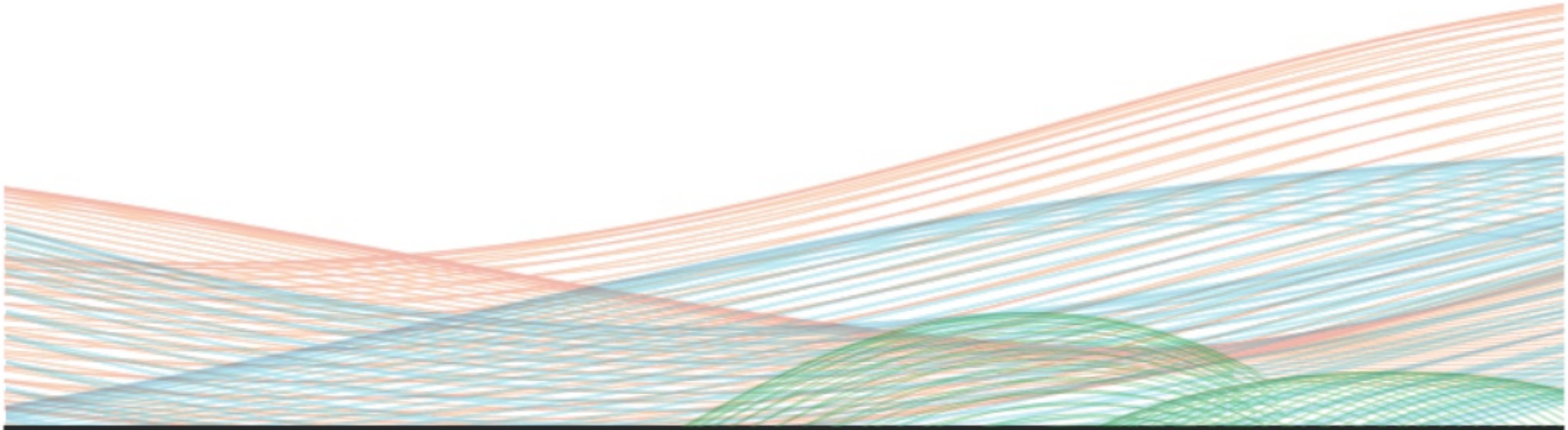


TRIFACTA

Amelia Arbisser
Adam Silberstein

Scalable and Incremental Data Profiling with Spark

Trifacta: Self-service data preparation

What data analysts hope to achieve in data projects

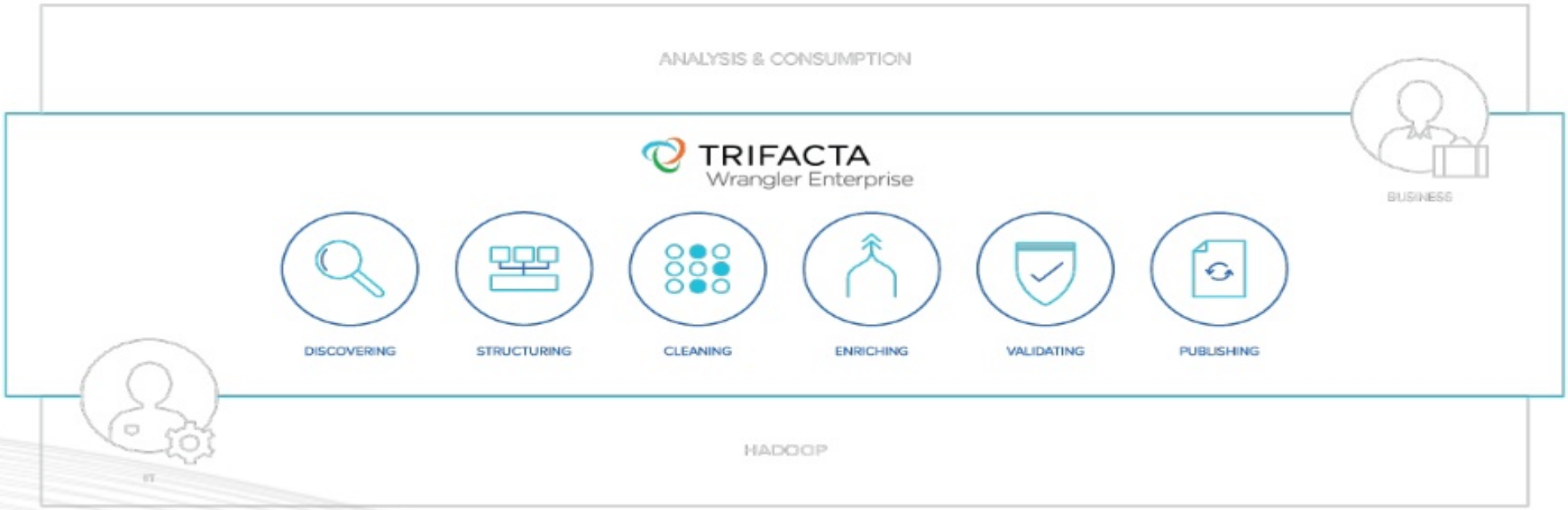
Trifacta: Self-service data preparation

What data analysts hope to achieve in data projects

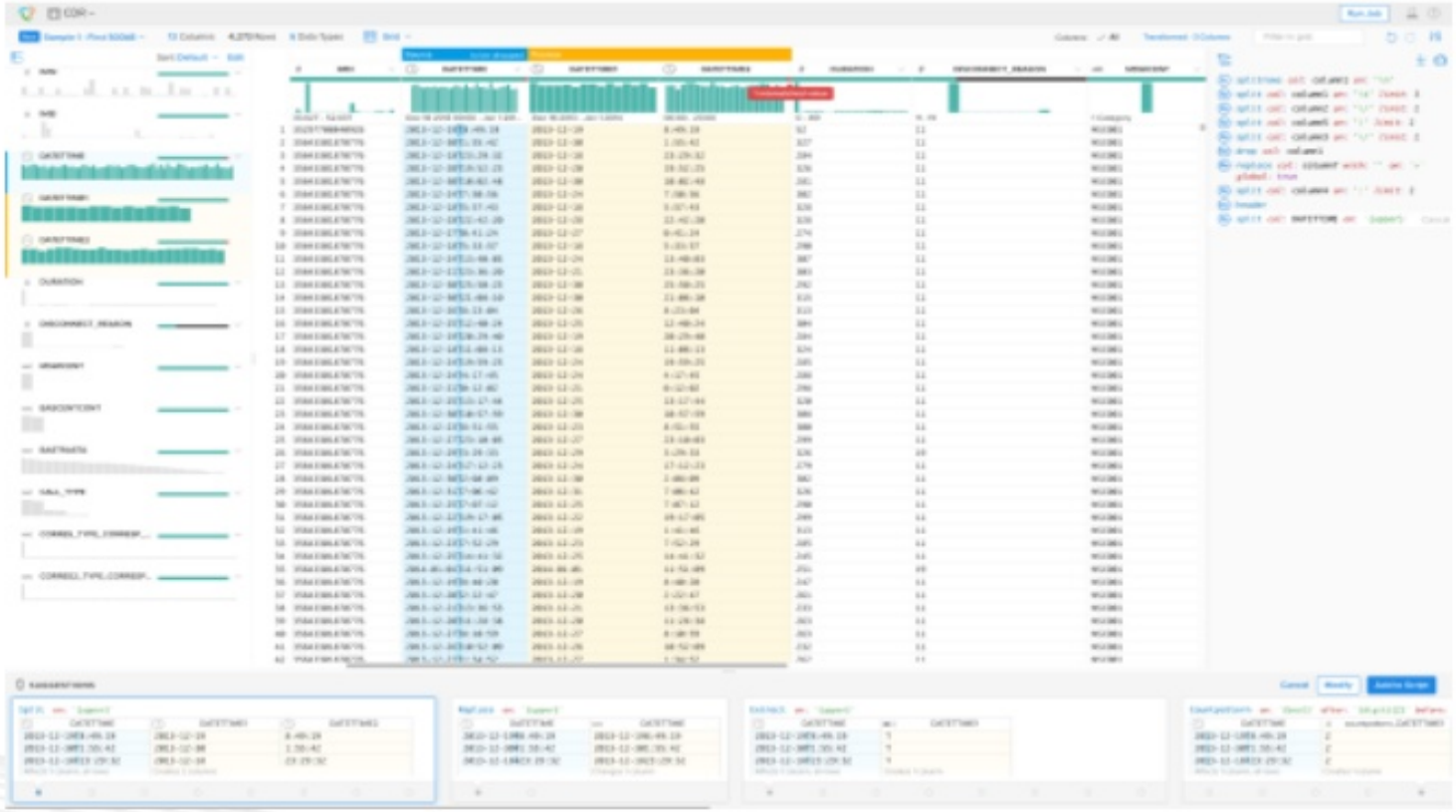
***80%** of time spent cleaning and preparing the data to be analyzed*

