Data modeling 101
Introduction

More and more companies are investing in a data warehouse and/or data lake and starting to centralize their data — or they’ve completed the centralization project and are working to socialize the data across the organization.

There are many benefits to this approach:

• You now have much more flexibility on how to analyze and interpret your data.
• You’re able to get a complete picture of your users - joining data from disparate sources that used to be siloed.
• Your data stack becomes more modular.

Flexibility is key when you’re getting more sophisticated with your data. You want to apply your own business logic to your data - for example, build custom attribution models and dashboards and reports that reflect the most important metrics for your business — not a vendor’s rigid view of your industry.

Modularity gives you choice and control (and limits vendor lock-in) — you’re able to choose best-in-class tools for each layer of your stack instead of relying on one vendor for collection, warehousing and visualization.

It also brings new challenges: lots of raw, event-level data from a plethora of sources and this is where data modeling comes in.
What is data modeling?

We'll start with a definition by Snowplow Co-Founder and CPO Yali Sassoon:

Data modeling is the process of using business logic to aggregate over event-level data to produce ‘modeled’ data that is simpler for querying.

When we’re doing data modeling, we’re typically aggregating our event-level data. Whereas each line of event-level data represents a single event, each line of modeled data represents a higher order entity, e.g. a workflow or a session, that is itself composed of a sequence of events. Data modeling includes cleaning the data, e.g. removing test records or internal traffic identified by IP addresses. It also includes creating metadata about data structures and their relationships that allow tools to act on the data with context (e.g. setting field data types in LookML)

Data modeling adds meaning to raw, event-level data using business logic. We might look at the data and decide that the page view recorded was the first page in a new session, or the first step in a purchase funnel. We might infer from the cookie ID recorded who the actual user is. We might look at the data point in the context of other data points recorded with the same cookie ID, and infer an intention on the part of the user (e.g. that she was searching for a particular product) or infer something more general about the user (e.g. that she has an interest in red pants). These inferences are based on an understanding of the business and product. This understanding is something that continually evolves; thus, as we change our business logic, we change our data models.

Companies typically write data models as SQL jobs that run inside their data warehouse. There are a number of reasons why these jobs are difficult to write and maintain, but one big issue is that aggregating event-level data is difficult. That is because:

- We’re often interested in understanding user journeys that are made of multiple different event types, with different fields collected across those different event types. These cannot be aggregated in a straightforward way.
- We’re often interested in the sequence of events. Knowing that a user performed A and B and C often is not enough — we care whether she did A then B then C, or A then C then B. SQL functions typically are no good at identifying different sequences, or letting analysts aggregate over different sequences.


---

1 Source: https://blog.getdbt.com/-how-do-you-decide-what-to-model-in-dbts-vs-lookml-/

---

---

---
Data modeling is an essential step in socializing event-level data around your organization.

The point of data modeling is to produce a data set that is easy for different data consumers, e.g. the marketing analyst, to work with using basic SQL skills. Typically, the modeled data will be socialized across the business using one or more business intelligence tools.

Limiting access to your raw, event-level data and consolidating your data models upstream means you can implement and maintain your business logic across the organization. It also democratizes data access — the threshold for querying raw data (e.g. having to join multiple tables) is much higher than having the option to query modeled data. Successful data teams spend time upfront empowering marketing, product teams and others to self-serve the reports that matter to their projects and use cases, using their favorite tools.

Once you’ve taken the leap to getting all your data into your data warehouse, your data models – if implemented correctly – will be your best friends.
When it comes to best practices around data management, Snowplow and our users take the following approach: your raw event stream should be immutable while your modeled data is mutable.

It’s always possible that some new data coming in, or an update to your business logic, will change the way we understand a particular event that occurred in the past and hence we can make an update to our data model to reflect this. This is in stark contrast to the event stream that is the input of the data modeling process: this is an immutable record of what has happened. The immutable record will grow over time as we record new events, but the events that have already been recorded will not change, because the different data points that are captured with each event are not in contention. It is only the way that we interpret them that might be, and this will only impact the modeled data, not the raw data.

