QlikView How to Guide
"Qlikview TreeView"

Description:

BISP is committed to provide BEST learning material to the beginners and advance learners. In the same series, we have prepared a complete end-to end Hands-on Beginner's Guide for Qlikview implementation. The document focuses on How to Series "Working with TreeView". Join our professional training program and learn from experts.

History:

Version	Description Change	Author	Publish Date
0.1	Initial Draft	Surbhi Sahu	21st Aug 2012
0.1	Review#1	Amit Sharma	29th Aug 2012

Table of Contents

The QlikView Treeview

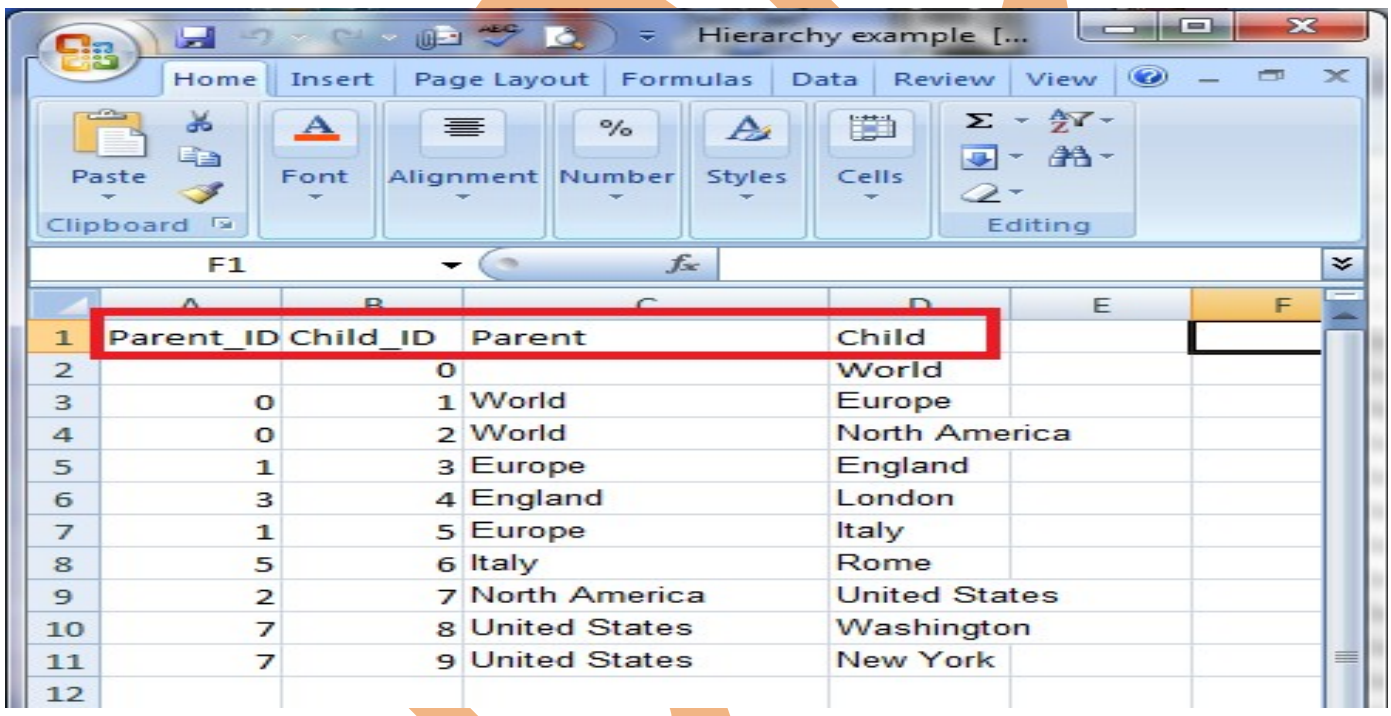
For implementing Treeview in the any QlikView app firstly we have to achieve hierarchy which discussed in the below procedure.

Expanding a hierarchy

The hierarchical nature of the table allows one value to be related to one or more values across the table, as a parent or as a child. In fact, one value can be related to one or more other values as a child and to one or more other different values as a parent.

The advantage of these tables is that they keep the information in a compact format, and QlikView is able to handle them, and expand its relations with a special function: Hierarchy. In technical terms, the original table format is called an Adjacent Nodes table, while the resulting table is called an Expanded Nodes. Some hierarchies are implied, like the ubiquitous calendar with Year ->Quarter->Month->Day, etc. In the case of the calendar, the hierarchy has levels that we all recognize and is typically stored in a denormalized table with a row for each value at the lowest level (say Day) and a column with repeating values for each higher level in the calendar hierarchy (Month, Quarter, Year, etc.).

We have given a table Hirarchy Example.

