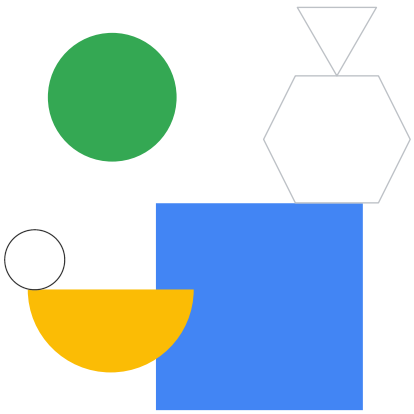
Google Cloud

Cleaning and Transforming your Data with Dataprep



Agenda

01 5 Principles of Dataset Integrity

02 Dataset Shape and Skew

03 Clean and Transform Data using SQL

04 Introducing Dataprep by Trifacta

Lab: Creating a Data Transformation Pipeline with Dataprep

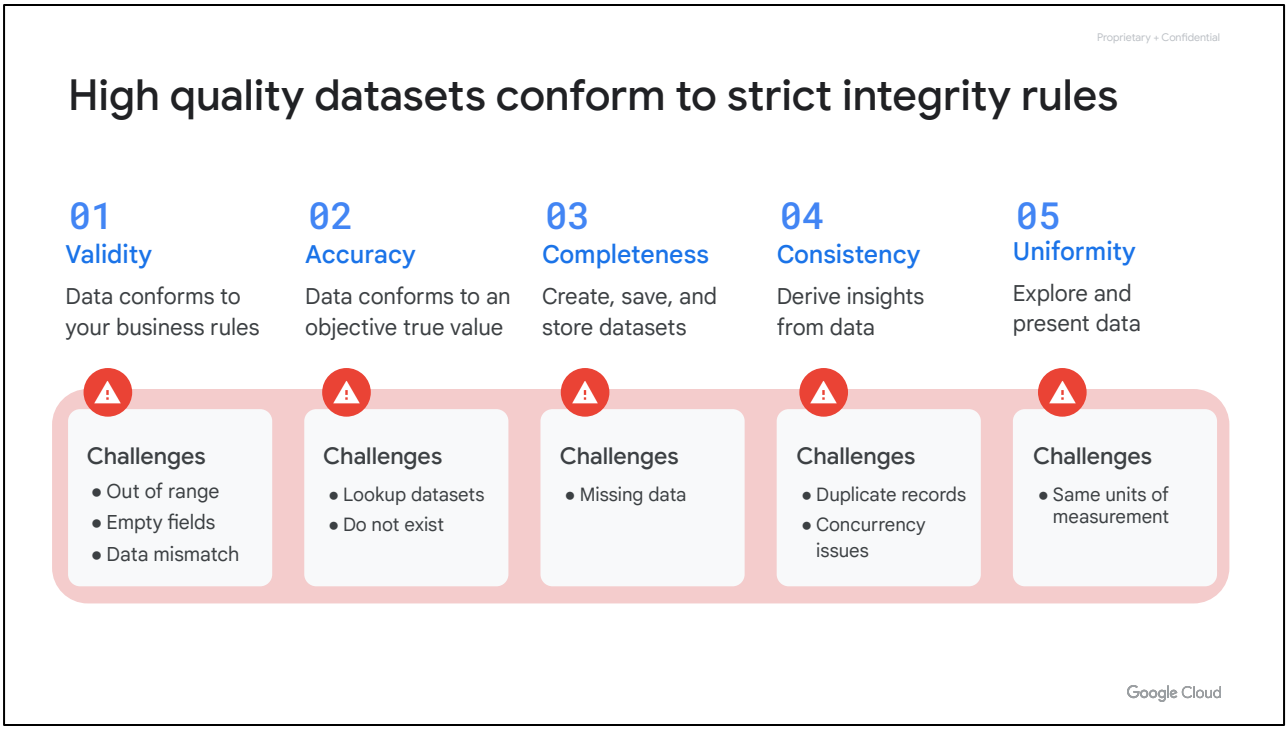
One of the most critical parts of data analysis happens way before you build your first visualization or machine learning model. While not necessarily the most glamorous, data preparation and transformation cannot be overlooked.

In this module we will cover the elements of good dataset hygiene and then look at two different ways to prepare your data: one through SQL in BigQuery and the other through a graphical user interface in DataPrep by Trifacta.

5 Principles of Dataset Integrity

“To him with a mess!” (German)

If you feed an ML model dirty data, you will get dirty results. Same goes with data analysis.