We therefore have two different data sets, both of which represent “what has happened”. The raw data is an un-opinionated description of everything that happened in your business. It’s a great, comprehensive record that should be optimized for auditability and completeness, rather than for specific patterns of querying. The modeled data, on the other hand, provides easy-to-consume tables that summarize key units of analysis and makes it easy to query.

<table>
<thead>
<tr>
<th>Raw data</th>
<th>Modeled data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unopinionated</td>
<td>Opinionated</td>
</tr>
<tr>
<td>Immutable</td>
<td>Mutable</td>
</tr>
</tbody>
</table>

The modeled data, on the other hand, provides easy-to-consume tables that summarize key units of analysis and makes it easy to query.
Centralizing this effort will avoid any confusion — especially if multiple teams are engaged with modeling efforts. Here are some key metrics to be sure to consider:

- Sessions
- Bounces
- Time spent
- Conversions
- Traffic source classifications

It will seem like more work upfront — after all, it’s much simpler to rely on logic and modeling packaged analytics vendors like Google Analytics and Adobe have already thought through for you, but you lose all of the flexibility you get from owning and developing your logic yourself.

For most retailers and ecommerce companies, Google and Adobe’s data model will suit their use case. These giants have built their platforms and logic for retailers — conversion and goal tracking, funnel analysis, etc. is optimized for a traditional ecommerce customer journey. That said, many businesses struggle to make Google and Adobe work for them, e.g. if you’re a two-sided marketplace with two distinct groups of buyers and sellers or a (mobile) subscription business that wants to understand retention.

Say you’re a recruitment marketplace and you have job seekers and recruiters interacting with your platform (two distinct user groups with different behavior). When a job seeker is looking for a job, one search on the site might result in five job applications. This means that the traditional funnel or conversion rate would make no sense.

A big part of data modeling is applying your business logic to your unopinionated data. Thus, before you start with modeling, you want to agree on your business logic and centralize this. How do you define conversions, micro conversions? Do you want to define a session in the most common way (30 minutes of inactivity) or do you have another way to define sessions more applicable to your business model?
In the following sections we’ll outline the basics of a few different data modeling examples.

### Modeling macro events from micro events (e.g. video views)

‘Macro events’ are made up of micro events. Creating macro events or summary tables make it a lot easier to ask and answer questions about the event stream by simple group by and aggregation functions.

For example, if we are a video media company, we may want to understand how individual users interact with individual items of video content.

In the example above we have modeled an event stream that describes all the micro-interactions that a user has with a particular video into a summary table that records one line of data for each video each user has watched. Note how the summary table includes:

- Information about the viewer (notably the user ID). This means we can easily compare consumption patterns between different user types.
- Information about the video (notably the video ID). This means we can easily compare consumption patterns by different video types.
- Information summarizing the user-video interaction i.e. has the user completed the video, has the user shared the video?

With the summary table above we can now easily compare, e.g. viewing figures or engagement rates by user types and video types.
Our summary table aggregates the event-level data down to an individual workflow level, where each workflow represents the journey that starts with a user performing a search, and ends when that user:

1. Makes a purchase of one of the search results, or
2. Performs another search, or
3. Drops out altogether.

Our aggregated data table contains a lot of interesting data, including:

- Data about the initial search that was performed. This is data that will be fetched from the early events in the workflow.
- Data about the actual results that the user interacts with: which results were selected, where they were ranked in the results.
- Which result (if any) the user selected and went on to buy. This data will be taken from later events in the workflow.

Again, note how the modeled data should be easy to work with. We can easily compare the number of searches to different destinations, compare conversion rates by destination and explore whether higher ranking results are more likely to be selected and then purchased, and see how this varies by user type, location and other key data points.

