# Prepare Your Data

# Concept | Recipe

In Dataiku, recipes are fundamental components within the Flow that encapsulate data transformation logic. They play a crucial role in separating processing logic from datasets, offering several advantages:

1. **Flexibility with Data Storage**: Recipes allow you to change underlying data storage technologies (e.g., switching cloud providers) without affecting the processing logic defined in the Flow's recipes. This flexibility ensures that your data processing workflows remain intact despite changes in data infrastructure.
2. **Clear Data Lineage**: By examining the Flow, you can trace the entire history of transformations applied to data, from its raw form to the final output dataset. This lineage visibility is critical for understanding how data has been manipulated and transformed over time.

**Recipe Categories**

Recipes in Dataiku are categorized based on their functionality and purpose:

* **Visual Recipes (Yellow circles)**:
  + These recipes offer a graphical user interface (GUI) for performing common data transformations. Users can apply operations like cleaning, grouping, or filtering data without needing to write code manually.
* **Code Recipes (Orange circles)**:
  + Code recipes provide the flexibility to define custom processing logic using languages such as Python, R, or SQL. They are ideal for tasks that require specific or complex transformations beyond what visual recipes offer.
* **Plugin Recipes (Red circles)**:
  + Plugin recipes extend Dataiku's capabilities through custom plugins. They combine the advantages of code recipes (freedom to define custom logic) with the accessibility of visual recipes (easy GUI-based interaction). Plugin recipes appear in the Flow only after relevant plugins have been manually installed.

**Choosing the Right Recipe**

* **Visual Recipes**: Suitable for common data transformation tasks that can be efficiently handled through a GUI, promoting ease of use and accessibility across team members who may not be proficient in coding.
* **Code Recipes**: Provide maximum flexibility for advanced transformations and custom logic, allowing data scientists and analysts to implement specific processing requirements using their preferred programming languages.
* **Plugin Recipes**: Offer a hybrid approach, where the power of custom code can be encapsulated within a visual interface, combining ease of use with the capability for advanced customization.

**Benefits of Recipe-Based Approach**

* **Modularity and Reusability**: Recipes can be reused across different projects or within the same project, promoting consistency and efficiency in data processing workflows.
* **Version Control and Collaboration**: Changes made to recipes are tracked, facilitating version control and collaborative work among team members working on the same project.

By leveraging recipes in Dataiku's Flow, organizations can streamline data transformation processes, maintain data lineage transparency, and adapt seamlessly to evolving data storage technologies and business requirements.

Top of Form

Bottom of Form

## Concept | Prepare Recipe

The "Prepare Recipe" in Dataiku is a powerful tool for creating data transformation scripts visually and interactively. Here's a breakdown of how you can use it effectively:

### 1. Adding Transformation Steps

To prepare your data, you add transformation steps to the recipe script. These steps can be added in several ways:

* **Using the Processor Library**: Dataiku provides a library of around 100 data processors. Each processor is designed to handle specific tasks such as filtering rows, rounding numbers, extracting regular expressions, concatenating or splitting columns, and more. For instance, there's a Formula language processor similar to spreadsheet functions for creating new columns.
* **Column Context Menu**: When working with a specific column, Dataiku suggests relevant steps based on the column's data type and context. For example, it might suggest parsing date columns or applying string transformations like converting to lowercase.
* **Analyze Window**: This window guides you through data preparation tasks such as merging categorical values, offering a structured approach to refining your data.
* **Manual Adjustments**: You can manually adjust column order by dragging columns, switch between table and column views for batch operations, and more.

### 2. Previewing and Applying the Script

When you add steps to the script, you can immediately preview their effect on a sample of the dataset in the Explore tab. This allows for quick iteration and modification of transformation steps. However, these changes are applied only when you run the recipe.

### 3. Managing Complexity

As scripts grow more complex, Dataiku offers features to manage them effectively:

* **Disabling Steps**: Temporarily disable steps to test different scenarios without deleting them.
* **Organizing Steps**: Group individual steps logically and add colors or comments to enhance readability and collaboration.
* **Copying and Pasting**: Easily copy steps within the same recipe or to another recipe, even across different projects or Dataiku instances.

### 4. Execution and Output

When you're ready, running the recipe applies all transformation steps to the entire input dataset, generating a new output dataset. The original input dataset remains unchanged.

# Concept | Date handling

In Dataiku, handling date parsing and working with date columns involves utilizing the Prepare recipe, which simplifies the process of recognizing and parsing date formats correctly. Here’s a step-by-step overview based on the provided text:

**Challenges with Date Handling**

Working with dates in data often presents challenges due to:

* **Multiple Formats:** Dates can be represented in various formats such as "1/5/19", "2019-01-05", or "1 May, 2019".
* **Time Zones:** Dates may include time zone information.
* **Extracting Components:** Extracting specific components like day of the week can be complex.

**Parsing Dates in Dataiku**

1. **Identifying Unparsed Date Columns:**
   * Dataiku's Prepare recipe helps in recognizing columns that contain unparsed dates.
2. **Using the Parse Date Processor:**
   * Dataiku suggests the Parse date processor when it identifies a column as containing unparsed dates.
   * This processor helps convert ambiguous date strings into a format that the system can understand as a date.
3. **Applying the Processor:**
   * Once identified, you can add the Parse date processor to the script.
   * This processor requires specifying the format of the date and optionally the time zone to ensure accurate parsing.
4. **Previewing and Confirming Changes:**
   * After applying the Parse date processor, Dataiku immediately shows a preview of how the date format has changed.
   * The column transforms from an unparsed string to a structured date format.
5. **Further Data Manipulations:**
   * With parsed dates, Dataiku suggests additional steps like computing time differences, extracting date components (e.g., day of the week), or filtering based on dates.

# Concept | Dataiku formulas

Dataiku's Formula language provides a powerful toolset similar to traditional spreadsheet formulas, enabling users to perform calculations, manipulate data, and enhance the functionality of visual recipes within the platform. Here’s a comprehensive overview of its capabilities and practical applications:

**Formula Processor in Dataiku**

The Formula processor within the Prepare recipe allows you to incorporate formulas to transform and analyze data:

* **Formula Step**: Add a Formula step to create new columns or modify existing ones based on specified expressions.
* **Editor Panel**: Offers support features like code completion and error alerts, enhancing ease of use and accuracy.

