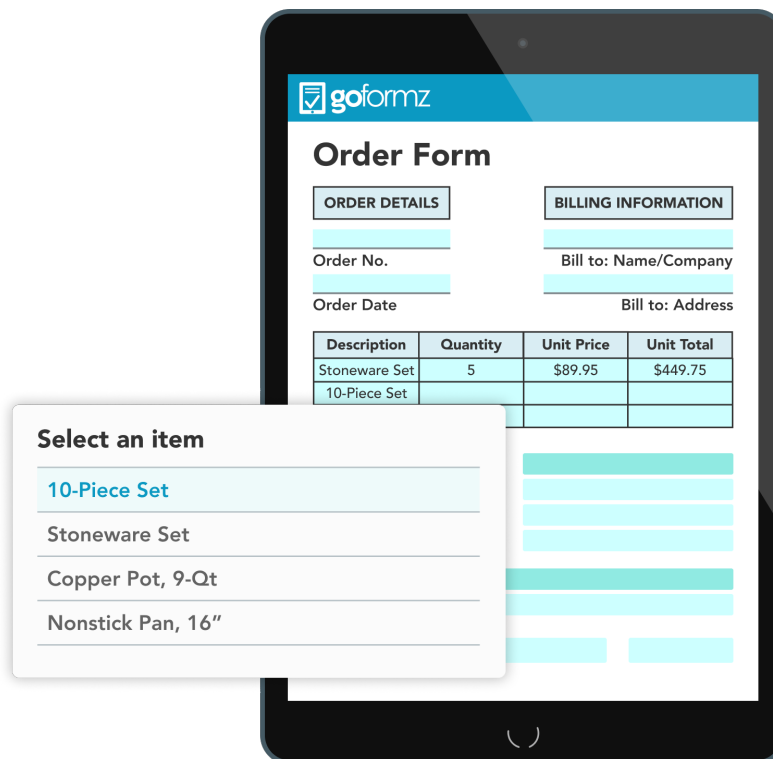




eBook

How to Use DataSources in Your Digital Form Template



www.goformz.com

Table of Contents

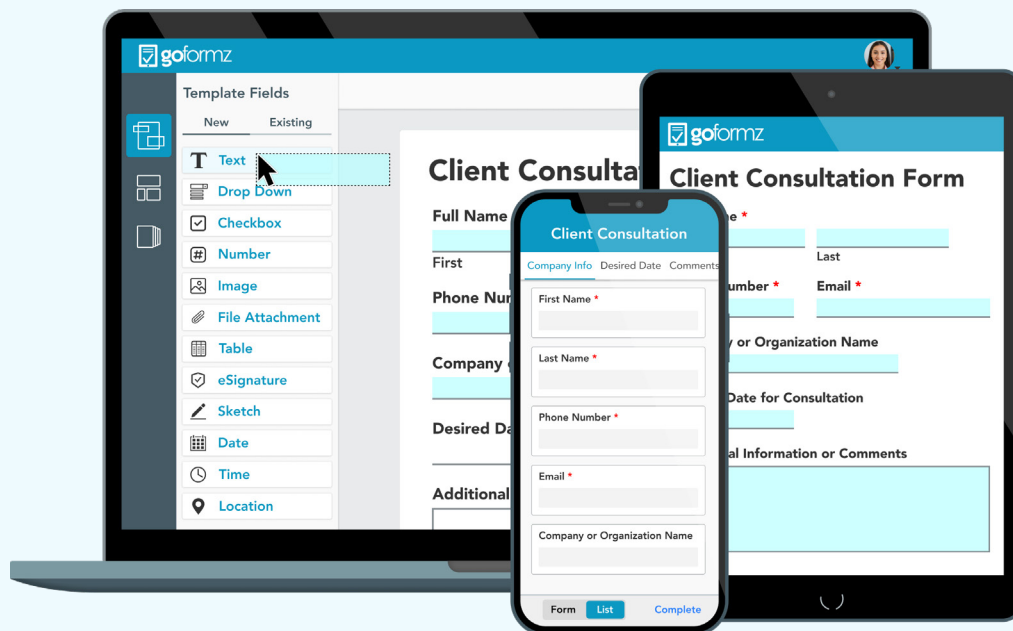
03	Introduction: Why use DataSources in your digital forms?
04	What are DataSources?
05	How to utilize DataSources within your digital form Template
06	How to create a DataSource
07	How to add DataSources in your digital form Template
10	3 DataSource Use Cases
11	Use Case One: Auto-Populating Fields
14	Use Case Two: Option Filtering
17	Use Case Three: Auto-Populating and Option Filtering
19	Frequently Asked Questions
20	Additional Resources

Introduction: Why use DataSources in your digital forms?

