Great Expectations & The Wonderful World of Data Quality Tools in Python

Sam Bail @spbail, Data Umbrella, March 2021

SUPERCONDUCTIVE

Hi, I’m Sam! (@spbail)

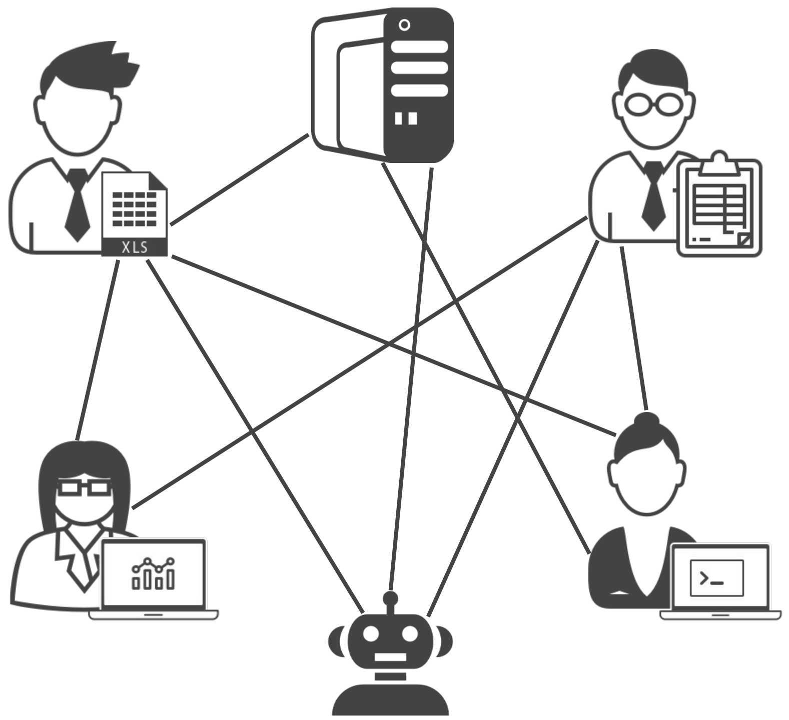
- I’m originally from Germany, currently based in NYC
- I have a PhD in semantic web technologies with a focus on data representation formalisms (Linked Data, OWL, RDF, in case that rings a bell…)
- I’ve been doing “data work” for a while, mostly working with 3rd party healthcare data
- And now I’m an engineering director at Superconductive, the core maintainers behind Great Expectations

Agenda

- Part 1: The Wonderful World of (Open Source) Data Quality in Python
  - Types of “data quality” tools
  - Overview of some prominent ones
- Part 2: Great Expectations: Overview and motivation
- Part 3: “Getting started” live demo of Great Expectations
- Q&A

SUPERCONDUCTIVE

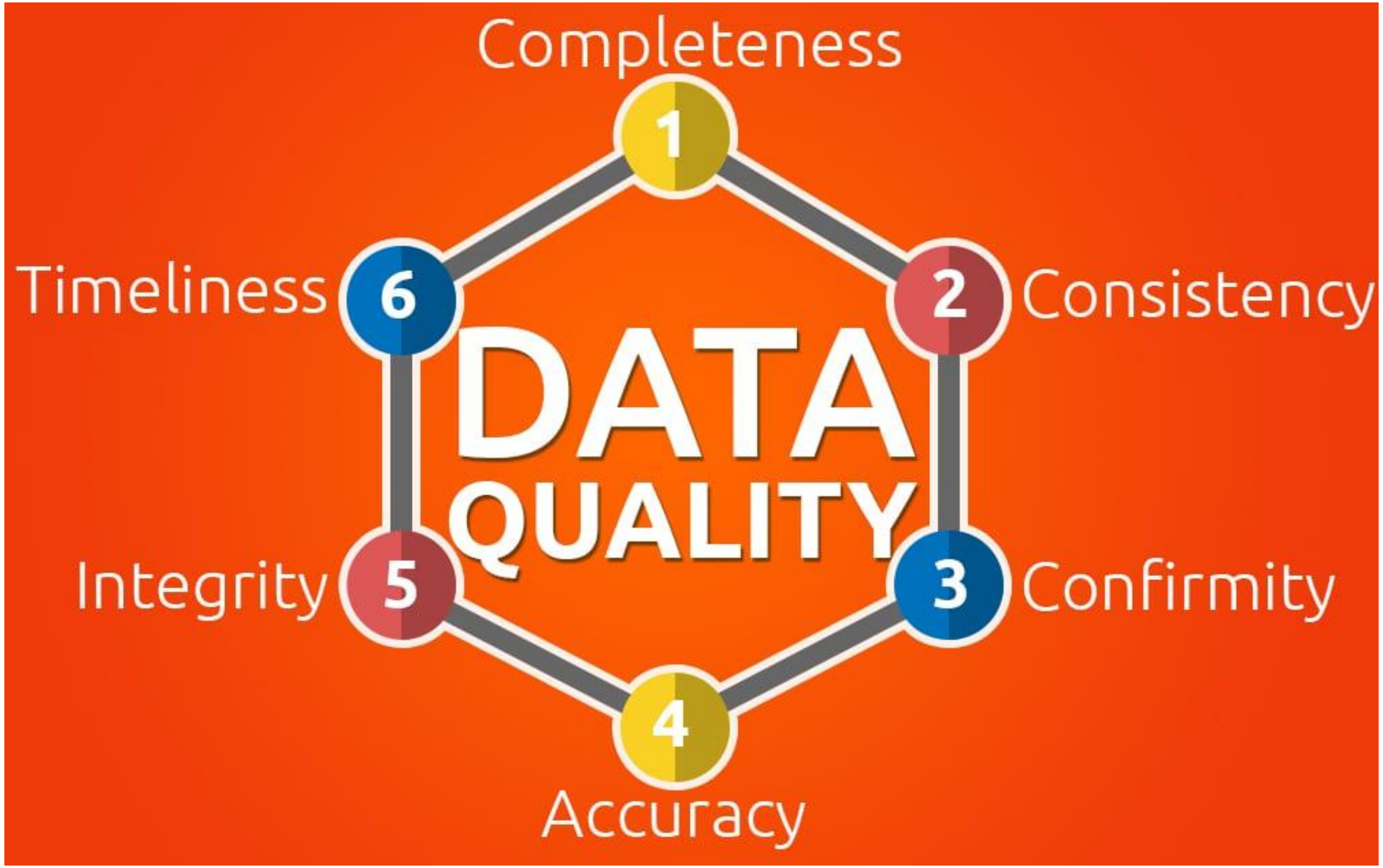
The challenge: Data workflows today are a mess



