EPISODE 1449

[INTRODUCTION]

[00:00:00] JM: Data infrastructure is a fast moving sector of the software market. As the volume of data has increased, so too has the quality of tooling to support data management and data engineering.

In today's show, we have a guest from a data intensive company, as well as a company that builds a popular data engineering product. James Densmore works at HubSpot, which produces tons of data and Lior Gavish works at Monte Carlo Data, which sells a data quality product.

[INTERVIEW]

[00:00:25] JM: Lior and James, welcome to the show.

[00:00:28] JD: Thank you. Thanks for having me.

[00:00:29] LG: Thanks, Jeff. Thanks for having me as well.

[00:00:31] JM: I'd like to start by talking about the modern data ecosystem, mostly in terms of the data warehouse. Can you describe how the data warehouse has become the centerpiece of data infrastructure? Why it has?

[00:00:52] JD: Sure. I can start. So I've certainly have seen a lot of changes in data warehousing technology over the last decade plus in analytics, and I think more recently, especially the advent of cloud data warehouses, starting with Amazon Redshift, going into Google BigQuery, Snowflake is now a massive player. I think both the cost to stand up a data warehouse being a lot smaller, you can start small and scale up, as well as the performance of data warehouses going from sort of leveraging application databases, row based databases to the calmer MPP databases we see today. For data warehouses, I think the combination of those two things, the technology, and the cost efficiency of the cloud, is why we are where we are today.

[00:01:40] LG: Yeah. I second James on that. It's definitely the move to the cloud has made it so simple to both get started and also to scale things. I think, historically, if you needed to process large

amounts of data, you would maybe have the temptation of putting together various technologies and try to build your own, do it yourself data warehouse, or use a kind of a legacy on-prem solution. But now with Snowflake and BigQuery, and Redshift being so cost effective and so scalable, that temptation is lower, and pretty much any data use case can be served by those oftentimes cost effectively. Just the convenience of having a relational database, that SQL friendly, that requires very little operational overhead is just – that package isn't sailing, compelling to most data teams out there.

[00:02:35] JM: How does the centrality of the data warehouse in comparison to all the other data sources like CRM tools, and Stripe, and all the other just voluminous sources of data? There are all these connection points between them, like you might want to do a join between data and Stripe and data in your data warehouse? You're doing ETL jobs between these different places. The fact that there's all these different data sources in addition to the weight of the data warehouse that's created a necessity for a variety of new data tools. Can you describe that data tooling landscape as you see it, and give an outline of the different problems that all this data tooling creates?

[00:03:28] JD: Yeah, I think it's been a blessing and a curse to have such an explosion of data tools come onto the scene, especially in the last five years or so. Before that, it was a lot of homegrown do a lot of exactly the things you're talking about. Having data in so many different systems is nothing new, but trying to pull it together and analyze it, is certainly something that is more prevalent today, even in organizations that aren't necessarily in tech or highly data driven. So, the tooling space, I think it's the blessing side of it is there's a lot of options for data teams. You don't have to either build everything yourself or use the limited tools that were of a particular ecosystem.

Microsoft was famous for kind of building their own BI ecosystem back in the early 2000s. Some other vendors took the same approach, and now you can mix and match. There are tools all the way up and down the sort of data pipeline, getting data into your warehouse, getting it out of your warehouse, orchestrating jobs and dependencies, building data modeling, data, observability, testing, validation, and all that. So, behavior it's become more of a mix and match, which I think is a good thing, mostly, but it is potentially overwhelming for a lot of teams, and I think has created even sports within the data world as far as what type of people you need on a data team as well. Certainly, the role of a data engineer is something that it's gone through many iterations, but it also means different things to different companies depending on their data stack, whether it's sort of no or low code heavy, or if it's still engineering, software engineering, heavy as well.

All in all a good thing, but certainly quite a different ecosystem, just in the volume of options, and just a few years ago.

[00:05:12] LG: Yeah, I agree with that, that the complexity has certainly increased. And the trend, you're talking about, Jeff, which is to kind of start bringing in data from a lot of different sources, and some of the some of the people that I talked, bring in hundreds of different data sources into the data warehouse, kind of creates a lot of needs or gaps in the stack. You need to be able to bring that data into the warehouse in the first place, so there's kind of a good choice of data ingestion tools that can move data around into their warehouse. You then need to, in order to get value out of it, and like you said, kind of join things together from different sources, you need to be able to kind of transform and model the data, essentially.