Digital forms are an incomparable tool for the modern workplace. Rather than grapple with tedious, manual tasks (like rekeying form data into spreadsheets and hunting through filing cabinets for the right document) teams now leverage digital forms to accelerate and simplify their data capture and processing. These digital efficiencies save teams precious time and budget and empower businesses to focus their manpower and resources on revenue-driving activities and customer care.

DataSources are among the most powerful and useful digital form features and play a major role in the most compelling GoFormz customer stories. Paperwork can be cumbersome, involving time-consuming tasks like hand-delivering, redundant data entry, correcting inaccurate information, and hand-filing forms, all of which slow daily operations and introduce opportunities for human error. *DataSources* directly address and resolve these challenges by accelerating and refining data entry and protecting the integrity of data collection. *DataSources* also work with offline forms, guaranteeing your functionality extends to the most isolated locations while still pulling data from your connected systems.

In this eBook, we'll explore how *DataSources* work, best practices for implementation, and popular use cases your team can leverage today.



What are DataSources?

Think of a GoFormz *DataSource* as a simplified database table. *DataSources* are made up of Rows and Columns. The rows are in a database table, while the Columns are equivalent to the headers.

Example Data Source - Customers

All	Key <i>IF</i>	Address	City	State	Zip
<input type="checkbox"/>	B V General Inc.	421 East Mission Avenue	Escondido	CA	92025-1909
<input type="checkbox"/>	Beverly Healthcare	421 East Mission Avenue	Escondido	CA	92025-1909
<input type="checkbox"/>	Birch Ptruck Convalescent Cntr	751 Medical Center Court	Chula Vista	CA	91911-6617
<input type="checkbox"/>	Casa Palmera Care Center	14750 El Camino Real	Del Mar	CA	92014-4204
<input type="checkbox"/>	Chase Care Center	1201 South Orange Avenue	El Cajon	CA	92020-7521
<input type="checkbox"/>	Community Health Group	740 Bay Boulevard	Chula Vista	CA	91910-5299
<input type="checkbox"/>	Country Hills Health Care Inc.	1580 Broadway	El Cajon	CA	92021-5124

← Columns

← Rows

These Rows and Columns collectively form a *DataSource*, which can be referenced by form fields to automatically populate corresponding form fields and dynamically filter options presented by a corresponding form field.

When working with *DataSources*, it is crucial to grasp two key concepts:

- » **Key Columns:** Every *DataSource* must include a unique key column, ensuring that no two records share the same key column value. The Key Column ensures that the correct *DataSource* record is referenced when populating fields with corresponding information.
- » **Indexing:** Indexing refers to configuring a specific column within a *DataSource* to be referenced for data retrieval. For example, the Key Column is always indexed, and you can index up to five additional columns per *DataSource*.

These two components are what make Auto-Populating and Option Filtering data possible, which we will cover in the next section.

How to utilize DataSources within your digital form Template

There are two ways GoFormz Template Builders can leverage *DataSources* within their digital forms:

- 1. Auto-Populating Fields:** DataSources can be used to automatically populate fields with information, based on the value of another field. For example, if a user selects a business location from a drop-down menu, a DataSource could be used to instantly fill the address and phone number field with the selected location's information.

This functionality can also be used when using Barcode fields. For example, if a user completing an inventory form were to scan the barcode of a product, a DataSource could be referenced to instantly fill the product's SKU, quantity available, price, and more.

This functionality eliminates unnecessary manual data entry, reduces data entry errors, and protects the integrity of your data collection.

- 2. Option Filtering:** DataSources can be used to present users with specific options based on the value of another field. For example, when completing an employee evaluation, you might have fields for the employee's department and supervisor. You will want to present the employee with the option to select a supervisor from within their department. So, based on their department selection, a different set of supervisors will be presented to the user.

Option Filtering ensures that only relevant choices are presented to users, reducing opportunities for confusion and error.

"I love it...GoFormz has made my life easier. The job is created in Smartsheet, and after, in GoFormz, they just need to select the job number and it populates the Work Order number, the PO number, the address, the state, and other pieces of information, and displays them accurately." - Terri Bryan, Accounts Manager, ARC American

How to create a DataSource

Prepare Your Data (and Spreadsheet)

To create a *DataSource*, you'll need to first prepare a CSV file of your data – follow these simple steps to export and configure your file:

1. Export your data from wherever it's stored (e.g. Salesforce, Quickbooks, Excel, Google Sheets, or an internal database) as a CSV file.
2. Ensure that your **Key Field is the first column of your spreadsheet**. This column will serve as a unique identifier for each record.