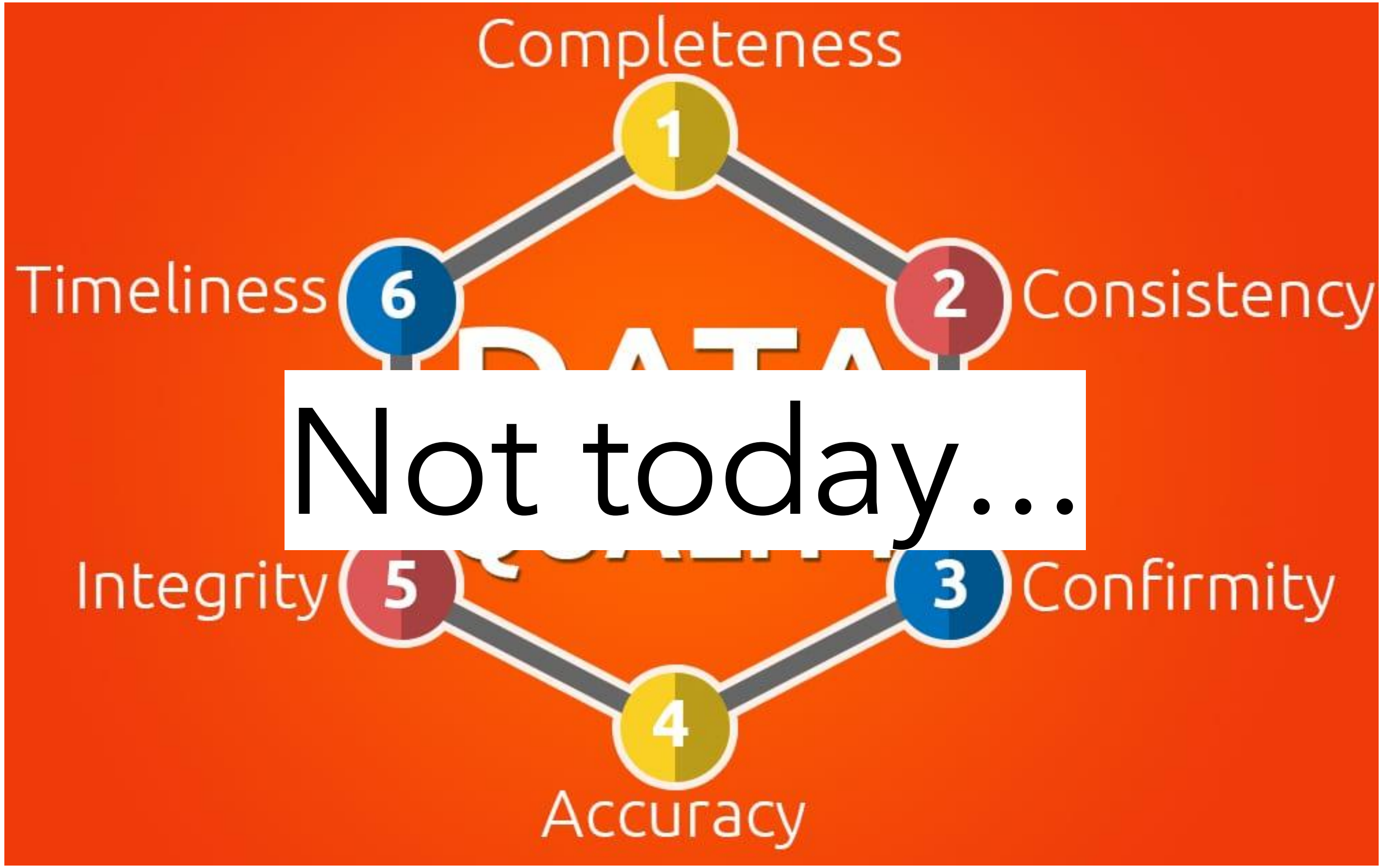
- Data pipelines are brittle and often fail, both loudly and silently
- Tacit knowledge scattered among domain experts, technical experts, and the code and data itself
- Maintenance is time-consuming, expensive, and morale-killing
- Documentation is chronically out of date and unreliable
- Trust in many data systems is low
- There are many different tools to help with this...

Part 1:

The Wonderful World of (Open Source) Data Quality Tools in Python



https://smartbridge.com/data-done-right-6-dimensions-of-data-quality/

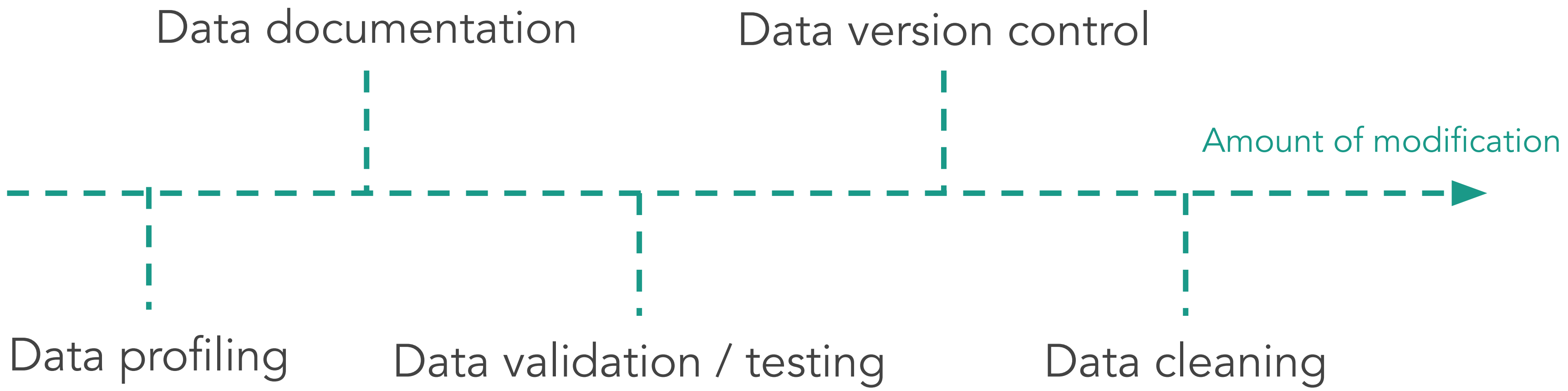


Not today…

SUPERCONDUCTIVE

https://smartbridge.com/data-done-right-6-dimensions-of-data-quality/

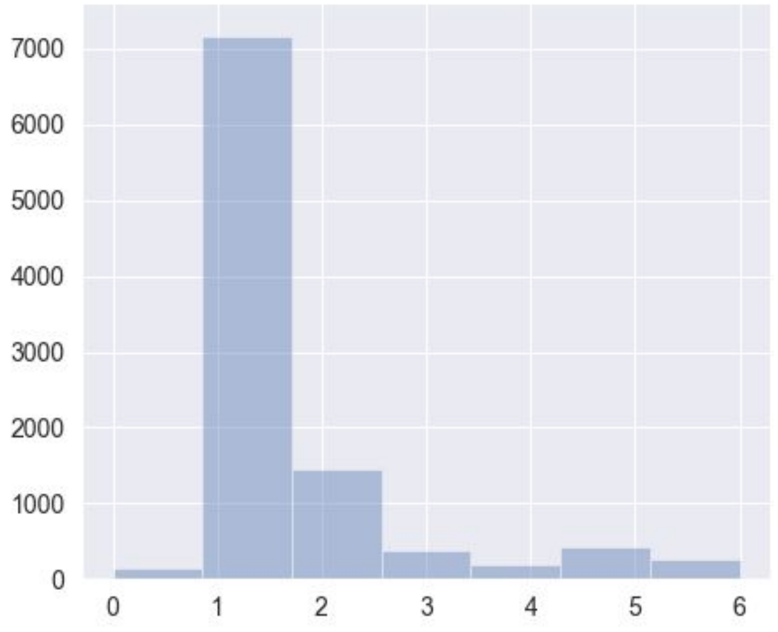
Different aspects of data "quality"



There are lots of different tools in the space...

