CREATING A COMPLEX LABEL USING GRIDFIELD WITHIN CODESOFT

CODESOFT offers the ability to create labels with multiple pages using the GridField object. This functionality is useful for businesses that print labels for cases, pick lists, bills of material, or purchase orders. The application can generate a list of products that are contained within a single shipment or container and the list can span multiple pages.
CREATING A COMPLEX LABEL

NOTE: The sample screenshots provided in this white paper use a sample database to reflect an inventory listing. However, please note that you can use whatever database connection you would like, or no database at all, depending upon your data entry method.

This white paper will detail two ways to create the table lookup that pulls the data onto the label. The first way uses a Key Field, which allows you to pull a record from a database based on a unique value that represents that record, like an Order Number or Serial Number. This is typically used when you want to pull one or more specific records onto your label. The second way does not use a Key Field, but instead pulls all records from the database or table onto the label. This is typically used when you want to display all the data from the datasource (such as a Purchase Order) on the label.

CREATING A TABLE LOOKUP FOR COMPLEX LABEL SETUP USING A KEY FIELD

STEP 1: CREATING A KEY FIELD
If the data on your label will be queried from a Key Field, you must first set up a lookup variable.

1. Go to Data Sources > When Printed > Add.
2. Enter a variable name (for example, OrderNumber).
3. If desired, enter sample data in the Value field (this will help confirm/deny the Table Lookup is working properly).
4. Click OK.

STEP 2: CREATING A DATABASE QUERY
Next, create a connection to your database and specify the fields you wish to use on your label.

1. Go to Data Sources > Table Lookup > Add.
2. Enter a lookup name relevant to the field you a querying (for example, Product Name Table).
3. Select or create an OLEDB connection (for example, your database connection).
4. Select SQL as the creation mode.
5. Enter an existing SQL query or create a new query using SQL Query Builder.
An example of a SQL query statement would look something like this:

```
SELECT [ID], [Product Name], [Unit Price], [Qty], [Discount], [Price] FROM [Header] WHERE [ID] = APPLICATION.DOCUMENT.OrderNumber
```

This method is for pulling values of data based upon a lookup field using a **Key Value**.

**CREATING A TABLE LOOKUP FOR COMPLEX LABEL SETUP USING NO KEY FIELD**

If the data on your label will be queried from a database connection, without using a **Key Field**, you must create your **Table Lookup** the following way.

**STEP 1: CREATING A DATABASE QUERY**

1. Go to **Data Sources > Table Lookup > Add**.
2. Enter a lookup name relevant to the data you are querying (for example, **Item Table**).
3. Select or create an OLEDB connection (for example, your database connection)
4. Select **SQL** as the creation mode.
5. Enter an existing SQL query or create a new query using SQL Query Builder.

An example of SQL query statement would look something like this;

```
SELECT Header.[Product Name], Header.[Unit Price],
Header.[Qty],
Header.[Discount],
Header.[Price] FROM Header
```

This method pulls all data from a database connection created within CODESOFT.

As you can see in the example above, the query is structured slightly differently than the previous connection using the **Key Field**. In this format, we are pulling every record from the current database named **GridField DB** and using the “database name.column” SQL statement structure.
As an example: if you had a database connection with only one (1) field/column called Serial Number and you imported new information from an SQL Server/ERP Database that also only had only one (1) field/column named Serial Number, you could effectively overwrite the current data with new data without changing your database connection. CODESOFT can automatically connect everything again if the column name(s) match. If they do not match however, you will need to manually direct the objects to their proper datasources once more. This would be common in an environment that generates Purchase Orders on a regular basis and does not re-use the data once it’s been printed.

In order to display only one (1) record at a time, instead of all available records, we need to format this variable properly. This can be done within the Table Lookup Properties. Right-click the object on the template and select Data Source Properties, or go to the Data Sources tab on the right side of your program and right-click the datasource and select Properties.

When using the Table Lookup function in CODESOFT, you can designate Field and Record separators to whichever character(s) you wish. This supersedes the format of the database itself. In this example the format is set up as follows:

1. **Field Separator** is designated by a pipe “|” (this divides the fields into respective columns)
2. **Record Separator** is designated by a hat “^” (this divides the records into respective rows)
3. **Relevant Records** is set to zero (0) as we do not want to define how many records we see. We want to see all of them, therefore we do not wish to designate a value to this field.