Additional Reading from https://en.wikipedia.org/wiki/Data_cleansing:

Validity: The degree to which the measures conform to defined business rules or constraints (see also Validity (statistics)). When modern database technology is used to design data-capture systems, validity is fairly easy to ensure: invalid data arises mainly in legacy contexts (where constraints were not implemented in software) or where inappropriate data-capture technology was used (e.g., spreadsheets, where it is very hard to limit what a user chooses to enter into a cell, if cell validation is not used). Data constraints fall into the following categories:

- *Data-Type Constraints* – e.g., values in a particular column must be of a particular data type, e.g., Boolean, numeric (integer or real), date, etc.
- *Range Constraints:* typically, numbers or dates should fall within a certain range. That is, they have minimum and/or maximum permissible values.
- *Mandatory Constraints:* Certain columns cannot be empty.
- *Unique Constraints:* A field, or a combination of fields, must be unique across a dataset. For example, no two persons can have the same social security number.
- *Set-Membership constraints*: The values for a column come from a set of discrete values or codes. For example, a person's gender may be Female, Male or Unknown (not recorded).
- *Foreign-key constraints*: This is the more general case of set membership. The set of values in a column is defined in a column of another table that

- contains unique values. For example, in a US taxpayer database, the "state" column is required to belong to one of the US's defined states or territories: the set of permissible states/territories is recorded in a separate States table. The term foreign key is borrowed from relational database terminology.
- Regular expression patterns: Occasionally, text fields will have to be validated this way. For example, phone numbers may be required to have the pattern (999) 999-9999.
- Cross-field validation: Certain conditions that utilize multiple fields must hold. For example, in laboratory medicine, the sum of the components of the differential white blood cell count must be equal to 100 (since they are all percentages). In a hospital database, a patient's date of discharge from hospital cannot be earlier than the date of admission.

Accuracy: The degree of conformity of a measure to a standard or a true value - see also Accuracy and precision. Accuracy is very hard to achieve through data-cleansing in the general case, because it requires accessing an external source of data that contains the true value: such "gold standard" data is often unavailable. Accuracy has been achieved in some cleansing contexts, notably customer contact data, by using external databases that match up zip codes to geographical locations (city and state), and also help verify that street addresses within these zip codes actually exist.

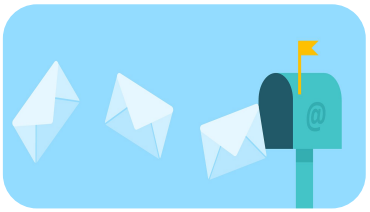
Completeness: The degree to which all required measures are known. Incompleteness is almost impossible to fix with data cleansing methodology: one cannot infer facts that were not captured when the data in question was initially recorded. (In some contexts, e.g., interview data, it may be possible to fix incompleteness by going back to the original source of data, i,e., re-interviewing the subject, but even this does not guarantee success because of problems of recall - e.g., in an interview to gather data on food consumption, no one is likely to remember exactly what one ate six months ago. In the case of systems that insist certain columns should not be empty, one may work around the problem by designating a value that indicates "unknown" or "missing", but supplying of default values does not imply that the data has been made complete.

Consistency: The degree to which a set of measures are equivalent in across systems (see also Consistency). Inconsistency occurs when two data items in the data set contradict each other: e.g., a customer is recorded in two different systems as having two different current addresses, and only one of them can be correct. Fixing inconsistency is not always possible: it requires a variety of strategies - e.g., deciding which data were recorded more recently, which data source is likely to be most

reliable (the latter knowledge may be specific to a given organization), or simply trying to find the truth by testing both data items (e.g., calling up the customer).

Uniformity: The degree to which a set data measures are specified using the same units of measure in all systems (see also Unit of measure). In datasets pooled from different locales, weight may be recorded either in pounds or kilos, and must be converted to a single measure using an arithmetic transformation.

Valid data follows constraints on uniqueness



What do these identifiers have in common?
Why were they set up that way?

What do all these images represent or have in common?

They are a unique way of identifying
- Phone = Phone Number
- Mail / Email = Address
- Car = License Plate Number

Why would it be terrible to have duplicative license plate numbers?
- Speeding tickets from someone else's car are fined to you and your as well!

In the same way with data, having **duplicate records** or having **stale records** (think of if your phone number or address changed but the post service did not recognize that change)

Valid data corresponds to range constraints

Roll #	Value
1	2
2	2
3	6
4	5
5	1
6	7

Which value(s) are out of range?

Which value looks out of place in the table?
(most answer 7)

Depends on how many dice you roll

I rolled two dice, which value is out of place now?
(correct answer: 1. Impossible to roll a 1 with two dice thrown)

Accurate data matches to a known source of truth

- U.S. States
- Washington
- Oregon
- California
- Hot Dog
- Florida
- Maine

There's a **set amount of known U.S. States**, **50,** and Hot Dog is not in that list.