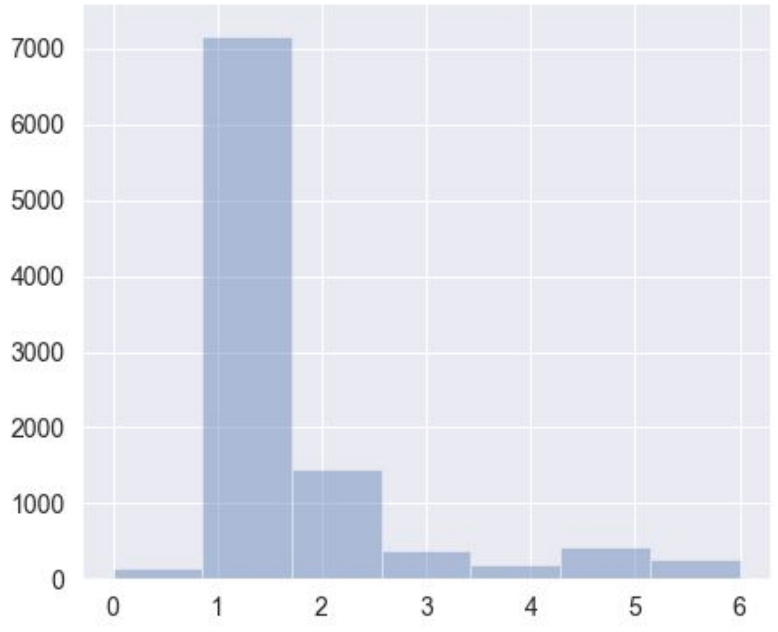
- I focused on "single purpose" tools rather than end-to-end data processing packages
- It's surprisingly hard to find a lot of open source "python data quality" packages
  - Note: The commercial space here is growing quickly
  - A lot of these aren't actively maintained
  - I marked active projects (=some activity in past 6 months) with *
- Let me know if I've missed anything!

Running example: NYC taxi data

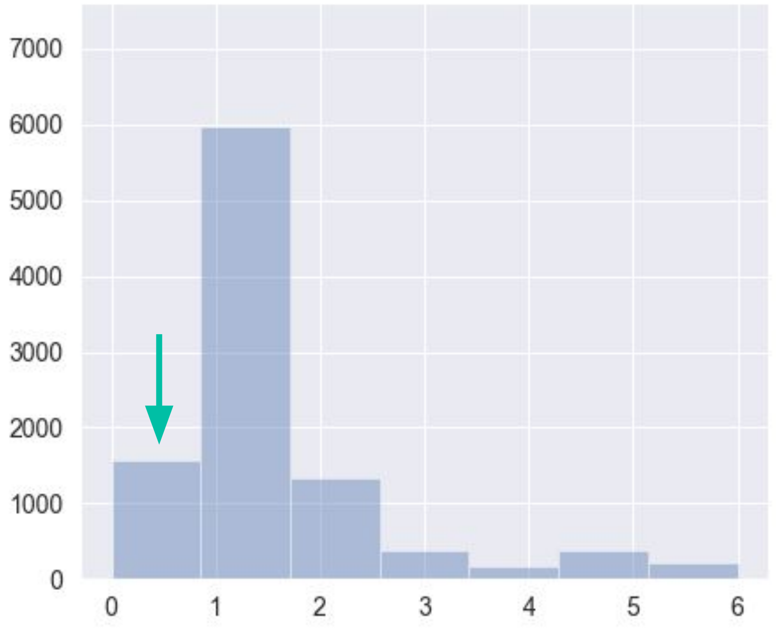


Number of passengers per ride in January 2019

How do we prevent against issues like this in our data?



Number of passengers per ride in January 2019



Number of passengers per ride in February 2019

SUPERCONDUCTIVE

Pure profiling tools

- Pandas Profiling*: Like a sophisticated extension of .describe() on Pandas dataframes, creates a more detailed profile report of the data.
  - https://github.com/pandas-profiling/pandas-profiling

Data profiling

Data documentation

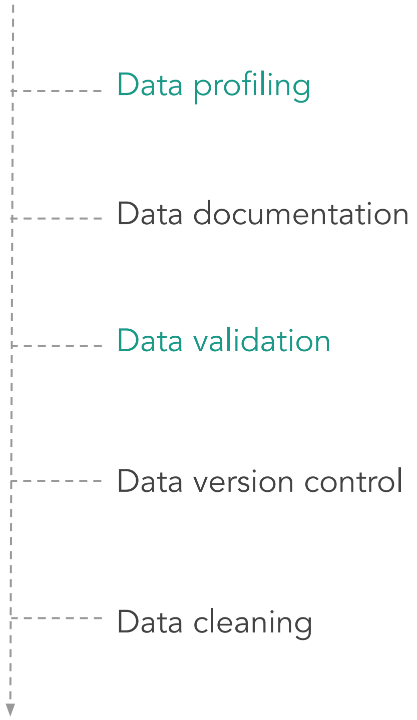
Data validation

Data version control

Data cleaning

Profiling + validation

- TDDA*: Allows specifying and verifying constraints on data, provides some lightweight profiling
    - https://pypi.org/project/tdda/
- pydqc: Does some data profiling (show basic stats about a table), compares columns between tables, compares tables for identity.
    - https://github.com/SauceCat/pydqc



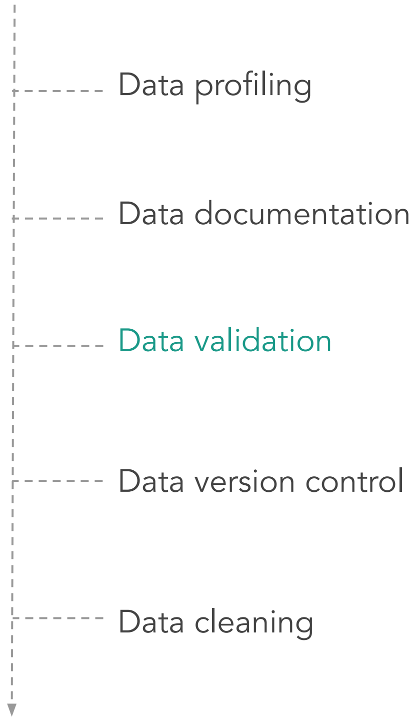
Profiling, validation, and documentation

- Great Expectations*: Allows you to write declarative data tests ("I expect this table to have between x and y number of rows"), get validation results from those tests, and output a report that documents the current state of your data
    - http://greatexpectations.io

- Data profiling
- Data documentation
- Data validation
- Data version control
- Data cleaning

Pure data validation / testing tools

- Bulwark*: Data testing framework that lets you add tests on methods that return Pandas dataframes. Has some built-in tests and allows custom methods for tests.
    - https://github.com/ZaxR/bulwark
- Engarde: A precursor to Bulwark, allows you to add decorators with data assertions to methods that return Pandas data frames.
    - https://engarde.readthedocs.io/en/latest