Best Practice Tip: It's best practice to include your column names in the CSV header. Most applications will have an option to do this.

Import Your Data into GoFormz

1. Log in to GoFormz from a computer and select 'DataSources' from the navigation bar. Then, click 'Create' (from the upper right-hand corner). This will prompt a pop-up window to appear.
2. Drag and drop your CSV file into the upload box or click anywhere within the upload box to launch the file selector. Then, select your CSV file.
3. In the DataSource Name field, enter the name of your new DataSource. This field will automatically populate with the name of your CSV file if left blank.
[OPTIONAL] Check the box for 'Make Available for Public Forms' if you would like this DataSource to be usable within forms generated using Public Form links. If left unselected, this DataSource will not be available within forms generated via Public Form link. Please note, this can also be done at anytime from the DataSource list page as well. Click Next.
4. The DataSource preview step will show the first 10 rows of your DataSource. By default, GoFormz will not designate your first row as a header row (as seen above). If your first row is a Header Row, select the checkbox next to 'This CSV has a header row'. This will automatically replace the top row of your preview with the contents of your first row on the CSV.
Note: If your CSV contains duplicate rows in the Key Column, the preview step will highlight these duplicates and not allow you to upload this CSV. Key Column values must be unique.
5. Once you are satisfied with your DataSource and there are no duplicate errors found, click Create.

Once these steps have been completed, the data from the CSV file will be uploaded into GoFormz and your new DataSource will appear in the DataSource lists as an option to pull data from.

How to add DataSources in your digital form Template

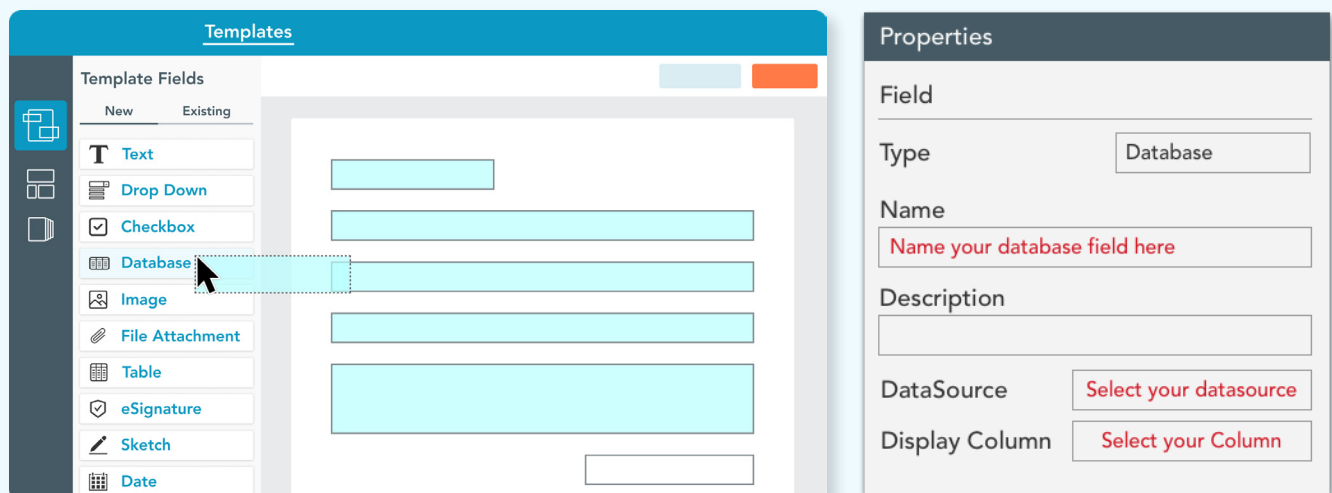
There are many ways to leverage DataSources within GoFormz. In this eBook, we'll showcase a few of the most popular DataSource use cases and how you can implement them within your digital form Template. Before diving into our *DataSources* use cases, let's first review how to implement *DataSources* when using popular form fields.

Using a DataSource with a Database field

A Database field works just like a Drop Down menu – it shows a list of items from which a user can make one selection. But, rather than displaying an ad hoc list of options, Database fields display items pulled directly from indexed DataSource columns. Your Database field can include all items in a DataSource column or a filtered list of items.