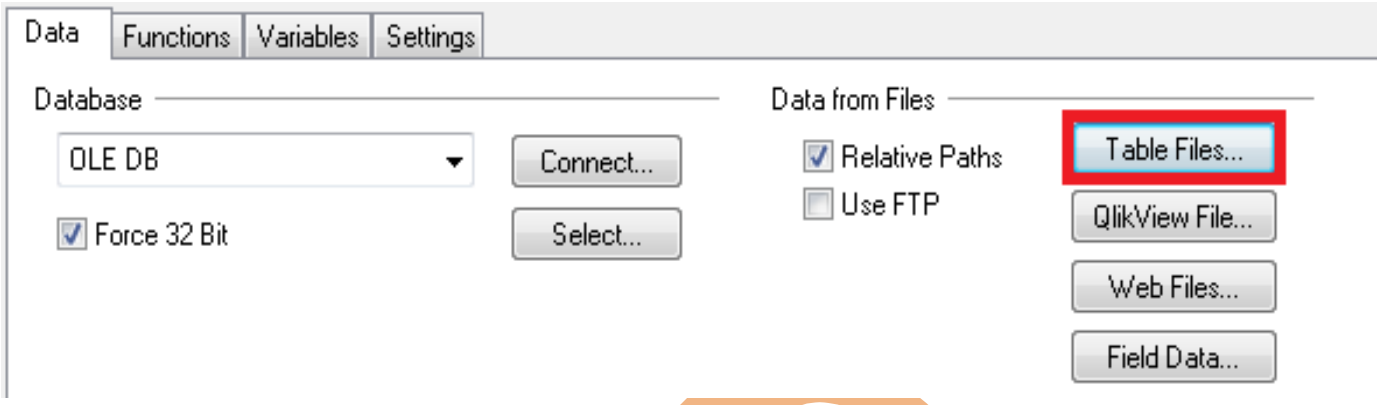


	A	B	C	D	E	F
1	Parent_ID	Child_ID	Parent	Child		
2		0		World		
3	0	1	World	Europe		
4	0	2	World	North America		
5	1	3	Europe	England		
6	3	4	England	London		
7	1	5	Europe	Italy		
8	5	6	Italy	Rome		
9	2	7	North America	United States		
10	7	8	United States	Washington		
11	7	9	United States	New York		
12						

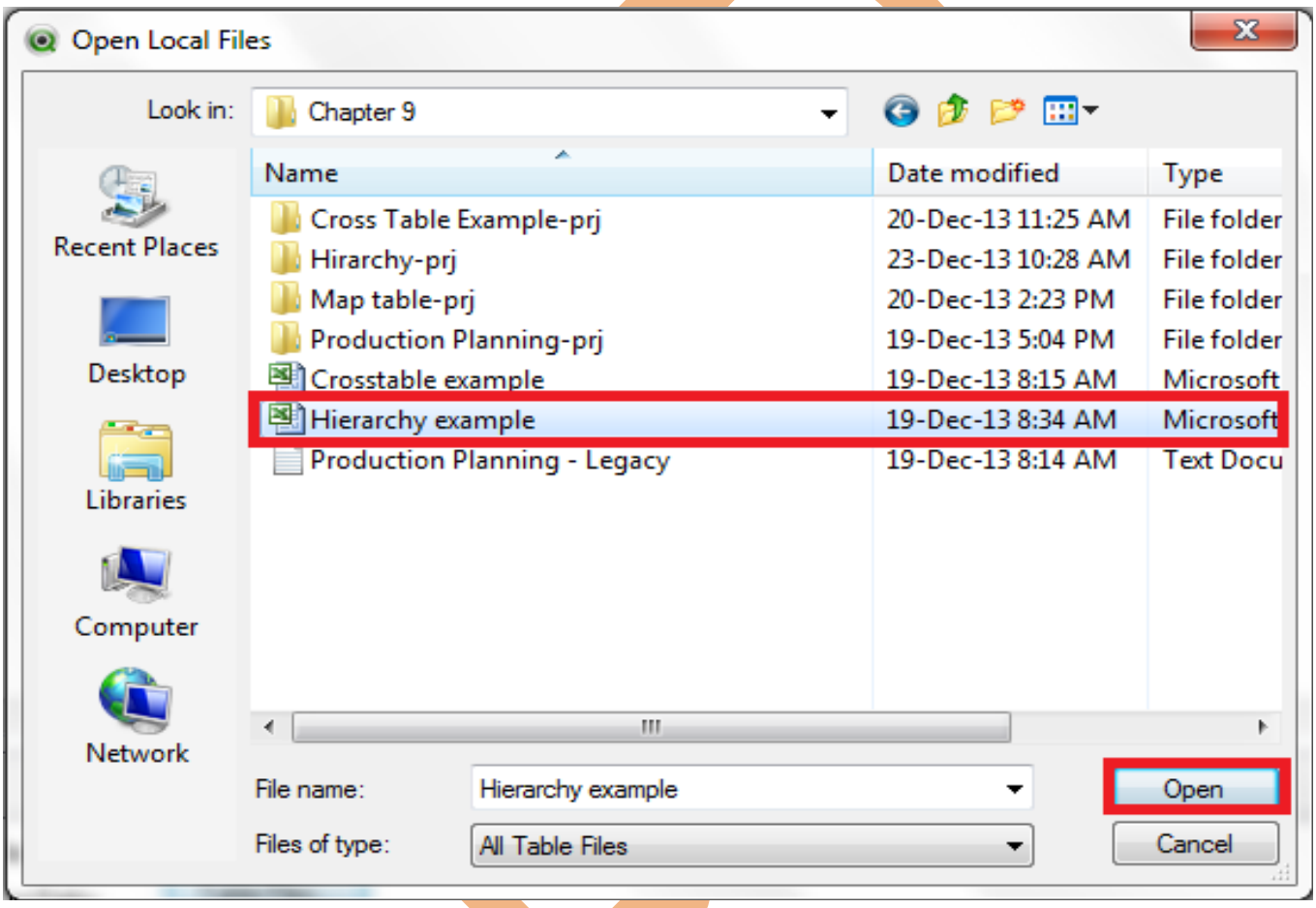
In this table 4 title fields are available. In which World, continent ,country and city name are present. we want to achieve the hierarchy as World – >Continent – >Country –>City for achieving this hierarchy we will follow these steps.

