**Data Documentation Best Practices**

**Save time, increase transparency, and provide high-quality data**



Photo by [Luisa Brimble](https://unsplash.com/@luisabrimble?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText) on [Unsplash](https://unsplash.com/s/photos/libraries?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText)

As an [analytics engineer](https://towardsdatascience.com/analytics-engineer-the-newest-data-career-role-b312a73d57d7), I am constantly tracking down owners of various datasets I see referenced in old data models. There are a ton of Google sheets floating around, being used in complex data models, yet nobody really knows *how* they are being used. I typically have to message multiple team…

The solution to this problem doesn’t have to be complex. In fact, it’s fairly simple. You need to document all of your data sources and data models from the very beginning of development. A lot of teams think of documentation as something that’s done after the fact, when the data is perfectly structured and organized. But, will your data ever be optimal?

If we wait until everything is “perfect” to implement best practices, we are doing the business a huge disservice. Documenting your data sources and models as you build out your data stack is what will help you achieve an optimal data culture. Data documentation is *the* solution, not something that results *from* the solution.

**Why data documentation is important**

Data documentation provides your business with benefits it wouldn’t receive otherwise. It fosters a better data culture, saving your data team’s precious time and increasing transparency within the business.

**Saves time**

Documenting your datasets upfront reduces future tech debt. Rather than looking back after your stack is built and trying to remember what all of your data means, you can document as you go. The *why* behind what you’re doing is still fresh in your mind. Every time I wait until I’m finished building something to document it, I forget the intricate details of the data. You learn so much tribal knowledge in the process of building a data model that it’s worth doing whatever you can to remember it.

**Increases transparency**

When you document all of your datasets, everyone within the business has a window into the data available to them. This fewer questions about what *can* and *can’t* be done from stakeholders. It also means that analytics engineers and data analysts won’t have to hunt down the owners of different datasets every time they go to use them. When datasets are documented once, the information needed is always there to reference.

**Ensures high-quality data**

As data practitioners, there is nothing more important than the quality of your data. Documenting your data ensures it is always being used in the right way. KPIs will be consistent across models, the same timezone will be used for different date fields, and code changes can be tracked (or at least that’s the goal). Documentation helps keep track of the freshness of datasets, potential errors, and dependecies between models- all key factors in data quality

**Data Documentation Best Practices**

Now that we know data documentation is a game-changer, let’s talk about some tips and tricks that you can implement into your data stack right now.

**Create a dbt style guide for your data models.**

After reading through lots of blog posts on [dbt](https://towardsdatascience.com/what-is-dbt-a0d91109f7d0)’s website, I discovered the best practice of creating a style guide. Even if you don’t use dbt to write your data models, you should create a style guide detailing the different code and naming conventions you wish to follow.

This is one of the first things I did, before even building any data models. I am so thankful I did this because it gave me a standard to closely follow. You can have a way of wanting to write everything in your head but until it’s written down somewhere, it is easy to forget. Every time I had a question when writing my data models, I was able to look at the guide and answer it without having to look at other data models and guess what standard I was going for.

What to include in your style guide:

* Naming conventions (the case to use and tense of the column names)
* SQL best practices (commenting code, CTEs, subqueries, etc.)
* Documentation standards for your models
* Data types of date, timestamp, and currency columns
* Timezone standards for all dates

Here’s an example of a few sections from my style guide that you’ll want to model yours after:

**dbt conventions**

Only base\_ models should select from sources.

All other models should only select from other models.

**Testing**

At a minimum, unique and not\_null tests should be applied to the primary key of each model.

Run the drop\_old\_relations script after completing a model in order to delete old views/tables

**Naming and field conventions**

Schema, table and column names should be in snake\_case.

Each model should have a primary key.

The primary key of a model should be named <object>\_id, e.g. account\_id – this makes it easier to know what id is being referenced in downstream joined models.

**Document column definitions while building a model.**

We already touched on how documenting your data as you build your data models will save you time. But, it will also help to increase the quality of your data. When creating definitions for your columns, you want to make sure they are consistent and accurate. You usually have the deepest understanding of the data *while* you are working with it, rather than after the fact. This is because you’ve learned the small nuances in the data as you wrote your data models.

I highly recommend documenting directly within dbt. If you aren’t familiar with source files within dbt, this is where you document your models and their columns. dbt allows you to define table, model, and column descriptions directly within these files. Writing and storing the documentation directly next to the code ensures a smooth workflow for analytics engineers and data analysts.

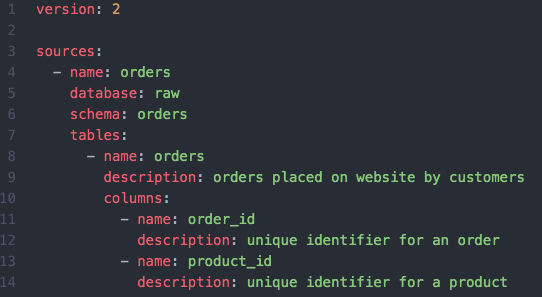
Data catalog tools like [Castor](https://www.castordoc.com/) take documentation a step further by allowing you to auto-populate these definitions defined within dbt across different datasets. That way, if you have the same column name in multiple datasets, you don’t have to keep defining it. Castor takes care of that with a click of a button, saving you precious time to focus on coding the data model instead.

