0-105189
Output Devices   File                          Form Class      Graphic
Main             Yes                           Program Name    AttractionsPerCategory
                                               Call Protocol   HTTP
                                               Parameters

Levels
For Each Attraction (Line: 6)
  Order:        CategoryId
                 Index: IATTRACTION2
  Navigation    Start from:  FirstRecord
  filters:      Loop while:  NotEndOfTable
  Join location:  Server
        =Attraction ( AttractionId)
            =Country ( CountryId)
            ~Category ( CategoryId)

    Break Attraction (Line: 14)
      Order:        CategoryId
                     Index: IATTRACTION2
      Navigation    Loop while: CategoryId = @CategoryId
      filters:
      Join location:  Server
            =Attraction ( AttractionId)
                =Country ( CountryId)
```

0 Errors | 0 Warnings | 1 Success

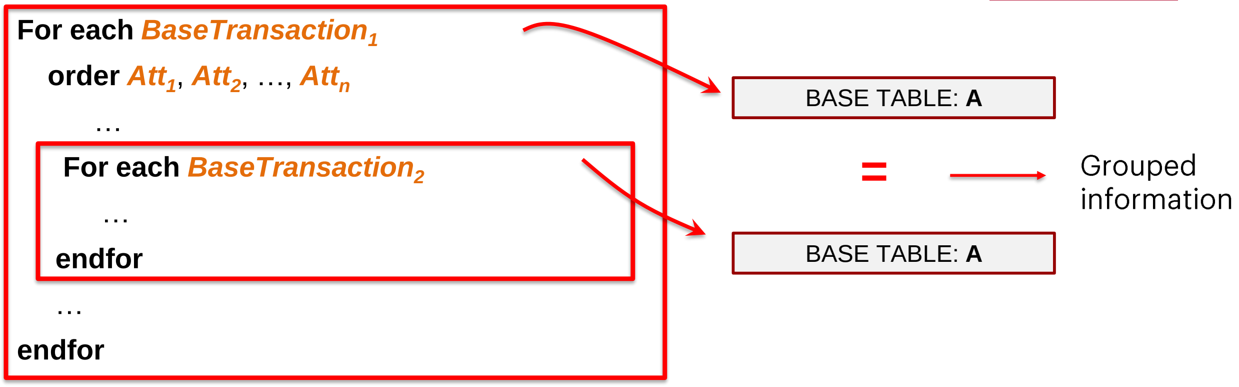
If we consider the resulting navigation list, we will see that it informs us about a For each to the Attraction table, ordered by CategoryId, which will be broken (Break) by the nested For each on the same table: Attraction. Note that in this break only the attractions in the category of that group are run through.

Conceptualization

- Break the information by some criteria

✓ Same base table for all

Break control



```
For each BaseTransaction_1
   order Att_1, Att_2, …, Att_n
        …
    For each BaseTransaction_2
         …
   endfor
   …
endfor
```

So let's now conceptualize the definition of a control break.

It relates to nested For eachs ...With the same base table for all the For eachs...

Conceptualization

- Break the information by some criteria

✓ **Break criteria: order clause**

```
For each $BaseTransaction_1$
  order $Att_1$, $Att_2$, …, $Att_n$
      …
    For each $BaseTransaction_2$
      …
    endfor
    …
endfor
```

Break criteria

BASE TABLE: **A**

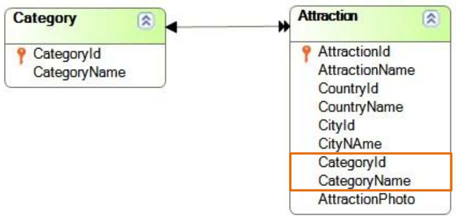
Grouped by…

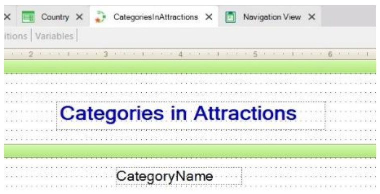
DEMO

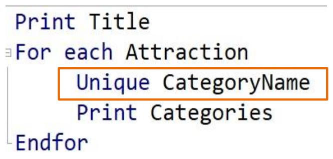
And the Order clause to establish the break criteria.

Another example: Unique clause

List only those categories that have registered tourist attractions. We don't want to see repeated names.