Step 1)Firstly create the new application and give the name Hierarchy.qvd.

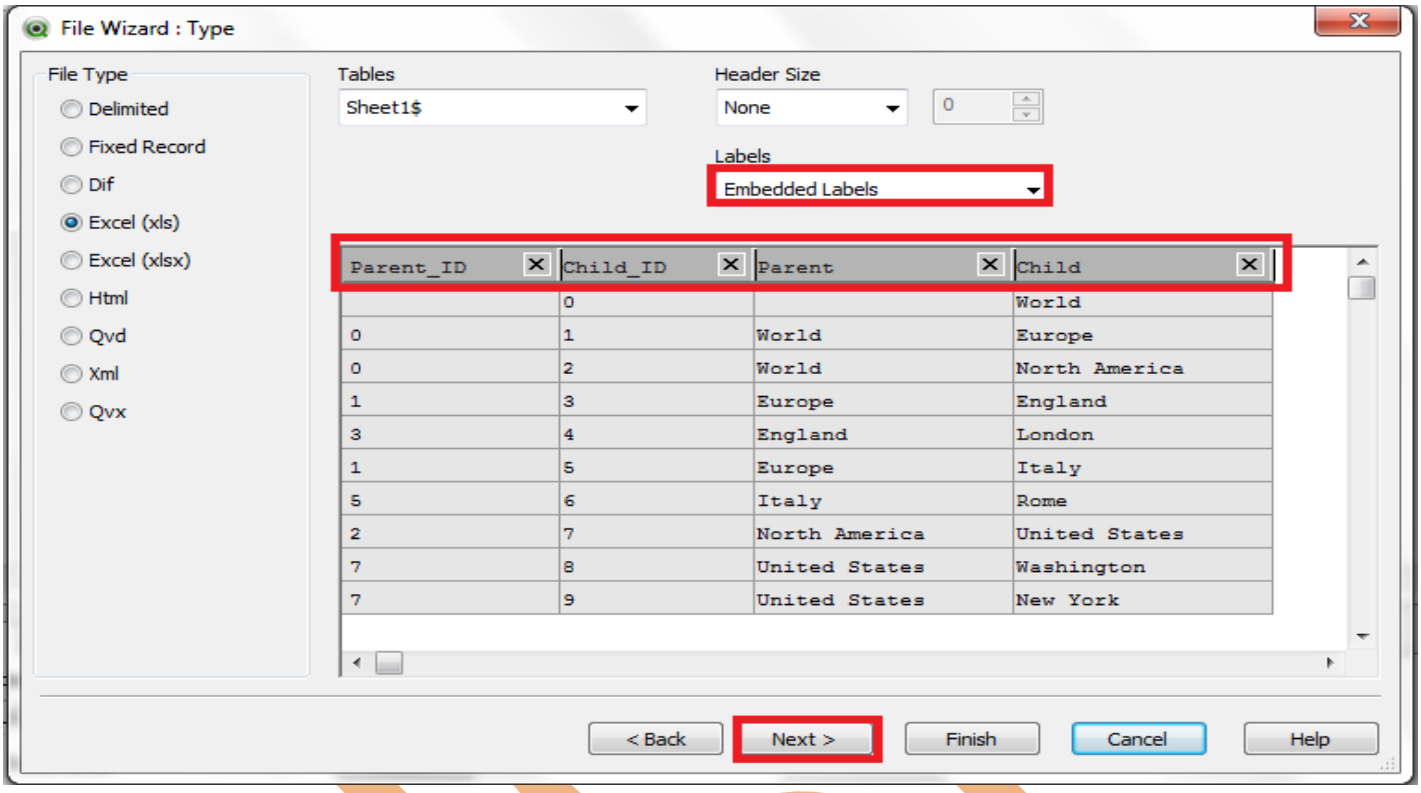
Step 2)Then go to the ->script editor and click on the option table files.



