Merging and splitting data

# Merging and splitting data#

In this chapter, you will learn how to merge and split data, and in what cases it might be useful to perform these operations.

## Merging data#

In some cases, you might need to merge (combine) and process data from different sources.

Merging data can involve:

• Creating one data set from multiple sources.

• Synchronizing data between multiple systems. This could include removing duplicate data or updating data in one system when it changes in another.

One-way vs. two-way sync

In a one-way sync, data is synchronized in one direction. One system serves as the single source of truth. When information changes in that main system, it automatically changes in the secondary system; but if information changes in the secondary system, the changes aren't reflected in the main system.

In a two-way sync, data is synchronized in both directions (between both systems). When information changes in either of the two systems, it automatically changes in the other one as well.

This blog tutorial explains how to sync data one-way and two-way between two CRMs.

In n8n, you can merge data from two different nodes using the Merge node, which provides several merging options:

• Append

• Combine  
Merge by Fields: requires input fields to match on  
Merge by Position  
Combine all possible combinations

• Merge by Fields: requires input fields to match on

• Merge by Position

• Combine all possible combinations

• Choose Branch

Notice that Combine > Merge by Fields requires you enter input fields to match on. These fields should contain identical values between the data sources so n8n can properly match data together. In the Merge node, they're called Input 1 Field and Input 2 Field.

Input 1 Field

Input 2 Field

Property Input in dot notation

If you want to reference nested values in the Merge node parameters Input 1 Field and Input 2 Field, you need to enter the property key in dot-notation format (as text, not as an expression).

Input 1 Field

Input 2 Field

Note

You can also find the Merge node under the alias Join. This might be more intuitive if you're familiar with SQL joins.

### Merge Exercise#

Build a workflow that merges data from the Customer Datastore node and Code node.

• Add a Merge node that takes Input 1 from a Customer Datastore node and Input 2 from a Code node.

Input 1

Input 2

• In the Customer Datastore node, run the operation Get All People.

• In the Code node, create an array of two objects with three properties: name, language, and country, where the property country has two sub-properties code and name.  
Fill out the values of these properties with the information of two characters from the Customer Database.  
For example, Jay Gatsby's language is English and country name is United States.

name

language

country

country

code

name

• Fill out the values of these properties with the information of two characters from the Customer Database.

• For example, Jay Gatsby's language is English and country name is United States.

• In the Merge node, try out different merge options.

## Looping#

In some cases, you might need to perform the same operation on each element of an array or each data item (for example sending a message to every contact in your address book). In technical terms, you need to iterate through the data (with loops).

n8n generally handles this repetitive processing automatically, as the nodes run once for each item, so you don't need to build loops into your workflows.

However, there are some exceptions of nodes and operations that will require you to build a loop into your workflow.

To create a loop in an n8n workflow, you need to connect the output of one node to the input of a previous node, and add an If node to check when to stop the loop.

## Splitting data in batches#

If you need to process large volumes of incoming data, execute the Code node multiple times, or avoid API rate limits, it's best to split the data into batches (groups) and process these batches.

For these processes, use the Loop Over Items node. This node splits input data into a specified batch size and, with each iteration, returns a predefined amount of data.

Execution of Loop Over Items node

The Loop Over Items node stops executing after all the incoming items get divided into batches and passed on to the next node in the workflow, so it's not necessary to add an If node to stop the loop.

### Loop/Batch Exercise#

Build a workflow that reads the RSS feed from Medium and dev.to. The workflow should consist of three nodes:

• A Code node that returns the URLs of the RSS feeds of Medium (https://medium.com/feed/n8n-io) and dev.to (https://dev.to/feed/n8n).

https://medium.com/feed/n8n-io

https://dev.to/feed/n8n

• A Loop Over Items node with Batch Size: 1, that takes in the inputs from the Code node and RSS Read node and iterates over the items.

Batch Size: 1

• An RSS Read node that gets the URL of the Medium RSS feed, passed as an expression: {{ $json.url }}.  
The RSS Read node is one of the exception nodes which processes only the first item it receives, so the Loop Over Items node is necessary for iterating over multiple items.

{{ $json.url }}

• The RSS Read node is one of the exception nodes which processes only the first item it receives, so the Loop Over Items node is necessary for iterating over multiple items.