**Key Features of Dataiku Formula Language**

1. Mathematical Functions: Includes round(), sum(), and max() for numerical operations.
2. Comparison Operators: Supports >, <, >=, <= for comparing values.
3. Logical Operators: Includes AND, OR for combining conditions.
4. Handling Missing Values: Functions such as isBlank() or isNULL() assist in managing null or empty values.
5. Conditional Statements: Supports if-then statements (if(condition, value\_if\_true, value\_if\_false)) for creating conditional logic.
6. String Functions: Includes length() to measure string lengths and substring() to extract parts of strings based on specified positions.
7. Random Number Generation: Utilizes rand(min, max) to generate random integers within defined ranges, useful for simulations.
8. Column Name Handling: Functions like strval() and numval() facilitate handling column names with spaces by ensuring correct syntax and data type handling.

**Practical Use Case: Fraud Detection**

An example scenario involves flagging large credit card purchases for potential fraud detection:

* **Objective**: Compare each transaction's amount (purchase\_amount) against the average purchase amount for the corresponding credit card (card\_purchase\_amount\_avg).
* **Formula Example**: Create a new column (higher\_than\_avg\_purchase) using an if-then statement:

if(purchase\_amount > card\_purchase\_amount\_avg, "yes", "no")

This flags transactions where the purchase amount exceeds the average as "yes", indicating potential fraud.

See the [Formula language](https://doc.dataiku.com/dss/latest/formula/index.html) page in the reference documentation.

# \* Tutorial | Prepare Your Data (text)

# Objectives

In this tutorial, you will:

1. Create a Prepare Recipe: Start a new data preparation workflow.
2. Add Various Processor Steps: Use different methods to add and configure steps within the Prepare recipe.
3. Parse Date Columns: Convert a date column from string format to a proper date type.
4. Use the Formula Language: Compute a new column using formulas.
5. Run the Recipe: Execute the preparation steps on the entire dataset.

**Step-by-Step Guide**

## 1. Create a Prepare Recipe

* Open the orders dataset in Dataiku.
* Click on the **Actions** button or the plus sign at the top-right.
* Under **Visual Recipes**, select **Prepare**.
* In the **New data preparation recipe** window:
  + Keep **orders** as the input dataset.
  + Accept the default output dataset name (**orders\_prepared**).
  + Click **Create Recipe**.

## 2. Add Steps in Different Ways

* **Drag Columns**: Drag the order\_id column in front of the pages\_visited column to reorder columns.
* **Analyze Window**: Cleanse the tshirt\_category column by merging similar categories using the Analyze window's Mass Actions feature.
* **Parse Date**: Convert the order\_date column from string format to date format using the **Parse date** processor.

**3. Use the Formula Language**

* Add a **Formula** step to compute the dollar value of each t-shirt order:
  + Click the yellow **+ Add a New Step** button.
  + Select **Formula** from the processors library.
  + Enter total as the name of the new output column.
  + In the expression field, type tshirt\_price \* tshirt\_quantity.
  + Click anywhere to see the total column appear.

**4. Rename/Delete Columns**

* After parsing the order\_date column:
  + Delete the original order\_date column.
  + Rename the parsed date column (order\_date\_parsed) to order\_date.

**5. Run the Recipe**

* Once all steps are defined:
  + Click **Run** in the lower-left corner of the page.
  + Wait for the job to complete.
  + Click **Explore dataset orders\_prepared** to view the output dataset.

### Group the Data

# Concept | Group recipe

The Group recipe in Dataiku is a powerful tool used for aggregating data based on specified group keys and performing aggregations on those groups. Here’s a structured overview based on the information provided:

**Concept**

The Group recipe aggregates values from columns based on the values of one or more key columns. This is useful for summarizing data into meaningful groups, such as summing transaction values by customer or finding maximum values by product category.

**Use Case**

In analytics, the Group recipe is essential for various aggregation tasks. For instance, it allows you to aggregate sales data by customer, product category, time period, or geographical unit.

**Group Components**

1. **Group Key**: Columns that define the groups. The output will have one row per unique combination of values in these columns.
2. **Aggregation**: Functions applied to columns within each group, such as sum, max, min, count, etc.

**Grouping with Dataiku**

**Group Step**

* **Select Group Key**: Choose the column(s) that will serve as the basis for grouping (e.g., customer, product category).
* **Set Aggregations**: Define what calculations you want to perform for each group (e.g., sum of sales, maximum order date).

**Output Step**

* **Result Columns**: After configuring the Group recipe, you can preview the output columns. These typically include:
  + The group key columns.
  + Columns with the aggregated values (e.g., max order date, sum of sales).

**SQL Query**

* For those familiar with SQL, Dataiku provides an SQL query view of the Group recipe's operations. This query corresponds to a SQL GROUP BY statement and allows for in-depth customization if needed.

**Output Dataset**

* After executing the Group recipe, the resulting dataset will contain only unique combinations of the group key columns and the calculated aggregations.

**Example**

If you have a dataset of transactions with columns like customer, product category, sale amount, and order date, you can use the Group recipe to:

* Group by customer to calculate total sales per customer.
* Group by product category to find the maximum sale amount per category.
* Group by order date (month and year) to aggregate sales for each month.

**Conclusion**

The Group recipe in Dataiku simplifies complex data aggregation tasks by allowing users to specify group keys and aggregations visually. It is particularly useful for summarizing and analyzing large datasets efficiently.