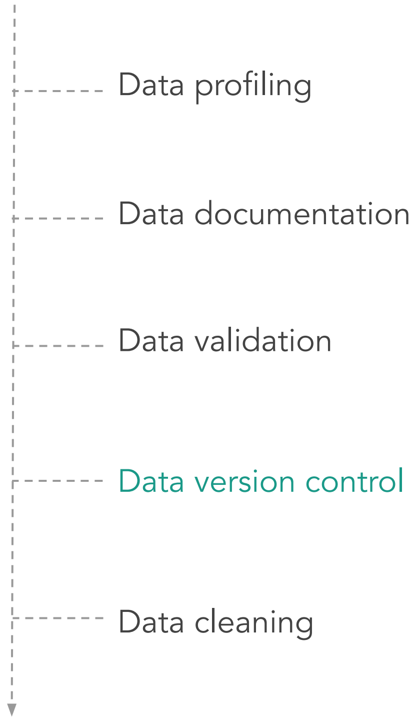
SUPERCONDUCTIVE

Pure data validation / testing tools

- Voluptous*: Allows you to specify a "schema" to validate JSON/YAML
  - https://github.com/alecthomas/voluptuous
- Opulent Pandas: A df-focused "version" of Voluptuous
  - https://pypi.org/project/opulent-pandas/
- mobydq*: Data validation web app that allows you to check for "indicators" such as completeness, freshness, latency, validity.
  - https://github.com/ubisoft/mobydq

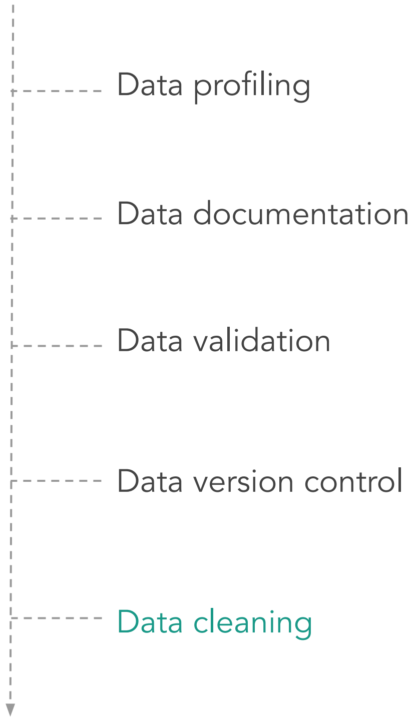
Data version control

- dvc*: Built on top of git, offers version control specifically for data and Machine Learning models.
- (Any others?)



Data cleaning

- dedupe*: Uses fuzzy matching to perform de-duplication and entity resolution in data.
    - https://github.com/dedupeio/dedupe
- datacleaner: Does some basic cleaning operations on data frames, e.g. dropping rows with missing values, codes strings as numeric values, etc.
    - https://github.com/rhiever/datacleaner
- (and many more… pyjanitor)



Part 2:

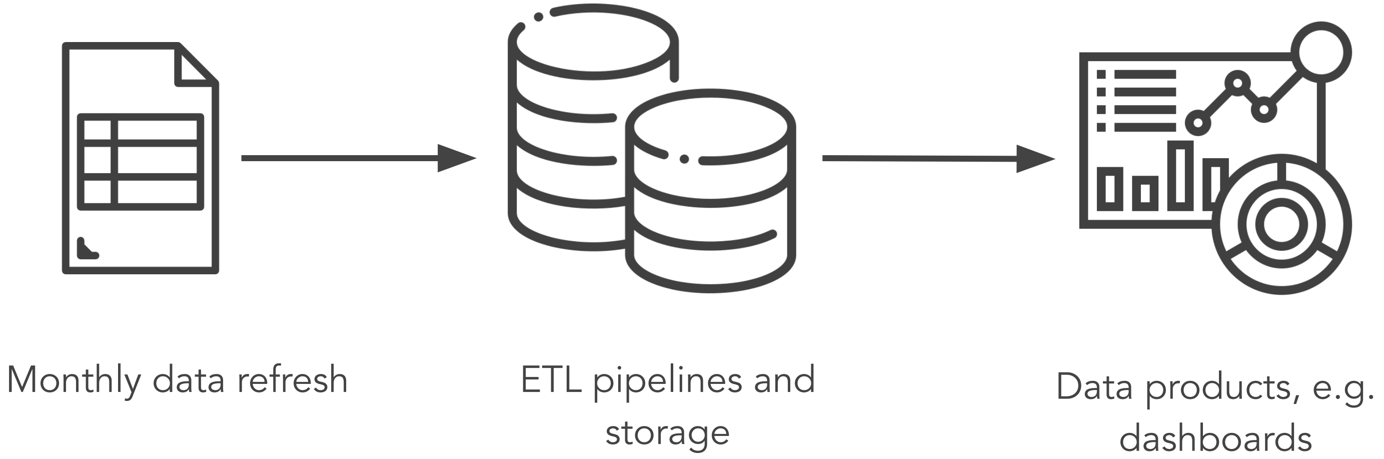
Deep Dive Into Great Expectations

“Our stakeholders would notice data issues before we did… which really eroded trust in the data and our team”