We can see how this organizes the data by the Preview section below:

**Product Name|Unit Price|Qty|Discount|Price^**

Mascarpone Fabioli|32.00|1|0|32.00^
# CREATING THE GRIDFIELD WITHIN THE CODESOFT LABEL TEMPLATE

**Example:** CODESOFT GridField Template (.lab)

<table>
<thead>
<tr>
<th>Product Name</th>
<th>Unit Price</th>
<th>Qty</th>
<th>Discount</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mascarpone Fabioli</td>
<td>32.00</td>
<td>1</td>
<td>0</td>
<td>32.00</td>
</tr>
<tr>
<td>Aniseed Syrup</td>
<td>10.00</td>
<td>4</td>
<td>0</td>
<td>40.00</td>
</tr>
<tr>
<td>Chef Anton’s Cajun Seasoning</td>
<td>22.00</td>
<td>1</td>
<td>0</td>
<td>22.00</td>
</tr>
<tr>
<td>Grandma’s Boysenberry Spread</td>
<td>25.00</td>
<td>1</td>
<td>0.02</td>
<td>24.50</td>
</tr>
<tr>
<td>Uncle Bob’s Organic Dried Pears</td>
<td>30.00</td>
<td>1</td>
<td>0.05</td>
<td>28.50</td>
</tr>
<tr>
<td>Northwoods Cranberry Sauce</td>
<td>40.00</td>
<td>2</td>
<td>0.1</td>
<td>72.00</td>
</tr>
<tr>
<td>Ikura</td>
<td>31.00</td>
<td>1</td>
<td>0</td>
<td>31.00</td>
</tr>
<tr>
<td>Queso Manchego La Pastora</td>
<td>38.00</td>
<td>2</td>
<td>0.05</td>
<td>72.20</td>
</tr>
<tr>
<td>Konbu</td>
<td>6.00</td>
<td>4</td>
<td>0</td>
<td>24.00</td>
</tr>
<tr>
<td>Tofu</td>
<td>23.00</td>
<td>1</td>
<td>0.03</td>
<td>22.55</td>
</tr>
</tbody>
</table>
STEP 1: CREATE GRIDFIELD OBJECT

1. Select the **Advanced Object Tool** within the **Creation Toolbar** on the left side of the program.
2. Select **GridField**
3. Place your **GridField Object** on the label template.
4. Expand/Collapse the size of the object to your desired width and height.

Once you have placed your GridField object on the label template, it will automatically open the **Properties** window for you to alter the properties to your specifications.

**NOTE:** Although **GridField** itself is a datasource, it does not appear in the datasources tab or within the datasources panel on the right-hand side of the program. The nature of the **GridField** is to have a dynamic interface that allows you to insert/remove datasources and variables created within the label template that affect the size/shape/appearance of the **Cells**. **Cells** are furcated sections of a **Grid**.
Click the **More Settings** button to view and edit the components of the GridField object.

In the above example, we can see the **Body** of the **GridField** contains one (1) **Grid**.

If we expand the contents of the **Grid** itself, we can see that it is comprised of six (6) **Cells**.

---

**STEP 2: FORMATTING THE GRID(S)**

There are nearly infinite formats one could construct to achieve their goal, and to specify every possible format in this white paper wouldn’t be feasible. However, we can use the sample format of our template to describe how to alter the format for your specific needs.

In this example, four (4) separate GridField objects make up the entirety of the label template. You may not need to create this many. You may have all variables tied to one (1) GridField object if you wish. However, certain functions are prioritized to whichever Grid they are tied to, such as Headers and Footers. In this respect, if you have a Header or Footer within a GridField object, it will only pertain to that GridField object you are currently in and no other objects.
The **Data Source** for this Grid is specified as a Formula. You can choose from multiple datasource types from this dropdown menu within the **Properties** of the Grid. This is very important, as this will determine if the values of the Grid/Cells are altered by changing a variable or if they are altered within the external datasource you’ve selected. Once you’ve defined where your data will be coming from, you can open the **More Settings** option.