To add a Database field equipped with a DataSource to your digital form Template, complete these two simple steps:

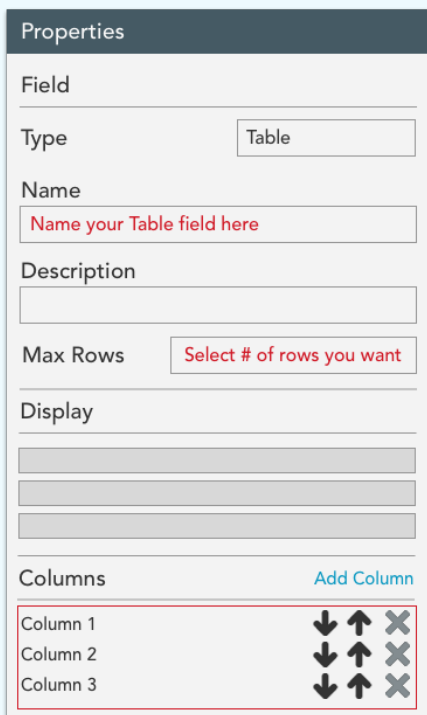
1. Drag and drop a Database field onto your Template.
2. Navigate to the Properties Panel and complete the following:
 - a. Add a Name for your Database Field.
 - b. In the DataSource property, select a DataSource from the drop-down.
 - c. In the Display Column property, select which column you wish to display (e.g. your DataSource 'Key' column).



Using a DataSource with a Table Field

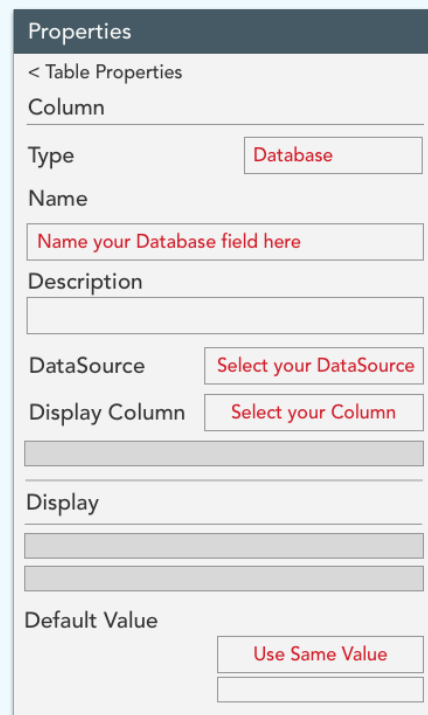
A Table field can hold various field types within each column. To include a DataSource within a column of a Table field, complete these two simple steps:

1. Drag and drop a Table field onto your Template.
2. Navigate to the Properties Panel and complete the following:
 - a. First, name your Table field.
 - b. Then, select a Table field column (the grey box above the Table field or found below in the Columns section of the Properties Panel).
 - c. In the 'Type' property, select 'Database'.
 - d. In the DataSource property, select your DataSource.
 - e. In the Display Column property, select which column you wish to display (e.g. your DataSource 'Key' column).



The screenshot shows the main 'Properties' panel for a Table field. The 'Type' is set to 'Table'. The 'Name' field contains the placeholder text 'Name your Table field here'. The 'Max Rows' field contains the placeholder text 'Select # of rows you want'. The 'Columns' section shows three columns: 'Column 1', 'Column 2', and 'Column 3', each with up/down arrows and a delete 'X' icon. A red box highlights the 'Columns' section.

Click one of the column's above



The screenshot shows the 'Individual Column properties preview' panel. The 'Type' is set to 'Database'. The 'Name' field contains the placeholder text 'Name your Database field here'. The 'DataSource' field contains the placeholder text 'Select your DataSource'. The 'Display Column' field contains the placeholder text 'Select your Column'. The 'Default Value' field contains the placeholder text 'Use Same Value'.