Trifacta: Self-service data preparation

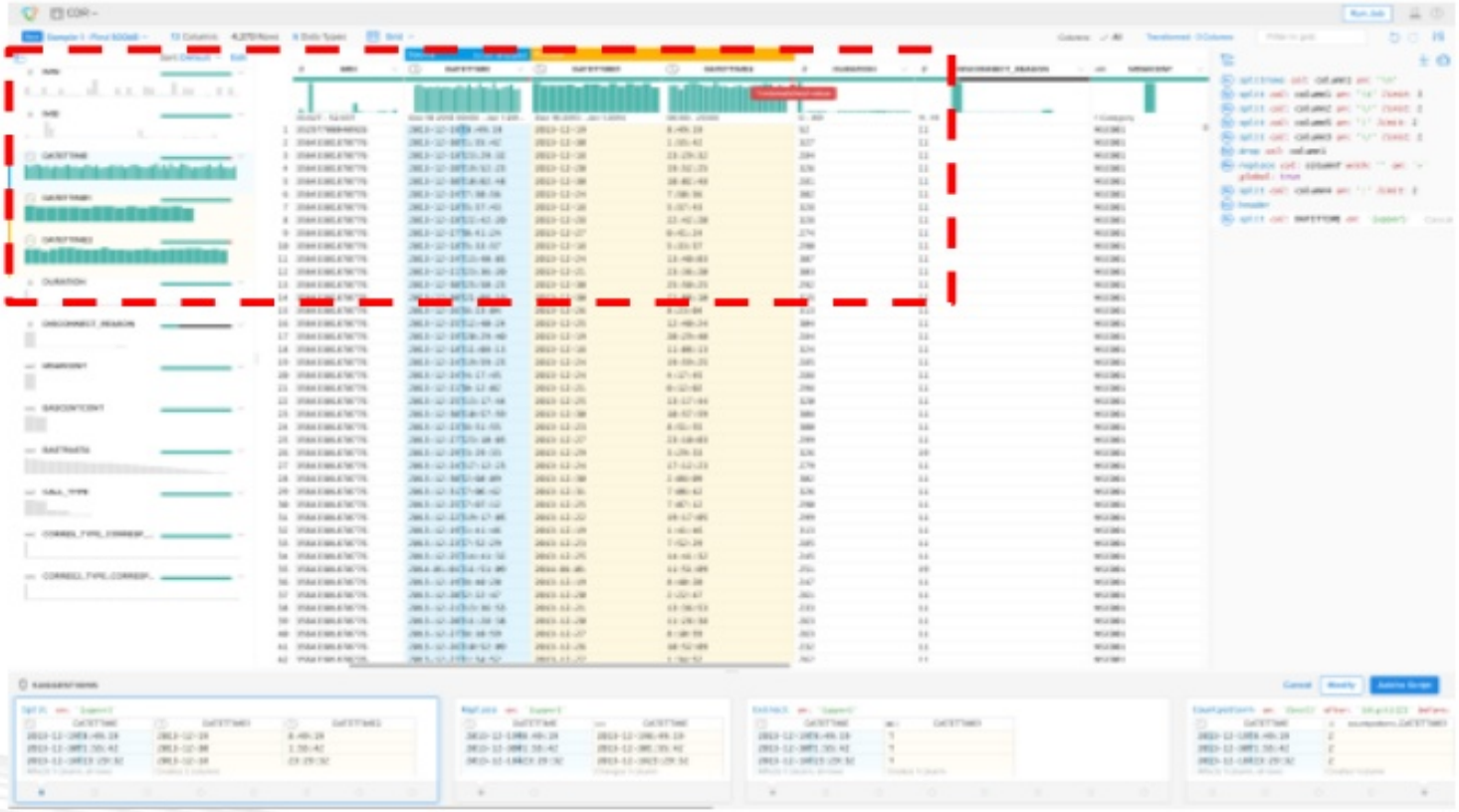
We speed up and take the pain out of preparation