---

**Event stream**

User searched for flights to Denver → Search results served → User selects first result → User returns to search results → User selects second result

**User flight purchase workflow**

<table>
<thead>
<tr>
<th>User</th>
<th>Destination</th>
<th># of results</th>
<th>Buy button presented?</th>
<th>Provider selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Denver</td>
<td>127</td>
<td>yes</td>
<td>American Airlines</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Modeling sessions

Building a sessions table is a way to aggregate your event-level data into smaller data sets. Sessions are meant to represent continuous streams of user activity. In web, it is common to define a session as ending when a period of 30 minutes passes and no event takes place.

At Snowplow, we don’t like treating sessions as the only unit of analysis (some analytics vendors still defer to sessions for everything). For example, if you’re a recruitment platform you might want to look at the time period from when the user first searched for the job until they apply, and simply aggregating sessions would tell you very little about how effective your platform is. For certain uses it obviously make sense to look at sessions, e.g. if you’re a newspaper it could be interesting to understand how many articles a user reads in a given session.

<table>
<thead>
<tr>
<th>user_id</th>
<th>session_id</th>
<th>mkt_source</th>
<th>session_start</th>
<th>session_end</th>
<th>events</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:jane.doe@gmail.com">jane.doe@gmail.com</a></td>
<td>5</td>
<td>google</td>
<td>2019-08-26 14:02:04</td>
<td>2019-08-26 14:04:37</td>
<td>[page_view, page_view, product_click …]</td>
</tr>
<tr>
<td>43ed06459</td>
<td>1</td>
<td>facebook</td>
<td>2019-08-26 15:15:22</td>
<td>2019-08-26 15:29:01</td>
<td>[app_open …]</td>
</tr>
</tbody>
</table>

Typical aggregations will include data on:

- How the user was acquired (source)
- Where the session started (e.g. the ‘landing page’)
- How long the session lasted
- What the session was “worth” (e.g. did a conversion/transaction occur? How much ad revenue was made?)
- What was the user trying to accomplish in the session?

Using our sessions table we can answer questions such as what number of sessions contain a conversion, how session length has changed over time and what type of sessions influence conversions or leads a user to churn.

Here’s an example of how to model sessions on Segment data (BigQuery) and on Snowplow data.
Modeling users

A very useful table will be your user table - aggregating a user’s event history over time. Such a table could be joined with data from your CRM system (for example to be made available to customer service agents to help users if they are having trouble with your product).

A typical user table will include data on:

- User account creation
- User activity
- And transaction data (e.g. number of transactions and LTV)

Based on the entire user history, we can bucket users into different cohorts based on when they were acquired or how they were originally acquired, categorize users into different behavioral segments and calculate actual and predicted lifetime value for individual users based on their behavior to date.

Here’s an example of a users table using Snowplow data.
Democratizing data with data modeling

The most successful data teams build a data infrastructure and platform that democratizes data access in the organization. Without it, the data team will often need to spend time supporting other teams (e.g. marketing or product) with reports and ad hoc analysis.

Especially if and when you’re embarking on a data centralization project, you want to make sure that your teams are empowered and feel comfortable with the tools and data at their disposal. For example, if your marketing department has been used to working in Google Analytics, you can’t replace that with a SQL-based visualization tool.

From our experience, the most successful data teams - large and small - succeed when a data centralization project is immediately followed by comprehensive data modeling (including all the relevant data sources) and a powerful visualization tool. Marketing will embrace your solution when they’re empowered to self-serve and see the opportunities they suddenly have; e.g. building a custom attribution model taking into account more sources like ad impressions, app installs, cancellations/returns, etc.

With solid data modeling and basic SQL training, data teams empower other teams to self-serve and can focus their time on building data-driven applications or products, not writing trivial SQL queries to support the product analyst’s weekly report.
Data modeling tools

Snowplow users can model their data as part of their pipeline using SQL Runner. SQL Runner is a simple open source SQL scheduling tool developed by the Snowplow team that can be used to script ETL or data modeling pipelines in SQL.