# Tutorial | Group recipe

In this tutorial on the Group recipe in Dataiku, the focus is on aggregating data based on specified keys and performing aggregations on those groups. Here’s a step-by-step summary based on the provided tutorial:

**Tutorial Steps: Using the Group Recipe**

1. **Select Dataset**: Start by selecting the orders\_prepared dataset from the Flow in Dataiku.
2. **Create Group Recipe**:
   * Click on the **Actions** tab in the right panel.
   * Click on the **+ button** to add a new recipe.
   * Choose **Group** from the list of Visual recipes.
3. **Configure Group Recipe**:
   * In the Group recipe dialog:
     + Choose customer\_id as the group key. This means we will aggregate data based on each unique customer\_id.
     + Change the name of the output dataset to orders\_by\_customer.
     + Select **Create Recipe** to proceed.
4. **Group Step**:
   * In the Group step of the recipe:
     + Specify the aggregations you want to perform for each group:
       - Minimum of order\_date to find the first order date (order\_date\_min).
       - Average of pages\_visited (pages\_visited\_avg).
       - Sum of total to get the total sum of orders (total\_sum).
       - The count aggregation is automatically included by default.
5. **Configure Output**:
   * Before running the recipe, navigate to the **Output step**.
   * Rename order\_date\_min to first\_order\_date for clarity.
6. **Run Recipe**:
   * Click **Run** to execute the Group recipe and create the new orders\_by\_customer dataset with aggregated results.
7. **Explore Output**:
   * After the recipe completes:
     + Open the orders\_by\_customer dataset.
     + Use tools like **Analyze** to explore columns, ensuring that each customer\_id now has unique aggregated data based on the specified aggregations.

**Notes from the Tutorial**

* **Column Usage**: Columns in the input dataset that are not part of the group key or selected for aggregation (e.g., order\_id, tshirt\_category) are excluded from the output dataset.
* **Data Types**: Aggregations like Min are based on the data type of the column (order\_date as a date, not string).

**Conclusion**

The Group recipe in Dataiku provides a straightforward method to aggregate data based on keys and perform essential aggregations such as sum, average, and minimum/maximum calculations. It is a fundamental tool for summarizing and analyzing datasets by different dimensions, making it easier to derive insights and prepare data for further analysis or reporting.

# Enrich Your Data With the Join Recipe

# Concept | Join recipe

In the context of Dataiku, the Join recipe serves the fundamental purpose of combining data from two datasets based on a common key column. Here’s a detailed overview of how you can utilize the Join recipe, based on the provided information:

### Concept: Join Recipe

The Join recipe in Dataiku enables you to enrich one dataset by combining it with columns from another dataset. This is achieved by matching values in a key column that is common to both datasets. The type of join you choose determines how the matching is performed and what the output looks like.

### Types of Joins

1. **Left Join** (Default):
   * Retains all rows from the left dataset and adds matching rows from the right dataset. If there is no match, the columns from the right dataset will be null.
2. **Inner Join**:
   * Keeps only the rows that have matching values in both datasets. This ensures that only rows with complete information from both datasets are included in the output.
3. **Outer Join**:
   * Keeps all rows from both datasets. Where there is a match, it combines the rows; where there isn’t, it fills with null values.
4. **Right Join**:
   * Similar to a left join but keeps all rows from the right dataset and adds information from the left dataset where there is a match.
5. **Left Anti-Join**:
   * Retains rows from the left dataset that have no match in the right dataset.
6. **Right Anti-Join**:
   * Retains rows from the right dataset that have no match in the left dataset.
7. **Cross Join**:
   * Matches every row of the left dataset with every row of the right dataset, resulting in a Cartesian product.
8. **Advanced Join**:
   * Provides custom options for row selection and deduplication when none of the other join types fit the requirement perfectly.

**Steps in Using Join Recipe**

1. **Change Key Column**:
   * Specify the key column or columns to match on. You can override automatically detected keys with your own selection.
2. **Add Dataset**:
   * For left, right, or inner joins, you can add a dataset to capture unmatched rows separately.
3. **Choose Output Columns**:
   * Select which columns from both datasets you want to include in the output dataset.
4. **Set Pre-Filters**:
   * Apply filters to exclude rows based on certain criteria before performing the join. This can help optimize the join operation by reducing unnecessary computations.
5. **Set Post-Filters**:
   * Apply filters to the resulting dataset after the join operation. This can include deduplication settings and conditions for returning only rows that match specific criteria.
6. **Review Output**:
   * Before executing the join, review the specifications and selected columns to ensure they meet your requirements.

**Tutorial Focus**

The tutorial focuses on using the Join recipe to enrich a dataset, specifically by demonstrating how to join datasets based on common keys and choose the appropriate join type depending on the desired outcome (e.g., retaining all rows from one dataset or only matched rows).

**Conclusion**

The Join recipe in Dataiku is crucial for integrating data from multiple sources or datasets within a project. It offers flexibility in terms of join types and customization options, making it a versatile tool for data preparation and analysis tasks in a workflow (Flow) environment

# Tutorial | Join recipe

**Getting started**

After grouping some data in the previous tutorial, let’s now demonstrate another key visual recipe, the [Join recipe](https://academy-content.dataiku.com/latest/data-preparation/visual-recipes/concept-join-recipe.html?opals=true), which allows you to enrich your data with columns from another dataset.

**Objectives**

In this tutorial, you will:

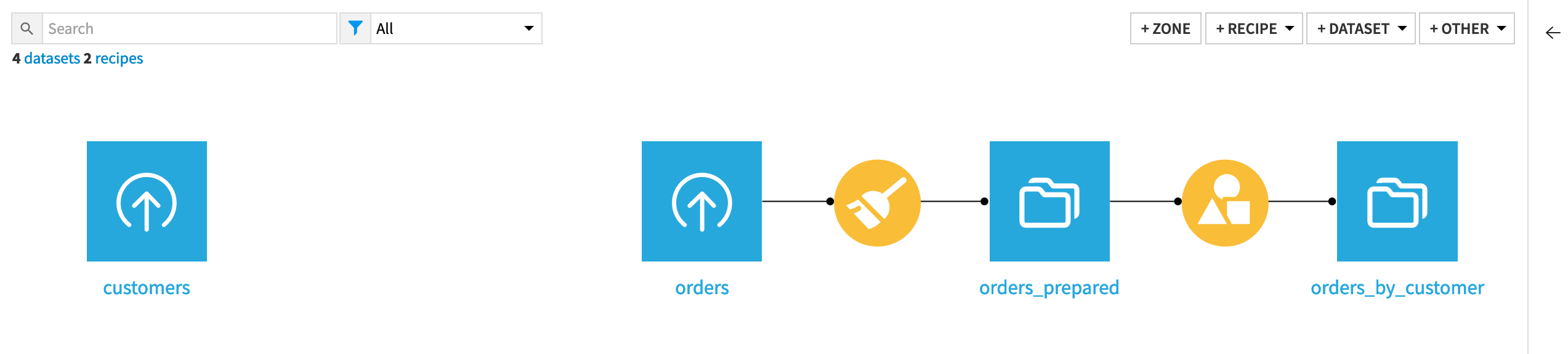
* Use a Join recipe to join two datasets.