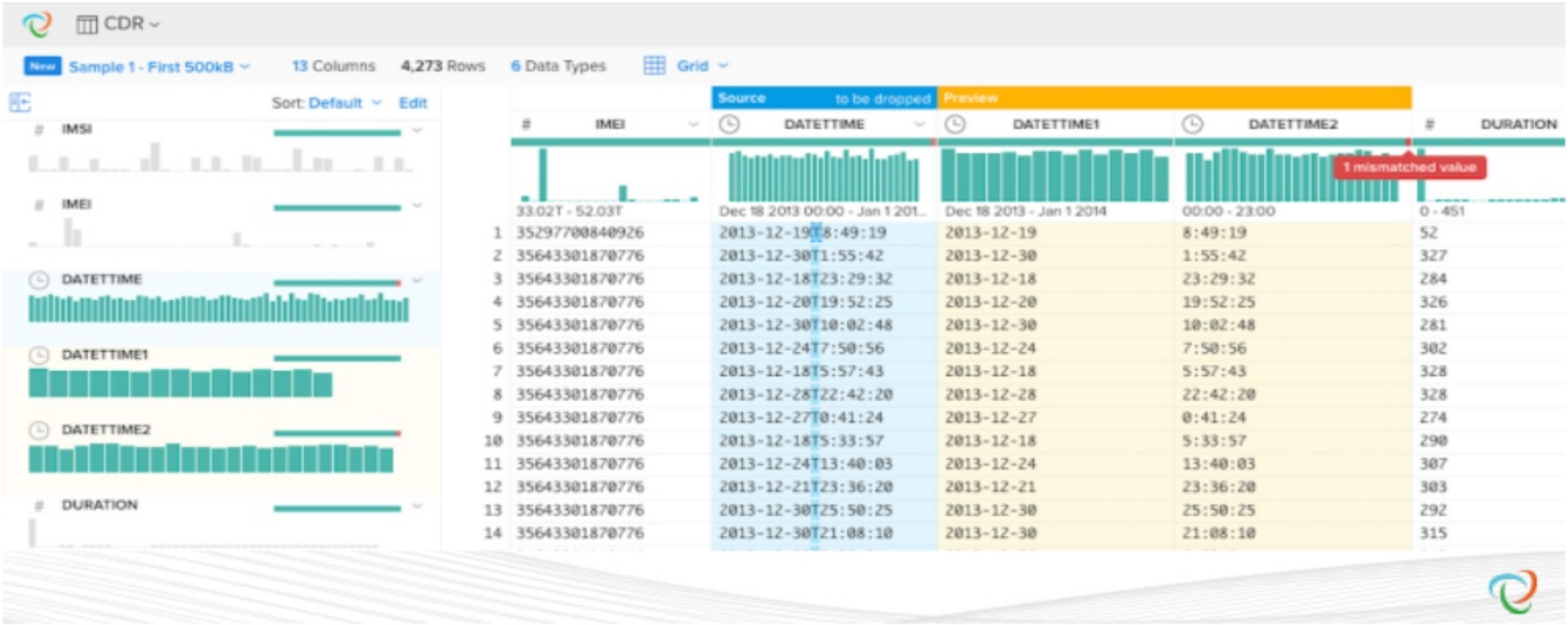
Transforming Data in Trifacta



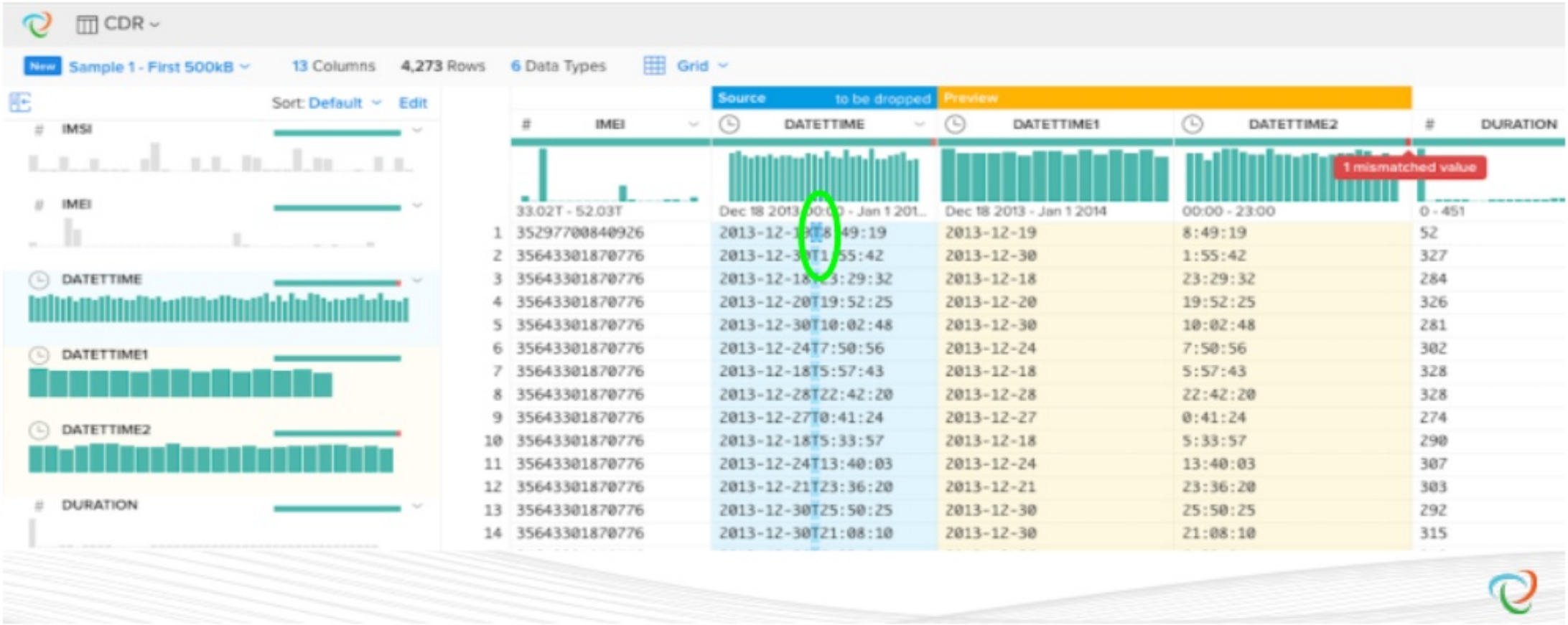
Transforming Data in Trifacta



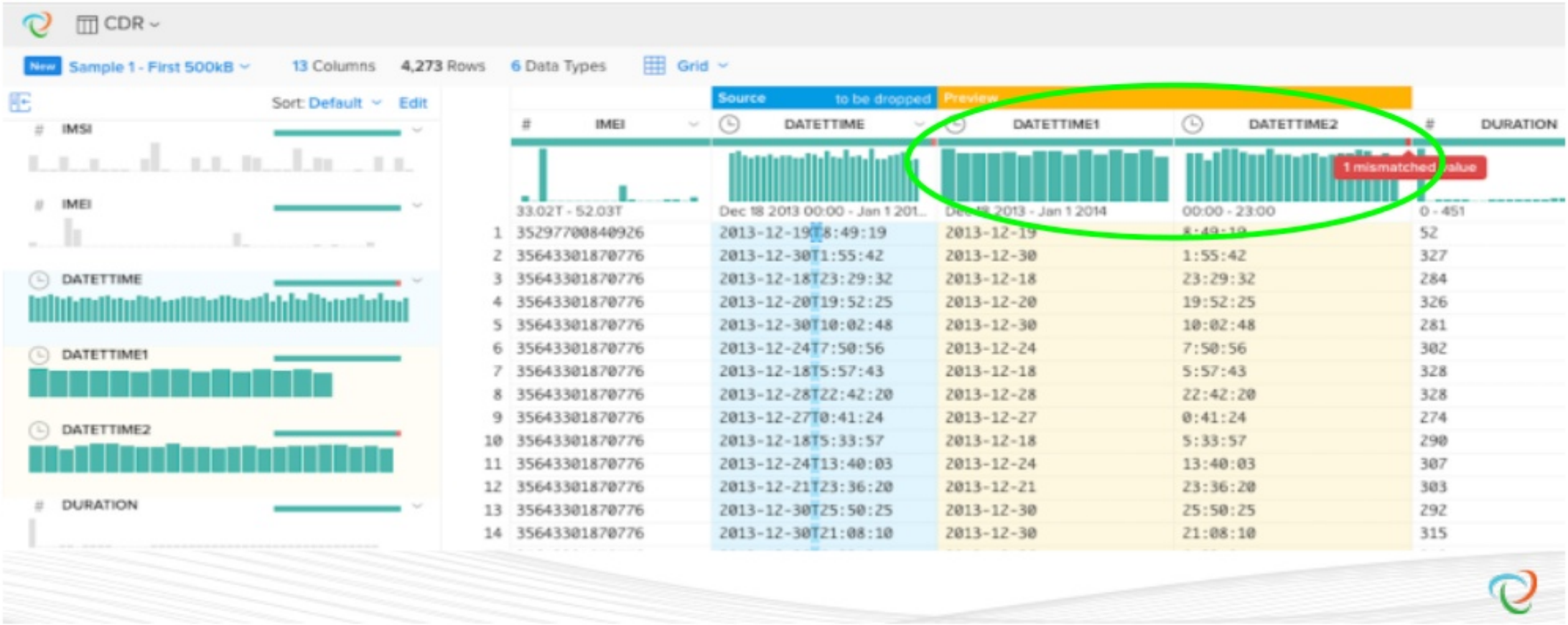
Predictive Interaction and Immediate Feedback