Individual Column properties preview

“GoFormz cut the time in half for building forms and populating the fields that we needed.” - Tony Tharp, Senior Technical Support Technician

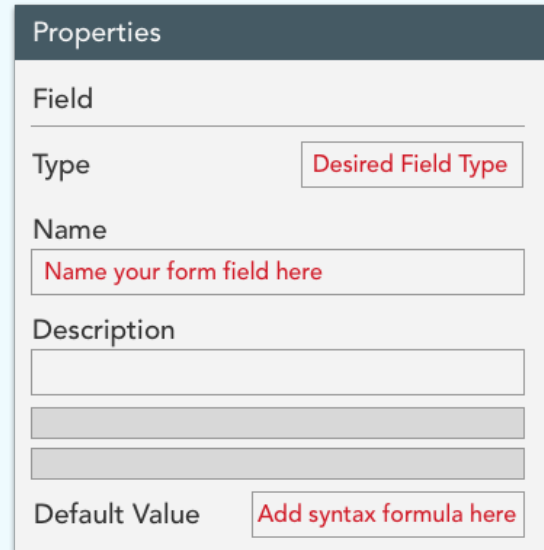
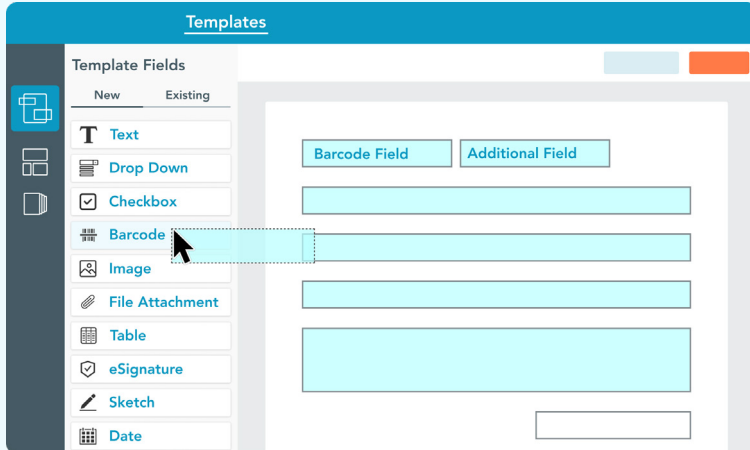
Using a DataSource with a Barcode Field

This field is used to scan a barcode and automatically populate corresponding form fields with DataSource information. Make sure that all expected barcode values are contained within the Key column of the DataSource you plan to use.

To include a Barcode field, follow these simple steps:

- » Drag and drop a Barcode field onto your Template.
- » Drag and drop an additional field (or locate an existing field) that directly correlates to your Barcode. With this field selected, navigate to the properties panel.
- » Within the Properties Panel, locate the Default Value and include this syntax formula: `=database("Datasourcename", [KeyField], "Column Header")`.

This syntax instructs this form field to reference the Key Field of the specific DataSource associated with your barcode and populate it with a corresponding value from the column specified by the "Column Header" value. Each of the syntax values ("Datasourcename", [Key Field], and "Column Header") should be customized to reference your specific DataSource.



Now that we've reviewed how to instrument your DataSource within your Template, let's explore three of the most popular use cases for digital forms equipped with DataSources: auto-populating fields, filtering options, and a combination of both functionalities.

3 DataSource Use Case Example Walk-Throughs

goformz

Order Form

ORDER DETAILS

Order No.

Order Date

BILLING INFORMATION

Bill to: Name/Company

Bill to: Address

Description	Quantity	Unit Price	Unit Total

Discount Rate

Tax Rate

Subtotal

Discount

Tax

Grand Total

Additional Comments or Instructions

Name (Print)

Signature

Date & Time

Use Case One: Auto-Populating Fields

Use Case Two: Option Filtering

Use Case Three: Auto-Populating and Option Filtering

Use Case One: Auto-Populating Fields

Automatically populating data is one of the most common methods for utilizing *DataSources* in GoFormz. In this use case, we will demonstrate how to use a Barcode field and *DataSource* to auto-fill the information of a specific product into a Table field.

To get started, open your Template in the Template Editor. For this example, we will use an Inventory Request Form. Then, follow along with the instructions outlined below.

Add and configure your Template fields

Drag and drop a Table field onto your Template and position it accordingly. It's best practice to name your Table field along with each of its Columns – allowing you to easily locate and manage your form's data and fields later on. For this example, we will use Item No., Description, Unit Price, QTY, and, Total to replicate the form.

