# Generate Primary Keys

**Contents:**

In database terms, a **primary key** is a column or set of columns that uniquely identifies rows in a table. Examples:

- For log data or other transactional data, the timestamp is typically a unique identifier.

  > **Tip:** If you think you need a primary identifier for your dataset, you should try to identify it or create it before you delete potentially useful columns.

- Product information typically contains an SKU identifier. If that is not available, you may need brand, make, and model combinations, which can be created using the method described below.

A well-organized source of data is likely to contain this information for you, but in some cases, you may be required to generate your own primary key.

> **Tip:** In the Transformer page, a quick way to check if there is a primary key in your dataset is to compare the count of categories in the data histograms for string-based data against the count of rows. If the numbers are equal, then the column is suitable for use as a primary key. However, if you ever join with another dataset, you must re-review the suitability of the field and may need to build a new primary key field. Keep in mind that counts apply to the displayed sample, instead of the entire dataset.

This section provides two methods for generating primary keys in your datasets.

## The unique row identifier method

When a dataset is loaded into the Transformer page for the first time, you can see a set of black dots along the left side. Hover over these dots to reveal the row numbers retrieved from the original source, if that information is still available. This method relies on these numbers for generating primary keys and is suitable when your final output contains a relatively few number of combined datasets.

> **NOTE:** Some transforms make original row order information unavailable.You cannot retrieve this information from relational sources.
>
> See *Source Metadata References*.

**Tip:** When you first load your dataset into the Transformer page, you should generate a column containing the original row information, such as the following:

| | |
|---|---|
| **Transformation Name** | `New formula` |
| **Parameter: Formula type** | `Single row formula` |
| **Parameter: Formula** | `SOURCEROWNUMBER()` |
| **Parameter: New column name** | `origRowId` |

This transform is useful to include after initial inference and structuring of each recipe for all of your datasets.

### Standardize formatting

The output of this column is a list of numeric values from 1 or 2 to the final row of your dataset. As a unique identifier, you might want to standardize these values. For example, you are transforming a set of orders. You may want to prepend your unique row identifiers with a code and to format them based on a fixed length, as in the following:

| origRowId | keyPrefix | primaryKey |
|---|---|---|
| 1 | ORD000 | ORD0001 |
| 2 | ORD000 | ORD0002 |
| ... | ORD000 | ... |
| 10 | ORD00 | ORD0010 |
| ... | ORD00 | ... |
| 99 | ORD00 | ORD0099 |
| 100 | ORD0 | ORD0100 |

This structuring generates primary keys of consistent length. You can use the following steps to standardize their formatting, assuming that you have already created the `origRowId` column.

**Steps:**

1. Change this column to be of String type. Select **String** from the data type drop-down for the column.
2. Create a column containing your prepended identifier and the proper number of zeroes. The following bit of logic generates a string with the proper number of zeroes depending on the length of the value in `origRowId`:

| | |
|---|---|
| **Transformation Name** | `New formula` |
| **Parameter: Formula type** | `Single row formula` |
| **Parameter: Formula** | `IF(LEN(origRowId) > 3, 'ORD', IF(LEN(origRowId) > 2, 'ORD0', IF(LEN(origRowId) > 1, 'ORD00','ORD000')))` |
| **Parameter: New column name** | `keyPrefix` |

> **NOTE:** The following works for up to 10,000 rows in the original dataset. You need to add additional `IF` clauses when your row counts exceed 10,000.

3. Now, you can merge these columns together:

| | |
|---|---|
| **Transformation Name** | `Merge columns` |
| **Parameter: Columns** | `keyPrefix,origRowId` |
| **Parameter: New column name** | `primaryKey` |

4. You can now delete the prefix column:

| | |
|---|---|
| **Transformation Name** | `Delete columns` |
| **Parameter: Columns** | `keyPrefix` |
| **Parameter: Action** | `Delete selected columns` |

These steps should be applied across all datasets that you intend to combine into your output dataset.

### Combine across datasets

After you have combined or enriched your dataset, you can combine these original row ID fields from each dataset to create a super primary key in the combined dataset using the method described below.

## The combined field method

If your final dataset contains more than a few combined datasets, this basic method for creating a primary key is to find a combination of fields that collectively represent a unique identifier from the final dataset. Columns:

- LastName
- FirstName
- TestNumber
- TestScore

Since there are multiple instances of test data for each person, there is no single column to use as a primary key.

**Steps:**

1. Load the dataset into the Transformer page.
2. Identify the columns that together can uniquely identifier a row. In the TestScores-All example, these columns are the following:
   a. LastName
   b. FirstName
   c. TestNumber

   > **NOTE:** It may be possible to set up a key using LastName and TestNumber, but that is not guaranteed. If the dataset changes over time, a working key based on these columns may become broken.

3. Use the `merge` transform to combine these columns together into a new column, such as the following:

|  |  |
|---|---|
| **Transformation Name** | `Merge columns` |
| **Parameter: Columns** | `LastName,FirstName,TestNum` |
| **Parameter: Separator** | `'-'` |
| **Parameter: New column name** | `TestID` |

The `with` clause identifies the delimiter between the merged column values.

4. Values should look like the following:

| TestID |
|---|
| Smith-Joe-2 |
| Doe-Jane-4 |
| Jones-Jack-1 |

5. In some cases, you may want to delete the source columns for the primary key.