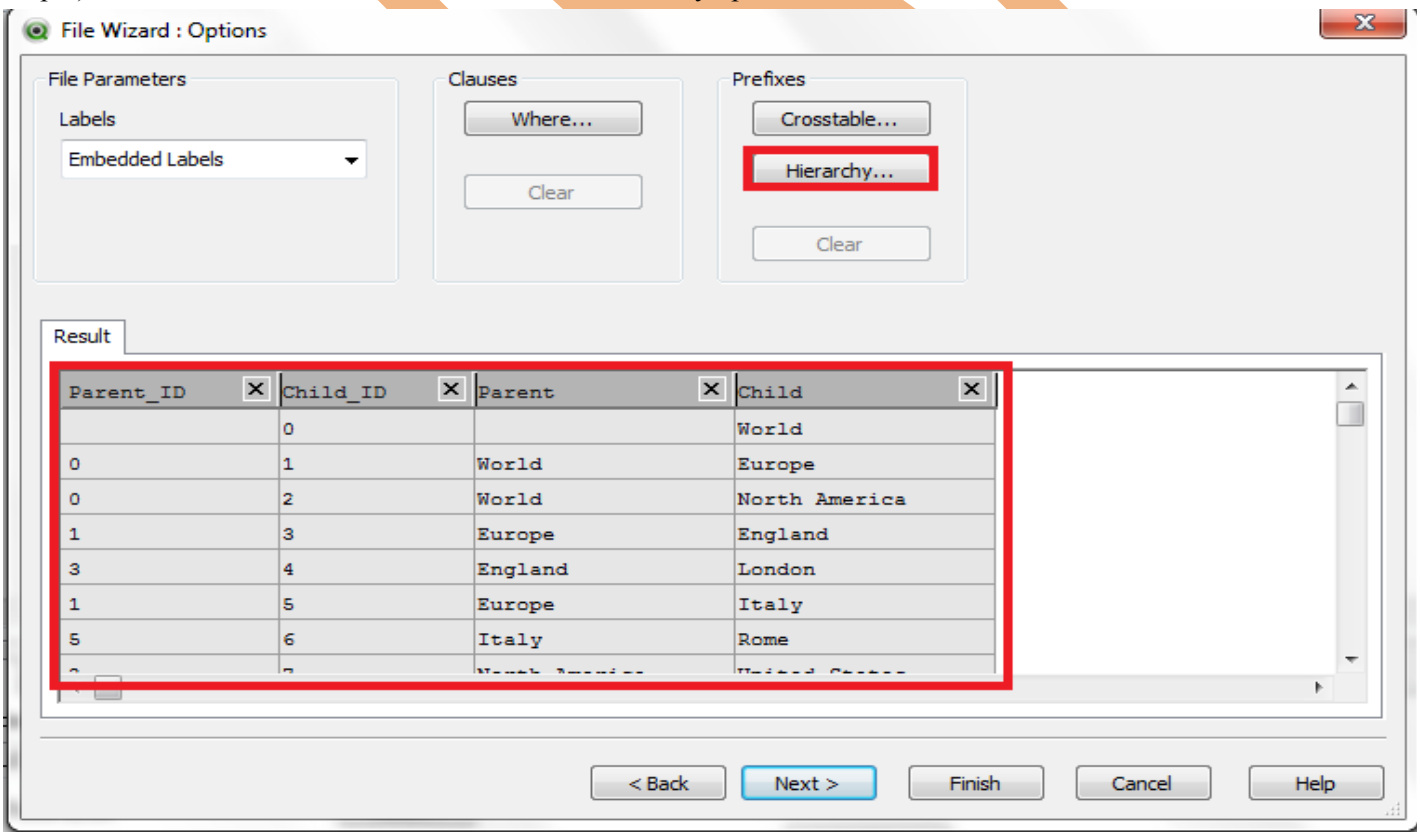
Step 3)Choose the file Hierarchy example and then open it.