Item No.	Description	Unit Price	QTY	Total
Item No.	Description	Unit Price	Quantity	Total
Item No.	Description	Unit Price	Quantity	Total
Item No.	Description	Unit Price	Quantity	Total
Item No.	Description	Unit Price	Quantity	Total
Item No.	Description	Unit Price	Quantity	Total
Item No.	Description	Unit Price	Quantity	Total
Item No.	Description	Unit Price	Quantity	Total
Item No.	Description	Unit Price	Quantity	Total

Next, you'll select which field type will occupy each Column cell. As we are using a Barcode for this example, we will select the Item No. Column to host our Barcode fields. For this example, our Table field will be configured with the following field types for each column:

- » **Column 1: Item No. = Barcode**
- » **Column 2: Description = Text**
- » **Column 3: Unit Price = Number**
- » **Column 4: Quantity = Number**
- » **Column 5: Total = Number**

Add and configure your DataSource

With our Table field configured, we can now connect our DataSource. If you have not yet uploaded your DataSource, please refer back to the 'How to Create a DataSource' section of this eBook. For this use case example, we will use a DataSource named *Inventory Request Form DataSource*.

For this example, we'll be configuring a DataSource to populate two columns of our table ("Unit Description" and "Unit Price") when a Barcode is scanned in the "Item No." column of our Table. First, we will set up our DataSource connection to the "Unit Description" column.

Select the gray header of the appropriate column of your Table field (for this example, the "Unit Description" column) and navigate to the Properties Panel. Locate the Default Value field and enter the following equation *customized for your specific DataSource*:

=database("Datasourcename", [KeyField], "Column Header")

For this use case example, our formula reads as follows:

=database("Inventory Request Form DataSource",[Inventory Table]![1][1],"Description")



As we are using a Table Field, we'll need to customize and apply this value for each row of the "Unit Description" column. To do so, change the Default Value Drop Down menu in the Properties Panel from *Use Same Value* to **Use Individual Values** and adjust the formula to match each row. For example:

Row 1: =database("Inventory Request Form DataSource",[Inventory Table]![1][1],"Description")

Row 2: =database("Inventory Request Form DataSource",[Inventory Table]![1][2],"Description")

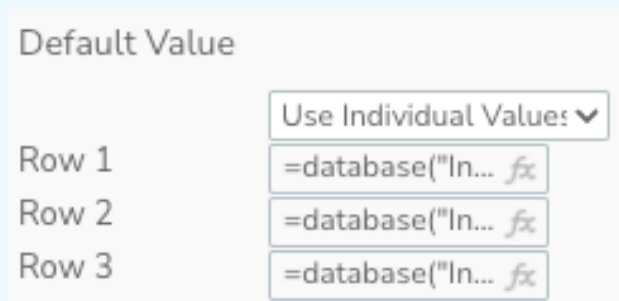
Row 3: =database("Inventory Request Form DataSource",[Inventory Table]![1][3],"Description")

As demonstrated in the example above, we have adjusted the formula for each row of the column to correspond with the appropriate row number. We'll apply the same customization to each of the rows of our column.

Now, we'll follow the same instructions to set up the next column of our Table to populate the Unit Price. Your Default Value equations should look as follows:



- Row 1:** `=database("Inventory Request Form DataSource", [Inventory Table]![1][1], "Unit Price")`
- Row 2:** `=database("Inventory Request Form DataSource", [Inventory Table]![1][2], "Unit Price")`
- Row 3:** `=database("Inventory Request Form DataSource", [Inventory Table]![1][3], "Unit Price")`



Now that you have successfully built a DataSource auto-populating data tied to a Barcode field, make sure to save your draft and run a test. It's best practice to run tests on your digital form Templates before Publishing them for other users to use. If you are satisfied with the result of the test, click Publish.

Use Case Two: Option Filtering

Option Filtering, also known as DataSource Filtering, is an excellent method for enhancing data accuracy and optimizing the data collection processes. As mentioned in the “Using a DataSource with a Database field” section, Option Filtering utilizes the Database field to retrieve data from a DataSource and present only relevant information to a form user.

To get started, open your Template in the Template Editor. For this example, we are going to use an Auto Repair Work Order, within which we will use a DataSource to filter options for the make, model, year, and transmission of a car.