**Use version control with your data models and data pipeline.**

Version control is an important software engineering best practice that can also be brought into your analytics workflow. Most teams use Github to keep track of changes made to any code. However, I also like to document different versions directly within tools like dbt and [Prefect](https://towardsdatascience.com/tired-of-airflow-try-this-c51ec26cd29d). These tools make it easy to display documentation alongside your code.

**dbt**

Within dbt, every source file has a “version” block. Here, you can change the version each time you make changes. It is so simple but it makes all the difference in keeping track of changes in your data over time.



src\_marketing.yml (image by author)

**Prefect**

[Prefect](https://towardsdatascience.com/tired-of-airflow-try-this-c51ec26cd29d) is my tool of choice for automating and deploying data pipelines. It is a Python-based tool and has a very clean UI, making it easy to use. When deploying a pipeline using the UI, you can add a ReadMe file that sits right next to the pipeline run code. I use this ReadMe as a chance to document all my different pipeline versions, including the data they sync and the models they run.

This is a very make-shift solution but it doesn’t downplay the importance that it has. Sometimes documentation can be as simple as keeping a document with a bunch of key information! There’s nothing wrong with making do with what’s available to you.

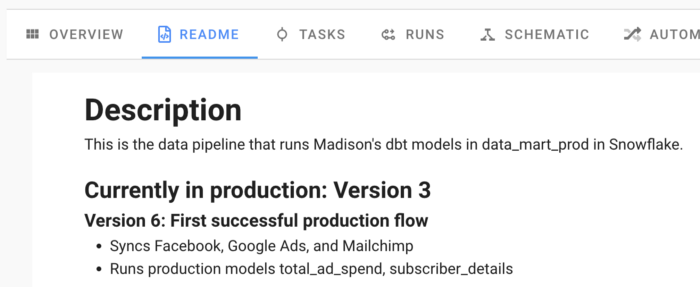


Image by author

As you can see, I document the version number along with the changes made in that version, data sources being synced, and data models scheduled to run.

**Integrate a data catalog tool into your stack.**

While my other data documentation best practices are more about making do with the tools you already use, you can also implement a tool into your stack that specifically focuses on data documentation. A data catalog integrates right within your data stack, syncing documentation from different tools like [Snowflake](https://towardsdatascience.com/how-i-organize-my-snowflake-data-warehouse-996965fe51dc?source=your_stories_page-------------------------------------), dbt, and Looker across your stack into one location. We are lucky to have great ones on the market that address the specific pain points I’ve mentioned thus far.

Data catalogs allow you to assign owners to your datasets as well as tag resources. This ensures you know the exact person to Slack when you have a question about the data or its quality status, eliminating the unnecessary back and forth. Tagging resources also allows you to separate datasets based on their business domain. This can be helpful in situations when business teams only want to view datasets in their domain. Tagging also acts as a form of data governance when used to restrict who can view certain datasets.

While these are core features of data catalogs, there are some powerful documentation features across different tools. One that I find super intriguing is Castor’s “tribal knowledge” automation. How many times have you found yourself posting in a team Slack channel, asking about a certain dataset? The people most knowledgeable on that dataset then respond in a Slack thread telling you all you need to know.

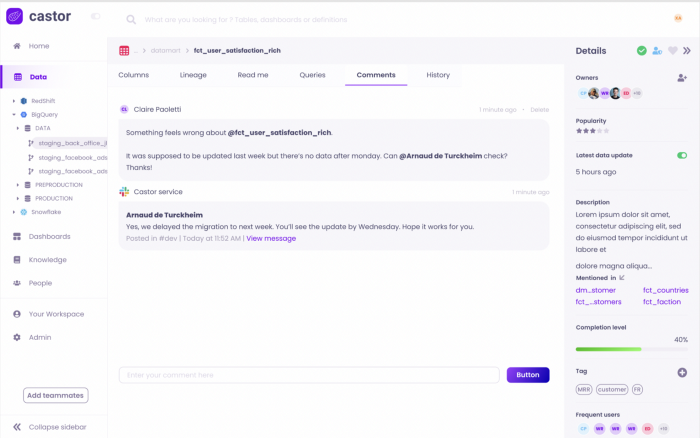


Image by author

[Castor](https://www.castordoc.com/) actually pulls the comments on a Slack thread and stores them in a section on your dataset within your catalog. This way the “tribal knowledge” is never lost! It is forever stored with the rest of your valuable documentation.

**Conclusion**

Let’s be honest. There’s nothing “glamorous” about documenting our data. It’s not the most fun thing for analytics engineers or data analysts. But, I argue that it is the most important. It is single-handedly the best way to change the data culture within a company.

When you focus on documentation, stakeholders can make decisions faster, data teams experience less friction when trying to get work done, and everyone has data they can depend on. Don’t push off documentation only to create more work for yourself and your team in the future. Prioritize it now and you’ll be happy you did.

For more on analytics engineering, the modern data stack, and best practices, [subscribe](https://madisonmae.substack.com/) to my free weekly newsletter.