```
Print Title
For each Attraction
    Unique CategoryName
    Print Categories
Endfor
```

For each base
Table: ATTRACTION

Now let's look at another situation.

Suppose we need to list only those categories that have registered tourist attractions. How can we go about it?

If we look at the transaction design, we clearly see that the categories related to some tourist attraction are those found in Attraction, as a foreign key.

The first thing we can think of is a For Each command with Attraction as the base transaction and then list the name of the Category.

Although this list effectively shows the name of the categories that have some associated attraction, those names are repeated because, for example, there are several attractions that are Monuments. Therefore, "Monument" as a category name is listed several times.

How can we control that these listed names are not repeated? That is, that they are shown only once?
Using the Unique clause.

This clause allows you to indicate the attribute, or set of attributes, whose value should not be repeated in the query output. These attributes must belong to the extended table of the For Each command.

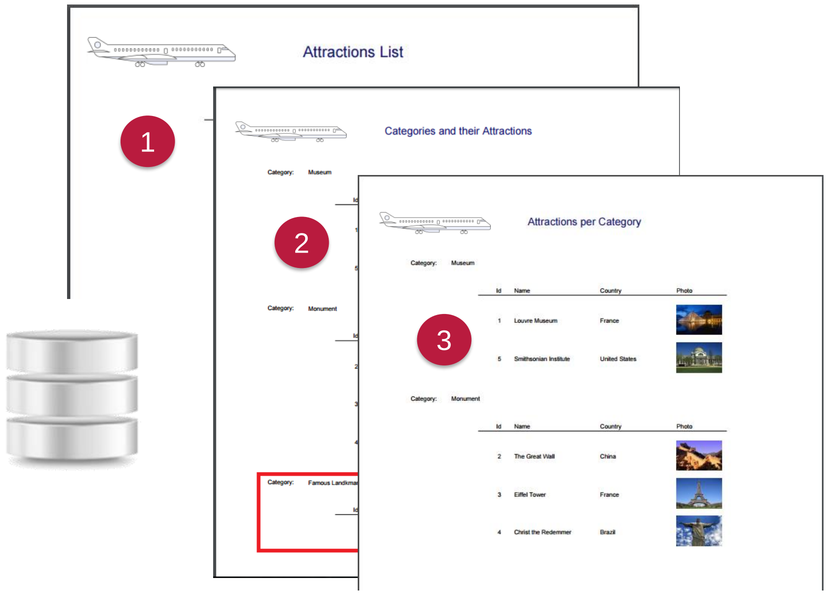
Summary

Summing up

- Listings

✓ Layout
✓ Source

For each command



Category: Museum

Id	Name	Country	Photo
1	Louvre Museum	France	
5	Smithsonian Institute	United States	

Category: Monument

Id	Name	Country	Photo
2	The Great Wall	China	
3	Eiffel Tower	France	
4	Christ the Redeemer	Brazil	

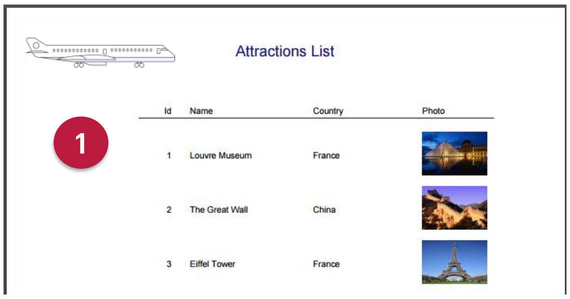
In these videos we have seen how GeneXus facilitates the determination of simple listings (that navigate a single table) or more complex listings that navigate information in several related tables (join), or in the same table, but grouped by some particular criterion (control break).

The command that we use in all cases for accessing the database is the **For each.**

Summing up

- Listings

✓ Layout
✓ Source



Attractions List

Id	Name	Country	Photo
1	Louvre Museum	France	
2	The Great Wall	China	
3	Eiffel Tower	France	

For each command

1

```
For each BaseTransaction
order Att_1, Att_2, …, Att_n
where condition_1
where condition_2
…
where condition_n
          MainCode
endfor
```

Simple For each

In the first case, we used a simple For each, where we inferred the table to be navigated through its Base Transaction.

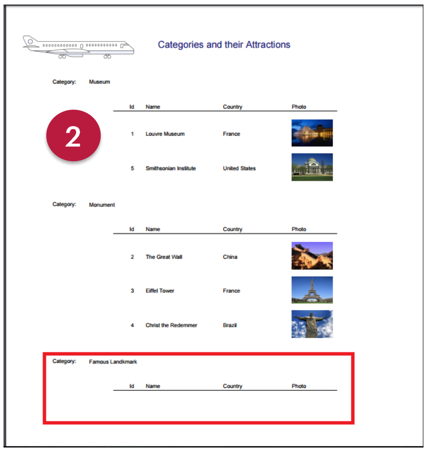
Summing up

- Listings

✓ Layout
✓ Source

For each command

Nested for eachs



2

Categories and their Attractions

Category: Museum

Id	Name	Country	Photo
1	Louvre Museum	France	
5	Smithsonian Institute	United States	

Category: Monument

Id	Name	Country	Photo
2	The Great Wall	China	
3	Eiffel Tower	France	
4	Christ the Redemmer	Brazil	

Category: Famous Landkmark

Id	Name	Country	Photo

2