*NOTE: You may only select one (1) datasource at any given time for a GridField object in this manner.*
As we can see from the example above, the Grid is comprised of several components:

1. Header
2. Body
3. Footer
4. Streams
5. Fonts
6. Pictures
7. Barcodes
8. Scripts

This allows you to build the Grid to your specifications. In this example, the Grid is labeled **Page Header** and it contains two (2) Cells. If we highlight the Grid within the Body named **Page Header** the options list on the right highlighted in the blue box will become selectable. From here we can select the **Edit** button to begin altering which data is displayed, as well as how it is displayed.
We can alter several details about the way the data is manipulated and displayed. This Grid is split into two (2) equal halves to display two (2) Cells, noted by the **Columns** section near the top of this window. You can add a **Frame Color** if you wish to have the borders of the **Cell** altered. Additionally, you can adjust the height of the Grid/Cells. By default, the **Row Height** is fixed, and must be manually selected to change the values. Much like the previous Table Lookup section without the Key Field, you can choose how the data is broken up with the Column/Line separators. The **Frame** and **Spacing** buttons affect each individual **Cell** within each **Grid**. If you wish to alter how the Grid behaves, we can open the **Default Cell** properties.
Within the first Cell we've selected the Grid to interpret the first Cell as an Image or Picture. In the second highlighted section near the bottom, this is where the “value” would be entered. The image in the example is tied to the Address Formula 2 that we saw in the first example given.

**NOTE:** Please remember to include the name of the file AND the extension when entering Image values.

This will now display the Teklynx.jpg image within the first Cell.

**IMPORTANT NOTE:** Please ensure that your Images, Text Files, Databases, and Serial Files are located within their respective Default Folder according to the Configuration section of the core CODESOFT program. You can navigate to this section by going to Tools > Configuration > Default Folders within CODESOFT.
Within the second Cell, we can see that it is to be interpreted as Text. If we continue the same path as before, you can also see that we’ve set our value as variable “Address Formula 2”. Although we’ve set this value within the Main Grid Properties, this must also be entered as the value to the Cell.

**NOTE:** Please note that there is a Carriage Return prior to the entry of the value given. This affects how it is displayed, by creating an actual Carriage Return within the Cell itself. It behaves normally as it would within a variable.

The result of the properties can be seen in the above output.
Moving on to the second **Grid**, we can see that no value for the **Data Source** has been selected. This will allow the variables full control over the values given within the **Cell(s) or Grid(s)**. In this example, we have two (2) Grids with different Fonts displayed in the above Preview. If we navigate to the **Properties** of this Grid we can see what is manipulating the data.

**NOTE:** You do not have to create Cells within a Grid to display data. Cells are another layer within a Grid to allow further manipulation of the data to a specific area of the Grid. In this example, we only have two (2) Grids that we are using to display the data we see. Therefore, whatever Properties we adjust within the settings will be applied to the Grid as a whole.

In the example to the right, we can see that the second Grid is comprised of two (2) Grids and a Font component. This will allow the user to define a particular Grid/Cell font type. It is important to
note that you can create multiple Fonts for the Grid, but you can generally only select one Font per Cell/Grid.

**NOTE:** Please keep in mind that if you choose a Font not normally found within the Windows OS, meaning you’ve installed it manually, you will need to choose the option within the *Interpretation* as *Rich Text Format*. This forces graphical emulation, so this will not work as a *Printer Resident Font* and may cause slower printing.

In the example to the right, we can see how we’ve defined the first Grid in the second GridField object. In the values, you can see the *Column Separator* pipe between the two Grids. This allows you to essentially “tab” over to the next Grid in the same row. In this example, in the first Grid half will be the “Shipping Address” and in the adjacent Grid half will be “Order”.

Highlighted in red below, you can see how the Grid is split into two (2) halves and displayed.
In the second Grid within this GridField object, we will add additional data to reflect the bottom half of the Grid.

**NOTE:** Please take note to the spaces prior to the values given within the data entry section. If you wish there to be spaces reflected in the output, you will need to enter them here. This works similarly to the Carriage Return section in the first GridField object.
Once you have set up the Properties of the Grid, you can alter the appearance of the overall GridField object through the **Shape Settings**. This can be done throughout all GridField objects independently. This will alter the Properties of the GridField itself, but it does not extend to the individual Grids/Cells within the GridField object.

In the **Default Cell Properties** we can select an array of options that can alter the way the Grid/Cell performs. These settings can be changed whenever you wish, but the variables/objects should be created properly first. This means any datasources you wish to include in the GridField object should be created before the GridField object.

In the second GridField object we have created, we can alter the Font to be larger and bold, as we see displayed in the first Grid of the second GridField object.
Within each Grid/Cell, there is an option to select the Default Cell Properties. When you select this option, it brings you to the window shown to the left. From here you can select multiple options that affect datasources such as:

1. **Interpret as**: How the Cell is interpreted. Options include Text, Picture, Barcode, Rich Text Format.
2. **Picture**: If there is one to be selected from a variable you have created/attached.
3. **Picture alignment**: The alignment of the Picture (if any).
4. **Font**: Will be selectable once you have added a Font to the Font section of the components.
5. **Text alignment**: Much like Microsoft Word’s text justification options.
6. **Orientation**: Rotates the data to the desired direction.