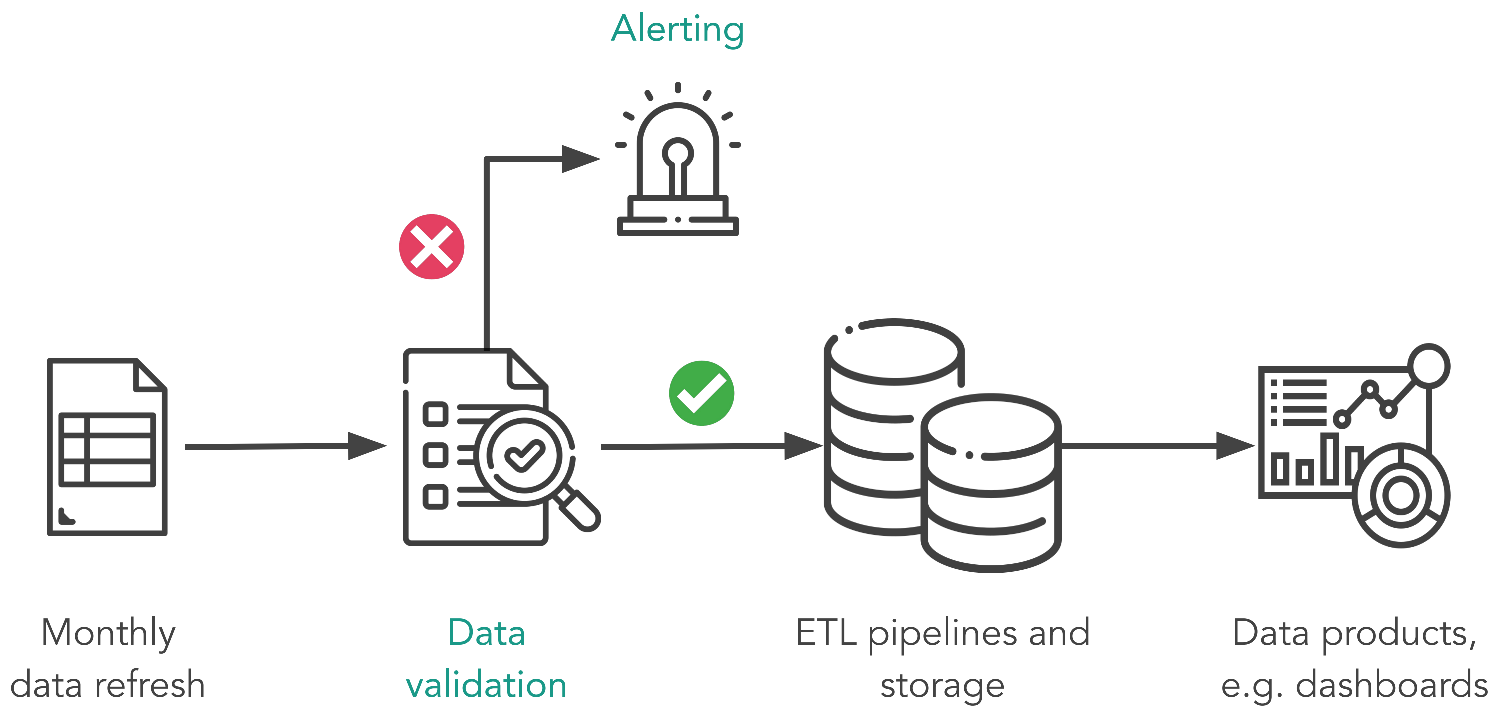
A Great Expectations user

SUPERCONDUCTIVE

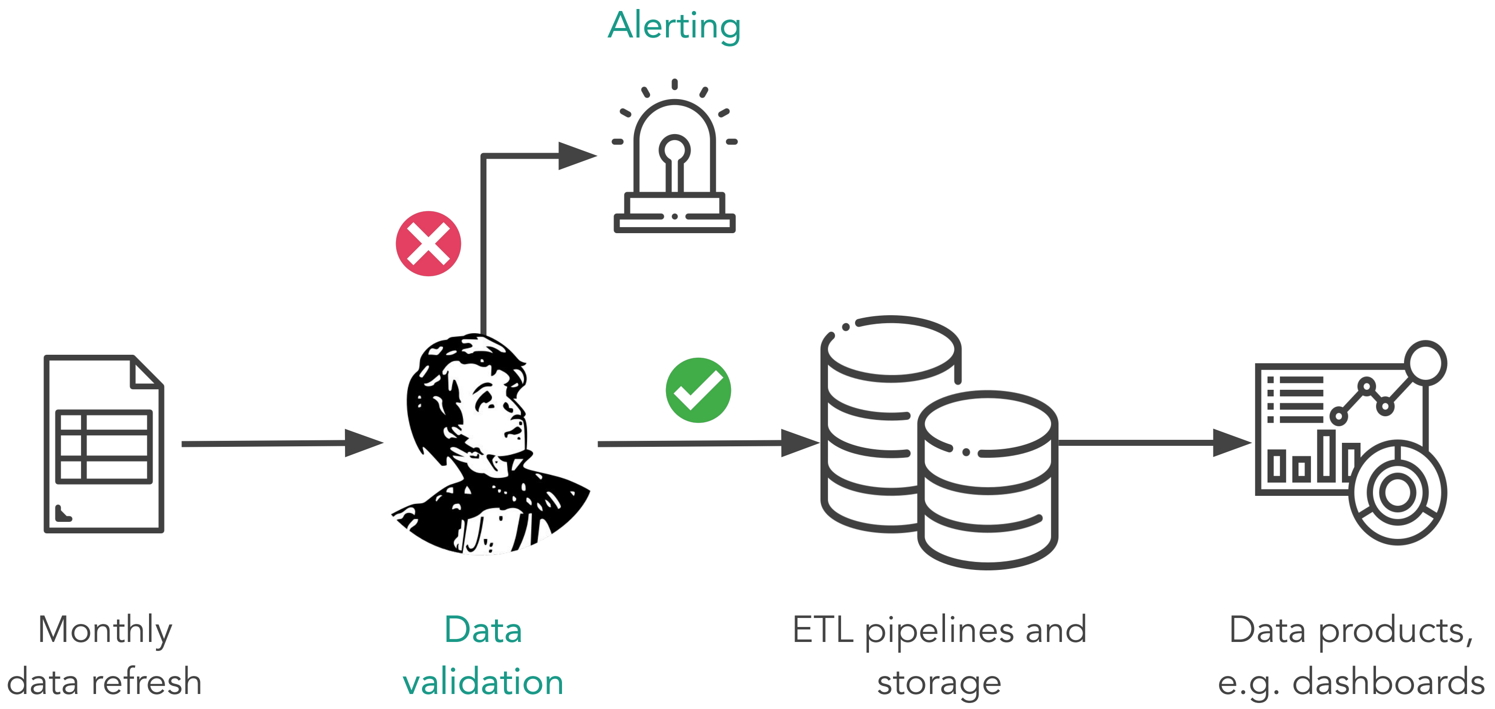
A typical data pipeline



A better data pipeline

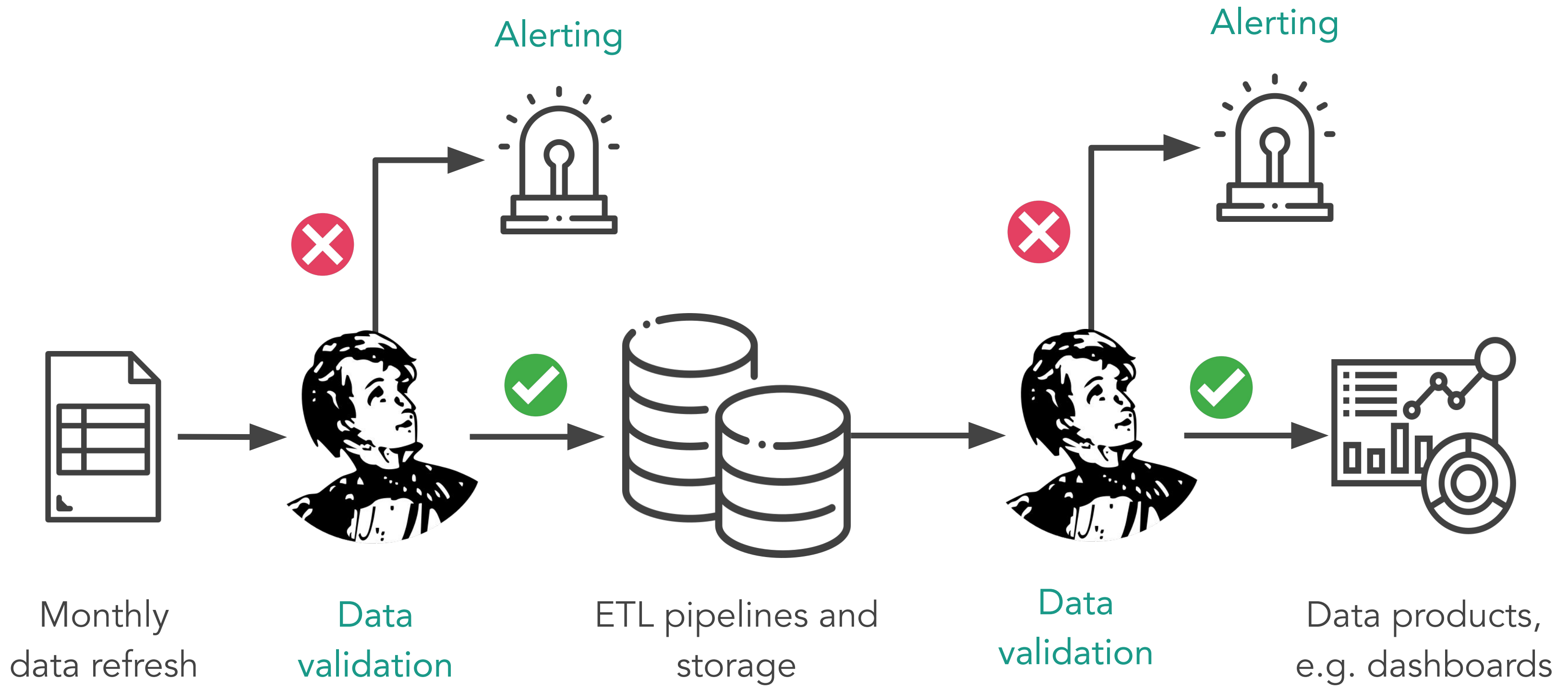


A better data pipeline...