Other tools worth mentioning are dbt and Dataform. Both tools are built for data analysts and operate in your cloud data warehouse, simplifying the execution of templated SQL scripts against a storage target. Both dbt and Dataform offer a free version of their product and largely offer the same features. The main differences are that Dataform also offers a web interface for teams to develop, test and schedule their SQL pipelines whereas dbt is a command line interface (CLI) tool only2.

If you’re modeling your data for the purpose of visualizing it in a BI tool, some tools offer built-in data modeling. In the case of Looker, LookML is their data modeling tool, basically teaching Looker how to navigate your data. Using LookML can be easier in the short run, but bear in mind that if you at one point decide to stop using Looker, your data models will be lost too. There are also limitations to what you can do in LookML, e.g. creating a sessions table: you can’t sessionize data without writing custom SQL— the query is too complex for Looker’s query model3.

---

1. Source: https://news.ycombinator.com/item?id=20441736

---

Snowplow users can model their data as part of their pipeline using SQL Runner. SQL Runner is a simple open source SQL scheduling tool developed by the Snowplow team that can be used to script ETL or data modeling pipelines in SQL.

Other tools worth mentioning are dbt and Dataform. Both tools are built for data analysts and operate in your cloud data warehouse, simplifying the execution of templated SQL scripts against a storage target. Both dbt and Dataform offer a free version of their product and largely offer the same features. The main differences are that Dataform also offers a web interface for teams to develop, test and schedule their SQL pipelines whereas dbt is a command line interface (CLI) tool only2.

If you’re modeling your data for the purpose of visualizing it in a BI tool, some tools offer built-in data modeling. In the case of Looker, LookML is their data modeling tool, basically teaching Looker how to navigate your data. Using LookML can be easier in the short run, but bear in mind that if you at one point decide to stop using Looker, your data models will be lost too. There are also limitations to what you can do in LookML, e.g. creating a sessions table: you can’t sessionize data without writing custom SQL— the query is too complex for Looker’s query model3.

---

1. Source: https://news.ycombinator.com/item?id=20441736

---

Snowplow users can model their data as part of their pipeline using SQL Runner. SQL Runner is a simple open source SQL scheduling tool developed by the Snowplow team that can be used to script ETL or data modeling pipelines in SQL.

Other tools worth mentioning are dbt and Dataform. Both tools are built for data analysts and operate in your cloud data warehouse, simplifying the execution of templated SQL scripts against a storage target. Both dbt and Dataform offer a free version of their product and largely offer the same features. The main differences are that Dataform also offers a web interface for teams to develop, test and schedule their SQL pipelines whereas dbt is a command line interface (CLI) tool only2.

If you’re modeling your data for the purpose of visualizing it in a BI tool, some tools offer built-in data modeling. In the case of Looker, LookML is their data modeling tool, basically teaching Looker how to navigate your data. Using LookML can be easier in the short run, but bear in mind that if you at one point decide to stop using Looker, your data models will be lost too. There are also limitations to what you can do in LookML, e.g. creating a sessions table: you can’t sessionize data without writing custom SQL— the query is too complex for Looker’s query model3.

---

1. Source: https://news.ycombinator.com/item?id=20441736
Introducing Snowplow Insights

**Snowplow** is an enterprise-grade data pipeline built for data teams. With Snowplow you can collect rich, high-quality event data from all your platforms and products.

Your data is available in real-time and is delivered to your data warehouse of choice where it can easily be joined with other data sets and used to power BI tools, custom reports or machine learning models.

Because of our laser-focus on data collection, Snowplow data is extremely granular and high quality. Snowplow’s pipeline is designed to minimize missing, lost and inaccurate data. Snowplow data is consistent (web, mobile, server, email and 3rd party) and well-structured making it ideal for modeling, no cleaning or structuring needed.

“Snowplow provides all of our event data in a data model we own end-to-end, and can easily shape to fit our organisational needs. Snowplow has really helped accelerate our analytics; now we can quickly answer questions that would have required a tremendous amount of engineering effort with our previous solution.”

Darren Haken, Head of Data Engineering, Auto Trader

---

**GET IN TOUCH**

to learn more about Snowplow Insights.