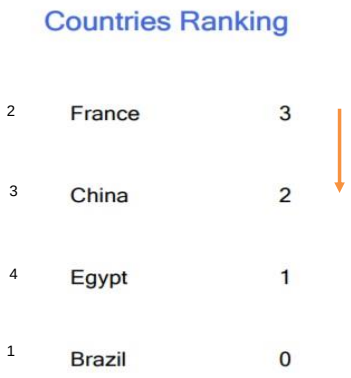
Compound data types

GeneXus™

Now let's consider that the Travel Agency has requested a ranking of all the countries, according to the number of tourist attractions they offer.

New requirement...

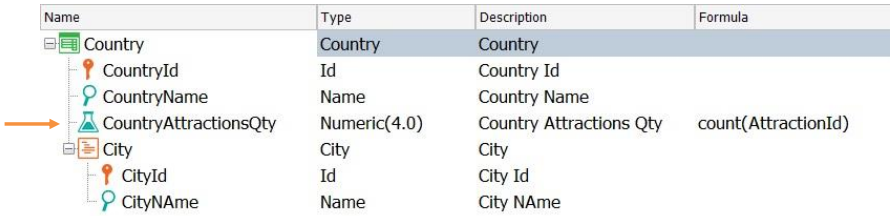
List all the countries, ordered from highest to lowest by the number of attractions they have registered.



We need to obtain a list similar to this one, which shows all the countries, ordered from highest to lowest by the number of attractions they have registered.

As we can see, each line will correspond to a country, showing its identifier, its name and the number of tourist attractions it has. The problem here is that we have to sort this information according to this last value, which is not in the database. It has to be calculated.

One option:



Name	Type	Description	Formula
Country	Country	Country	
CountryId	Id	Country Id	
CountryName	Name	Country Name	
CountryAttractionsQty	Numeric(4.0)	Country Attractions Qty	count(AttractionId)
City	City	City	
CityId	Id	City Id	
CityNAme	Name	City NAme	

```
For each Country order (CountryAttractionsQty)
     Print Countries
Endfor
```

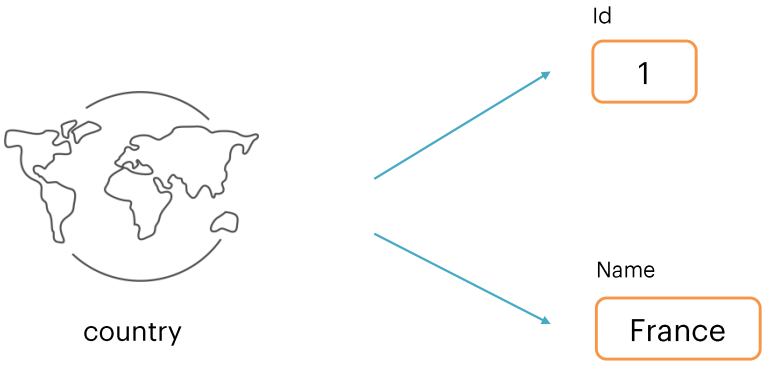
CountryName CountryAttractionsQty

How can this list be implemented?

One option, applying the concepts we already know, would be to define a formula attribute at the level of the Country transaction structure, and then run a For Each command, with Country as the base transaction, ordering from highest to lowest by that calculated attribute.

This solution is completely valid, but it is also valid to solve a requirement without the need to add new attributes to a transaction just to solve a certain query.

Simple data types



So, we are going to solve this requirement in another way, adding new concepts that will be very useful for more complex cases. Let's start talking about Structured Data Types.

So far, we have always used **simple** data types. We have defined attributes and domains of Numeric, Character, Date, Image type, etc.

If, for example, we now want to store the identifier and name of a country in variables, then we need two variables.

But we will see next that we can also use **compound data types.**

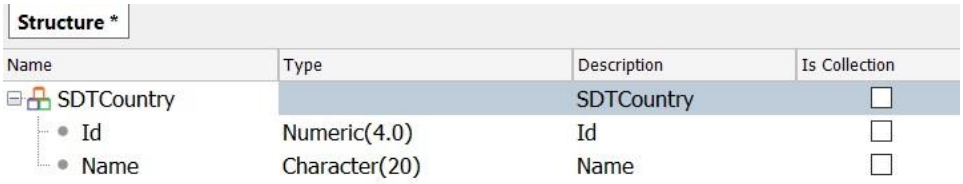
Compound or Structured data types

A compound data type can only be assigned to a variable, never to an attribute..

A compound data type allows several data items to be stored together in a single variable. To put it simply, it's like grouping several simple variables together under one name.

A structured data type is created by means of a GeneXus object of this type. The data type created can only then be assigned to a variable, never to an attribute.

GeneXus object: Structured Data Type (SDT)



we will then create an object of SDT type and name it SDTCountry. Note that we place SDT before the word because a transaction and a structured data type with the same name cannot exist within the same KB.

Remember that we already have the Country transaction.

For now, we only want to save the Identifier and the name of the countries so we define those items, or those members in the structure:

- Id, of Numeric type
- and Name, of Character type.

For this definition we've just made, GeneXus created the data type SDTCountry, so we can already start creating variables based on this data type.

The question we now ask ourselves is how can we load values into a structured variable?

Loading an SDT

1) Manual loading

```
&CountryItem.Id = 1
&CountryItem.Name = "France"
```

2) Load inside a For each

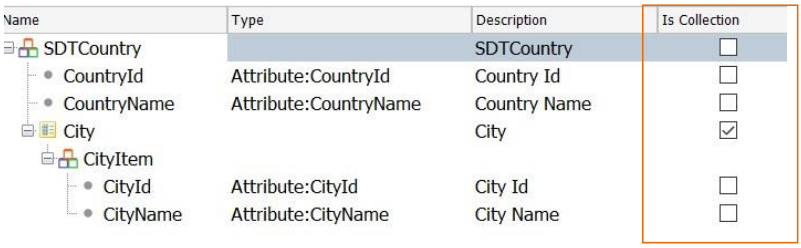
```
For each Country
     Where CountryName = "France"
     &CountryItem.Id = CountryId
     &CountryItem.Name = CountryName
     print printblock1
Endfor
```

As a first example we will load the data from France, and we will do it manually.
Note that by typing the variable &CountryItem, and pressing the period key, we can already see the items that make up the data type.

We will then load the Id 1 that corresponds to France, and the name "France." This way we load the variable manually.

Another option would be through a For Each command by positioning ourselves in the record corresponding to France, and then loading the variable &CountryItem with the value of CountryId for the ID item, and with the value of CountryName for the Name item.

Defining collections



Collection SDT

Collection variable

Let's see now that when the structure to be created matches completely, or partially, the structure of a transaction, we can drag this transaction into the structure of the SDT without generating any kind of ambiguity, because GeneXus can distinguish between the attributes of the transaction and the items of an SDT, even if their names match.

. Remember that a structured data type can only be assigned to variables and not to attributes.

Note also that the structure of an SDT can be very complex! For example, each country has a collection of cities, and it is clear that the data types of the SDT members are derived from the attributes.

What if we need to keep a collection of countries? How can it be done?

One is to define the SDT as a collection, and to do so we only need to check this box that says IsCollection.
Thus, when saving the changes, GeneXus will create the SDTCountry data type for the collection, and the SDTCountry.SDTCountryItem data type for the collection item.

Another way is to leave our SDT as we have originally defined it and mark the collection at the level of the defined variable.

In the next video, we will talk in detail about the collection variables.

GeneXus™

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