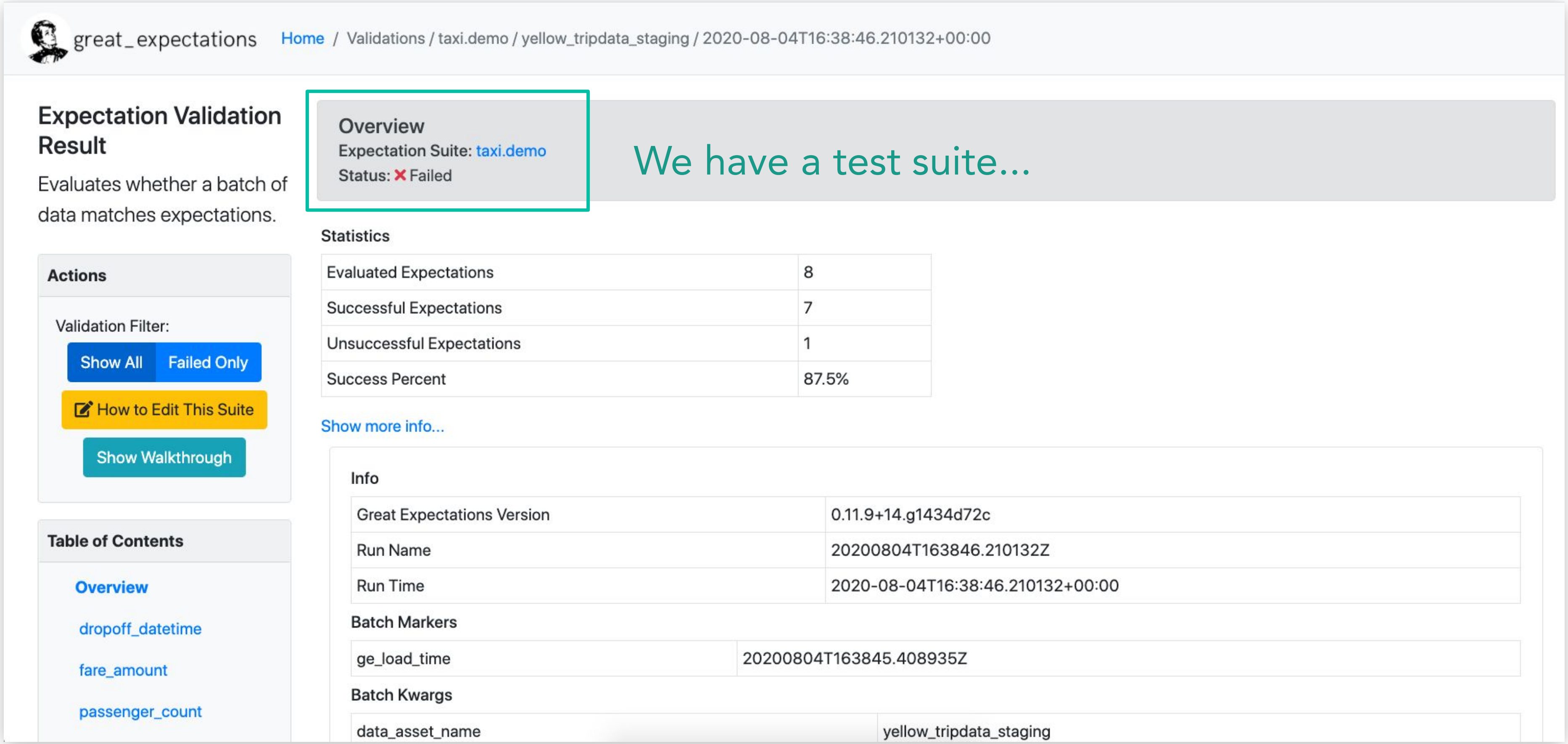


SUPERCONDUCTIVE

Or even better...



SUPERCONDUCTIVE



great_expectations Home / Validations / taxi.demo / yellow_tripdata_staging / 2020-08-04T16:38:46.210132+00:00

Expectation Validation Result

Evaluates whether a batch of data matches expectations.

Actions

Validation Filter:

Table of Contents

Overview

Expectation Suite: taxi.demo

Status: ✗ Failed

We have a test suite...

Statistics

Evaluated Expectations	8
Successful Expectations	7
Unsuccessful Expectations	1
Success Percent	87.5%

Show more info...

Info

Great Expectations Version	0.11.9+14.g1434d72c
Run Name	20200804T163846.210132Z
Run Time	2020-08-04T16:38:46.210132+00:00