So there's a good choice of tools out there to do that, and to really derive meaning and insight out of data. And then of course, you have to deliver that value to the business ultimately, right? That comes through BI tools and visualization tools that help people understand the data and make decisions based on it. There's another whole ecosystem around creating machine learning models that will presumably make automated decisions for the business. And then we're also increasingly seeing cases where people use the data platform to really build product features. When you go into a modern application and see analytics and reporting, on top of how you use that SaaS service, that's oftentimes driven by the company's data platform by the data warehouse. So, there's a whole stack of solutions to help with that.

One of the challenges, and I agree with James 100% here, is just the complexity of the system is growing. The more data sources you have, the more pieces to the puzzle. In terms of your stack, the more difficult it is to understand what's happening and how the platform is being used, and it creates a bunch of different challenges. One of them that's obviously near and dear to my heart is reliability. So how do you even know that all these things are working as you expect them? Monte Carlo is a data observability company to try to solve that, with that approach. There are challenges around data discovery, actually, like how are you assuming the team is creating all these different products on top of the data platform. How do you help people find what they need, find the data that they need, find the dashboards they need? And then there's also challenges around security and access control and governance. How do you make sure you're exposing things in a responsible manner?

So, the potential is just huge, and there's so much that you can do today with a modern data warehouse and a modern data platform. But at the same time, forces the data engineering teams to shift and to start tackling new challenges that come with that scale.

[00:08:04] JM: So James, in your work at HubSpot, there's tons of data to deal with, and is your work mostly focused on organizing like metadata within HubSpot? Or like metadata that belongs to HubSpot? Or do you also spend time architecting how user data flows through the system?

[00:08:32] JD: Yeah, it's a little bit of both to be honest. My team, I weighed this central data engineering and data warehousing functions kind of like a data platform team, and we're serving a lot of a number of internal analytics engineers, data analysts, that are embedded throughout different parts of the business and they have different needs. So our job, a big part of it is just what we talked about in terms of getting data into a warehouse and kind of joining it up and combining it for value. The majority of the data that we're working with is created internally for us. So from the HubSpot application, our billing systems, all of our internal tools, that is the majority. And then we of course, we're a SaaS company, we use other SaaS products, and we pull data from some of those into our warehouse to join up with our internal data as well.

And then from there, there's different use cases. There's certainly the more traditional reporting for finance or for sales to track their workflows and their weeds and all that. There's also marketing, using data to send better emails and customer engagement. And then, as far as the product itself, there are multiple teams at HubSpot focused on using data in the HubSpot application as well. So we're using it both to run our own business and then to also help our customers who are HubSpot customers, in their businesses. I think something that is definitely on the sort of trending side, it has come up is, that using the data warehouse as a source for operational needs that was something that was not necessarily unheard of, but very risky. I think not that long ago, to have your data warehouse behind either a production application, or some kind of operational workflow, perhaps like an email workflow or something like that. That's become much more, not just popular, but also possible and safe, given a lot of the tooling that's being built, and just the more mature engineering processes on data teams now.

So not that long ago, I always felt the data warehouse was something you didn't want to consider production. It was very much, you wanted to trust the data for reporting and more traditional analysis of data. But those operational workflows and products use cases, those always gave me pause, and I think a lot of data leaders pause. But with the advent of both some of the modern data tooling, as well

as, again, more mature practices of data engineers, and similar folks on data teams, works a lot more like the software development world now, that has given us that confidence to start branching out beyond the more traditional analytics and into some of those other use cases that are more either customer facing or operational within our own business.

[00:11:23] JM: Can you talk about what the data landscape looks like? I'm specifically interested in OLTP databases, ETL tools, and your OLAP setup, what you give to your data scientists. I just like to get a sense of the data landscape.