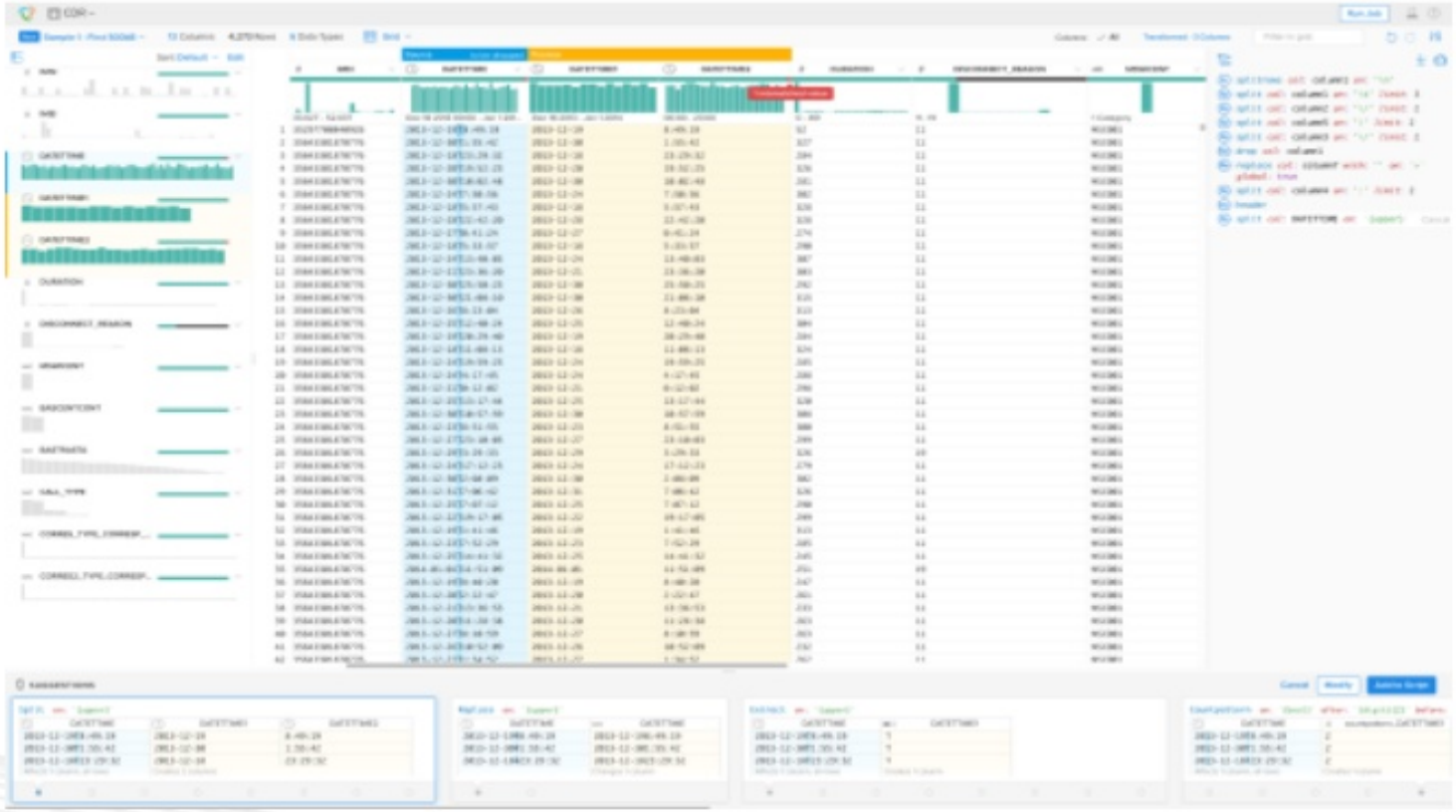
Predictive Interaction



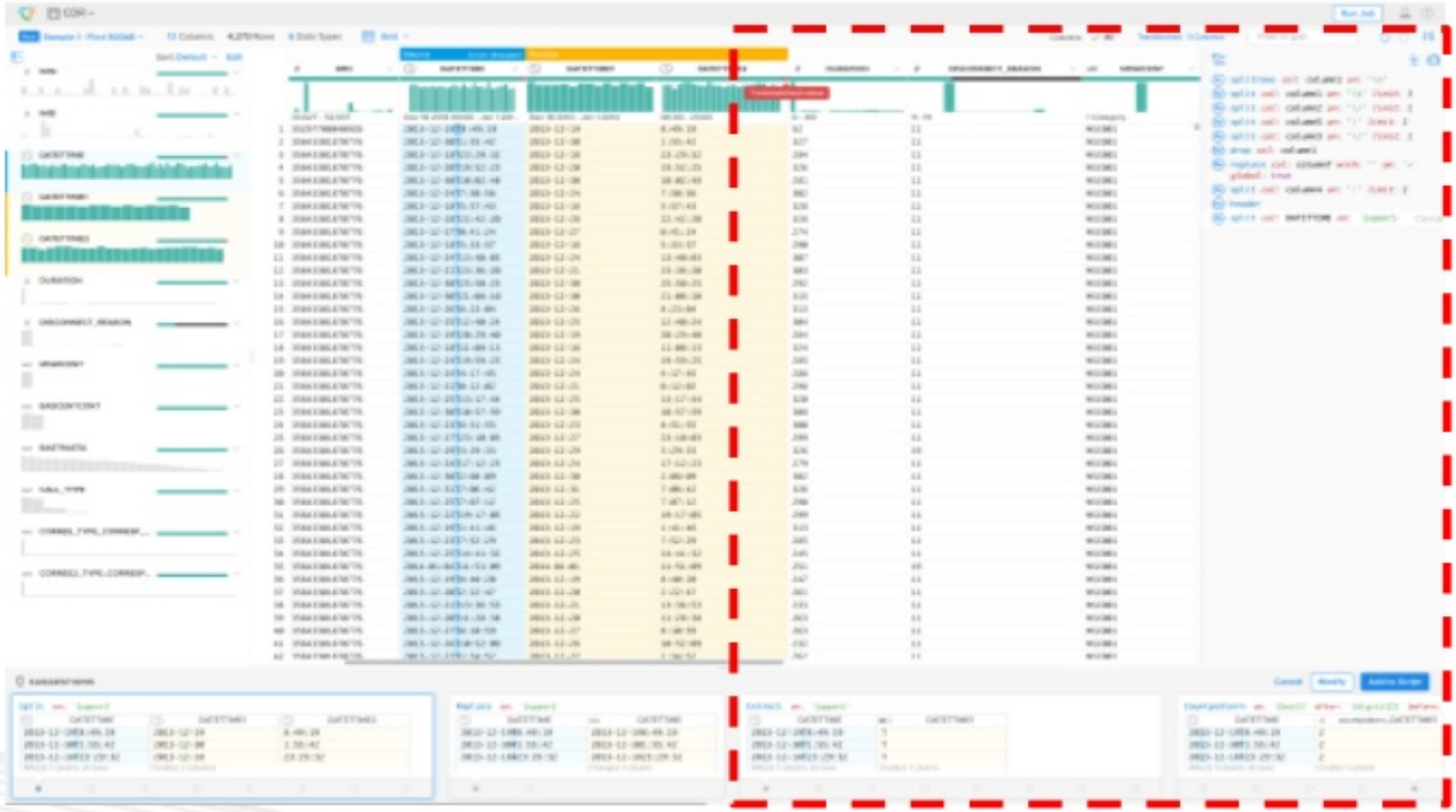
Immediate Feedback

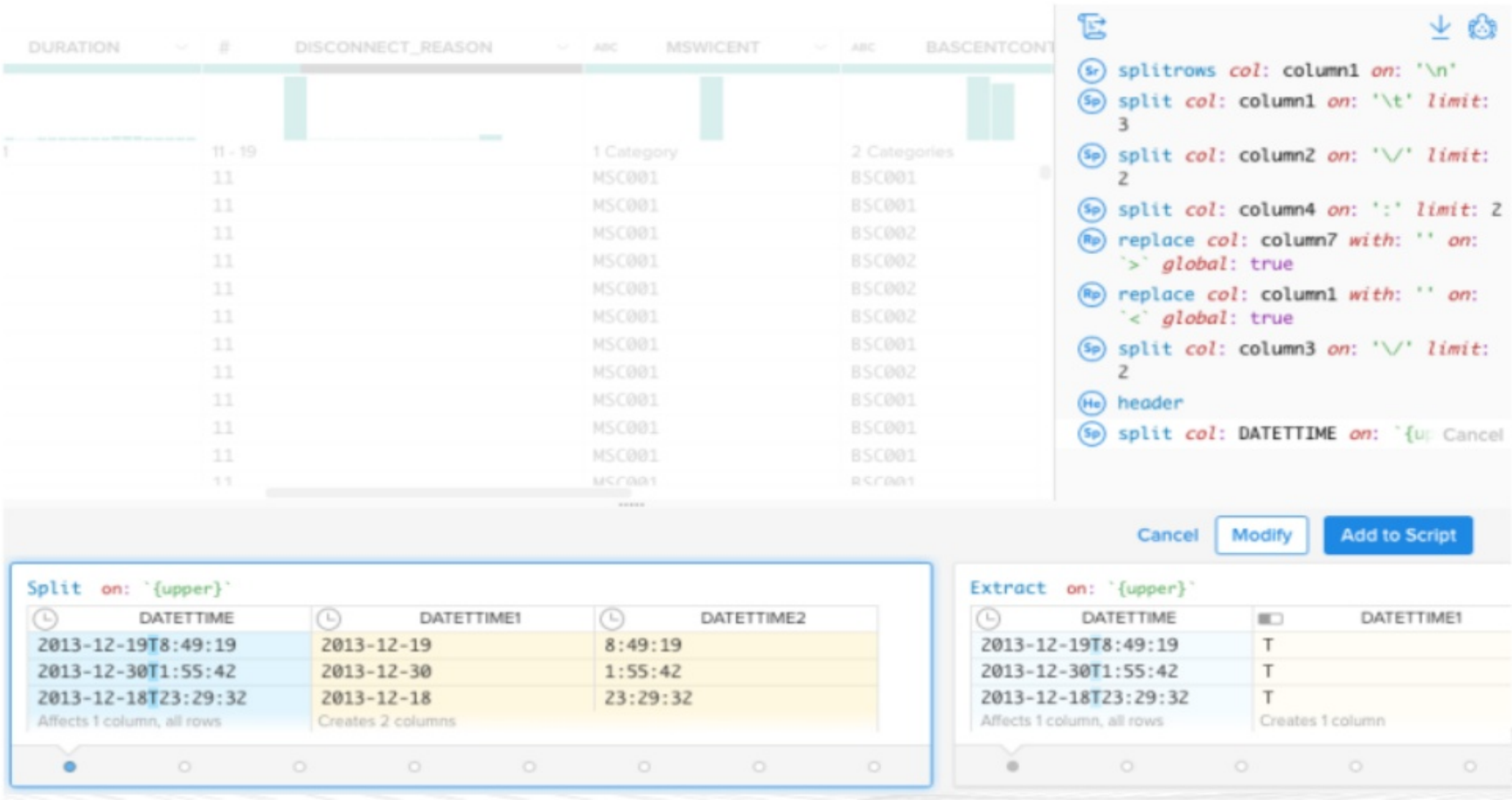


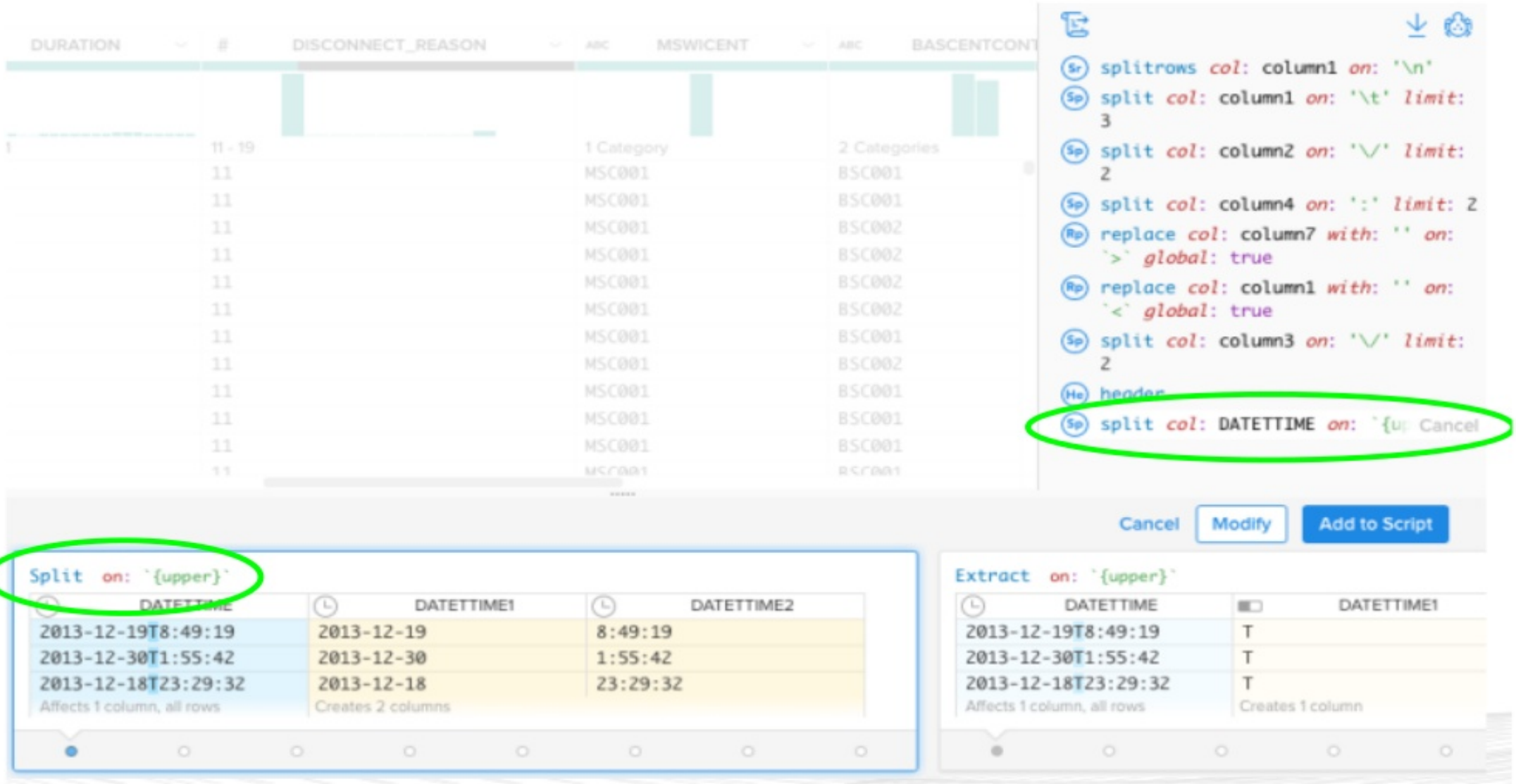
Transforming Data in Trifacta



Transforming Data in Trifacta

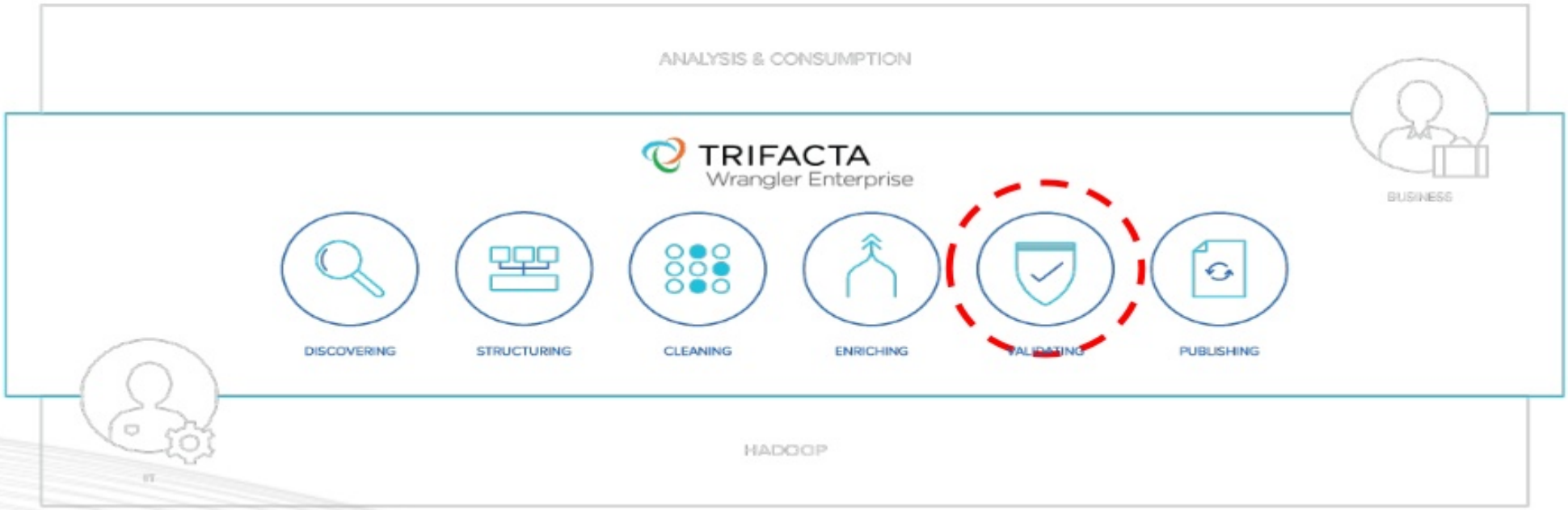



DURATION
#
DISCONNECT_REASON
MSWICENT
BASCENTCONT
splitrows col: column1 on: '\n'
split col: column1 on: '\t' limit: 3
split col: column2 on: '\/' limit: 2
split col: column4 on: ':' limit: 2
replace col: column7 with: '' on: `>` global: true
replace col: column1 with: '' on: `<` global: true
split col: column3 on: '\/' limit: 2
header
split col: DATETTIME on: `{u
Cancel
Modify
Add to Script
Split on: `{upper}`
DATETTIME
DATETTIME1
DATETTIME2
2013-12-19T8:49:19
2013-12-19
8:49:19
2013-12-30T1:55:42
2013-12-30
1:55:42
2013-12-18T23:29:32
2013-12-18
23:29:32
Affects 1 column, all rows
Creates 2 columns
Extract on: `{upper}`
DATETTIME
DATETTIME1
2013-12-19T8:49:19
T
2013-12-30T1:55:42
T
2013-12-18T23:29:32
T
Affects 1 column, all rows
Creates 1 column