Even this is up for debate because there are additional territories like Guam, Puerto Rico, military bases (so it depends on what your source of truth is for accuracy. What standard are you using?)

You may see this come up in the future (instructor hint: Dataprep wants to invalidate GU (Guam) and AP (Armed Forces Pacific) since they aren't in the standard it uses for U.S. States)

Are we missing the complete picture of our dataset? What do you see here?

What are we looking at? Unless you an expert traveller or know the area, it may be hard to pick out from just these two images.

Visually, it is very clear when we aren't seeing the whole picture.

Get the complete picture -- question the source of where you got the data. **Is there any more? (next slide)**

We’re in London!

With incomplete data is hard to get a real sense of what is going on. We inherently trust data because if it comes to us in row and column format somehow everything we need is in there if we just query it. Not true!

Consequences of incomplete data

- Insights over rely on the small segment of data (e.g. Lamp Clock Land)
- Biased results
- Interpreting patterns that are not there (overfitting in ML models)

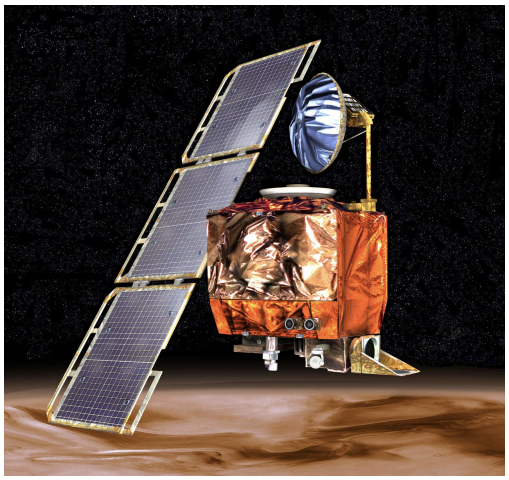
Consistent data ensures harmony across systems

House Address	Owner ID
123 ABC St	12

Owner ID	Owner Address
15	123 ABC St
12	53rd Ave.

Who owns the house?

Uniformity in data means measuring the same way



=$125 Million

In November 1999, NASA lost a Mars climate orbiter because of English vs metric system measurements

(speaking of teams using the same data types…)

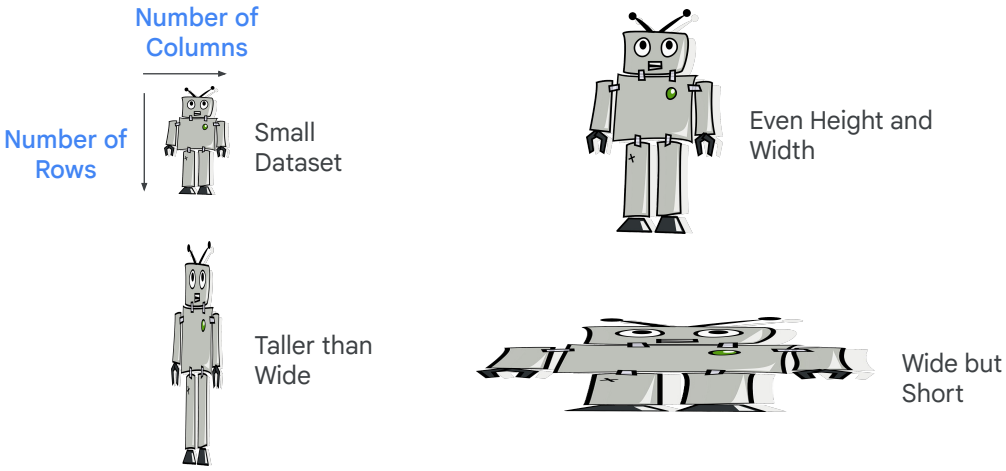
On November 10, 1999, NASA lost its $125-million Mars Climate Orbiter in 1999 because spacecraft engineers failed to convert from English to metric measurements when exchanging vital data before the craft was launched, space agency officials said Thursday.

A navigation team at the Jet Propulsion Laboratory used the metric system of millimeters and meters in its calculations, while the contractor in Denver Colorado, which designed and built the spacecraft, provided crucial acceleration data in the English system of inches, feet and pounds.