[00:11:44] JD: Yeah, absolutely. So, I like to think are the central sort of point of all of our data infrastructure is our data warehouse. And for us, that's Snowflake. When we think about where we're getting data from, now, as I mentioned, a lot of our data being a software company, we just generate a lot of our own data, and that's coming from a number of sources across the rest of our engineering infrastructure. So those could be my SQL databases, that could be internal rest API's. We make heavy use of Kafka, so we have a lot of event data coming through. There's vault data dumps, and S3 buckets. There's just a number of different sources internally, that our other engineering teams besides my team, are using to build the HubSpot application internal tools, and you name it. We're building on that same platform.

So, we're probably – I don't know if we're in the minority, but certainly in the minority who are on the more modern data stack, and that we primarily have built our own ingestion pipelines from those sources into Snowflake. We do utilize some frameworks and reusable ways of doing that. But we have a data engineering team in my department, there are really backend software engineers that are very focused and knowledgeable about data pipelining, getting data from all those sources into snowflake, ensuring that it's on time and reliable and valid, but they are building those pipelines themselves. And so, that's something for us that we've chosen to be very, very custom on that end of our pipeline on the initial sort of ingestion piece. And then we're using a lot of, I would say, more third-party tools further down the stack.

We're using Apache Airflow as an open source orchestrator for a lot of the jobs that we run, can handle dependencies through DAGs. We leverage DBT, for our data modelling. So another open source tool that is really become more and more popular out there in the last few years. And then we're using Monte Carlo for our data observability, data validation and testing, and that's something that's newer to

our stack, but proven to be very valuable in terms of like finding where there are issues and data that we may not have bought the right tests.

Before I mentioned, data engineering is becoming a lot more similar to the software engineering workflows and life cycle and everything. And so, those engineers are writing tests on their data ingestions. The data warehousing team is writing tests, on their data models that they're building in the warehouse, but we're only writing tests for the things that we can think about. So having a more general observability tool, it's been great for us. And I think it's something that a lot of teams are investing in. And then sort of like past that, we think about tools, like Lookers our more like enterprise BI tool. That's where a lot of analysts and power users in the business are building dashboards and reports. And then there's a number of teams, data scientists, or some data analysts are a little bit more code savvy that are doing work in Python or R. And so we have – they're using a lot of Jupyter notebooks, things like that.

So, we do have some infrastructure as well that teams like that can deploy their scripts and jobs, scheduled on, through some of our orchestration tooling, and run those models as well. Of course, you never want to leave out things like Excel and old-fashioned tools that people are using and still get a ton of value out of. But we do try to centralize as much as possible on our infrastructure, mostly for single source of truth, but also for security and privacy reasons. The more that we can govern appropriately, the easier it is when we're going through and making sure we're securing our data properly. So all in all, compared to other teams that I've worked on and with, our team is very software engineering heavy compared to some data team. I think that speaks to us being part of a software company, but also us deciding that there are certain parts of our data stack, we really want to own ourselves. So, we still do some custom tooling in those departments.

[00:15:51] JM: So given that Lior is on the call, and he works at Monte Carlo, I'd like to get an understanding of what data observability means in practice for a giant organization like HubSpot, and why it's important, what you're validating, and yeah, just give me a sense of how data observability fits into what you do?

[00:16:16] JM: For us, data observability, that term is something that is relatively new to our organization, but the need and the practice of it is not. So what I mean by that is being a newer, an area where we're seeing a lot of tools like Monte Carlo come up, that's not something that came out of nowhere in terms of like making up a problem. I'm a big believer in – there are plenty of problems out

there to solve, and I love when vendors are coming in and solving those problems. And in this case, I think data observability is something that we have struggled with and every data team I've been on a struggle with, and not really been able to pin down and name as an overall problem.

So, what it really means for us is, we have hundreds of different data sources that we're adjusting into Snowflake. The majority internal, some third party, those have been built out over the years. It's just not feasible for data engineers to not only build the pipelines to get the data into Snowflake, but to think about how to test and monitor the voided of all that data. So of course, when they first build out that pipeline, that ingestion, they can go in and say, “Hey, does this data match the source system that I'm pulling it from?” That's kind of a basic day one test. They'll certainly be able to write some basic tests, perhaps in sequel to check to see unique values in a column if there's not already a key in there. Are there things like doing some basic tests around row count, step changes, base? Did we get 100 rows yesterday and 6,000 rows today, or 6 million rows today? Maybe something's wrong there.