DURATION
#
DISCONNECT_REASON
MSWICENT
BASCENTCONT
splitrows col: column1 on: '\n'
split col: column1 on: '\t' limit: 3
split col: column2 on: '\/' limit: 2
split col: column4 on: ':' limit: 2
replace col: column7 with: '' on: `>` global: true
replace col: column1 with: '' on: `<` global: true
split col: column3 on: '\/' limit: 2
header
split col: DATETTIME on: `{u
Cancel
Modify
Add to Script
Split on: `{upper}`
DATETTIME
DATETTIME1
DATETTIME2
2013-12-19T8:49:19
2013-12-19
8:49:19
2013-12-30T1:55:42
2013-12-30
1:55:42
2013-12-18T23:29:32
2013-12-18
23:29:32
Affects 1 column, all rows
Creates 2 columns
Extract on: `{upper}`
DATETTIME
DATETTIME1
2013-12-19T8:49:19
T
2013-12-30T1:55:42
T
2013-12-18T23:29:32
T
Affects 1 column, all rows
Creates 1 column

Where Profiling Fits In

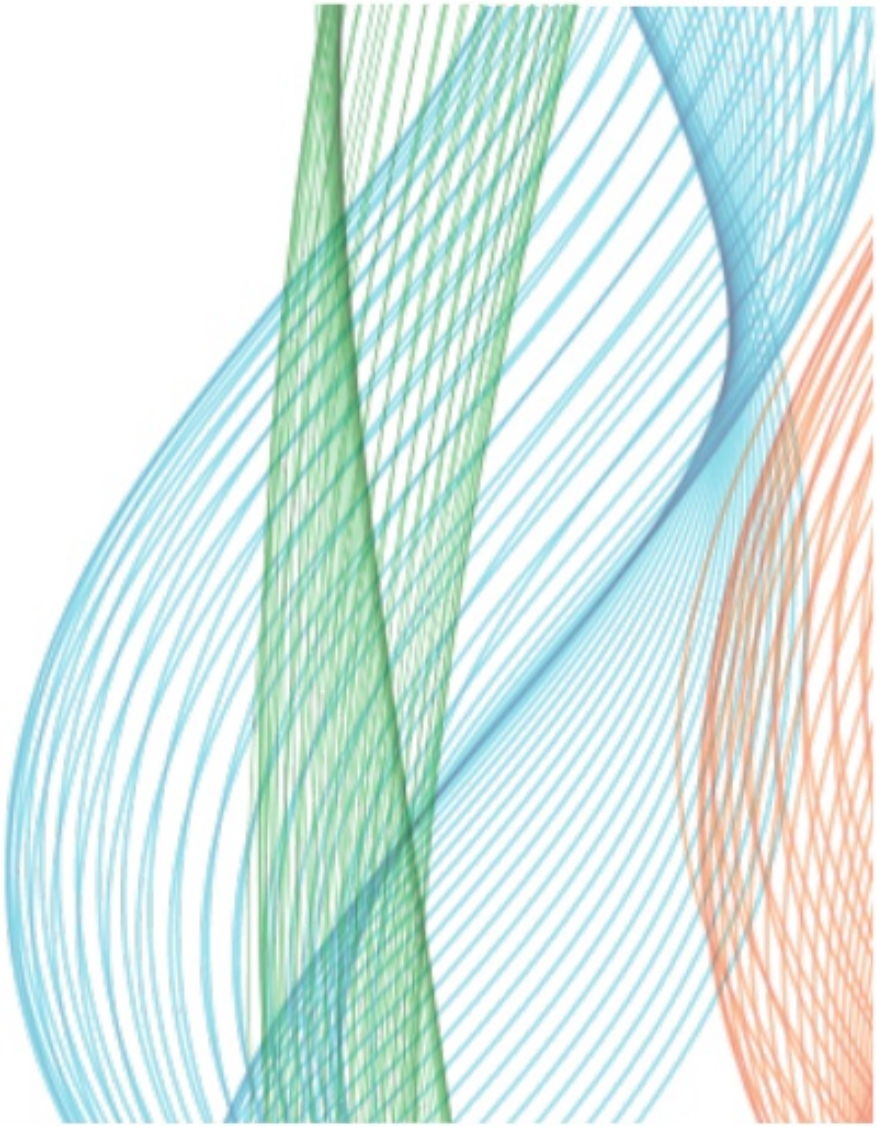
We validate preparation via *profiling*



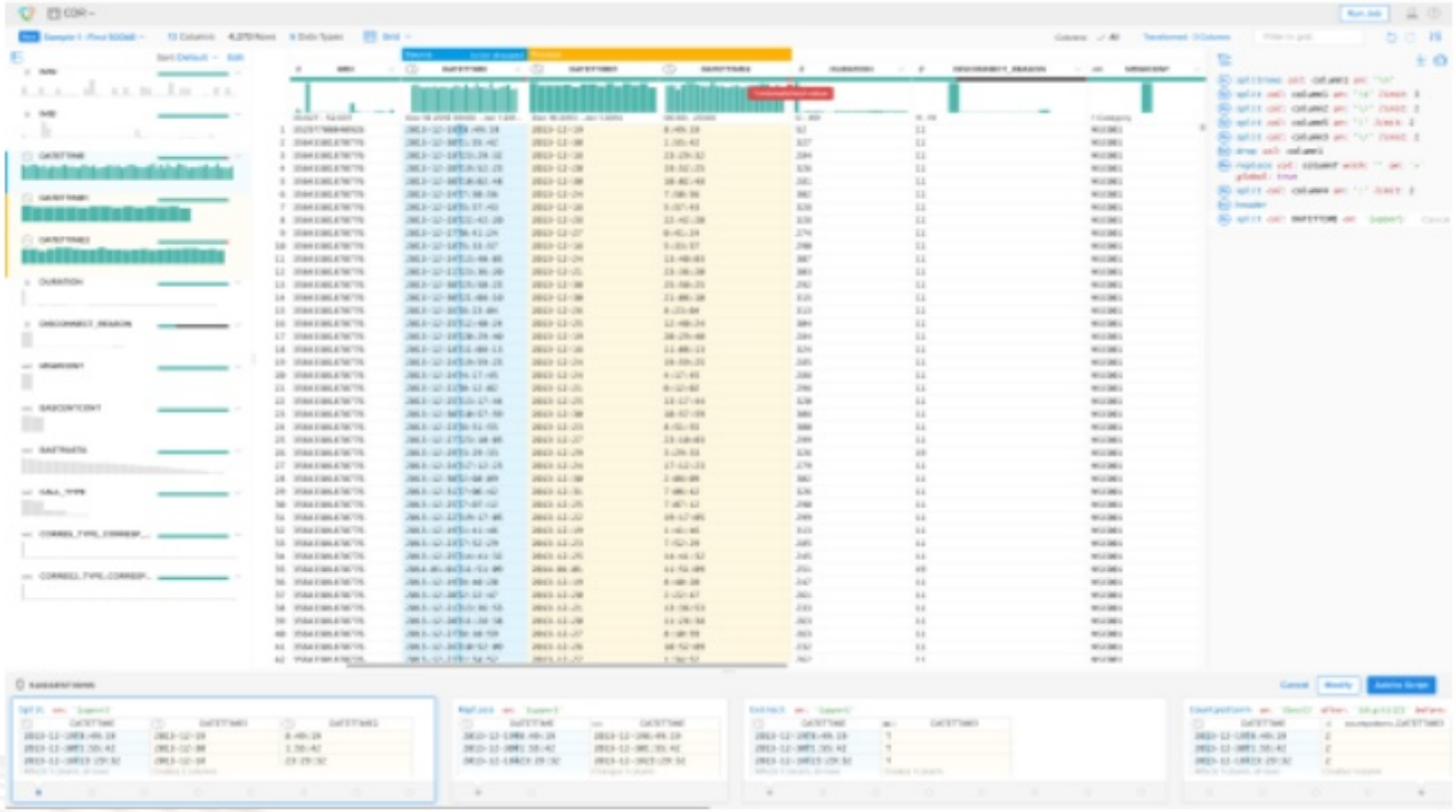
Spark Profiling at Trifacta

- Profiling results of transformation at scale
  - Validation through profiles
- Challenges
  - Scale
  - Automatic job generation
- Our solution
  - Spark profiling job server
  - JSON spec
  - Pay-as-you-go