### Prepare the project

After grouping the data by customer ID in the previous tutorial, you have a project with the *orders\_by\_customer* dataset. You now need to import one new dataset to that project.

1. Download the [customers CSV file](https://downloads.dataiku.com/public/website-additional-assets/data/customers.csv).
2. From the Flow of your existing project, click **+ Dataset > Upload your files**.
3. Click on **Select Files**, and choose your **customers.csv** file.
4. Finish importing the dataset by clicking the **Create** button or using the shortcut **Cmd/Ctrl+S**.

Once you have the dataset *customers* and *orders\_by\_customer* in your Flow, you’re ready to start!

[](https://academy-content.dataiku.com/latest/_images/starting-flow-103.png)

**Explore the customer data**

Tip

A screencast at the end of the page recaps the instructions described here.

Let’s explore the *customers* dataset.

1. From the Flow, double-click on **customers** dataset to open it.
2. Explore the dataset. As you can see, each row in this dataset represents a separate customer, and records:
   * The unique customer ID
   * The customer’s gender
   * The customer’s birth date
   * The user agent most commonly used by the customer
   * The customer’s IP address
   * Whether the customer is part of Haiku T-Shirts’ marketing campaign

Note

Take a few minutes to explore it with tools like Analyze. Also, note the gray portion of the *gender* column’s data quality bar representing missing values.

#### Create a Join recipe

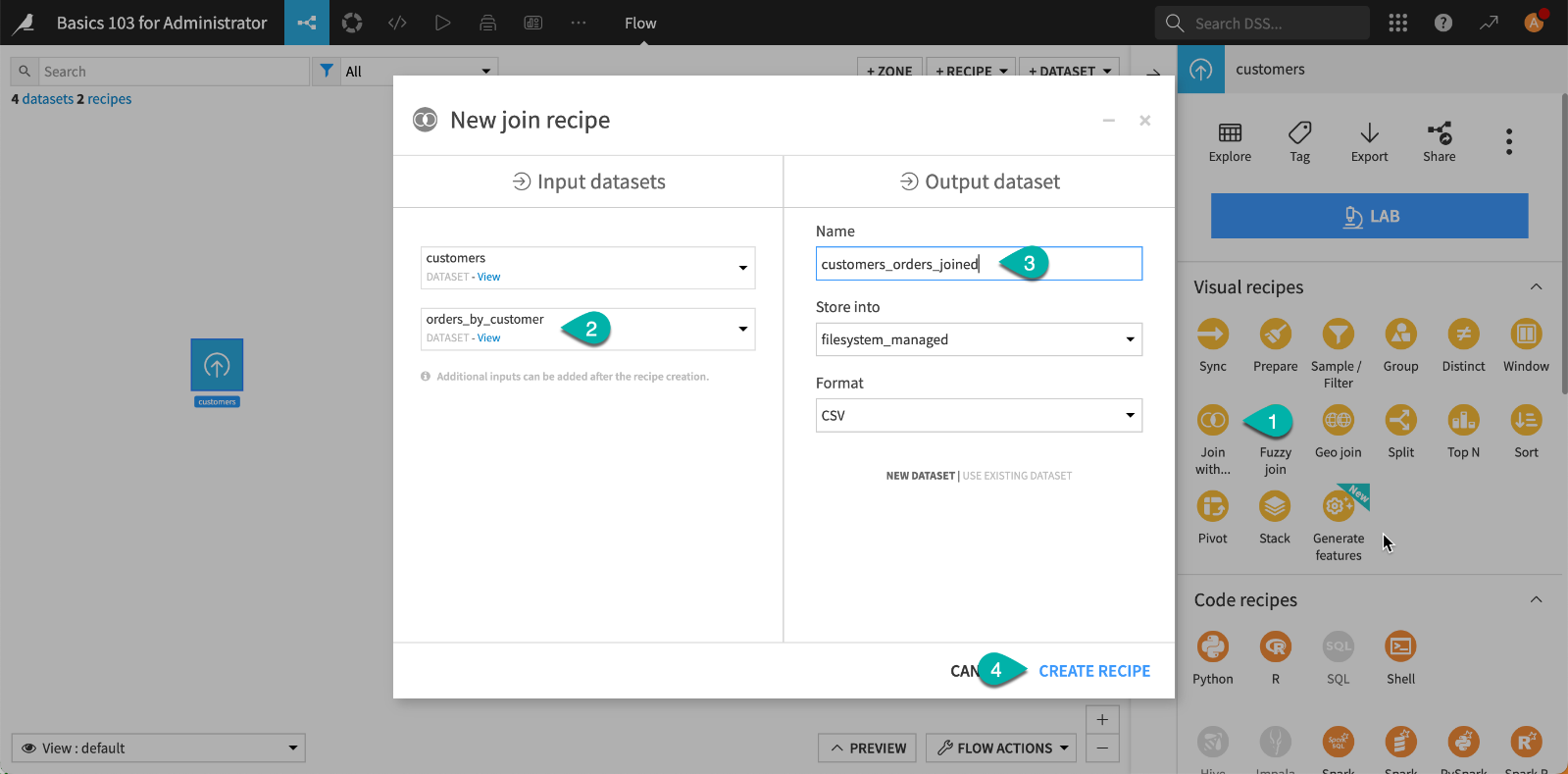
We can use the **Join** recipe to enrich the *customers* dataset with the information about the aggregate orders customers have made from the *orders\_by\_customer* dataset.