We've done basic things like that, but it's just not scalable when you think about the diversity of data sources that not just HubSpot, but all organizations are leaning into. I say diversity and data sources, because maybe 10 years ago, data volume, felt like the biggest challenge. And now to me, it's not data volume, it's that breath of data, and just the thought of trying to come up with ways to keep track of what are these hundreds of different data sources coming into my data warehouse? How do I know they're valid? How do I trust them? How do I monitor their growth over time? How do I know who's using them? That's another piece.

So, even if the data is valid, who's using that downstream? What is the lineage of that data all the way from where I ingest it, into those different data models and Looker dashboards? And then when something does go wrong, how do I know who to tell? HubSpot is a fairly good sized company. Now, if we have one of our data ingestion jobs fail, do we send a note out to hundreds of analysts and software engineers? Or can we use something like Monte Carlo to figure out, “Okay, who's consuming that data? What are they using?” And then have that award them more targeted, otherwise, it just becomes a word fatigue, which is something that we were suffering from, not that long ago, where we got better at writing tests and alerting. But when you have so many different jobs, and you're checking little things that could go wrong, and it just becomes too much. So targeting the alerting, understanding ownership.

And the last thing I would say is really understand what data matters, what data is even being used. What are your key data assets over time, over years, as you build out a data warehouse. You sort of

build some tech debt into it, but you also build up things that were important a few years ago, that may not be as important now. So looking at usage metrics, and things like that has been really valuable for us to go back and say, “Do we need to shut down from these old jobs and use cases? They might be causing more confusion and they are value.” So to me, overall data observability is that, great, we have all this data, that's not our problem anymore, we can process that amount of data. It's not a computing problem anymore. It is understanding and trusting our data once it's in our warehouse. So that's been the big shift for us on data observability.

[00:20:17] JM: So, could you go a little bit deeper into how that works in practice? Maybe give an example of a common data error that could occur, and how data observability remediates that.

[00:20:37] JD: One that comes up quite a bit is, a change in one of those source systems that we're ingesting data from, that we're not expecting. I don't mean a breaking change like one of the software engineers, who's building that source system did something that broke our data ingestion. So it's not like a hard error type situation where we're expecting a certain JSON structure or something like that, and we didn't get it. That might throw a hard error and we know something is wrong. The ones that are really scary, are ones where that's fine, but the meaning of the data has changed. It could be that we've started doing hard deletes, instead of soft deletes at table, or tracking a new event is being logged that we didn't know about, or data is duplicated, something like that.

So when we use something like Monte Carlo, we pointed out the history of those tables, and that data is telling us, “Hey, something’s different here.” It doesn't always tell us why. But it might point us to the volume in this table has shot way up or way down, and we've seen that in terms of change to a hard delete versus a soft delete in the source system. And sometimes when we go back, we're doing a full load of that table. Other times, we're doing incremental, but when we see these changes in either volume, or makeup of the data, that alerts us, literally alerts us, via email or Slack. And then we have an analyst, usually, we try to have those alerts go to the analyst, that is the primary owner of those tables, if there's a default, it comes back to my team, and they're looking into and going, “Something's wrong here.” It may be something – without a tool like this, a lot of times the way you found that was somebody ran a report or a dashboard, weeks or months later and said, “Something's not right here.” It's sort of the nightmare of every data team is getting that question from a stakeholder saying, something's not right. There’s not a whole lot of information. But it just doesn't work right. And we would rather get to the points that someone consuming a report or a dashboard coming to us. We actually want to go to them and say, “Hey, be careful today. Something is – we're not sure what it is yet, but

we're looking into it, something's not right in the data. So just kind of pause any critical activity or just, you know, keep an eye on that dashboard."

And that, in itself builds a lot of trust with our stakeholders in the business, that we're finding things before they are because that is completely the opposite of how BI analytics is kind of always work. It's always the stakeholder seeing something wrong and going to the data team. If that happens enough, you just have a distrust in the data that even if it is right, people start to question. So, it really is those source system changes that I think are the ones that are quite common. And often, sometimes it's no, nothing is wrong. But there was either a big sale or something happened in the business. Other times there is something wrong, and we just want to know as soon as possible, before it actually does break something. Or we make a bad decision down the line because we're working with invalid data.

