Data Pipeline Maintenance

Day 1 Lecture
Maintain Data Pipelines like Airbnb and Netflix

Transcript:

0:00
data pipeline maintenance is an inevitable part of data Engineering in this 2-hour course we're going to be covering all of the things like how to
0:06
do a good data migration how to set up runbooks for your on call rotation what are different ownership models that you
0:13
can do so that your data Engineers do not become burnt out and we'll also cover common ownership problems and
0:20
maintenance problems that come up that I've seen throughout my career at Facebook Netflix and Airbnb I hope you
0:25
enjoy the course today and if you want to do more of this stuff in the cloud you should check out data expert Academy
0:31
in the description below where I can get you 20% [Music]
0:38
off if you're a high performing data engineer this uh is something that is a
0:44
fear that you will have as you grow deeper and deeper into your career is um
0:52
every time you build a pipeline it's not like you just build it and it's done right if you build it and
0:58
then you have to maintain it there's like this continual cost so like in some
1:03
ways like there's this I I I've always had this sphere where it's like uh doing data engineering is inherently
1:10
unsustainable because every pipeline you write has uh an added cost so that means
1:17
that like as you build more pipelines eventually you're going to have too many pipelines that you can't maintain them
1:23
all like that's just like the nature of the Beast that like literally there's like it's an inherently unsustainable
1:29

pattern because of that and so uh I I that's why I have this uh the way I like
1:34
this right every pipeline you build slowly crushes you with maintenance costs because if you think about it like
1:41
okay if you have one Pipeline and then uh say every pipeline you have has a 10%
1:47
chance of failing on a given day so if you have one pipeline that means that on
1:52
call is going to have to do something three days a month uh if you have 10
1:58
pipelines and there's a 10% chance of one of them failing that means the on Call's going to have to do something every single day so uh that's the thing
2:06
to think about right as like you have um more and more pipelines and then it's like if you have 100 pipelines okay now
2:13
now on call is projected to do 10 things every day they have to unblock 10 pipelines all right okay what are we
2:18
talking about today so there's going to be four pieces that we're going to talk about today and we're actually going to uh be breaking out into breakout groups
2:25
a little bit today as well so um we're going to be talking about uh the difficult parts of the data engineering
2:30
job we're going to be talking about the ownership models of data sets uh we're going to talk about team models for data
2:35
engineering and then uh also common problems who run into maintaining pipelines those are going to be the four
2:41
kind of big areas that we're going to be covering today and uh hopefully this can shed some light on some stuff and uh in
2:48
that uh ownership models of data sets that's where we're going to be covering a little bit more around what on call
2:53
looks like and all that kind of stuff especially in regards to what it looks like in big Tech so let's uh let's dive
3:00
a little bit into this like first problem here the difficult parts of the data engineering job so um I think that
3:07

a lot of times uh uh people look at data engineering data science and they're
3:12
like wow those guys they write like five SQL queries and then they're just like done they're done for the day and then
3:17
they just go sip sip on martinis on the beach and then they're they're vibing and uh that's actually not the case like
3:24
uh it's very far from that actually uh if you Google uh data engineer burnout
3:30
uh the numbers there say that like something like 90 90 plus% of data Engineers are experiencing burnout right
3:38
now so that's like welcome right welcome to the burnout club uh you I hope you
3:46
enjoy it it's worth it in some ways so uh that's a big thing to remember as
3:51
you're like going through this kind of data engineering journey is that you need to protect your peace you need to protect your well-being and that is
3:58
going to be something that will be very good for you as you kind of go through your career because if you don't protect
4:05
your piece like data engineering is going to eat you alive it's going to eat you alive it's not like it's it's not a
4:10
forgiving feel in that way so uh that's one of the things that I think is why uh
4:16
not as many people are doing it because you kind of have this combination of things of like why I think data
4:22
engineers get paid really well uh even though like to ramp up as a data engineer does not take that long maybe
4:28
takes like five or six months but my perspective on why data engineers get paid significantly well is one side of
4:35
it is uh you don't get that much recognition for your work because you're kind of in the backgrounds and the and
4:42
like the data scientist or the data analyst is going to be the one who presents the work and like you don't get the recognition and a lot of people want
4:48