1. Click the **Actions** button in the top right of the **Explore** tab of the *customers* dataset. The **Actions** tab opens in the right panel.
2. Under **Visual recipes**, choose **Join with…**.
3. Select **orders\_by\_customer** as the second input dataset.

Note

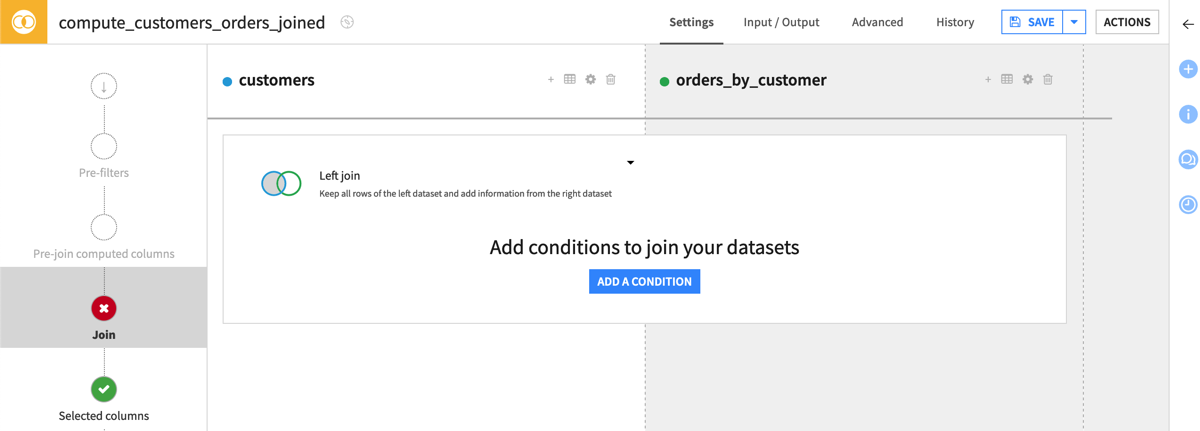
Although only two datasets can be added in the Join recipe creation dialog, more datasets can be added at the **Join** step after creating the recipe.

1. Change the name of the output dataset to customers\_orders\_joined.
2. Click **Create Recipe**.

[](https://academy-content.dataiku.com/latest/_images/join-dialog.png)

#### Define the join condition

The core step of this recipe is the **Join** step, where you choose how to match rows between the datasets. In this case, we want to match rows from the *customers* and *orders\_by\_customer* datasets that have the same value of *customerID* and *customer\_id*, respectively.

[](https://academy-content.dataiku.com/latest/_images/join-recipe.png)

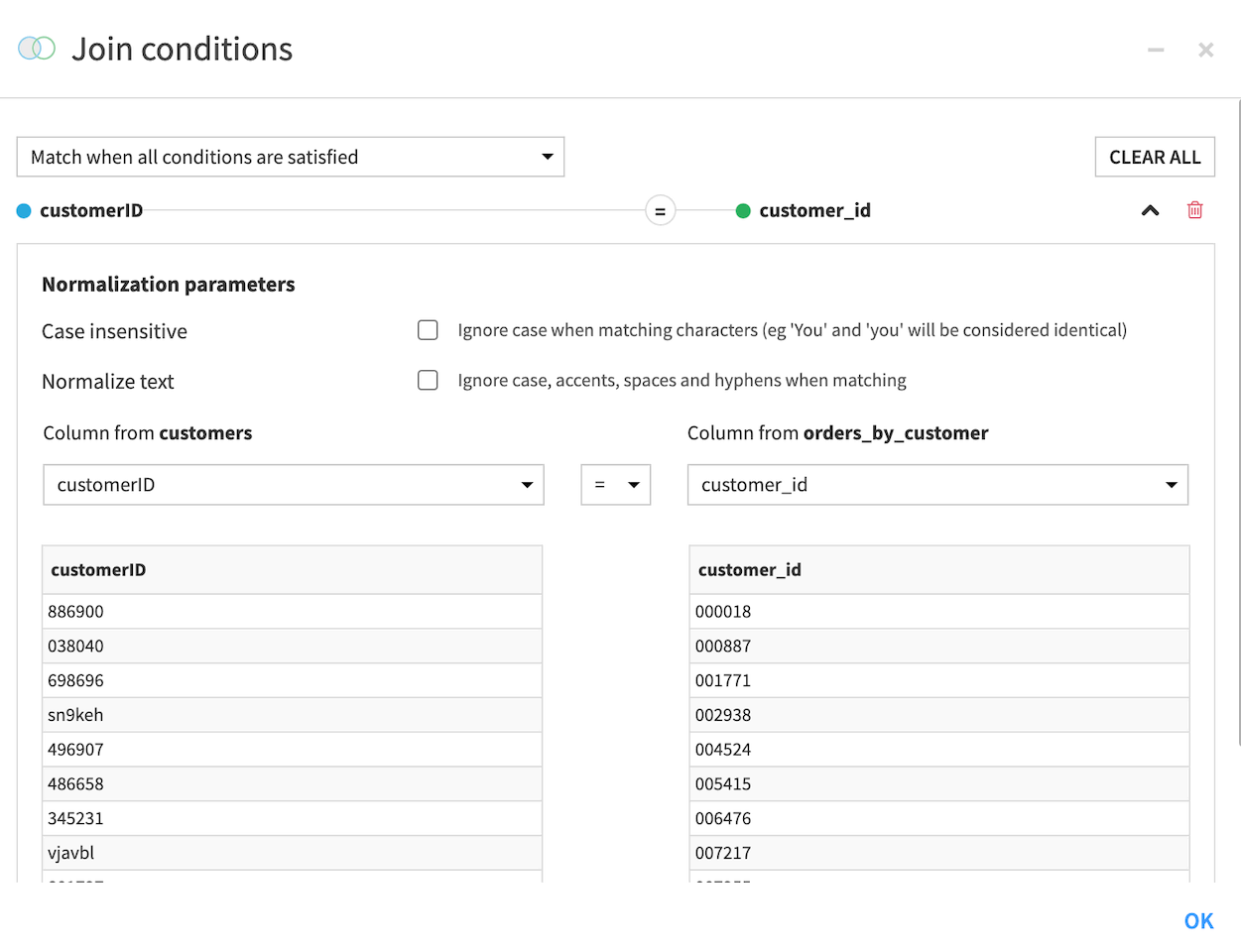
Note

Notice the **+** button at the top right of each dataset in the Join step. You can use this button to add more datasets to join with the *customers* and the *orders\_by\_customer* datasets.