The Case for Profiling

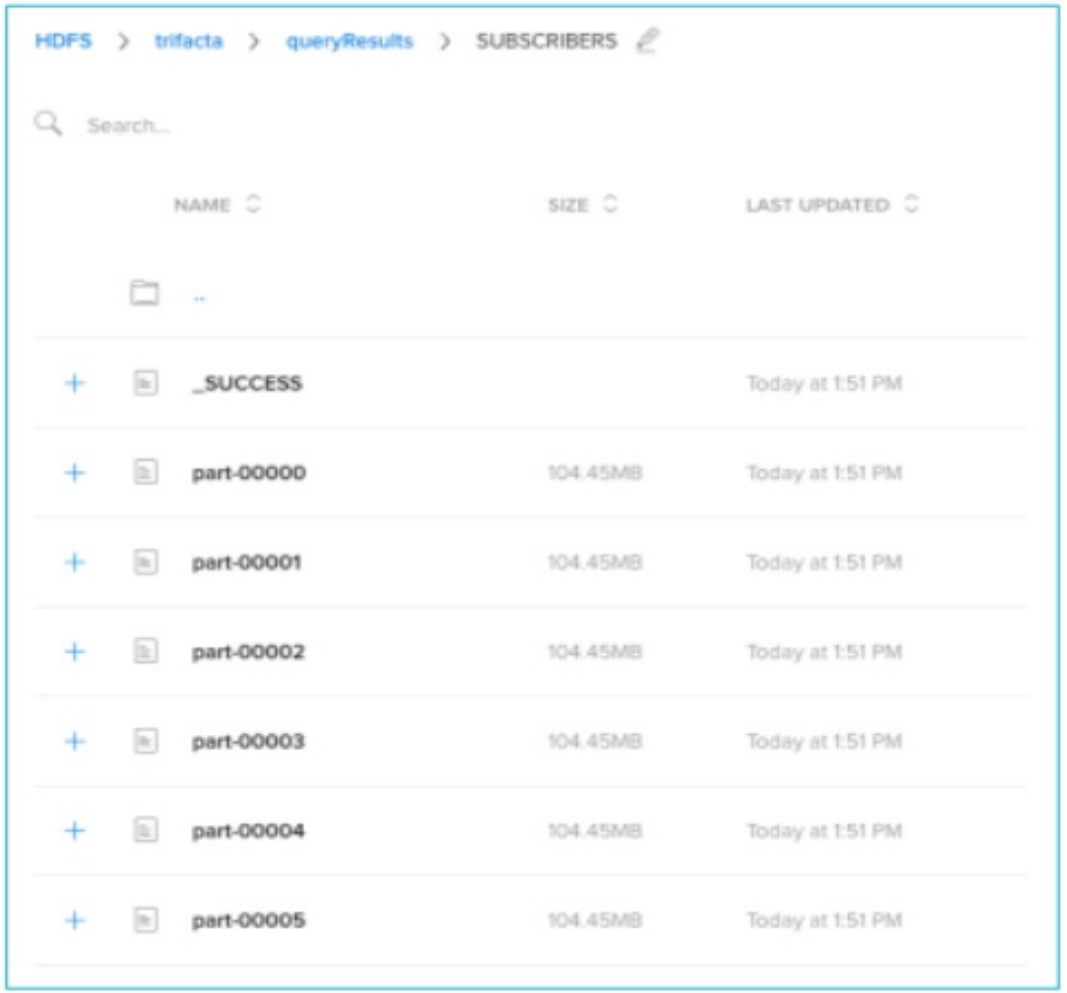


Transforming Data in Trifacta

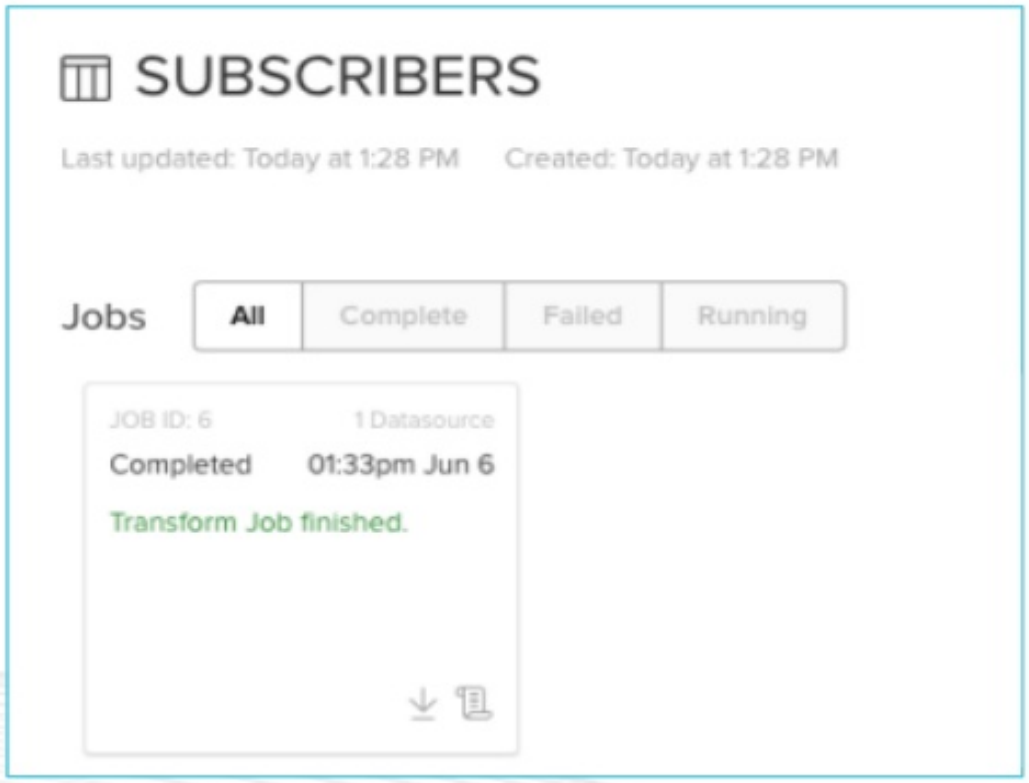


Job Results

- Even clean raw data is not informative.
  - Especially when it is too large to inspect manually.
- Need a summary representation.
  - Statistical and visual
- Generate profile programmatically.

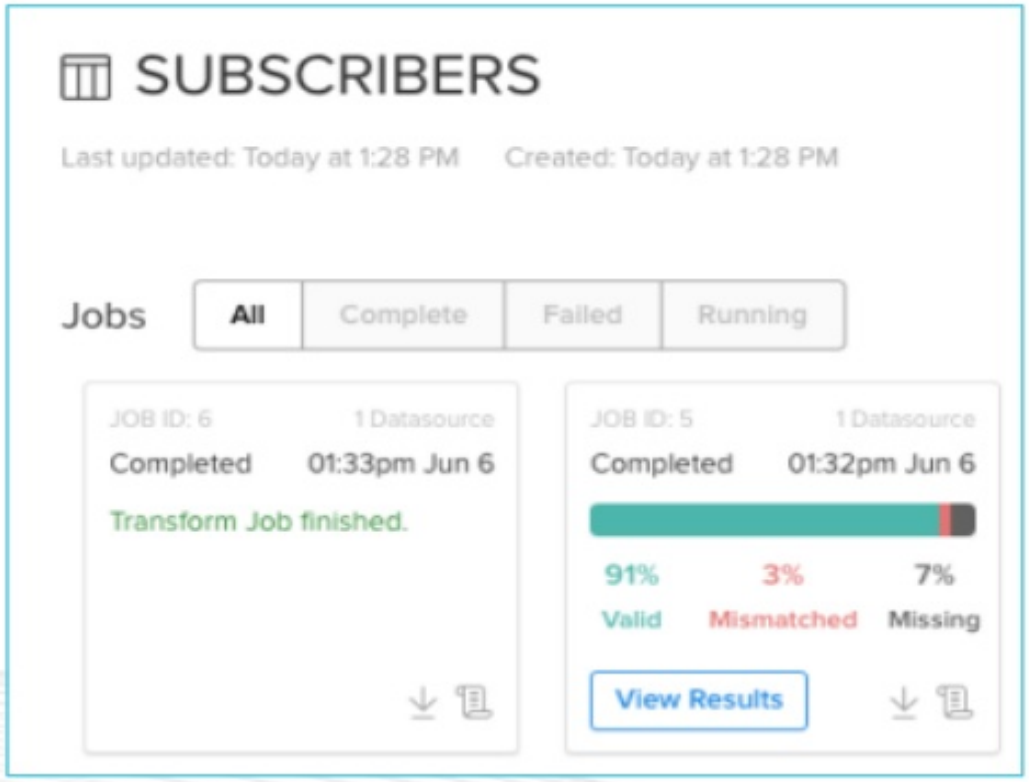


Visual Result Validation



- Similar to the profiling we saw above the data grid.
- Applied at scale, to the full result.
- Reveals mismatched, missing values.

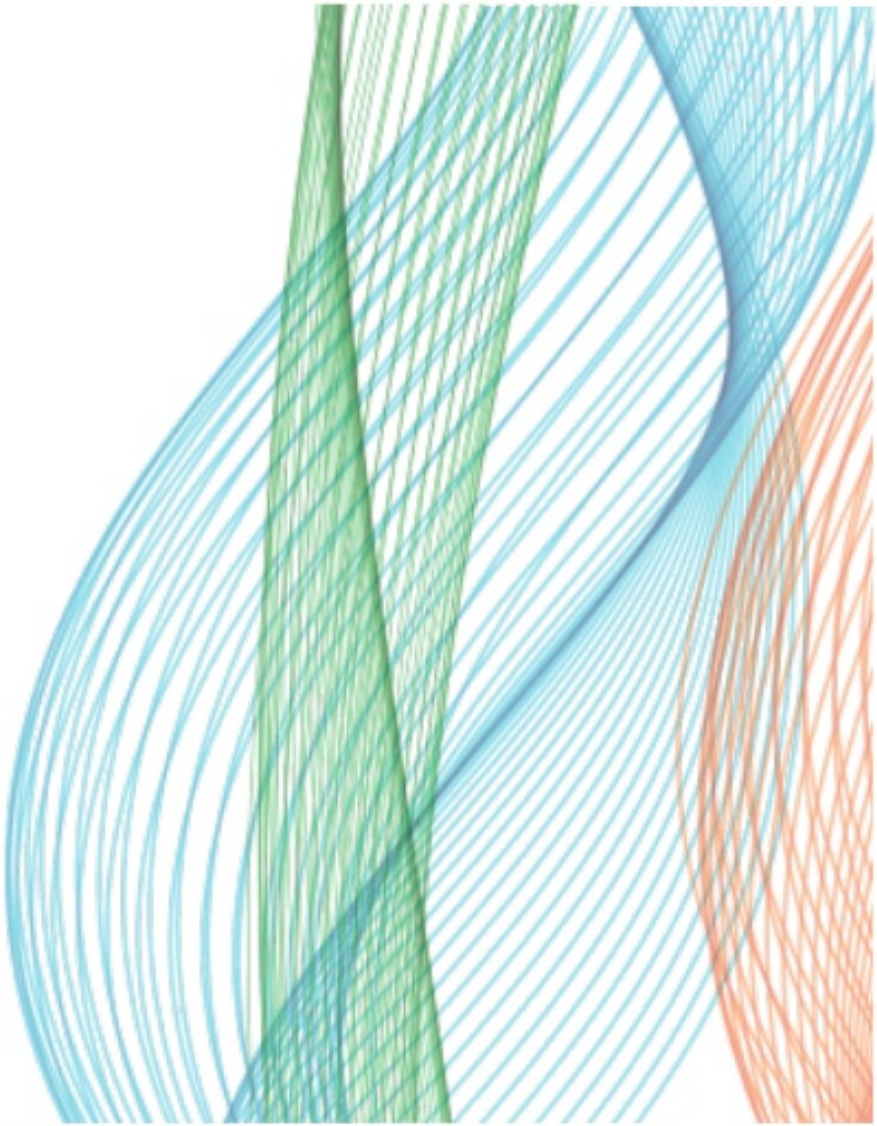
Visual Result Validation



- Similar to the profiling we saw above the data grid.
- Applied at scale, to the full result.
- Reveals mismatched, missing values.