Additional Reading:
Mars Climate Orbiter
https://en.wikipedia.org/wiki/Mars_Climate_Orbiter

Dataset Shape and Skew

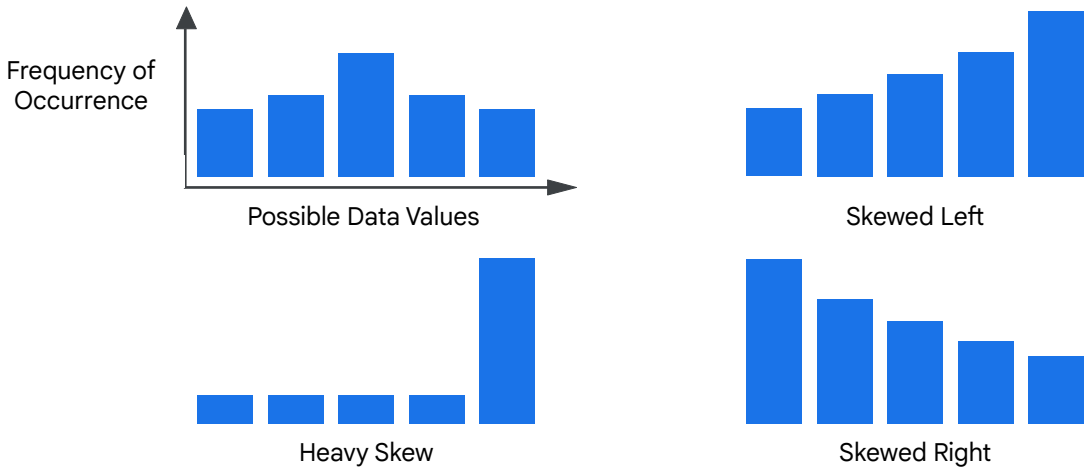
Understanding dataset shape



Wide datasets mean you have a lot of columns and fields but not that many records to analyze. This is likely the worst case scenario.

What can do you?
Wide but short? **Ask for more data!**

Understanding dataset skew (distribution of values)



Does it correspond to what you were expecting? Are there clear outliers or anomalies in your data?

How would you analyze this in SQL? (GROUP BY, ORDER BY). Soon we'll show you how to quickly visualize this in Dataprep.

Centered around one value (e.g. you have an international business but sales orders data is only showing one country)

Clean and Transform Data using SQL

Clean and transform data with SQL

- Setup field data type constraints
- Specify fields as NULLABLE or REQUIRED
- Proactively check for NULL values
- Check and filter for allowable range values
  - SQL Conditionals: CASE WHEN, IF ( )
- Require primary keys / relational constraints in upstream source systems (remember, BigQuery is an analytics warehouse not your primary operational database)

01

Validity

Data conforms to your business rules

Conditionals:
https://cloud.google.com/bigquery/docs/reference/standard-sql/functions-and-operators#casting

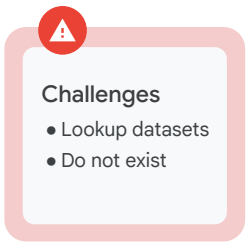
Clean and transform data with SQL

- Create test cases or calculated fields to check values
  - SQL: (quantity_ordered * item_price) AS sub_total
- Lookup values against an objective reference dataset
  - SQL: IN( ) with a subquery or JOIN

02

Accuracy

Data conforms to an objective true value



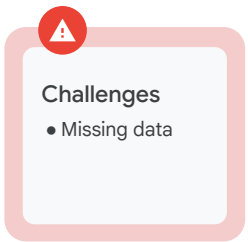
Clean and transform data with SQL

- Thoroughly explore the existing dataset shape and skew and look for missing values
  - SQL: NULLIF( ) , IFNULL( ), COALESCE( )
- Enrich the existing dataset with others using UNIONs and JOINs
  - SQL: UNION, JOIN
  - Example: Multiple years of historical data are available for analysis

03

Completeness

Create, save, and store datasets.

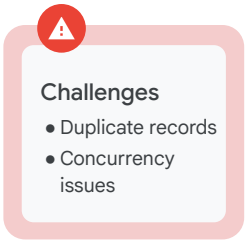


Clean and transform data with SQL

- Store one fact in one place and use IDs to lookup
- Use string functions to clean data
  - PARSE_DATE( )
  - SUBSTR( )
  - REPLACE( )

04
Consistency

Derive insights from data



Documentation on string functions:
https://cloud.google.com/bigquery/docs/reference/standard-sql/functions-and-operators#string-functions

Clean and transform data with SQL

- Document and comment your approach.
- Use FORMAT ( ) to clearly indicate units.
- CAST( ) data types to the same format and digits.
- Label all visualizations appropriately.

05

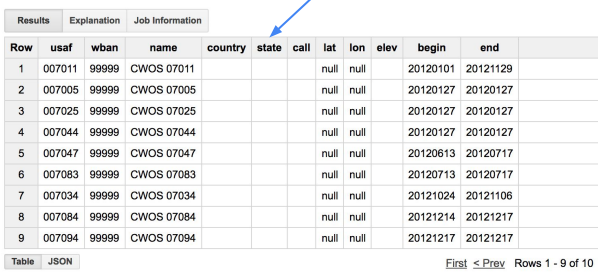
Uniformity

Explore and present data

Tricky NULLs when filtering out missing values

```
#standardSQL
SELECT * FROM
`bigquery-public-data.noaa_gsod.stations`
WHERE state IS NOT NULL
LIMIT 10
```

Why does the below query still show blank state values when we clearly filtered on IS NOT NULL?