1. Click on **Add a Condition** to tell Dataiku which columns to match.

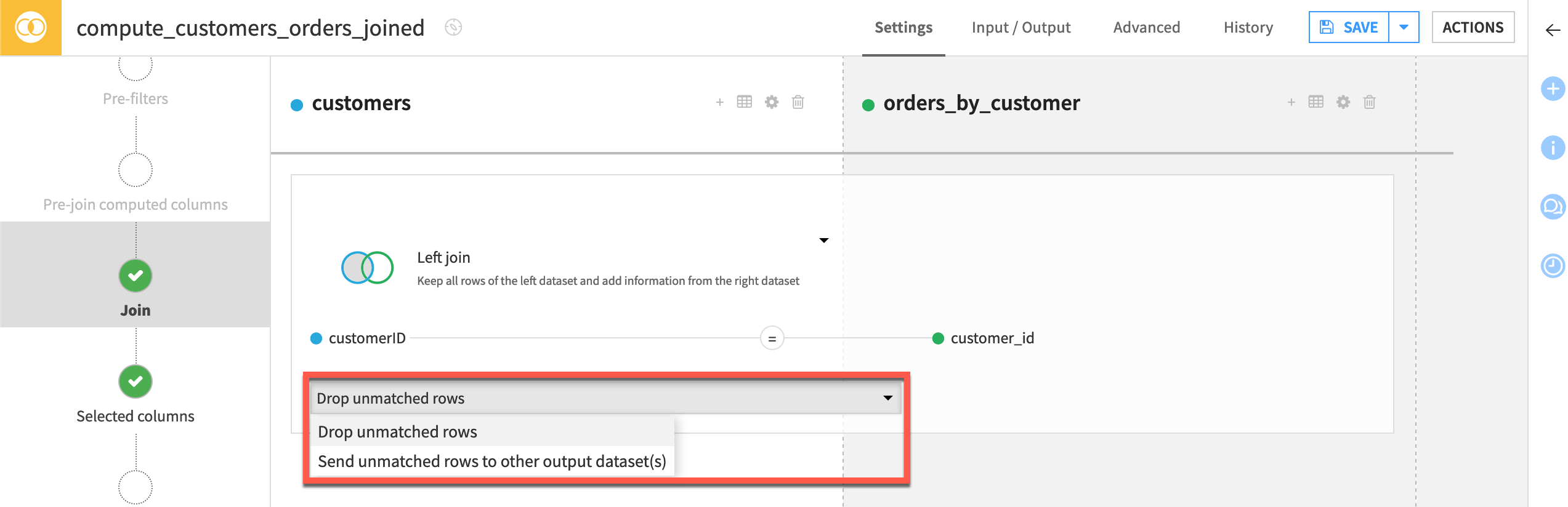
The **Join conditions** dialog opens, where Dataiku automatically recognizes that the ID columns are the join key, even though they have different names. This is the only condition we need to add here.

1. Select **OK** and return to the Join recipe.

[](https://academy-content.dataiku.com/latest/_images/join-conditions-dialog.png)

Note

By default, the recipe will drop unmatched rows, but we can also configure the recipe to send unmatched rows to another output dataset. This can be helpful to confirm your expectations of the output.

[](https://academy-content.dataiku.com/latest/_images/join-drop-unmatched-rows.png)

#### Set the join type

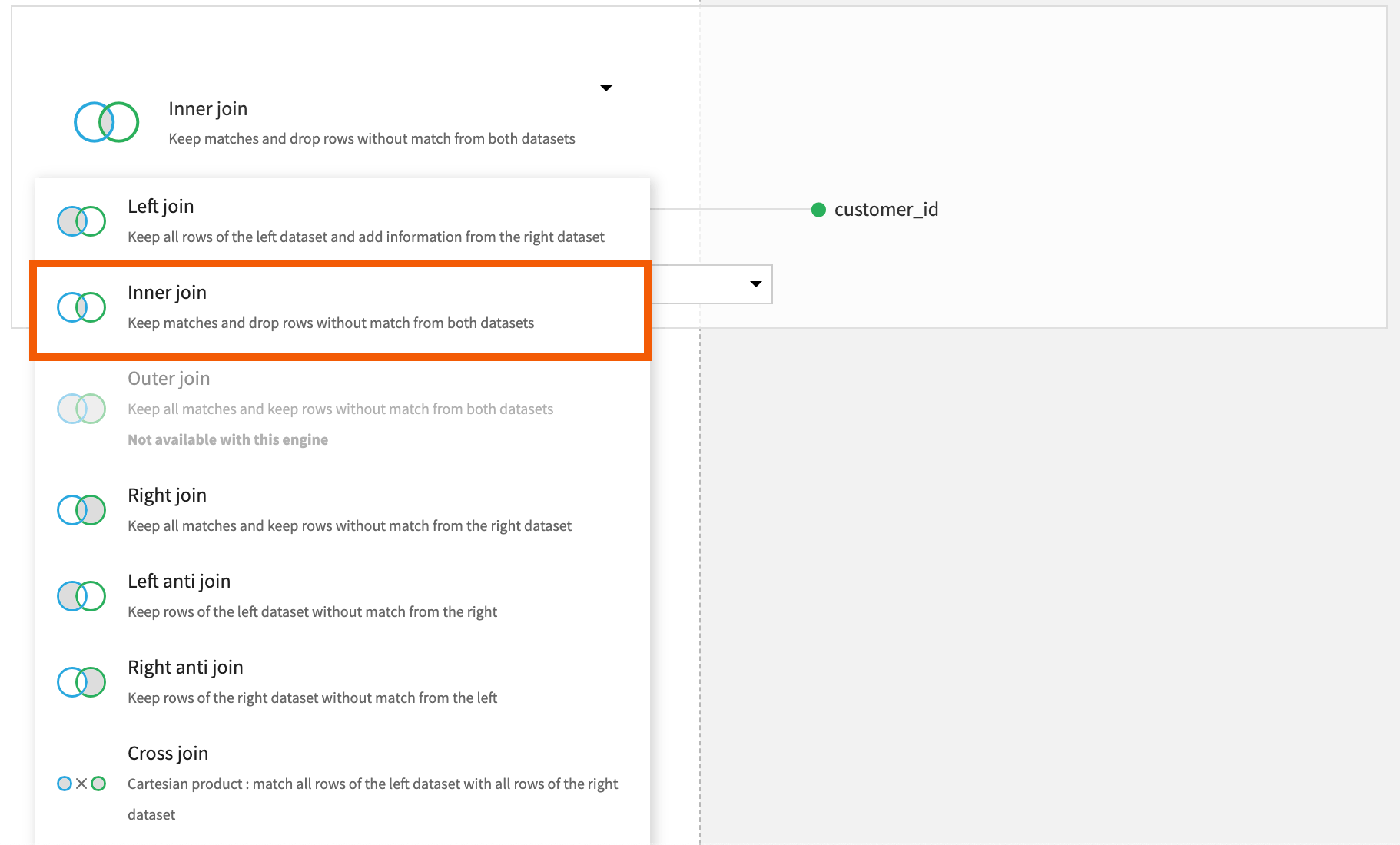
By default, the Join recipe performs a left join, which retains all rows in the left dataset, even if there is no matching information in the right. Since in this case we only want to work with customers who have made at least one order, let’s modify the join type to inner join, which keeps only the rows that match in both datasets.