Visual Result Validation

Challenges



The Goal: Profiling for Every Job

- We've talked about the value of profiling, but...
- Profiling is not a tablestakes feature, at least not on day 1.
  - Don't want our users to disable it!
- Profiling is potentially more expensive than transformation.

Profiling at Scale

- Large data volume
  - Long running times
  - Memory constraints

Performance vs. Accuracy

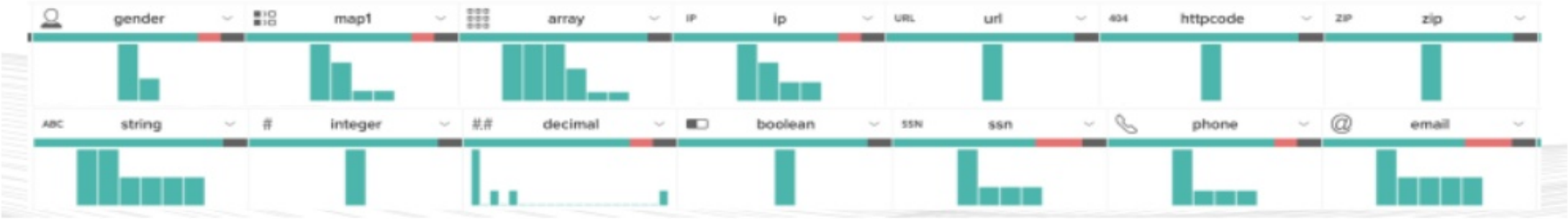
- Approximation brings performance gains while still meeting profiler summarization requirements.
- Off-the-shelf libraries (*count-min-sketch*, *T-digest*) great for approximating counts and non-distributive stats.
- Not a silver bullet though...sometimes approximations are confusing.

Flexible Job Generation

- Profile for all users, all use cases: **not** a one-off
  - Diverse schemas
  - Any number of columns
- Calculate statistics for all data types
  - Numeric vs. categorical
  - Container types (maps, arrays,...)
  - User-defined

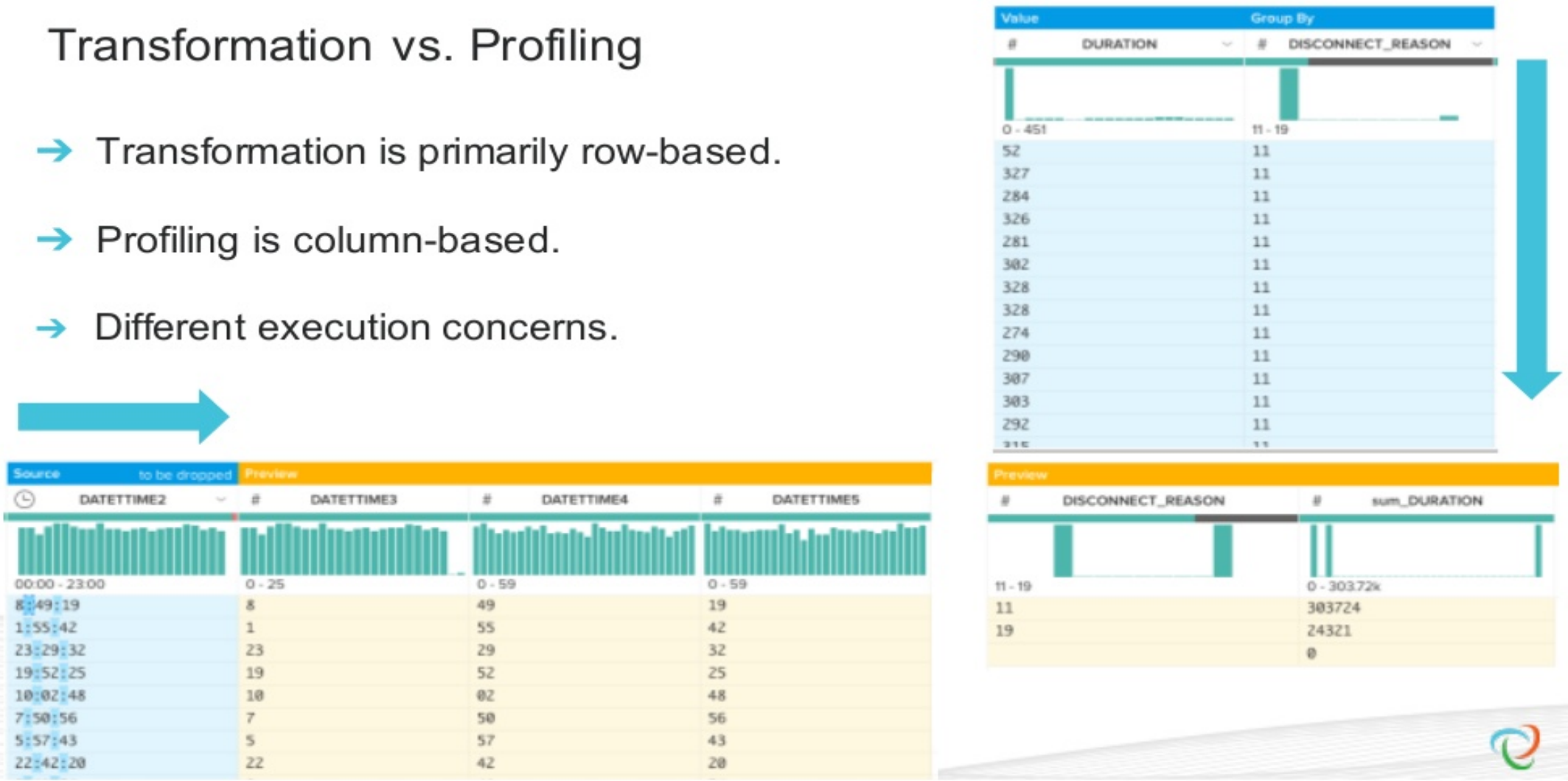
Flexible Job Generation

- Profile for all users, all use cases: **not** a one-off
  - Diverse schemas
  - Any number of columns
- Calculate statistics for all data types
  - Numeric vs. categorical
  - Container types (maps, arrays,...)
  - User-defined

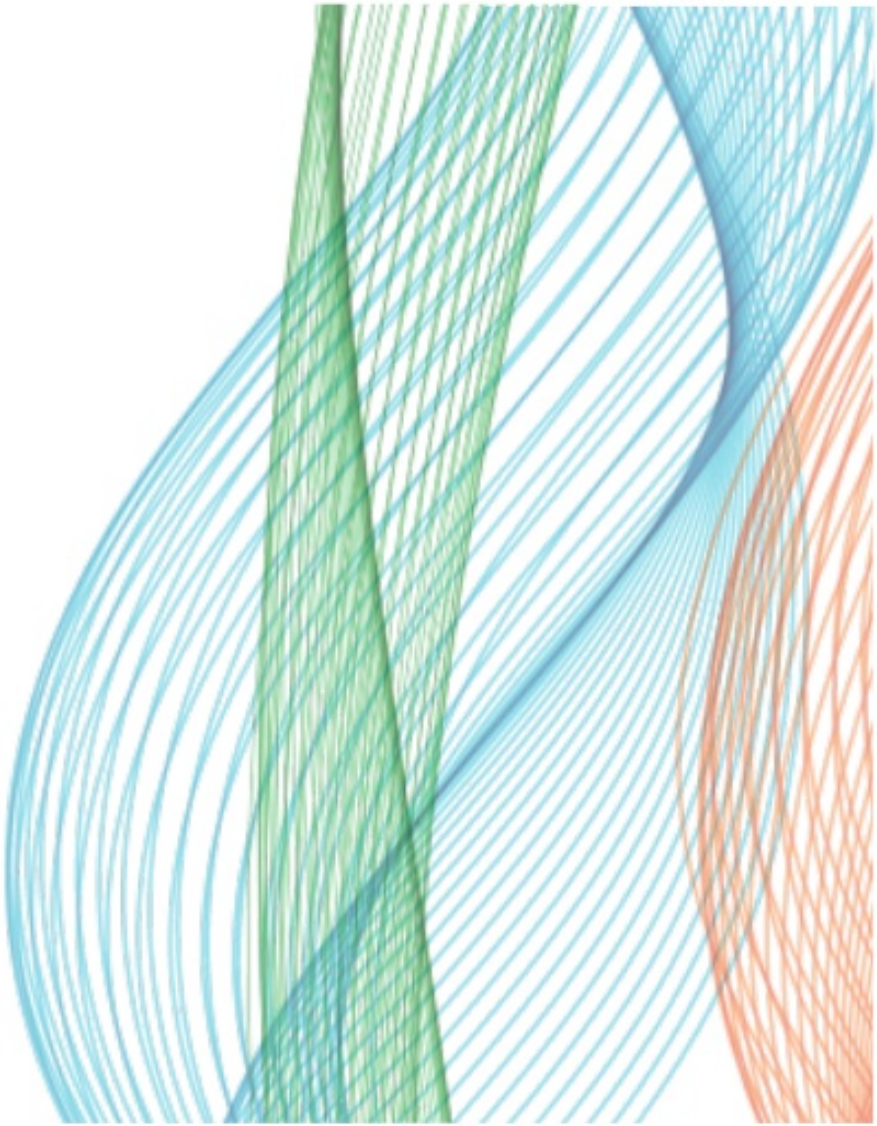


Transformation vs. Profiling

- Transformation is primarily row-based.
- Profiling is column-based.
- Different execution concerns.



