Improvements to the Type System

Contents:

- *Release 8.11*
  - *Mismatched values are no longer published as null values in CSV outputs*
- *Release 8.2*
- *Release 8.0*
  - *Data type inference and row split inference run on more data*
- *Release 7.5*
  - *PII - Improved matching for social security numbers*
  - *PII - Improved and expanded matching for credit card numbers*
  - *Improved type inference for relational sources*

This section provides information on improvements to the Trifacta® type system.

Release 8.11

Mismatched values are no longer published as null values in CSV outputs

In prior releases, when a file was published in CSV format, any values that were mismatches for a column's data type were written as null values, which could lead to loss of data that was meaningful to downstream systems.

Beginning in this release, mismatched values are written out in CSV format as String values by default.

NOTE: The ability to write out mismatched values in CSV outputs is enabled by default in new flows and CSV publishing actions. For existing CSV outputs, the prior behavior is maintained.

NOTE: This capability applies only to CSV outputs at this time. In the future, it will be applied to other non-schematized outputs, such as JSON.

Tip: When visual profiling is enabled, you can still identify the values in the generated results that are mismatched for their column data types.

As needed, you can configure the ability to write out mismatches for CSV outputs for individual publishing actions. For more information, see *File Settings*.

Release 8.2

None.

Release 8.0

Data type inference and row split inference run on more data

When an dataset is imported into the Trifacta application, a larger volume of data is read from it for the following processes:

NOTE: The following is applied to datasets that do not contain schema information.

- **Split row inference:** Patterns in the data are used to determine the end of a row of data. When a larger volume of data is read, there should be more potential rows to review, resulting in better precision on where to split the data into separate rows in the application.
- **Type inference:** Patterns of data in the same column are used to determine the best Trifacta data type to assign to the imported dataset. A larger volume of data means that the application has more values for the same column from which to infer the appropriate data type.

NOTE: An increased data volume should result in a more accurate split row and data type inferencing. For pre-existing datasets, this increased volume may result in changes to the row and column type definitions when a dataset is imported.

Tip: For datasets that are demarcated by quoted values, you may experience a change in how columns are typed.

If you notice unexpected changes in column data types or in row splits in your datasets:

1. **Type inference:** You should move your recipe panel cursor to the top of the dataset to see if you must reassign data types.
2. **Split row inference:** Create a new imported dataset, disabling type inference in the import settings. Check the `splitrows` transform to see if it is splitting the rows appropriately. For more information, see *Import Data Page*.

Release 7.5

PII - Improved matching for social security numbers

In prior releases, Personally Identifiable Information (PII) for social security numbers was identified based only on the length of values, which matched too broadly.

In this release, the constraints on matching of SSN values has been tightened when applied to PII.

Tip: PII detection is applied in generated log entries and in collaborative suggestions. When matching PII patterns are detected in data that is surface in these two areas, a mask is applied over the values for security reasons.

For more information, see *Social Security Number Data Type*.

For more information, see *Data Type Validation Patterns*.

PII - Improved and expanded matching for credit card numbers

In prior releases, PII for credit card numbers was identified base on 16-digit values.

In this release, the matching constraints have been expanded to include 14-digit credit card values.

Also, the constraints around valid 16-digit numbers have been improved with better recognition around values for different types of credit cards. In the following table, you can see lists of valid test numbers for different credit card types and can see how detection of these values has changed between releases.

Test Number	Credit Card Type	Is Detected 7.4?	Is detected 7.5?

2222 4053 4324 8877	Mastercard	No	Yes
2222 9909 0525 7051	Mastercard	No	Yes
2223 0076 4872 6984	Mastercard	No	Yes
2223 5771 2001 7656	Mastercard	No	Yes
5105 1051 0510 5100	Mastercard	Yes	Yes
5111 0100 3017 5156	Mastercard	Yes	Yes
5204 7400 0990 0014	Mastercard	Yes	Yes
5420 9238 7872 4339	Mastercard	Yes	Yes
5455 3307 6000 0018	Mastercard	Yes	Yes
5506 9004 9000 0444	Mastercard	Yes	Yes
5553 0422 4198 4105	Mastercard	Yes	Yes
5555 5555 5555 4444	Mastercard	Yes	Yes
4012 8888 8888 1881	Visa	Yes	Yes
4111 1111 1111 1111	Visa	Yes	Yes
6011 0009 9013 9424	Discover	Yes	Yes
6011 1111 1111 1117	Discover	Yes	Yes
3714 496353 98431	American Express	Yes	Yes
3782 822463 10005	American Express	Yes	Yes
3056 9309 0259 04	Diners	No	Yes
3852 0000 0232 37	Diners	No	Yes
3530 1113 3330 0000	JCB	Yes	Yes
3566 0020 2036 0505	JCB	Yes	Yes

For more information, see *Credit Card Data Type*.

For more information, see *Data Type Validation Patterns*.

Improved type inference for relational sources

In prior releases, when you generated outputs, the typecasting for the outputs was determined by the data types that were inferred by the application. So, if a column contained only "Yes" or "No" values, then the application is likely to have inferred the column data type as Boolean.

The above presented problems for relational sources for the following reasons:

1. Relational sources could include schema information, which should override what the application inferred. Type inferencing can optionally be disabled for schematized sources from with in the application. For more information on disabling type inference for a schematized source, see *Import Data Page*.
2. For some relational sources, datasets are ingested into the backend datastore and stored there as CSV files. These CSV files are then used as the source for the imported datasets. In these cases, the original source schema information was lost, which meant that the application's type inference was applied. This applied to the following data sources:
    a. Snowflake
    b. Redshift
    c. SQL Datawarehouse
    d. Alation
    e. Waterline

Beginning in this release, the schemas from relational datasources that are ingested to the backend datastore are now used for generated outputs, unless the type was being forcibly set to something else during the recipe step. At the time of original import, the schema of the relational datasource is stored as part of the ingest process; this schema is stored separately.

> **NOTE:** If you created recipe steps that forcibly change a column's data type from within the application to be different from the source data type of your relational source, you may need to revise these recipe steps or remove them altogether.

During publication, Dataprep by Trifacta maps its internal data types to the data types of the publishing target using an internal mapping per vendor. For more information, see *Type Conversions*.