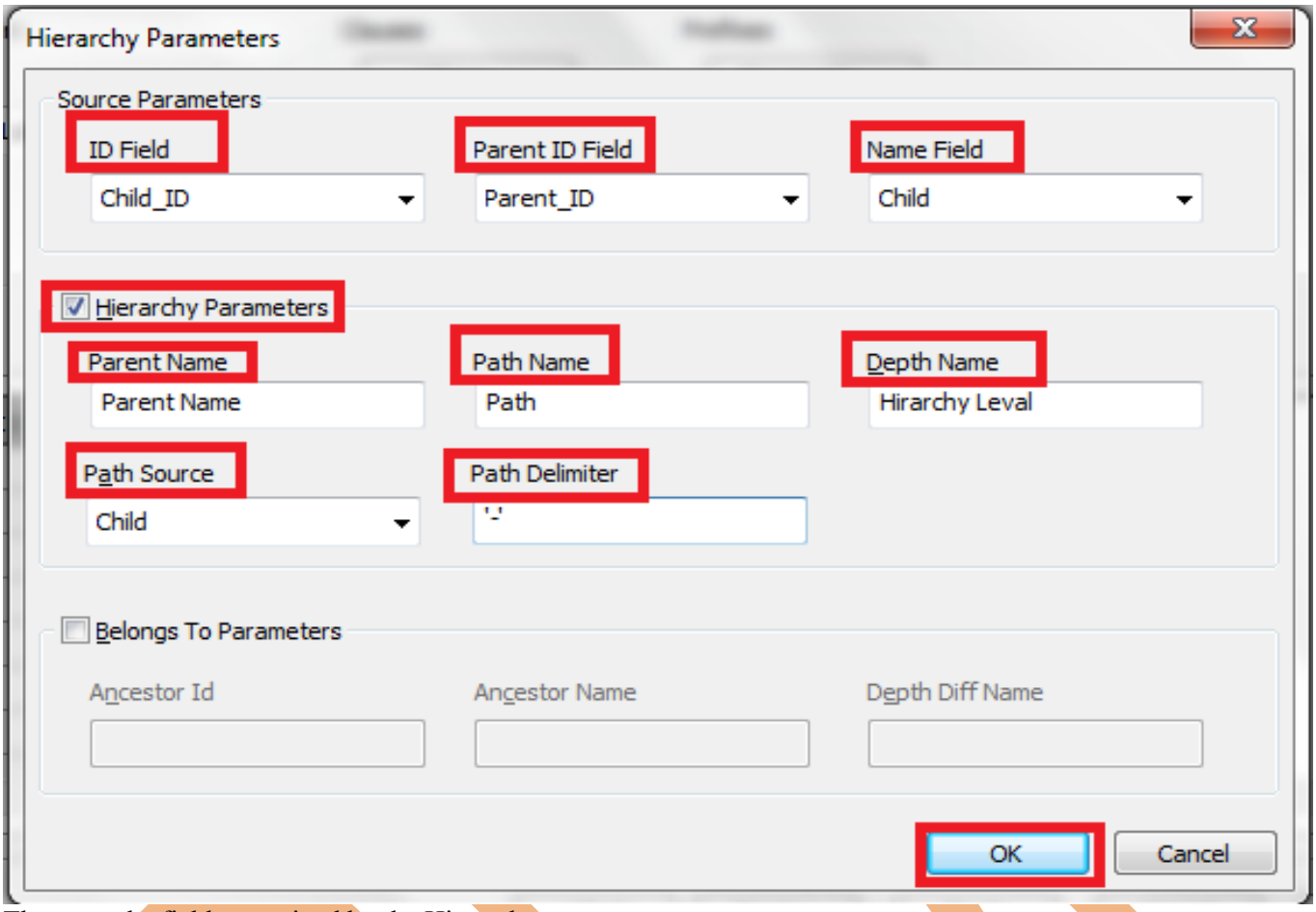


Step 4)Then the table wizard will open, check the Labels it should be Embedded labels. Then->Next



Step 5)Now click on the ->Next. Then choose the Hierarchy option.





These are the fields contained by the Hierarchy parameter.

- **ID Field:** This is the field that stores the IDs corresponding to the child nodes
- **Parent ID Field**: This is the field that stores the ID of the parent node.
- **Name field:** This is the field that stores the name of the child node.
- **Parent Name:** This is a string used to name a new field that will be created containing the names of the parent nodes.
- **Path Name:** This is a string used to name a new field that will be created containing the list of nodes from the top level to the corresponding node.
- **Depth Name:** This is the name to be assigned to a new field that will hold the number of levels for each expanded node.
- **Path Source:** This is the field from the source table that contains the value that should be used to populate the hierarchy path.
- **Path Delimiter:** This defines the string that should be used to separate the hierarchy values in the path.

->Then click ok

Step 6) Then the script will be added in the script editor. Here the HIERARCHY function comes default with the script this had come because we have used the hierarchy function via file wizard.

Main | Hirarchy

```
1  Directory;
2  HIERARCHY(Child_ID, Parent_ID, Child, ParentName, Child, Path, '-')
3  LOAD Parent_ID,
4       Child_ID,
5       Parent,
6       Child
7  FROM
8  [Hierarchy example.xls]
9  (biff, embedded labels, table is [Sheet1$]);
```

Give the HierarchyTable in the place of Hierarchy.

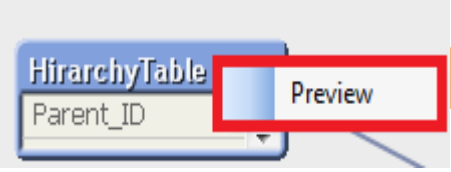
Hierarchy() function

The Hierarchy() takes these parameters.