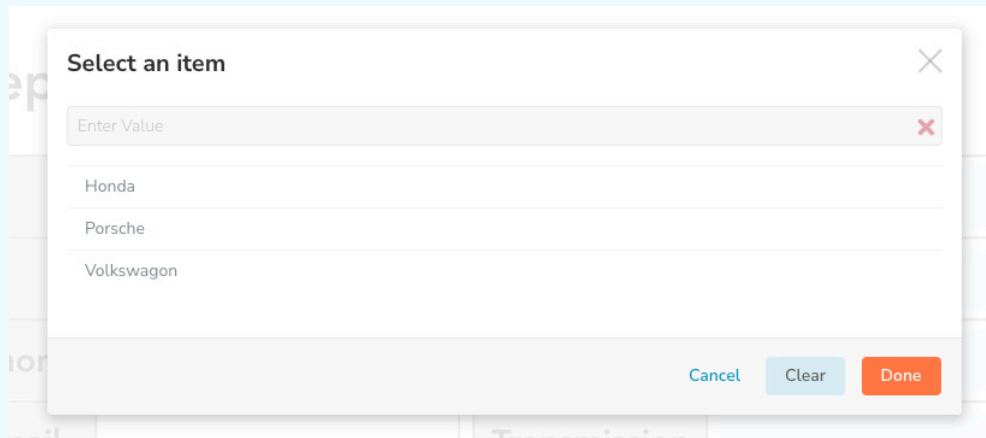
Auto Repair Work Order			
Name		Car Make	Car Make
Address		Car Year	Car Year
Contact Phone		Car Model	Car Model
Contact Email		Transmission	Transmission

Drag and drop four Database fields onto your Template and name each field. For this example, our Database fields will be named:

- » Car Make
- » Car Model
- » Car Year
- » Transmission

Now, select the first Database field, Car Make. In the Properties Panel, locate the DataSource property. From the drop down, select the name of the DataSource you would like referenced by your field. For this use case example, our DataSource is named “Option Filtering Auto Repair Work Order”.

Next, select your Display Column using the Display Column property. For this field, we will select Car Make as our Display Column. This means everything that appears in the Car Make column of your DataSource will be visible in a Drop Down menu.



Next, select the Car Year Database field. Follow the same steps to select your DataSource, but for this step change the Display Column to Car Year. You will see a Filter property directly beneath the Display Column property. Click into the field and add the following formula (customizing the formula as needed for your use case and DataSource):

=“Car Make”=[Car Make]

This equation instructs the Car Year field to reference the selection made in the Car Make field and only display drop down menu options that correlate to the specific type of car selected.

Properties	
Field	
Type	Database
Name	Car Year
Description	
DataSource	Option Filtering Auto...
Display Column	Car Year
Filter	=“Car Make”=[Car Make]

Similar equations will be utilized in our final two fields, **Model** and **Transmission**. To do so, select the Car Model field and return to the Properties Panel. As **Car Model** is already the Key Column for our DataSource, we will change the Display Column property to Key, before clicking into the Filter property and adding the following formula:

=AND(“Car Make”=[Car Make], “Car Year”=[Car Year])

This formula instructs the *Car Model* field to only display options pertaining to selections made in the two previous fields (*Make* and *Year*).

Finally, select the *Transmission* Database field, update the Display Column to *Transmission*, and type this formula into the Filter section:

=AND("Car Make"=[Car Make], "Car Year"=[Car Year], "Key"=[Car Model])

For reference here is how each field should look:

	DataSource	Display Column	Filter
Car Make	Option Filtering Auto Repair Work Order	Car Make	
Car Year	Option Filtering Auto Repair Work Order	Car Year	= "Car Make"=[Car Make]
Car Model	Option Filtering Auto Repair Work Order	Key	=AND("Car Make"=[Car Make], "Car Year"=[Car Year])
Transmission	Option Filtering Auto Repair Work Order	Transmission	=AND("Car Make"=[Car Make], "Car Year"=[Car Year], "Key"=[Car Model])

After configuring the filtering equations, test the form to ensure the filtering operates correctly. Once confirmed, save, preview, and publish your digital form Template.

"...You can get down to business faster, especially if everything is filled out the right way the first time." - **Dr. Brad Wilkinson, Solo Practitioner, Pediatric Dental Specialties**

Use Case Three: Auto-Populating and Option Filtering

Now that we've reviewed two distinct examples of using *DataSources*, we can now utilize them in conjunction within the same use case. In this example, we will employ Option Filtering and Auto-Populating to rapidly fill out an entire section of a digital form.

For this example, we are using an Employee Evaluation Form that will filter the Company Name, Department, and Employee Name. The Employee Name selection will then auto-populate the Employee ID Number, Employee Title, and Employee Manager.

There are multiple ways to accomplish Option-Filtering and Auto-Populating using *DataSources*. For this example, we will use two separate CSV files:

- » Our first CSV File, **Company Names**, contains information about each employee, what company they work for, and their department.
- » Our second CSV, **Employee Info**, contains the Employee Name, Employee ID Number, Employee Title, and Employee Manager.