[00:23:43] JM: Lior, when you heard James talk about the proclivity of data problems within HubSpot, does that align with what you see from other enterprises that you talk to? Or where do you see the most common data problems, data inconsistencies emerging from?

[00:24:07] LG: Yeah, actually, thank you, James, for kind of laying that out. It's always exciting for me to hear when we're able to kind of make an impact on our customers and help them the challenges that they have. And to your question, Jeff, it definitely aligns a lot with what we're hearing across the board. I think James is definitely ahead of the curve in many, many ways. But other teams will have similar challenges. I think there's a lot of factors that go into creating trust and data, right? One factor that James mentioned is you're just ingesting data from hundreds of sources that you don't control. The example that James gave is where there's a change that another team is making. If you're lucky, they'll let you know that the change happened, but it's extremely hard to scale. And you're typically also not – you're taking data not only from other engineering teams are, which you may be close with, but you're also taking data from your marketing team, and from your sales team, and from your operations team. They may not all be attuned enough to letting you know when things change and things happen, and that can actually impact the analytic stack, sometimes in very unpredictable ways, in those soft ways that James mentioned.

So that's a big issue that we're seeing. Another thing is as data engineering teams are scaling, they kind of run into some of the challenges that software engineers oftentimes encounter in a complex environment, in a kind of maybe a microservices environment where it's sometimes hard to understand the implications of a particular change, right? Someone may be changing one other transformations, or

creating a new one, gets consumed downstream, and it can sometimes be hard to anticipate how that change and really the code that's transforming the data might affect the system, and how it will apply to production data that it's always hard to test on during development, and how it might affect downstream consumers that you don't fully understand, or you don't fully understand how they use the data.

So that's another source of kind of errors or problems that we see customers struggling with. Another area where we see issues coming up, there's a lot of – we talked about it earlier, that the stack is now pretty complicated and it only works if a lot of different tools work together, right? In James's case, starting from the Kafka streams, and then then later on with Airflow and Looker, and the data warehouse, and all these things have to work well together, and they have to be configured properly.

I'll give a very simple example. Someone might change permissions on one of those systems and cause data to be available, or someone might change the scheduling on Airflow and that might have unintended consequences, right? So changes to that operational environment, changes the configuration, or kind of transient failures there can really impact the correctness of data, and trust in data. So, we're seeing that happening, too.

Now, some teams might have very good observability and monitoring on those systems individually. But it can sometimes be challenging to do that at scale, and to put the proper alerting on each one of those systems and to understand how these interactions impact the data that's being consumed. That's where data observability really comes in. It allows you to monitor the system as a whole and understand how all those different pieces interact, to create the end result, the dashboard or the model or the product experience.

[00:28:08] JM: So when you look at the potential insertion points for Monte Carlo, ensuring data observability. Can you tell me where the best insertion points are? Because I mean, I think people listening probably understand that there's a lot of different places where data can go. There's a lot of different places where data can end up, and presumably you would want to be able to check or validate the health of data everywhere in the pipeline. So I just like to get a sense for when you look at the “data mesh” as a whole, where do you want to place validation checks?

[00:28:56] LG: It's a great question. We've fundamentally believe that what you said is true. That really to understand data health, you need to understand how things evolve and change throughout the process, right, all the way from those Kafka streams and through the data warehouse and all the way to

the end products, to the reporting layer, or machine learning. So we from our side, taking the effort of creating integrations with all those different pieces. So the people can understand how the system functions, and to also be able to track problem in the earliest possible moment, right? Like you definitely want to catch issues as upstream as you can. Having said that, you did ask what's like the kind of best place to start and it really depends on the kind of data architecture. But in general, my advice would be to start as close as possible to the end product to the thing that gets consumed. And that typically means starting from the data warehouse, and perhaps the BI layer.

For most teams, this would be pretty much the thing that gets consumed by end users. The reason to start there is A, it's the closest possible to an end product. So, you're going to get a lot of signal there, right? If something breaks, there's something is wrong there, it's very likely impact for the business, and very likely something that you need to take care of. Whereas if you look at the more kind of upstream parts, there could be a lot of different issues that are happening or breakages, some of them may be important, some of them might affect like your most precious products, some of them may not. So, that's a place to scale as you kind of mature with data observability.