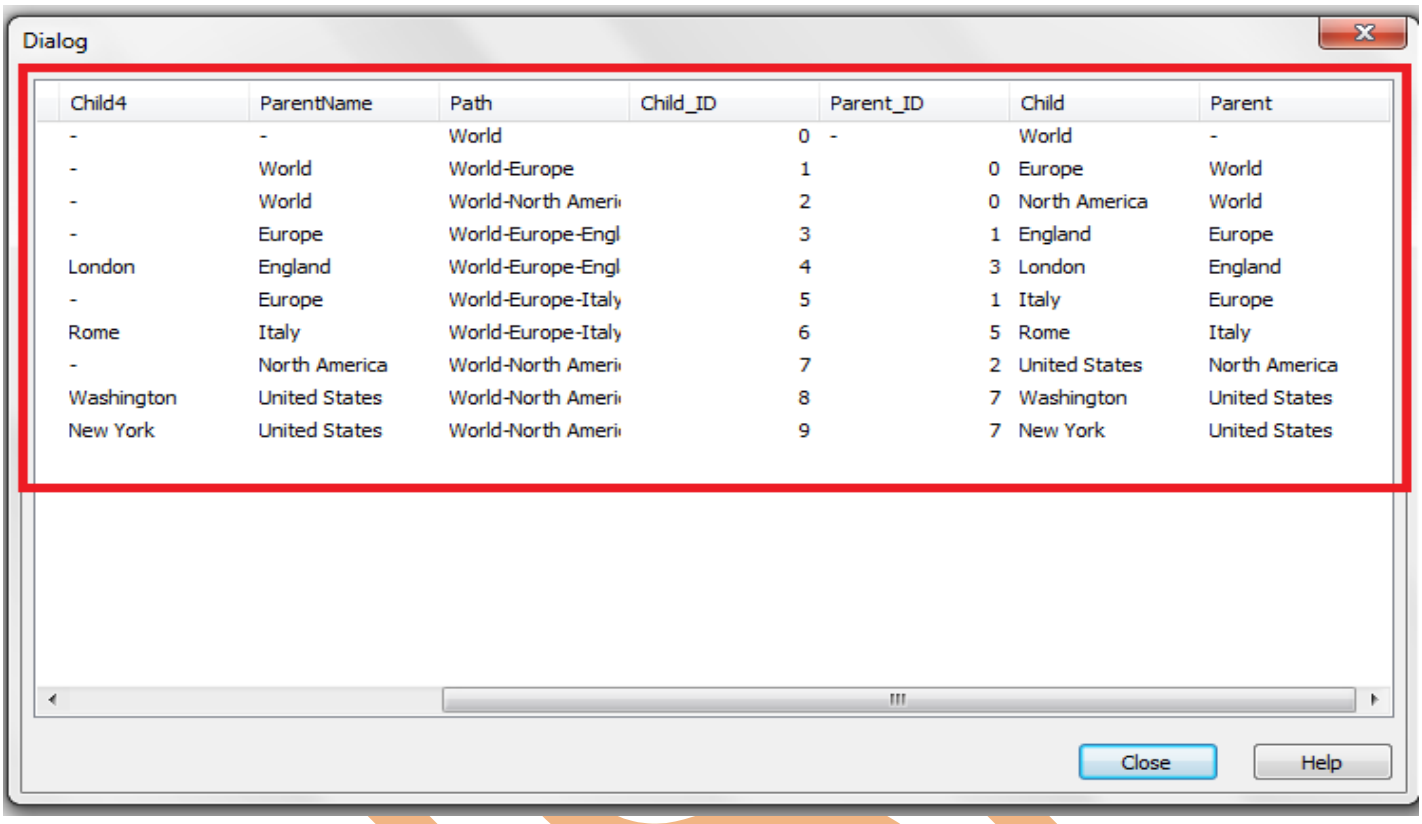
Hierarchy (NodeID, ParentID, NodeName, [ParentName], [PathSource], [PathName], [PathDelimiter], [Depth] **)**

- **NodeID** –This field takes Name of the field that contains the node id. This field must exist in the input table here NodeID is Child_ID.
- **ParentID** – This field takes name of the field that contains the node id of the parent node this is Parent_ID.
- **NodeName** – This field takes the name of the field that contains the name of the node as child is taken.
- **ParentName** – A string used to name the new ParentName field. If omitted, this field will not be created. If the field name has spaces or special characters then it needs to be in brackets or single quotes .
- **PathSource –** Name of the field that contains the name of the node used to build the node path. This parameter is optional. Here path source is child.
- **PathName –** A string used to name the new Path field, which contains the path from the root to the node. It is an Optional parameter. If omitted, this field will not be created. If the field name has spaces or special characters then it needs to be in brackets or single quotes. Path name is Path taken.
- **PathDelimiter** – A string used as delimiter in the new Path field. Optional parameter. If omitted, '/' will be used. If a single character is provided then the parameter value does not need to be in single quotes.
- **Depth** – A string used to name the new Depth field, which contains the depth of the node in the hierarchy.It is Optional parameter.

Step 7)Now Save the setting and reload Then go to the table viewer.



Here the table is created with the name of HirarchyTable. Then go to the Preview option.



If any of the optional parameters are left blank, the new field that uses the missing parameter will not be created when expanding the hierarchy the resulting expanded nodes table has one field for each hierarchy level, and one record for each node. Additionally, new fields have been created to show Path and Depth information.

If one node has multiple parents, the expanded nodes table will have several records for these nodes and the expanded nodes table will exclude any orphan nodes, that is, nodes that have no connection to a top-level node. Only nodes connected to the highest hierarchy level will be kept in the final table. Once we have a table with this structure, it is easy to use it on the frontend of the QlikView document, for example, within a pivot table or in a hierarchy dimension group.

We can implement this hierarchy in the form of list box by using the tree-view list-box.

The tree-view list-box

A tree-view list box is good at representing hierarchical levels, and provides an easy way to collapse/expand the hierarchy with the plus and minus icons to the left of each parent value.