Note

For more information about join types, visit [Concept | Join recipe](https://academy-content.dataiku.com/latest/data-preparation/visual-recipes/concept-join-recipe.html?opals=true).

1. Click on the **Left join** indicator.
2. Select **Inner join**.

This will retain only the customers who have made an order, and remove the others from the output dataset.

[](https://academy-content.dataiku.com/latest/_images/join-type.png)

#### Define output columns

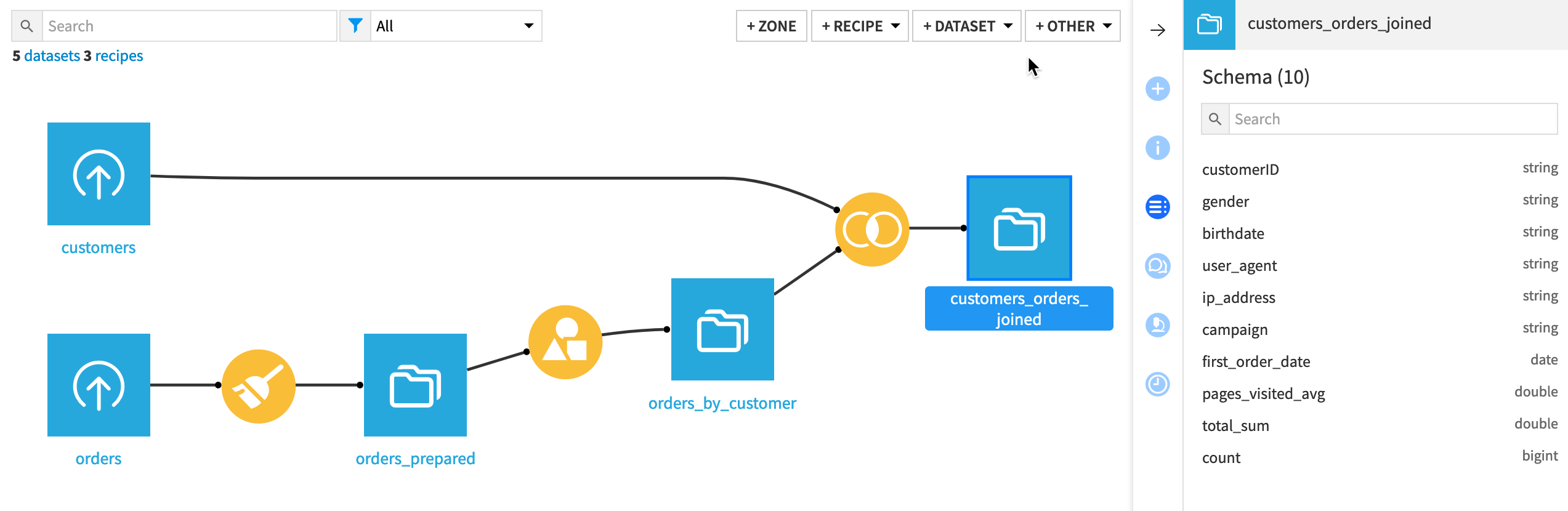
The **Selected columns** step of the Join recipe allows you to choose which columns to retain from the input datasets.

In this tutorial, let’s skip this step and keep all the columns.

**Execute the recipe and explore the output**

Now, let’s run the recipe and view the output.

1. Click **Run** to execute the recipe.
2. When the recipe has finished running, click **Explore dataset customers\_orders\_joined** at the bottom of the screen to explore the **customers\_orders\_joined** dataset.
3. From the top navigation bar, go back to the Flow to check your progress.

[](https://academy-content.dataiku.com/latest/_images/customers-orders-joined.png)

See also

For more information on the Join recipe, see also the [Join: joining datasets](https://doc.dataiku.com/dss/latest/other_recipes/join.html) article in the reference documentation.

So far all of your work has been in the Flow. Now, it’s time to learn about the statistics!

# Explore the Flow

# Tutorial | Explore the Flow[#](https://academy-content.dataiku.com/latest/getting-started/dataiku-ui/tutorial-explore-flow.html?opals=true#tutorial-explore-the-flow)

The **Flow** is the visual representation of your project pipeline.

After following the previous tutorials on [preparing the data](https://academy-content.dataiku.com/latest/data-preparation/prepare-recipe/tutorial-prepare-data.html?opals=true), [grouping the data](https://academy-content.dataiku.com/latest/data-preparation/visual-recipes/tutorial-group-data.html?opals=true) and [enriching it](https://academy-content.dataiku.com/latest/data-preparation/visual-recipes/tutorial-enrich-dataset.html?opals=true), our project contains a few datasets and recipes, and so let’s review the Flow.