Open your Template in the Template Editor, add your desired fields, and give them names. For this example, we will use three Database fields and three Text fields.

Database Fields		Text Fields	
Company Name	Company Name	Employee ID Number	Employee ID Number
Department	Department	Employee Title	Employee Title
Employee Name	Employee Name	Employee Manager	Employee Manager

Configure option filtering

First, we will configure Option Filtering within the Database fields. For this use case example, we will configure option filtering for the following Database fields:

- » Company Name
- » Departments
- » Employee Name

Select your first Database field, for this example, we will begin by selecting Company Name. Navigate to the Properties Panel and select your DataSource from the DataSource property. For this example and field, we will select the Company Names DataSource and will adjust the Display Column property to Company Name.

Next, select the Department Database field, add the DataSource Company Names, change the display column to Department, and then add this formula to the Filter section:

=“Company Name”=[Company Name]

Finally, go to the Employee Name Form Field, add the DataSource, change the display column to Key, and then add this formula to the filter section:

=And(“Company Name”=[Company Name],“Department”=[Department])

Configure auto-population

Now, it's time to set up our Text fields to automatically populate based on selections made using the Database fields. For this example, we will configure the following Text fields to populate with information based on the Company, Department, and Employee Name fields selected above:

- » Employee ID Number
- » Employee Title
- » Employee Manager

For each of your Text fields, navigate to the Properties Panel, and locate the Default Value formula bar. For each of the Text fields listed above, we will add a formula instructing the field to populate with information pulled from the Employee Info DataSource.

- » Employee ID Number: **=database(“Employee Info”, [Employee Name], “Employee ID Number”)**
- » Employee Title: **=database(“Employee Info”, [Employee Name], “Employee Title”)**
- » Employee Manager: **=database(“Employee Info”, [Employee Name], “Employee Manager”)**

Important Note: Referencing two DataSources will only work if the values match exactly. For example, the Employee Names in the *Company Names DataSource* must perfectly match the Employee Names listed in the *Employee Info DataSource*.

Finally, save your digital form Template as a Draft and test your formula. For this use case, this test should confirm that only relevant options are presented within the Department and Employee Name Database fields and that the correct employee information is instantly populated within the appropriate Text fields (e.g. ID and Manager) once a selection has been made. Once your test is complete and successful, publish your digital form Template!

Frequently Asked Questions

1. Is there a limit to the number of columns and rows a DataSource can have?

We can support up to 200 columns and 10,000 rows in your DataSource. If you wish to use more, please reach out to customer support.

2. Can you do a look-up on any other DataSource column than the key?

When using a Database field, you can choose to look up and populate the field from a list of any of the indexed columns (one per DB field). But in order to populate other fields based on that DB field value, you must use the key.

3. How do you add/remove a column from a DataSource?

The only way to add a column to a DataSource is to import a CSV including the new column. You cannot remove a column from a DataSource at this time.

4. Is there a limit to how many DataSources I can have linked to a Template?

No!

5. What does the Last Updated Date column mean in GoFormz?

We recently added a default column to all DataSources that show the last updated date of that row. This column looks like it's part of the DataSource but it's not able to be used on the Template like other columns are and it is not included in the CSV export of a DataSource.

Additional Resources

If you found this eBook helpful and are looking for additional information to kickstart your Data Source needs, [contact our team](#) or check out the helpful resources listed below:

Support Documentation

- » [DataSources Overview](#)
- » [How to Use Database and Barcode Fields](#)
- » [How to Create a DataSource](#)
- » [How to Edit DataSources](#)
- » [Best Practices for Managing DataSources](#)

Helpful Blogs

- » [ProTip: Auto-Fill Form Fields with Database Information](#)
- » [FAQ: Can DataSources be used with Public Forms?](#)
- » [How to Reduce Data Entry Errors with Auto-Populated Form Fields](#)
- » [FAQ: Can DataSources be used while offline?](#)

How GoFormz Users are Using DataSources

- » [Case Study – ARC American and Mobile Energy Forms](#)
- » [Case Study – Budinger & Associates and Mobile Engineering Forms](#)
- » [Case Study – DESHAZO & Construction Mobile Forms](#)
- » [Case Study – DiCAN and Mobile Manufacturing Forms](#)
- » [Case Study – Knights Spraying and Mobile Field Service Forms](#)
- » [Case Study – Pacific Seafood and Mobile Food Processing Forms](#)
- » [Case Study – Rhino Cable Services and Mobile Telecommunication Forms](#)