With the resulting data from the above example, we will create a new list-box object by following these steps:

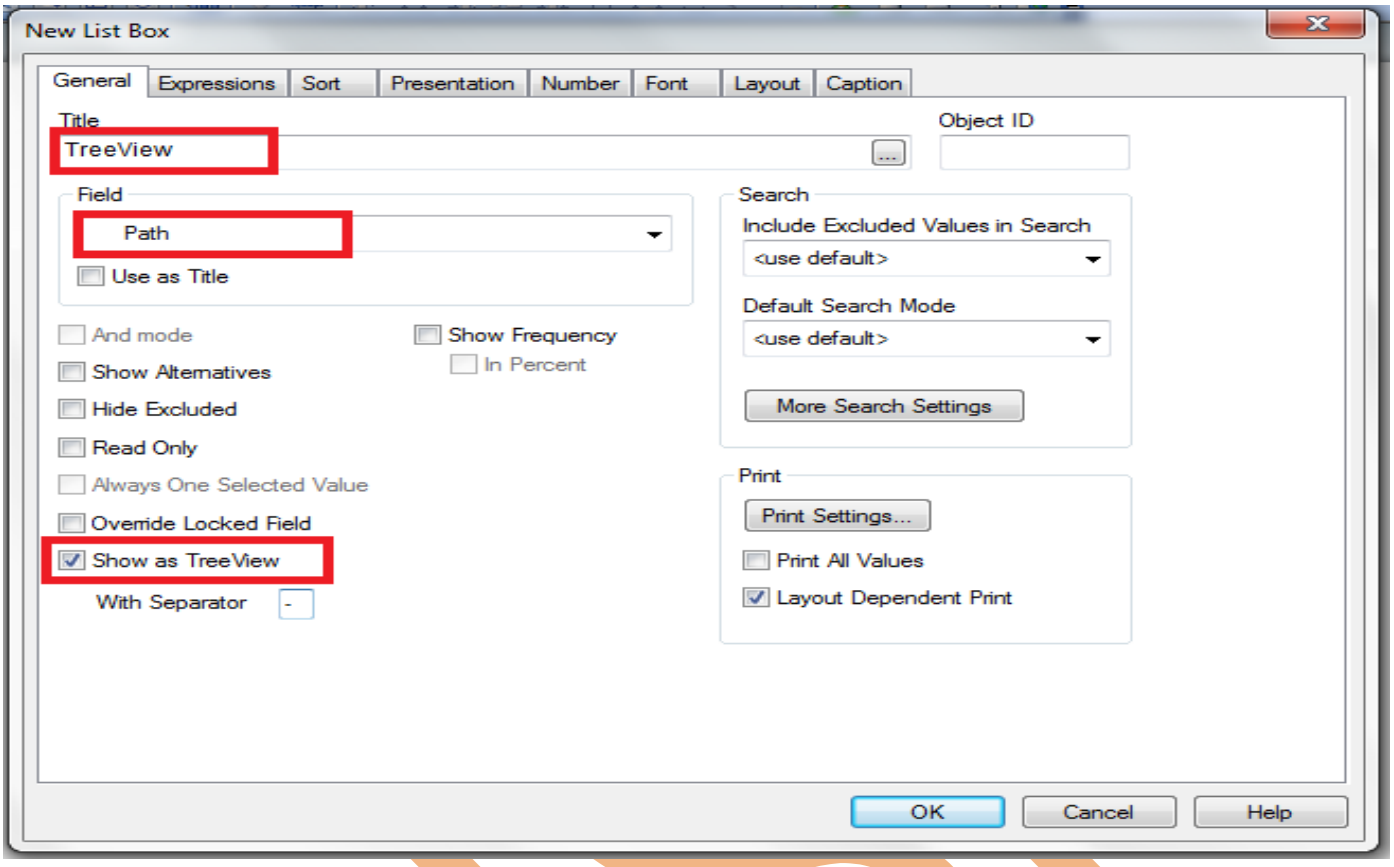
Step 1)Go to the sheet->New sheet object ->List box

Step 2)Then the window will open give the title name as TreeView

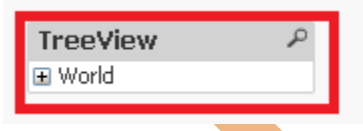
->Scroll down the field option and choose the path in the field option.

->Check the option Show as TreeView this will create the tree structure in the listbox