Results | Explanation | Job Information

Row	usaf	wban	name	country	state	call	lat	lon	elev	begin	end
1	007011	99999	CWOS 07011				null	null		20120101	20121129
2	007005	99999	CWOS 07005				null	null		20120127	20120127
3	007025	99999	CWOS 07025				null	null		20120127	20120127
4	007044	99999	CWOS 07044				null	null		20120127	20120127
5	007047	99999	CWOS 07047				null	null		20120613	20120717
6	007083	99999	CWOS 07083				null	null		20120713	20120717
7	007034	99999	CWOS 07034				null	null		20121024	20121106
8	007084	99999	CWOS 07084				null	null		20121214	20121217
9	007094	99999	CWOS 07094				null	null		20121217	20121217

Table | JSON

First < Prev Rows 1 - 9 of 10

Blank or “” is not the same thing as a NULL. An empty string or blank is a valid data value whereas a NULL is the absence of any data. Keep this in mind when cleaning data. If the value is properly null, BigQuery will show “null” as you see in the lat / long fields above.

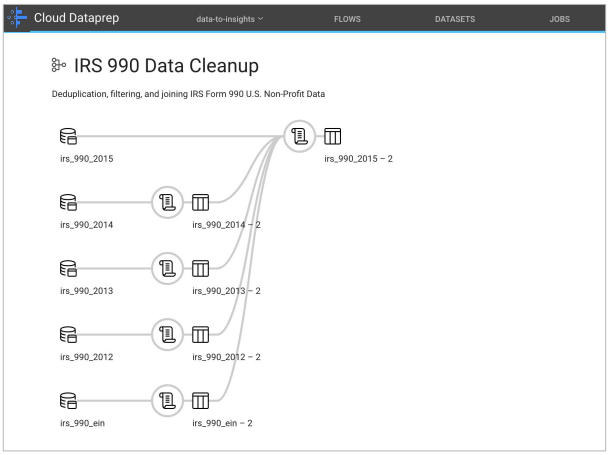
How could we adjust the query above to filter out blanks as well?
WHERE state IS NOT NULL AND state <> “”

Introducing Dataprep by Trifacta

Create repeatable data **transformation flows with the Dataprep UI**

Use Flows to wrangle your data.

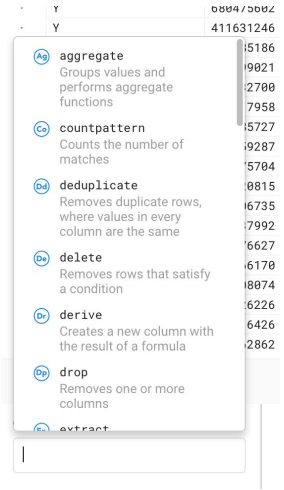
Create Flow



Dataprep flows are a collection of recipes that transform your dataset. They are visually represented in this directed acyclic graph (or DAG) which is common among workflow tools.

Transform data with a variety of predefined wranglers

- Use the Dataprep GUI to create and preview data preparation steps
- Chain together multiple wranglers into a repeatable recipe
- Common tasks like record deduplication and derived fields



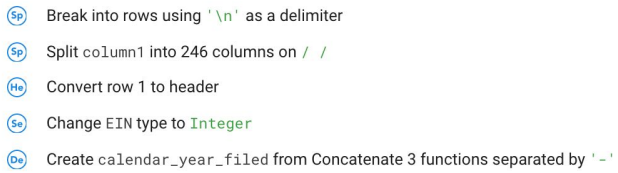
Dataprep is a graphical user interface (GUI) for creating data transformation steps. You can pick from predefined wranglers to process your data.

Dataprep guides:
https://cloud.google.com/dataprep/docs/

Access Dataprep
https://clouddataprep.com/

Chain transformation rules together into a **recipe**

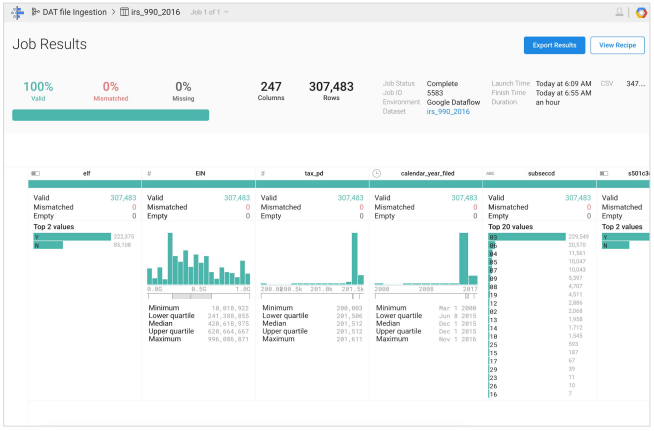
- Repeatable set of transformation steps build by chaining data wranglers together
- Jobs run against recipes
- Can include end-to-end steps from ingestion, transformation, aggregation, save to BigQuery



Recipes are a collection of wranglers. When you are done writing your data transformation recipes, you will then run your Dataprep **job** which processes your recipes against your entire dataset (not just the 10MB sample brought into the transformer).