Once you have set the Properties to achieve the data presentation you wish, we can move on to the third GridField object we see displayed in our sample.

The third GridField object we need to create contains our Inventory listing for this example. As you can see in the example, we do not have a Data Source selected.
As we know already what uniform Properties we wish to apply to the GridField object, we can apply them quickly.

As shown above, we have added a **Header** to this GridField object. This allows you to have a static Grid/Cell composition that will print with every label printed. This will be shown in the output in the final example.
Highlighted above, we see the **Header** displayed. This will be displayed with every subsequent record shown below for every label printed. This behavior extends to the **Footer** as well.

The setup for the **Header** section is in the example below.

**NOTE:** Please take note of the five (5) columns we have created for each Field/Column within the Table Lookup from which we are pulling our data. If your output looks like several columns mashed together, it could be because the number of columns within the GridField object are not equal to the number of columns you wish to display from the database.

**NOTE:** Remember to use your **Column Separator** to divide your Fields/Columns within this window to organize your data appropriately to match the number of Cells you have created within the Grid Properties.

**NOTE:** In addition to using the **Column Separator**, you want to ensure that you are adding your Grid Variables to the highlighted section in red in the same order that the data is appearing from your external datasource or entry method.

For example: Inventory PN | Inventory Unit Price | Inventory Qty | Inventory Discount | Inventory Price

<table>
<thead>
<tr>
<th>Product Name</th>
<th>Unit Price</th>
<th>Qty</th>
<th>Discount</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mascarpone Fabioli</td>
<td>32.00</td>
<td>1</td>
<td>0</td>
<td>32.00</td>
</tr>
<tr>
<td>Aniseed Syrup</td>
<td>10.00</td>
<td>4</td>
<td>0</td>
<td>40.00</td>
</tr>
<tr>
<td>Chef Anton’s Cajun Seasoning</td>
<td>22.00</td>
<td>1</td>
<td>0</td>
<td>22.00</td>
</tr>
<tr>
<td>Grandma’s Boysenberry Spread</td>
<td>25.00</td>
<td>1</td>
<td>0.02</td>
<td>24.50</td>
</tr>
</tbody>
</table>
Much like the first GridField object, now that we have set up our format for the Grid/Cells we can import our datasource into the Body of the GridField object.

It is highly recommended to use the same number of **Columns** you have designated within the **Header** portion of the GridField object, or this could cause potential issues.

In the example above, we have selected our Table Lookup without a Key Field, named “Item Table”. This was created prior to generating the GridField objects to ensure the data was being pulled properly into CODESOFT.

**NOTE:** Please note that you should ensure to change your values for the **Column Separator** and **Line Separator** to match what you have designated within the **Data Format** section of your Table Lookup without a Key Field.
STEP 3: CREATING MULTIPLE CELLS WITH INTERPRETATION VARIANCES

In the example given above, we can see that we have created our Grid, added a Font, and created our Barcode within the GridField object. For this GridField object however, we will be using two (2) separate cells as they will need to be defined in separate formats; Text and Barcode.

NOTE: Please keep in mind that when mixing different object types, you will need to create Cells within the Grid in order to display them properly. Below are the examples of the datasource types as explained previously within the Default Cell Properties section.

1. Text
2. Image
3. Barcode
4. Rich Text Format

When you create a Cell within the Grid, you have the option to change the Interpretation of your data to display how you wish. This allows for a flexible environment to manipulate your data in whatever way you require. If we look at the examples below, we can see just how we get our Grid to format our output as we saw in the first example of this Grid section.
To construct a **Cell** we need to highlight the **Grid** we have created within the **Body** and select **New**. Once we have done so, we will be able to manipulate the data as we see fit.

**NOTE:** Please keep in mind that much like the previous section dealing with the database connection, if you wish to have this data pulled from a database for the barcode values, you must create the same number of Cells as there are Columns within the database to properly display the values within the Grid. Additionally, it worth noting that the number of Cells created should match the number of sections you wish to display. In this example, we have two (2) objects we wish to display, therefore we must create two (2) Cells.