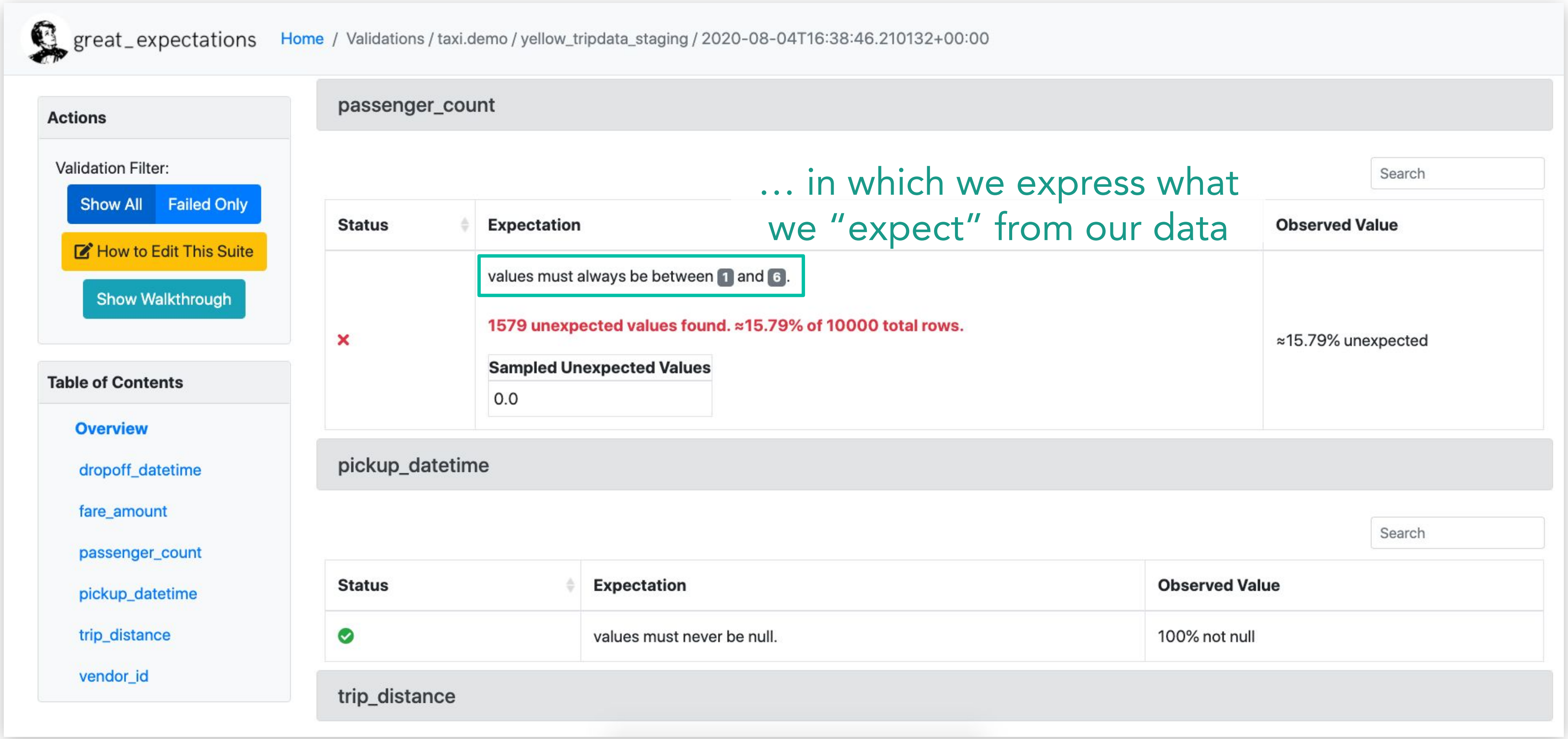
Batch Markers

ge_load_time	20200804T163845.408935Z

Batch Kwargs

data_asset_name	yellow_tripdata_staging

SUPERCONDUCTIVE



great_expectations Home / Validations / taxi.demo / yellow_tripdata_staging / 2020-08-04T16:38:46.210132+00:00

Actions

Validation Filter:

Table of Contents

passenger_count

… in which we express what we "expect" from our data

Status	Expectation	Observed Value
✗	values must always be between 1 and 6. 1579 unexpected values found. ≈15.79% of 10000 total rows. **Sampled Unexpected Values** 0.0	≈15.79% unexpected

pickup_datetime

Status	Expectation	Observed Value
✓	values must never be null.	100% not null

trip_distance

SUPERCONDUCTIVE

What is Great Expectations?

```
> pip install great_expectations
```

It's an open source Python library

```
my_data_project
└── great_expectations
    ├── checkpoints
    ├── expectations
    │   └── taxi
    │       └── demo.json
    ├── plugins
    ├── uncommitted
    └── great_expectations.yml
```

That operates by creating tests in code and storing a collection of them them as a "suite" in JSON

SUPERCONDUCTIVE

What is an Expectation?

```
expect_column_values_to_be_between(
      column='passenger_count',
      min_value=1,
      max_value=6
)
```

It's a statement about what we expect from our data, expressed as a method in Python

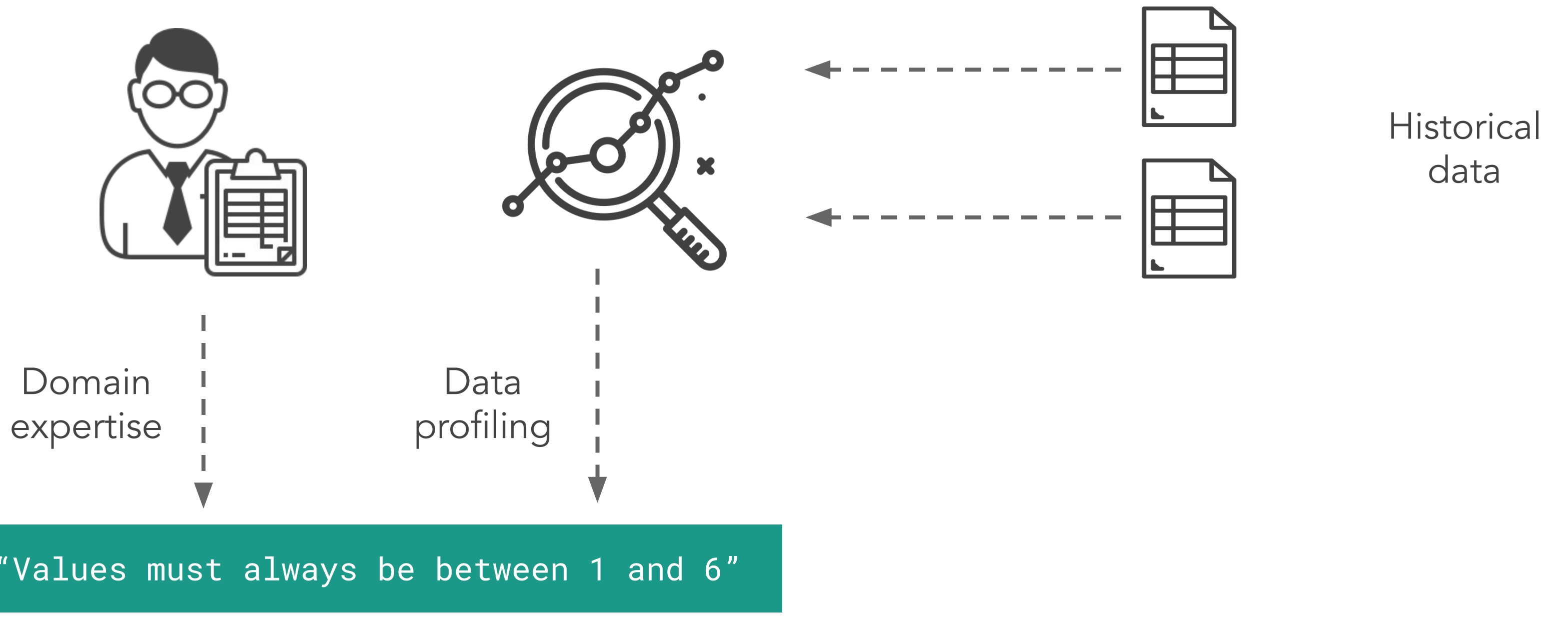
```
{
  "expectation_type": "expect_column_values_to_be_between",
      "kwargs": {
        "column": "passenger_count",
        "min_value": 1,
        "max_value": 6
      },
}
```