Monitor jobs and save results as a new table in BigQuery

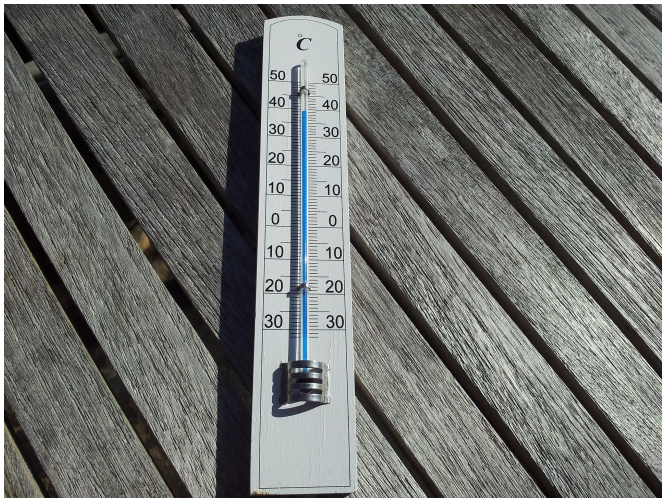
- Track completed and ongoing jobs
- See the data quality metrics for transformed datasets
- View histograms with summary statistics for each field



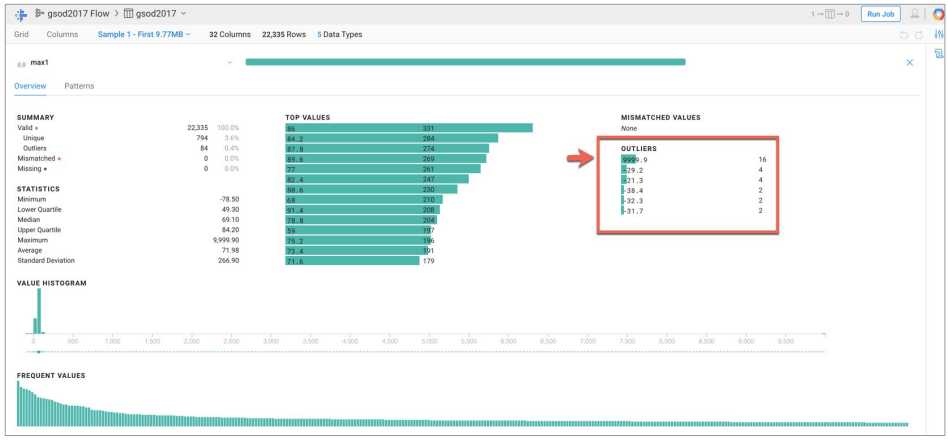
Google Cloud

https://cloud.google.com/dataprep/pricing

Cleaning NOAA temperature data with Dataprep

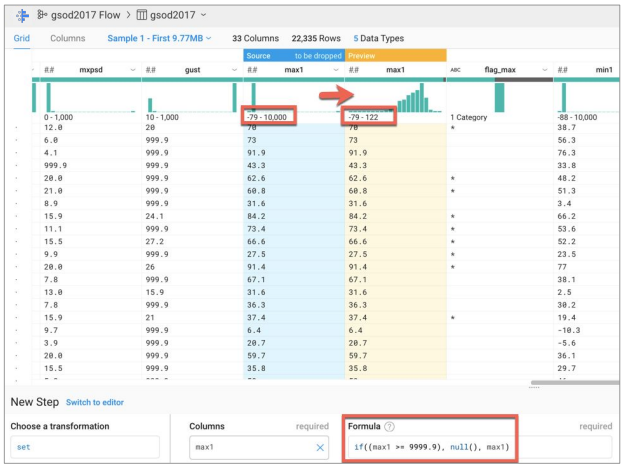


Using **column details** statistics reveals outlier max temperature



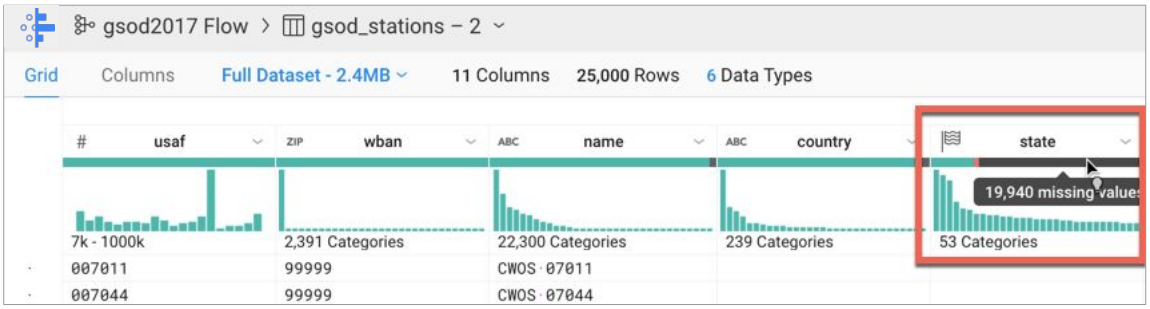
Click on a column, then Column Details to view key stats about that data field.

Set the anomalous 9999.9 temp value to NULL with a formula