**STEP 4: CREATING A PAGE NUMBERING OBJECT/FORMULA**

In the bottom section of the label template, we can see that there are two (2) objects that have different types of datasources. In the example shown, one (1) is a **Text Object**, while the other is a **Barcode Object**. To recreate these objects within the Grid, please use the instructions below.
1. Navigate to Data Sources > Formulas > Add.
2. Enter a formula Name relevant to the variable (E.G. PageNumbering).
3. Use the following statement/expression using our Control Variables:

   “Page:” & “ ” & {MLOCURPAGE} & “/” & {MLOMAXQTY}

4. Click OK.

This will allow you to have a “box counter” based upon the current page being printed and the maximum number of pages being printed. It will automatically increment until it has reached its maximum page count. If you print more than the total number of pages being printed, as in the example given above, it will return to the start value of the counter and increment accordingly.

The variables we are using in this Formula are Control Variables. These are datasources that override all other functions of their category. This can be denoted by the “@” symbol prefix to the datasource.

EXAMPLE: If you were to enter the value of six (6) into the “number of labels to be printed” section within the print screen, you should receive two full copies (3 pages each) of this template we’ve created today.

We have gone over how to incorporate advanced Formulas using Control Variables and how to create and use a barcode within the GridField objects. Next, we will look at the current settings we have for this object.

NOTE: Please keep in mind that dependent upon the height your GridField object, the barcode may not appear immediately. It is suggested that you choose a fixed height for the barcode to appear, to allow it to be properly created within the GridField object.

EXAMPLE: If my GridField object is one (1) inch in height and my barcode has a height of .75 inches, more than likely you will not see the barcode appear within the Grid or Cell. The reason being is that barcodes need specific dimensions in order to work properly. It is recommended to leave .25 inches worth of “white space” or blank area on the left and right sides of the barcode for it to be scanned properly.
NOTE: This applies to most linear and a few 2D barcodes. Omnidirectional barcodes, this generally does not apply to however, it would be beneficial to leave some “white space” for the barcode to be scanned properly if you have a thick border surrounding the Grid/Cell.

In the Cell properties to the left, we can see that we have selected the Cell to interpret the data as Text, chosen a Font, and added our datasource (Page Numbering formula) to the value section of our Cell. In this Cell, we will also be using the Margins properties to show how to manipulate the display of the data even further.

NOTE: Please keep in mind that using Margins will push the data in the desired opposite direction of the margin value and Borders will affect the height and width of the visible area within the individual Cell. Additionally, it is important to note that when using the Margins properties, it will collapse or expand the size of the datasource you have displayed within the Cell.

EXAMPLE: If I were to have a 1x1 Grid/Cell where we would like a .25 margin around the entire Cell, this would only allow me .5 inches worth of Cell that will have the ability to display the values of the datasource. This will force the value to become smaller or larger comparatively.
STEP 5: CREATING A BARCODE

Now let us look at the Barcode creation within the next Cell.

![Barcode Properties Screen]

First, we must create our datasource within CODESOFT to apply it to the Cell we are about to format. We will create the simple formula shown above.

Then, we must define the Symbology.

Once you have selected the Barcodes component, click New. This will open your Barcode Properties screen.
For this barcode, we will select the **Symbology** PDF417 and set the **Fixed Height** to 0.500.

**NOTE:** Please note the “Fixed Height” section of the **Barcode Properties**. This will ensure that it will show up properly within the Cell. However, in order to fixate it within the Cell as we have showing in the preview of the fourth Grid, we will need to manipulate the **Margins Properties** as well.

The **Margins Properties** will be located within the main Cell’s Properties located in the previous example.
Once the barcode is defined, we can now create our **Barcode Cell**.

In the example given above, we can see that we have set the **Interpretation** to “Barcode”, selected the proper barcode **Symbology**, and added the datasource value within the proper section. Once this has been completed, we can now begin the final stages of testing and approving the GridField label template.

**STEP 6: FINALIZATION AND TESTING**

Now that you have created the label template in the desired manner you wish, it is now time to test. If you have followed this guide closely, you should have been able to either recreate what we have displayed here, or a version of your own. If we initiate a few test printouts, we should be able to quickly discern if we have configured the GridField objects properly.
## Shipping Address