that kind of recognition when they do their job that's one piece of it and then the data analyst and data scientists they don't have to do on call
4:54
so they also don't have to wake up at 3: in the morning to fix the pipeline and so uh they also get so they get all the
5:01
credit for presenting the visualization and they don't have to do any of the work so that's where I think that
5:06
there's like a lot like data Engineers really are kind of like the unsung heroes of the business analytics
5:13
environment and uh and I think that that's something that is really great and uh can like but it's
5:22
also reasons why like being that unsung hero who wakes up at 3: in the morning who doesn't get recognition like it can
5:28
be it can it suck it can honestly suck really bad too right and I've been there as well I've been there as well in my
5:35
job and when I've been like dude I freaking hate my job right now I don't want to work I don't want to do this anymore right and I mean that's why I
5:41
you know like I've jumped around a lot and I feel like that's been a part of why I have been job I've job hopped
5:47
quite a bit it's not just for like I'm chasing money or chasing new titles or getting promotions or whatever but like
5:53
also just from the angle of like at least at the first like you know the first couple months of your job they
5:58
don't expect you to be on call right like usually you join the on call rotation like like 6 months in so at
6:06
least the first 6 months like you're uh like not burnt out because you don't have to worry about the maintenance and
6:12
that's kind of nice and that's something that I I think that's one of the other benefits of like job hopping in data
6:17
engineering is that like you can avoid the maintenance like that's always really nice but let's go let's
6:24
dig a little bit deeper into like why data engineering sucks okay so we're going to be doing uh
6:30

some breakout rooms here hopefully we're going to have some really nice discussions here because I know there's a lot of people in here who have data
6:35
engineering jobs and if you don't have data engineering jobs some of these like you might uh like I want you to think
6:41
about these more maybe in a hypothetical situation of like what would potentially be the hardest for you um so what we're
6:47
going to talk about today is or what we want people to discuss in kind of a break in breakout sessions is um we want
6:55
you to introduce yourself so that you can meet some new friends obviously and then uh we want to talk about like what's
7:01
your favorite part of data engineering and then uh what's the hardest part of data engineering and then what are some
7:07
strategies that you've uh used to implement uh to make data engineering less difficult and minimize burnout and
7:12
make sure to not forget uh to talk about what your favorite data type is and like
7:17
you know there's a lot of choices there you know you got like Boolean int big int struct array like all those
7:23
different uh but you but you can't just say array as your favorite data type you have to you have to at least say like
7:28
why like like at least a little bit a little bit more on like what's going on there so um that's kind of like where
7:34
we're going toh start today and I because I really think this is going to give uh the the more Junior people in
7:40
this call a lot more perspective on what they're going to be going into and I think the more senior people they could
7:46
learn some more strategies potentially on uh on like how to manage this stuff and minimize burnout my summary of the
7:53
three that I think of when I'm like I don't like data engineering is you have like high expectations and that kind of
7:58

lead to Bur turnout you have data quality issues and then you have unclear priorities and ad hoc requests those are
8:05
going to be the three kind of buckets of things and we're we're we're going to go more into detail on each one of these
8:11
different topics this is kind of the idea uh like these are the three kind of areas right we have high expectations
8:17
data quality issues and unclear priorities are going to be the three kind of areas that uh make data
8:24
engineering hard and uh and I think are going to be some of some some of the the
8:30
most uh difficult things and why I think a lot of people leave data engineering
8:35
is because these problems like even at the best companies with the best people like they still do not manage this
8:44
correctly okay uh one of the things that I think is very interesting about data engineering is
8:51
uh building pipelines and building out uh infrastructure uh for companies it
8:58
should be viewed as a marathon on it's similar to like you imagine like when uh you know back in the 1950s and 60s when
9:05
the US was building out the highway system and uh they instead of like uh
9:11
there was like a big pressure to build it as quickly as possible and then they're like just slap the highways together just do it just just just
9:16
freaking it doesn't matter the quality just freaking slap it together and we're just way because we want these highways built as quickly as possible and if you
9:23
imagine like if that was the priority of things like uh you're going to um we'd have like really terrible roads and uh
9:32
that would be kind of a gnarly sort of problem and I don't recommend uh doing
9:37
that that like I don't recommend slapping together your pipelines I think that every time that you do that you're
9:44
going to eat pain every time like every time that I've cut corners on a pipeline
9:51
to get a to get a data set or to get an answer to someone faster I've always