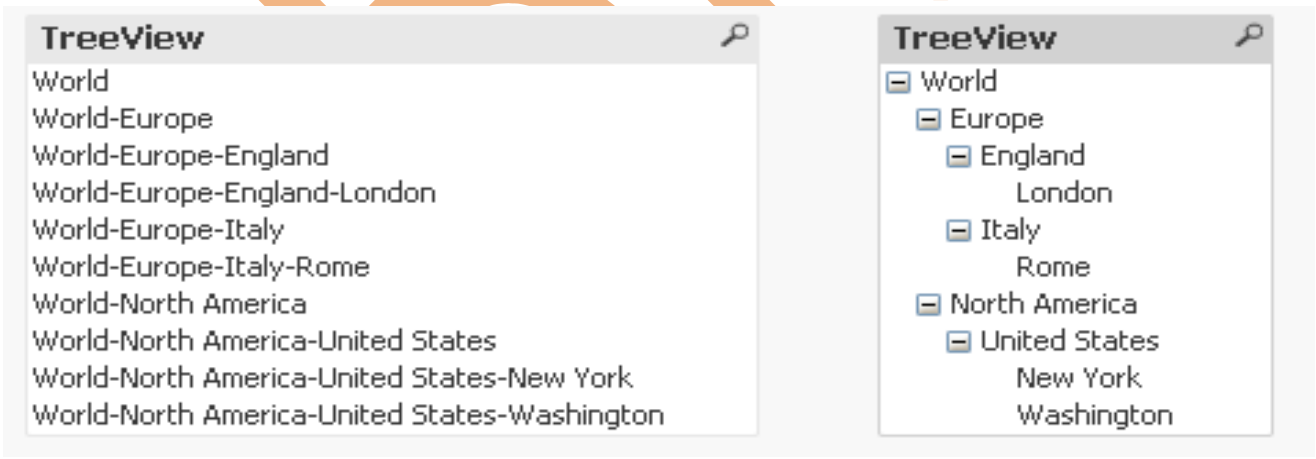
Then->ok



Step 3)Then the new list box will be created. Here we are taking advantage of the hierarchical path created in one of the field. The tree-view list box only requires a field with the hierarchical definition for a set of values, and with each hierarchical level separated by a specific character. The separation symbol can be any character.



After this explore the list box and create new list box without using the checked Show as TreeView option.



You can see here the difference between both the listboxes. See the first list box created having very complex type structure but in the second list box it has simplest structure.

HierarchyBelongsTo() function

The Hierarchy() can be used for creating a table that can be used to drive pivot tables or drill groups but it isn't appropriate for searching or selecting because the ancestors or a node are in separate fields. Here the HierarchyBelongsTo() load takes place. It also loads from the adjacency list table. It creates a table with all of the ancestors of a node in a single column. Here is the syntax for the HierarchyBelongsTo() load .

HierarchyBelongsTo(NodeID, ParentID, NodeName, AncestorID, AncestorName, [DepthDiff] **)**

NodeID – The name of the field that contains the node id. This field must exist in the input table It is named as Child_ID in the table HierarchyAncector.

ParentID – The name of the field that contains the node id of the parent node it is used as Parent_ID.

NodeName – The name of the field that contains the name of the node. It is the ParentName.

AncestorID – A string used to name the new ancestor id field, which contains the id of the ancestor node and the id is used as ParentName.

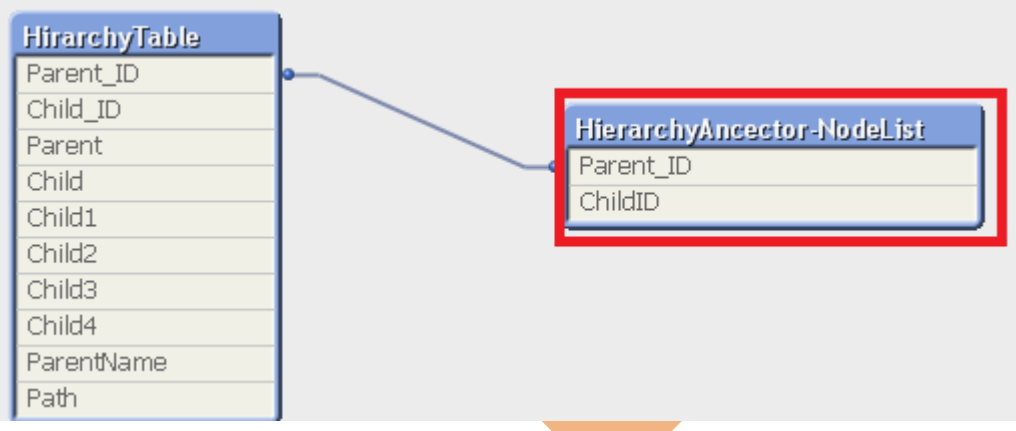
AncestorName – A string used to name the new ancestor field, which contains the name of the ancestor node.

DepthDiff – A string used to name the new DepthDiff field, which contains the depth of the node in the hierarchy relative the ancestor node.It is optional parameter. If omitted, this field will not be created.

```
HierarchyAncector:
HIERARCHYBELONGSTO(Child_ID, Parent_ID,ParentName, ParentName,AncectorName)
LOAD Parent_ID,
     Child_ID as ChildID
     Resident HirarchyTable;
```

And we have kept two fields in our HierarchyAncector table, These fields are taken which are Parent_ID and child_ID. we have renamed the child_ID field thats why the synthetic key is not created. We are fetching these records from the table HirarchyTable.

Then save the table and reload.



The whole script is used in the function as:

```
HirarchyTable:
HIERARCHY(Child_ID, Parent_ID, Child, ParentName, Child, Path, '-')
LOAD Parent_ID,
     Child_ID,
     Parent,
```

```
     Child
FROM
[Hierarchy example.xls]
(biff, embedded labels, table is [Sheet1$]);

HierarchyAncector:
HIERARCHYBELONGSTO(Child_ID, Parent_ID,ParentName,
ParentName,AncectorName)
LOAD Parent_ID,
     Child_ID as ChildID
     Resident HirarchyTable;
```