Teklynx America  
501 W. Northshore Drive, Suite H-380  
Glendale, WI 53217

## Order

ORDER ID 123456  
DATE: 05/10/2018

<table>
<thead>
<tr>
<th>Product Name</th>
<th>Unit Price</th>
<th>Qty</th>
<th>Discount</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mascarpone Fabioli</td>
<td>32.00</td>
<td>1</td>
<td>0</td>
<td>32.00</td>
</tr>
<tr>
<td>Aniseed Syrup</td>
<td>10.00</td>
<td>4</td>
<td>0</td>
<td>40.00</td>
</tr>
<tr>
<td>Chef Anton’s Cajun Seasoning</td>
<td>22.00</td>
<td>1</td>
<td>0</td>
<td>22.00</td>
</tr>
<tr>
<td>Grandma’s Boysenberry Spread</td>
<td>25.00</td>
<td>1</td>
<td>0.02</td>
<td>24.50</td>
</tr>
<tr>
<td>Uncle Bob’s Organic Dried Pears</td>
<td>30.00</td>
<td>1</td>
<td>0.05</td>
<td>28.50</td>
</tr>
<tr>
<td>Northwoods Cranberry Sauce</td>
<td>40.00</td>
<td>2</td>
<td>0.1</td>
<td>72.00</td>
</tr>
<tr>
<td>Ikura</td>
<td>31.00</td>
<td>1</td>
<td>0</td>
<td>31.00</td>
</tr>
<tr>
<td>Queso Manchego La Pastora</td>
<td>38.00</td>
<td>2</td>
<td>0.05</td>
<td>72.20</td>
</tr>
<tr>
<td>Konbu</td>
<td>6.00</td>
<td>4</td>
<td>0</td>
<td>24.00</td>
</tr>
<tr>
<td>Tofu</td>
<td>23.00</td>
<td>1</td>
<td>0.03</td>
<td>22.55</td>
</tr>
<tr>
<td>Product Name</td>
<td>Unit Price</td>
<td>Qty</td>
<td>Discount</td>
<td>Price</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------</td>
<td>-----</td>
<td>----------</td>
<td>--------</td>
</tr>
<tr>
<td>Pavlova</td>
<td>17.00</td>
<td>2</td>
<td>0.03</td>
<td>33.85</td>
</tr>
<tr>
<td>Chang</td>
<td>19.00</td>
<td>24</td>
<td>0.2</td>
<td>364.80</td>
</tr>
<tr>
<td>Tunnbrod</td>
<td>9.00</td>
<td>2</td>
<td>0</td>
<td>16.00</td>
</tr>
<tr>
<td>Original Frankfurter</td>
<td>13.00</td>
<td>2</td>
<td>0</td>
<td>28.00</td>
</tr>
<tr>
<td>Chartreuse Verle</td>
<td>18.00</td>
<td>2</td>
<td>0.05</td>
<td>34.20</td>
</tr>
<tr>
<td>Clam Chowder</td>
<td>10.00</td>
<td>3</td>
<td>0</td>
<td>28.95</td>
</tr>
<tr>
<td>Spaghetti</td>
<td>12.00</td>
<td>3</td>
<td>0.02</td>
<td>35.28</td>
</tr>
<tr>
<td>Filo Mix</td>
<td>7.00</td>
<td>2</td>
<td>0</td>
<td>14.00</td>
</tr>
<tr>
<td>Pate Chinois</td>
<td>24.00</td>
<td>2</td>
<td>0</td>
<td>48.00</td>
</tr>
<tr>
<td>Pierrot</td>
<td>34.00</td>
<td>2</td>
<td>0.06</td>
<td>48.00</td>
</tr>
<tr>
<td>Summer Sausage</td>
<td>33.00</td>
<td>1</td>
<td>0</td>
<td>17.00</td>
</tr>
<tr>
<td>Baklava</td>
<td>15.00</td>
<td>2</td>
<td>0.01</td>
<td>29.70</td>
</tr>
<tr>
<td>Product Name</td>
<td>Unit Price</td>
<td>Qty</td>
<td>Discount</td>
<td>Price</td>
</tr>
<tr>
<td>------------------</td>
<td>------------</td>
<td>-----</td>
<td>----------</td>
<td>--------</td>
</tr>
<tr>
<td>Pizza (Pepperoni)</td>
<td>8.00</td>
<td>4</td>
<td>0</td>
<td>32.00</td>
</tr>
<tr>
<td>Marmalade</td>
<td>81.00</td>
<td>1</td>
<td>0.04</td>
<td>77.76</td>
</tr>
</tbody>
</table>
CONCLUSION
GridField is a powerful tool that provides a vast number of options for label template configuration. If you have questions or run into issues, please contact TEKLYNX in your region.

To download the latest version of CODESOFT label design software, visit the TEKLYNX website.