So, for most teams, I'd say that the best place to start is a data warehouse in the BI layer, and then extend that to more and more pieces of the stack. Another reason to extend it to other pieces of the stack is that it's critical in terms of really handling those issues and getting to resolution. I'll give you an example, in James' case, they might get an alert about something on Snowflake that is breaking, to really be able to get to the bottom of it, they might need to understand the Kafka stream from which the data came from. They might need to understand what's happened on the most recent Airflow run? So, having visibility into those parts of the stack is helpful, not only from a kind of validation alerting standpoint, but also from the standpoint of how to efficiently and effectively address those problems and investigate them and get to resolution.

[00:31:48] JM: Coming back to James, the end users of data, the biggest data consumers at high volume that I see from, from my perspective, are the data scientists and the people who are consuming dashboards and reports, which is, in some sense, most of the company. But the people who are interacting with the data the most, the big data, are those people who are designing the dashboards and the people who are performing big queries against large datasets. Are these people responsible for the data integrity? Are they responsible for the data cleaning process and the data maintenance process? Or are the upstream producers of the data more responsible for that data cleaning?

[00:32:41] JD: We have it built out, we're clear on like the roles and responsibilities throughout that. So I don't think that people you're referring to in particular, we started to shift to like a job title called analytics engineers, for those folks. That's the title, I know, the team over a DBT, I think coin, so I'll give them credit. But a data analyst with a bit more engineering, like software engineering and technical ability, at least as far as using things like Git and some command line and some a little bit of Python or scripting. Those folks, they're the ones in between the data that my team is delivering to the data warehouse, or my data engineering team is, and the folks building those dashboards and consuming them, as you mentioned.

So those analytics engineers are the ones working with those high volumes of data, writing massive queries and building the data models that are then used in dashboards or further analysis or reporting. They're responsible. They're sort of in the middle. So let me take a step back. Our data engineers are responsible for ensuring the data delivered into Snowflake matches those source systems that they are ingesting data from. So, that's their job, and that's something that they're on the hook for writing tests and handling the words that come in, and all that. Once that data is delivered, and those analytics engineers are turning it into something that is consumable and understandable for analysis, that's their responsibility on data integrity there. They might start building a model and say, "Well, something's wrong here." And they might go to data engineering and say, like, "Hey, everything's good?" "Yeah, everything's good." And then they might have to go back to that source team and say, "Hey, can you tell me a little bit more about how this data is structured or what this means?"

But once they've built that, they're writing kind of like unit tests, if you will, for all their data models, and then also relying on words from something like Monte Carlo, after things are built as well. So they're responsible for those data models, and we break it up basically, almost like by schema in Snowflake. So we have different owners for the data integrity of different schemas. And within those schemas, individual teams might break it down even further. We even do things like look at GitHub commit history and see who made the last change. We might send an alert based on that. And then the next phase out is okay, those data models are great. We're monitoring them. What about all the dashboards and reports that people build? That's actually where it gets a lot harder to be honest, because there's so many more people, to your point, like almost the entire business at that point, is not even just like consuming dashboards, and many of them are filtering them and building their own custom ones, and sharing reports and all that.

So that's been a little bit more difficult to govern. I think it's still not just at HubSpot, but in general, it still feels like a frontier that is a bit harder. But there are data analysts like specialists embedded in all the different teams at our company, at least, that are the primary source for ensuring that the given departments, dashboards and reports are valid. So they'll do the same thing. If something is wrong, don't look at their own dashboards first, their own analysis, their code and say, “Hey, is this right?” Something what's incorrect downstream from that, or upstream. They'll go back to the data analysts or the analytics engineer and look at their data models. And then back through data engineering, and it's a source system.

We've tried to carve out those roles and responsibilities, and also make sure that we have ways for people to understand who that next person in the chain is one way or the other. That's something we use Monte Carlo for. There's some other tools we've used in the past for data lineage to say, like, “Who do I talk to when this isn't right? And I've already done my investigation in my part of it.” When you have a large company, or just a lot of data sources, that in and of itself can be a challenge just to know who to talk to, but at least by defining that ownership, it's good organizationally. But it also helps us target those alerts, build in different rules around code changes. So different rules in our PR process in GitHub, ensuring that certain tests are run or certain people review code before it goes to production, depending on who owns it and where it is. I found that to be very important, and rather than just having sort of collective ownership of everything, really, it's important for us, at least to define that very clearly for everyone.