That is stored in JSON

```
"Values must always be between 1 and 6"
```

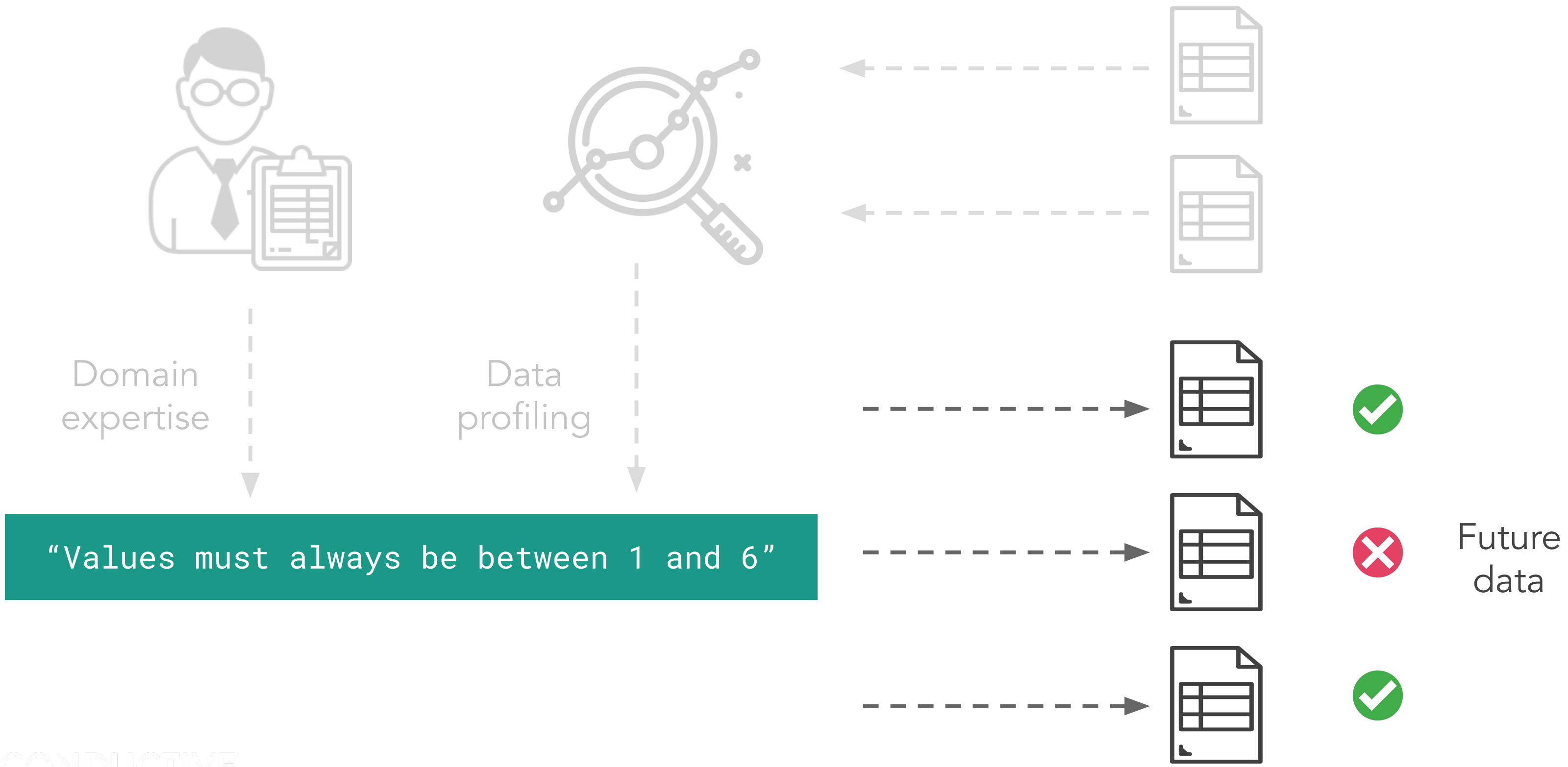
And can be translated into a human-readable format

Built-in profiling to "scaffold" Expectations



SUPERCONDUCTIVE

Validation of new data using Expectations



Different types of Great Expectations workflows

Interactive workflows

Import the GE library in a notebook (local, Databricks, etc.)

Connect to a batch of data

Profile and validate data on-the-fly and iteratively using Expectations

Batch processing workflows

Create and store Expectation Suites

Load Expectation Suites to validate incoming batches of data

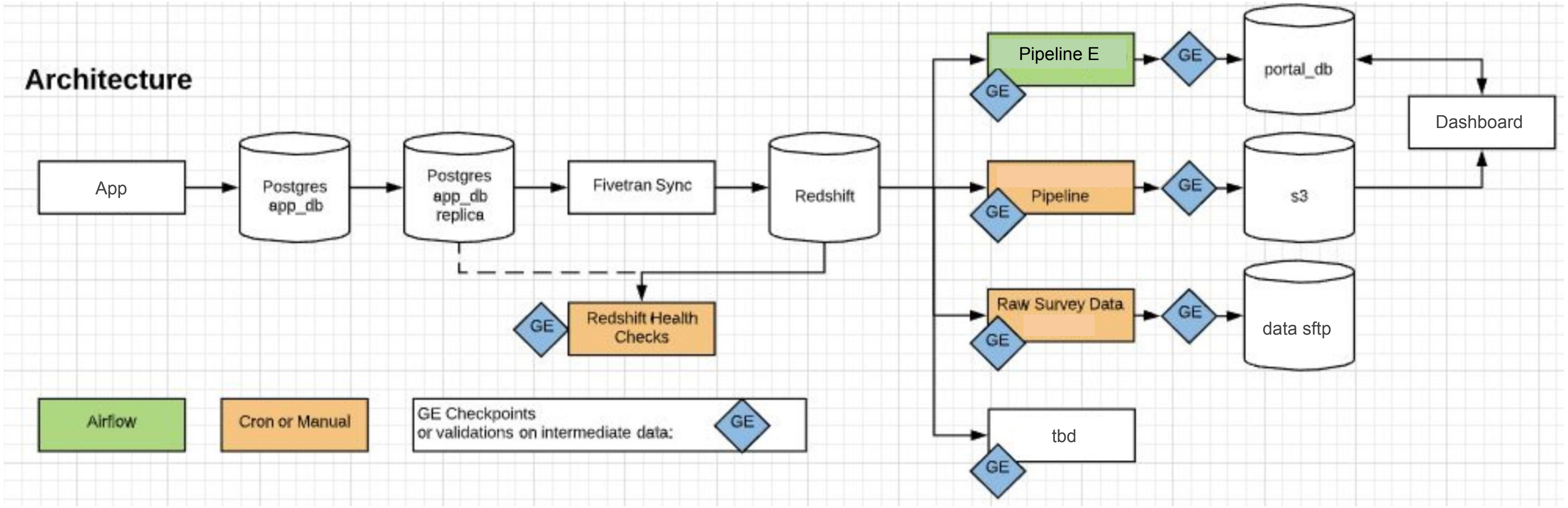
Validation results determine alerting and pipeline behavior

CI/CD workflows

Local tests of pipeline changes against input fixtures during code development process

Automated testing against input fixtures, e.g. when opening a PR

Sample architecture



Integrations

Orchestration

Apache Airflow

DAGSTER

Kedro

PREFECT

Compute

APACHE Spark

pandas

DASK

HIVE

Amazon Athena

presto

Storage + RDBMS

snowflake

Powered by MySQL

SQLAlchemy

PostgreSQL

Microsoft SQL Server

amazon REDSHIFT

Google BigQuery

10 01

amazon S3

Part 3:

Demo time! “Let’s do it live”

In this demo, I’ll show you a quick workflow

- Initialize a data context
- Connect to a Datasource
- Create an Expectation Suite with an automated profiler
- (If time allows: Create a Checkpoint to validate new data)
- Navigate Data Docs

That's it! Questions?
Sam Bail @spbail

Great Expectations

Join our Slack channel!
greatexpectations.io/slack

SUPERCONDUCTIVE