```
For each BaseTransaction
order Att_1, Att_2, …, Att_n
where condition_1
where condition_2
…
where condition_n
   …
   For each BaseTransaction
         order Att_1, Att_2, …, Att_n
         where condition_1
         where condition_2
         …
         where condition_n

           …
          endfor
        …
endfor
```

1

≠

N

Join

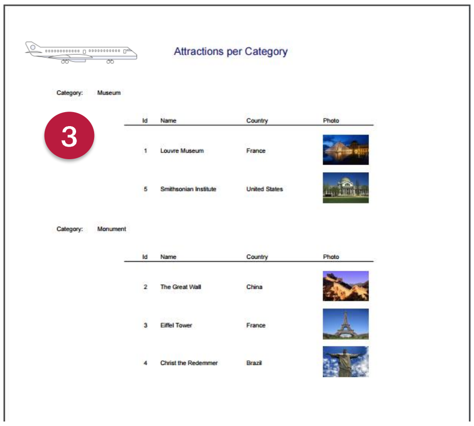
In the second case we have a pair of nested For eachs, where, from different base transactions we discover a relation of one to many amongst the information of each For each.

Summing up

- Listings

✓ Layout
✓ Source

For each command



3

Attractions per Category

Category: Museum

Id	Name	Country	Photo
1	Louvre Museum	France	
5	Smithsonian Institute	United States	

Category: Monument

Id	Name	Country	Photo
2	The Great Wall	China	
3	Eiffel Tower	France	
4	Christ the Redemmer	Brazil	

3

```
For each BaseTransaction
order Att1, Att2, …, Attn
where condition1
where condition2
…
where conditionn
    …
    For each BaseTransaction
        order Att1, Att2, …, Attn
        where condition1
        where condition2
        …
        where conditionn

          …
         endfor
        …
endfor
```

=

Nested For eachs

Break control

While in the third case we also have a couple of nested For eachs, but in this case their base transactions **are the same.** GeneXus then understands that we want to "break" or "group" the information from **the** table that is to be run through, by the attribute or the group of attributes specified in the Order clause of the external For each.

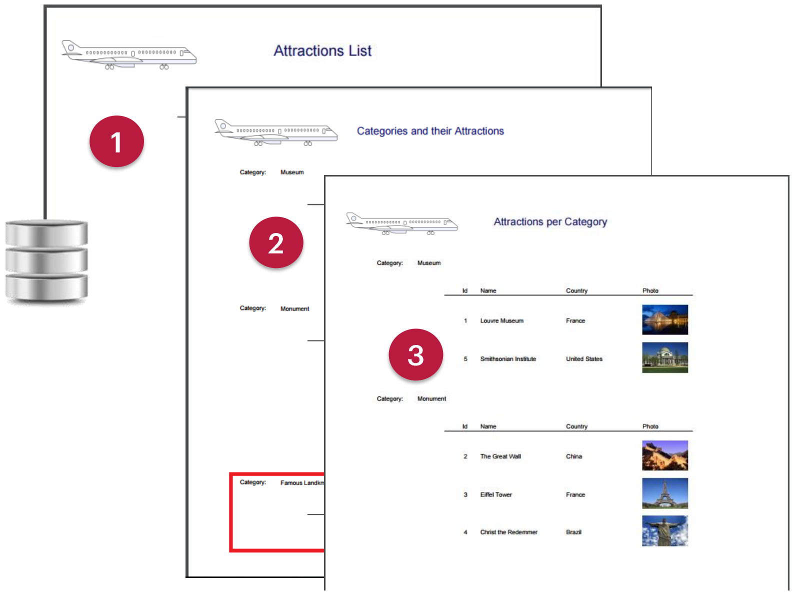
And more…

- Listings

✓ Layout
✓ Source

comando
For each

- Panels
- Query object



Further ahead we will see other mechanisms to implement queries to the database for obtaining information in a flexible and appealing manner.

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