Solution: Spark in the Trifacta System

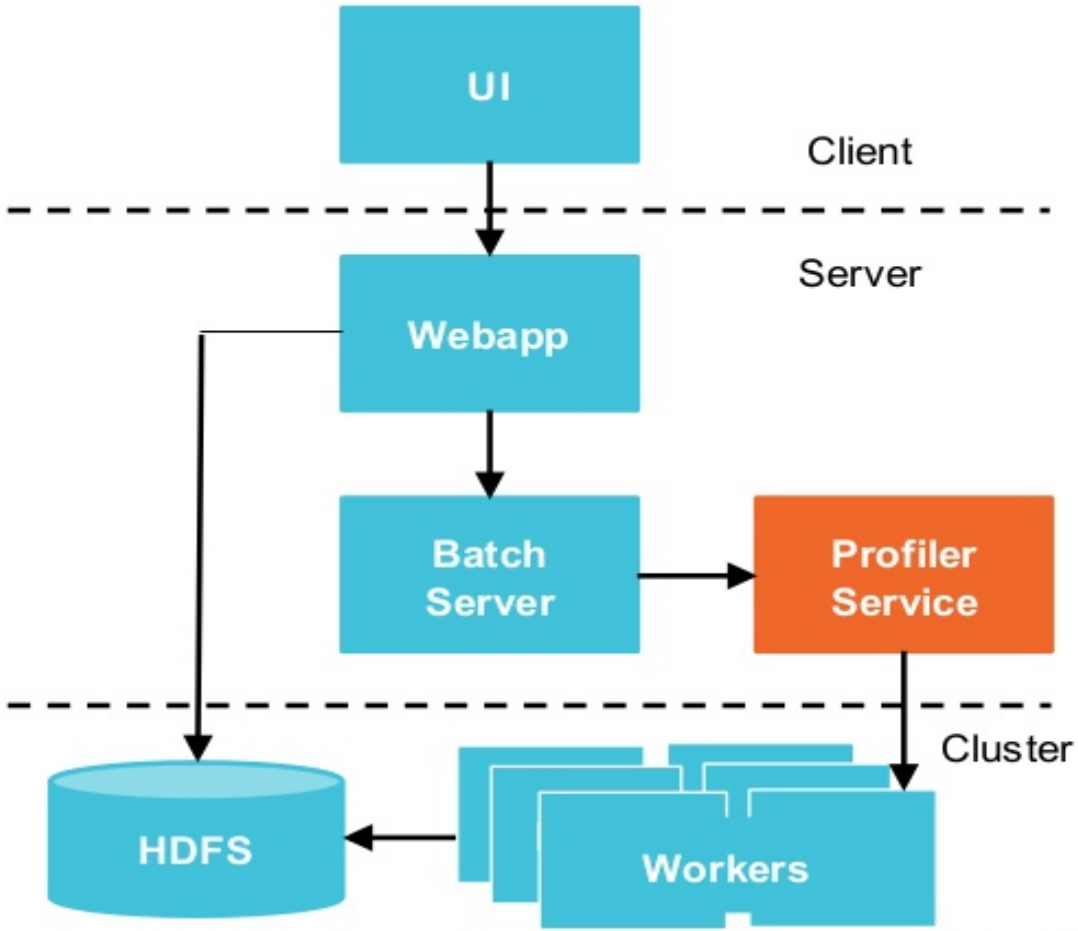


Why Spark

- Effective at OLAP
- Flexible enough for UDFs
- Not tied to distributions
- Mass adoption
- Low latency

Spark Profile Jobs Server

- We built a Spark job server
- Automatically profiles output of transform jobs
- Outputs profiling results to HDFS
- Renders graphs in Trifacta product



Profiler Service

- Which columns to profile and their types
- Uses Trifacta type system, eg:
  - Dates
  - Geography
  - User-defined
- Requested profiling statistics
  - Histograms
  - Outliers
  - Empty, invalid, valid
- Extensible
  - Pairwise statistics
  - User-defined functions

Profiling DSL

```
{
        "input": "hdfs://hadoop.trifacta-dev.net:8020:/trifacta…",
        "schema": {
        "order": ["column1", "column2", "column3"],
        "types": {
            "column1": ["Datetime", {"regexes": , "groupLocs": {...}}],
            "column2": [...],
            "column3": [...]
        }
    },
    "commands": [
        {         "column": "*",
            "output": "hdfs://hadoop.trifacta-dev.net:8020:/trifacta…",
            "profiler-type": "histogram",
            "params": {...}       },
        {         "column": "column1",
           "output": "hdfs://hadoop.trifacta-dev.net:8020:/trifacta…",
           "profiler-type": "type-check",
            "params": {...}       }        ]
}
```

Profiling DSL

```
{
        "input": "hdfs://hadoop.trifacta-dev.net:8020:/trifacta…",
        "schema": {
        "order": ["column1", "column2", "column3"],
        "types": {
            "column1": ["Datetime", {"regexes": , "groupLocs": {...}}],
            "column2": [...],
            "column3": [...]
        }
    },
    "commands": [
        {         "column": "*",
            "output": "hdfs://hadoop.trifacta-dev.net:8020:/trifacta…",
            "profiler-type": "histogram",
            "params": {...}       },
        {         "column": "column1",
           "output": "hdfs://hadoop.trifacta-dev.net:8020:/trifacta…",
           "profiler-type": "type-check",
            "params": {...}       }        ]
}
```

Profiling DSL

```
{
        "input": "hdfs://hadoop.trifacta-dev.net:8020:/trifacta…",
        "schema": {
        "order": ["column1", "column2", "column3"],
        "types": {
            "column1": ["Datetime", {"regexes": , "groupLocs": {...}}],
            "column2": [...],
            "column3": [...]
        }
    },
    "commands": [
        {       "column": "*",
            "output": "hdfs://hadoop.trifacta-dev.net:8020:/trifacta…",
            "profiler-type": "histogram",
            "params": {...}         },
        {       "column": "column1",
           "output": "hdfs://hadoop.trifacta-dev.net:8020:/trifacta…",
           "profiler-type": "type-check",
            "params": {...}       }        ]
}
```

Profiling DSL

```
{
        "input": "hdfs://hadoop.trifacta-dev.net:8020:/trifacta…",
        "schema": {
        "order": ["column1", "column2", "column3"],
        "types": {
            "column1": ["Datetime", {"regexes": , "groupLocs": {...}}],
            "column2": [...],
            "column3": [...]
        }
    },
    "commands": [
        {       "column": "*",
            "output": "hdfs://hadoop.trifacta-dev.net:8020:/trifacta…",
            "profiler-type": "histogram",
            "params": {...}       },
        {       "column": "column1",
            "output": "hdfs://hadoop.trifacta-dev.net:8020:/trifacta…",
            "profiler-type": "type-check",
            "params": {...}       }       ]
}
```

Performance Improvements

Spark profiling speed-up vs. MapReduce

	Few Columns	Many Columns
High Cardinality	2X	4X
Low Cardinality	10X	20X

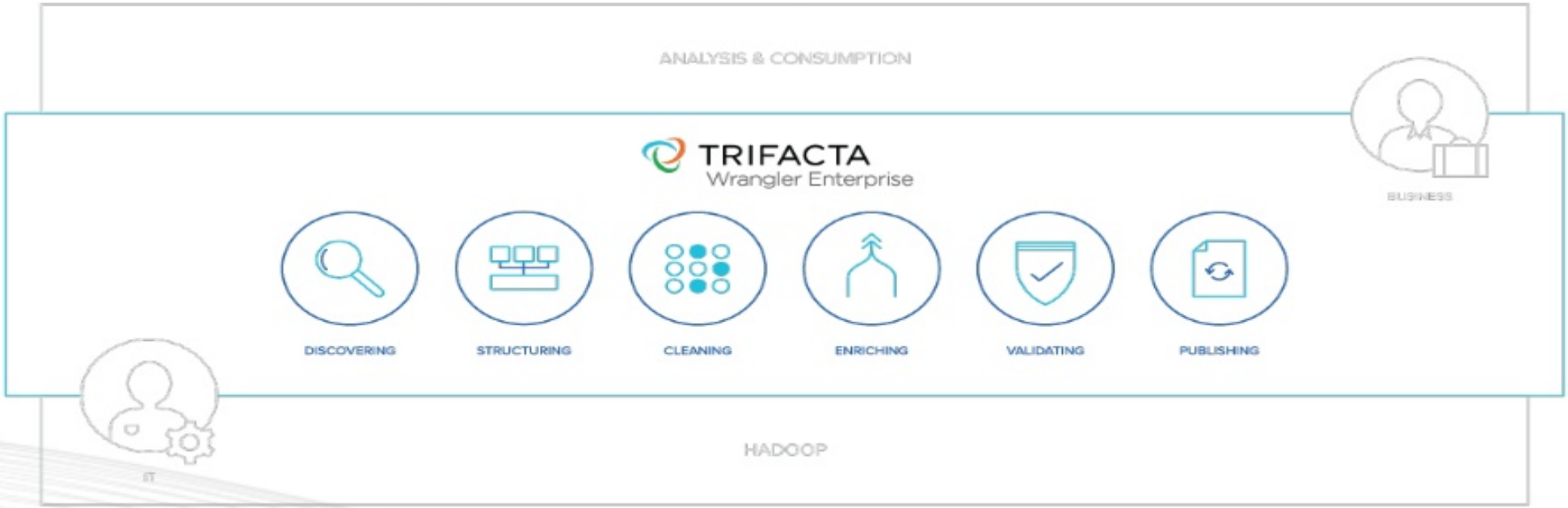
- A few troublesome use cases:
  - High cardinality
  - Non-distributive stats
- Huge gains for large numbers of columns

Pay-as-you-go Profiling

- Users do not always need profiles for all columns
- More complex and expensive statistics
- Drilldown
- Iterative exploration

Conclusion

Profiling informs data preparation.



Questions?