Dataprep will often suggest transformations if you click on data values (e.g. filter out NULLs or create derived values).

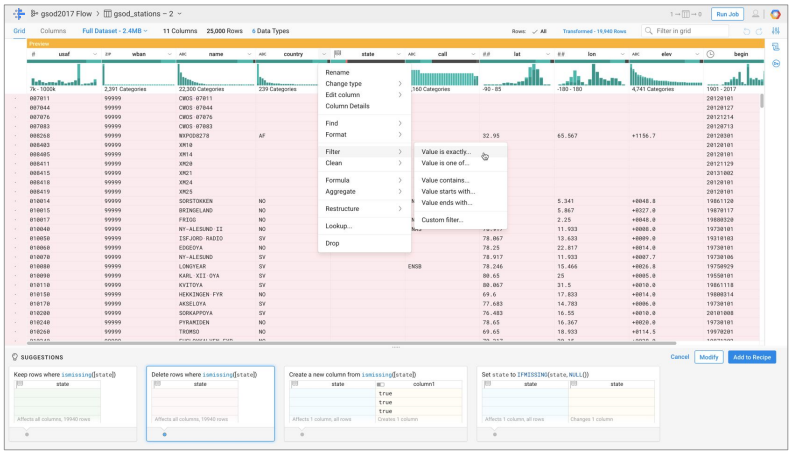
Looking at the **Data Quality bar** shows many **states missing**



The data quality bar compares the data values against known rules for data type: valid, invalid, missing.

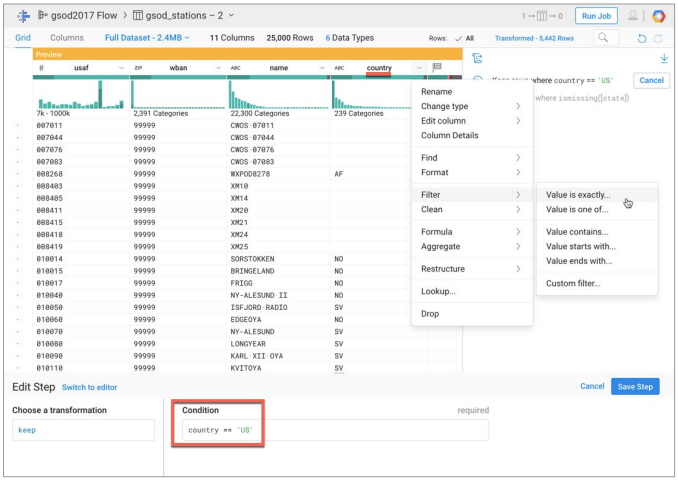
If there are too many columns to view in this horizontal Grid view, click on Column view.

Filter on U.S. only weather recordings

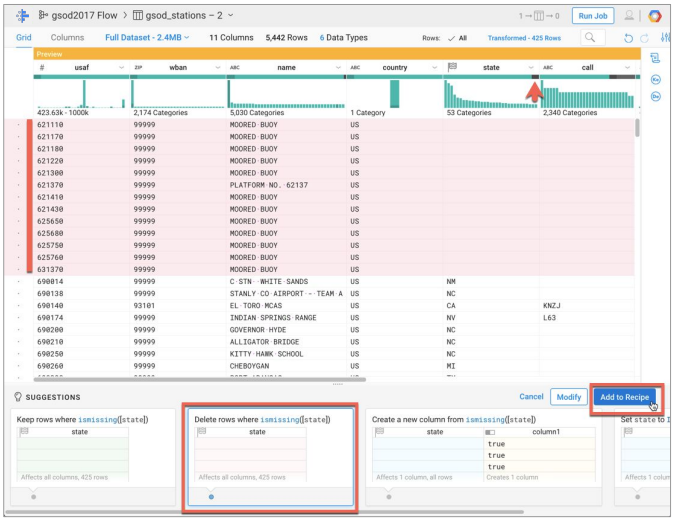


Apply filters on records based on defined conditions (much like a WHERE clause in SQL)