[00:37:10] JM: Can you take me inside a specific engineering problem at HubSpot, specific data engineering problem, that maybe will shed some light on how you solve data problems that can exist across the organization? Maybe you can just give an example of a real world data problem.

[00:37:30] JD: We have plenty to pick from. I think that one of our typical challenges is the dependencies of all these different systems, and keeping everything in sync. And so our data engineers, we actually have an entire team dedicated to what we call data orchestration. And when you think about – so, first, those hundreds of different data sources we're ingesting from, downstream from those, people are building thousands of different data models, and then thousands of different either dashboards or reports, or they're running Python scripts, or doing ad hoc analysis, and keeping everything timed and in order. It's very important, but it's also very hard. Back in sort of the beginning of the business intelligence days, everything ran in batch often once a day. We still have some things like

that. But I clearly remember many jobs where it was around 6, 7 AM, you're sort of waiting for the daily update and everything in the data warehouse.

Again, there are some things that we still do that way that, that's how we started at HubSpot. And some things only make sense to update daily. You don't need one of your financial reports to be updating every hour, or every minute or something like that. But there are others, especially operational needs, that you do need more frequent updates. What is challenging is that everything's now, moving on from this daily batch to everything having their own sort of cadence and needs, but yet, having all these dependencies. A single data model in the warehouse might have dependencies on six different source system data ingestions. Some might be more streaming ingestion, or micro batch. It might be all the collect Kafka data and ingest it every minute or two. Others are big batches from legacy systems, like a Hadoop cluster, or an API or something like that. That might take an hour, might take two hours. And you don't want those data models to be updating every 10 minutes when you have something upstream taking two hours.

So big problem for us is, how do we tie all that together? How do we ensure that we run things when they should be running also in the correct order, are also not running something downstream if an upstream job has either failed or failed a test? I mean, we don't want to refresh with improper data. So the orchestration component is a big – that’s a big engineering problem for us. It's not even really a data problem as much as it is. We have all these different jobs and processes that run on a lot of different infrastructure, some in-house, some third party, and they all have different ways of alerting and throwing errors and connecting with each other. Some are tightly coupled, some are uncoupled. That's for us, a great example of a pure engineering challenge in our data engineering space, and one that we have to invest heavily, just given the complexity of our infrastructure.

[00:40:25] JM: Well, it's been great talking to you both. Any closing thoughts on the modern data landscape and predictions for the near future?

[00:40:37] JD: I'm more optimistic than I have been, ever. I think just in terms of the maturity of the space, and I think it's been pointed out a few times here that there's – the software engineering space, certainly has matured over the years, and I think the data teams and data organizations are learning a lot of those lessons. And because of that, able to like mature even a little bit faster at a faster pace. So I'm really excited about just the adoption of more software engineering practices in data engineering, and even analytics engineering, and then the tools to support that. So I couldn't be more excited, but

especially on that, the sort of lower layers and more engineering data engineering, aspects of it just feels like now's the time, where we finally hit the maturity we need to really grow beyond being more of like a back office type organization, and really being, at least in my case, even part of like, the product engineering organization. So really feels exciting. I think the next couple of years are going to be just more of the same.

[00:41:40] LG: Yeah, same here. I'm super excited about the fact that data platforms are now creating products and are getting more and more integrated into the kind of core offerings and core software solutions that companies provide, and are driving real customer value in a very direct way. That's super exciting, and that's supported by exactly what James said, which is the ability to adopt software engineering practices in the data stack. And to really productize the thing and get to the confidence level where software products feel comfortable serving it directly to customers. That's very exciting, and that opens up a lot of possibilities both in terms of offering analytic capabilities and kind of machine learning based solutions that should really accelerate, or abilities as an industry to do that, and I'm very excited about that.

[00:42:32] JM: Cool. Well, guys, thank you for coming into the show. It’s been a real pleasure.

[00:42:35] JD: Yeah, thanks for having me.

[END]