#### Objectives

In this tutorial, you will:

* Learn about features for understanding a Flow.
* Understand Dataiku’s visual grammar.

#### Navigate the Flow

There are several ways to navigate to the Flow. You can do it by:

* Clicking the **Go to Flow** button on the project homepage.
* Selecting the first menu to the right of the project title in the top navigation bar.
* Using the keyboard shortcut G+F.

#### See also

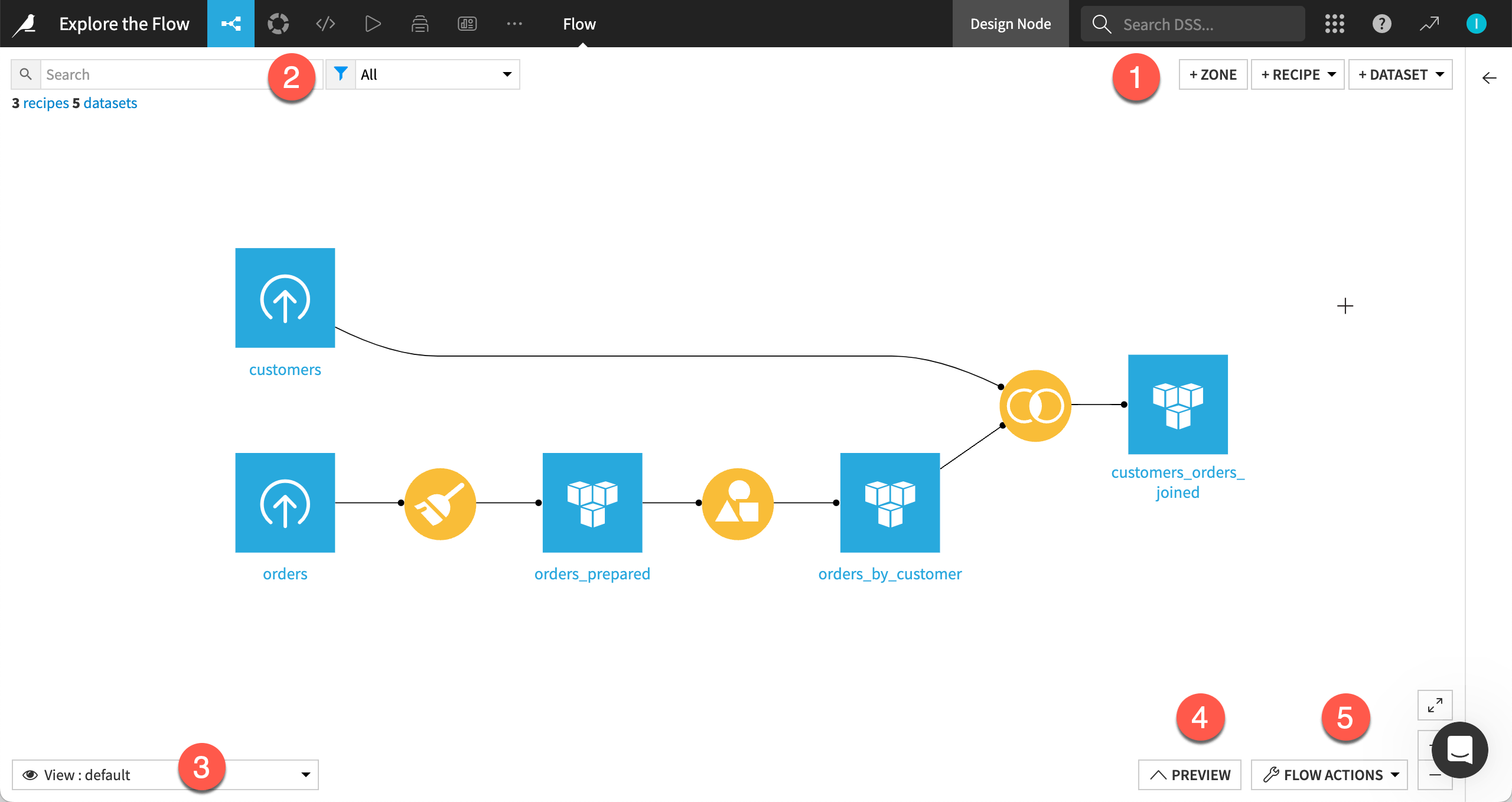
See the [reference documentation](https://doc.dataiku.com/dss/latest/accessibility/index.html) to get a full list of keyboard shortcuts.

In this tutorial, let’s follow this procedure:

1. Click on the **Flow** icon next to the project name in the top navigation bar to go to the project **Flow**.
2. Explore it.

As you can see, the Flow here is quite basic, but it can be helpful to be aware of a few features for navigating more advanced Flows:

* Flow zones (1) for separating stages of a project.
* The search box and filter icon (2) to find objects.
* The Flow views menu (3) for many different thematic visualizations of your project.
* The Preview menu (4) that allows previewing a dataset when you select one of them.
* The Flow actions menu (5) for building and exporting the Flow.

[](https://academy-content.dataiku.com/latest/_images/flow-navigation.png)

#### Explore Dataiku’s visual grammar

Take some time to study the Flow. You’ll notice that shapes and colors have specific meanings in Dataiku.

* A blue square represents a dataset.
* A yellow circle represents a visual recipe.
* The icons on the objects indicate the type of dataset or recipe.

#### Note

As you build more advanced projects, you’ll see more of this visual grammar on display:

* A diamond shape for machine learning models.
* The color green for all machine learning elements.
* Orange circles for code recipes and (often) red for plugin recipes.
* Black squares for datasets shared from other projects.

#### What’s next?

Congratulations on building your first Flow! Soon you’ll build much more complicated data pipelines with new kinds of objects, such as models.

Continue to [Basics 103](https://academy.dataiku.com/basics-103) to learn more about statistics and dashboards (our reporting tool).