Keep U.S. only weather recordings



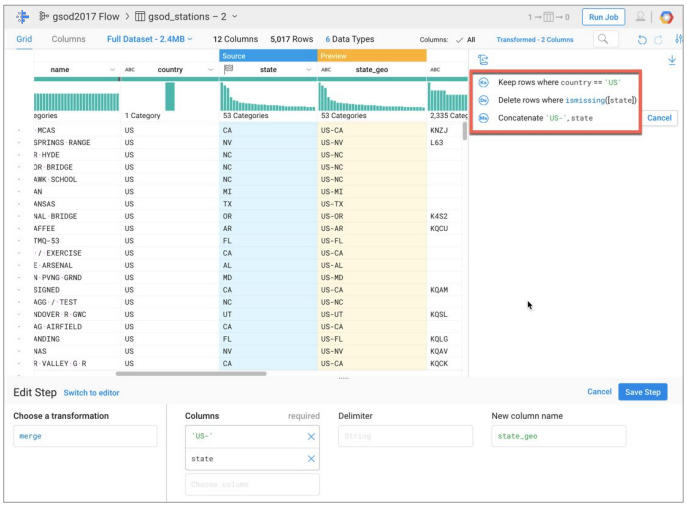
Delete missing data for the State field



- Browse through automatic **suggestion** cards for transformation
- **Modify** to customize your own logic
- **Add to Recipe** when ready

Add wranglers to recipes after previewing the transformation results.

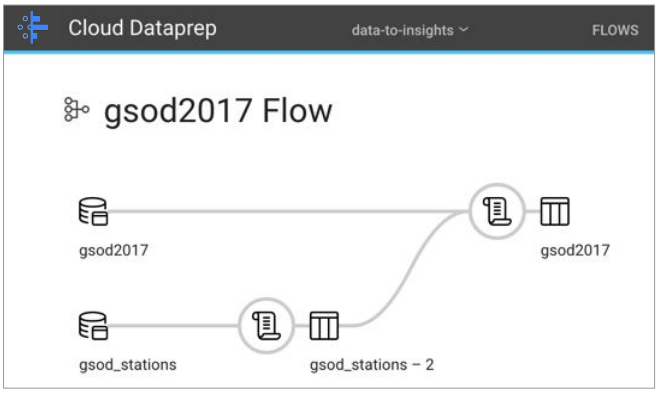
Review final recipe and save



- Toggle open the right side bar to view the steps in your recipe
- Modify or remove steps as needed
- Click **Run Job** when you want to execute

Review the final recipe by opening the recipe panel on the right side

Run the Flow which includes our recipes and outputs a table



Review the final data flow and then run the job which will then kickoff the a new Dataflow job (different tool) behind-the-scenes.

Summary: Create clean datasets with SQL and/or Dataprep

Dataset integrity includes validity, accuracy, completeness, consistency, and uniformity

Explore your data to determine if there is heavy skew which could impact performance

Clean and transform your dataset by writing SQL statements

Clean and transform your dataset through the Dataprep UI

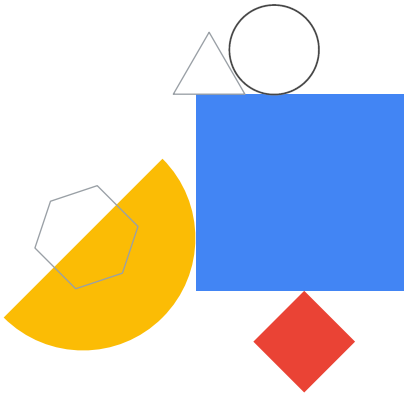
Cleaning a dataset is the first step to truly understanding all the possibilities available. You cannot even hope to get to advanced insights like building machine learning models without ensuring that you are getting the accurate and complete picture of the data.

As we discussed, exploring your dataset can uncover data skew which, when discovered early, can help prevent performance problems in the future.

Lastly we compared two ways of transforming your dataset: (1) building SQL statements as part of your Extract Transform and Load process and/or (2) using Dataprep flows, wranglers, and recipes to automatically create pipelines through a fun GUI.

Lab Intro

Creating a Data Transformation Pipeline with Dataprep



Lab objectives

01 Connect BigQuery datasets to Dataprep

02 Explore dataset quality with Dataprep

03 Create a data transformation pipeline with Dataprep

04 Schedule transformation jobs outputs to BigQuery