9:56
regretted it it's like never been something I've been like wow I'm so happy that I I I got that to them two
10:02
days early to get them to get everything ready to go but because then it's like always they'll come back to me later and
10:09
they're like a quality issue or like uh there or it's missing a column or like
10:15
there's like you almost always like if you rush a data pipeline you're going to eat um pain in the quality bucket in the
10:23
completeness bucket or in the maintainability bucket where like maybe
10:28
you maybe you do ride it complete and uh and it has the quality but like then you it's not optimized and it's like not the
10:35
fastest it could be or like it fails I've generally found that like you're going to eat pain in one or more of
10:42
those buckets if you rush your pipelines but on the flip side analytics um is a
10:49
Sprint if you are a data analyst and you're trying to answer questions about the business the faster you can answer
10:55
the questions about the business the better there is a a side of that that is
11:00
important for analytics to answer things rapidly so there is this kind of like push and pull between uh data
11:08
engineering and data analytics there's like this push and pull and uh one of
11:13
the things that I've noticed in my career when I've done this stuff is and I was a lot more naive at one point in
11:19
my career I remember 20 2017 is the year that really uh stands out to me where I did not obey this rule
11:27
right here where uh I'm just going to go over a little bit about like a quick anecdote so um I used to work in
11:33
notifications at Facebook right and then um in the end of 2017 uh Facebook saw
11:39
its first drop in growth um ever and uh people were freaking out they were like

11:44
what's going on like how do we fix this problem and analytics team starts freaking out and then they're like and they're like Zach you're you're a good
11:50
data engineer you're a rockstar data engineer like we're going to move you you're not going to work on notifications anymore you're just going to work on this hard analytic problem of
11:58
like trying to diagnose why growth is not going back in the US and I was like okay and um and the thing that happened
12:06
for me that I realized was uh that I treated that problem like a Sprint and
12:14
then that Sprint lasted like seven months and I was like essentially uh you
12:21
know that's why I have this rule here that says say no to sprinting the marathon um we're not like that one that
12:28
one guy who can run you know a sub 2hour Marathon like there's like there's a couple genetic freaks out there who can
12:34
you know Sprint the marathon but they're they're few and far between most of us need to have a little bit more balance
12:39
in our lives and uh that's one of the things that end up happening for me when I was working at Facebook was like I uh sprinted that Marathon got really burnt
12:46
out I ended up hating my job and then I quit I left because I was like I don't want to do this anymore and um and I got
12:53
like just so much emotional turmoil in my soul from doing that and uh so a big
13:00
thing here that I've noticed in my own life is uh and this is something that I
13:07
changed my perspective on when I worked at Airbnb and I feel like I actually had more impact at Airbnb in some regards
13:13
when I made this change where one of the things was was like when I was working at Facebook and trying to diagnose this
13:20
growth problem I literally felt like it was life and death like it felt very uh
13:26
fight ORF flight very like intense I had a lot of anxiety about like I must solve this problem like otherwise like my
13:32

company's going to die and like I felt like a very like kind of personal responsibility for the problem even
13:38
though obviously Facebook has tens of thousands of employees and I'm just one data engineer and it's like looking back on it I'm like wow I was like really
13:45
naive and really like young and stupid but uh one of the things that I it's one
13:50
of the quotes that I say now nowadays when I whenever I feel this pressure to Sprint uh one of the things that I
13:57
always say at least to myself and sometimes to my analytics Partners like but usually just to myself is that we're
14:03
not saving lives here we're not saving lives like we're not uh like this is not a life or death situation if I get you
14:10
the data two days late it's probably going to be okay like it might even be exactly the same like uh whether like
14:17
because if I got you the data two days early it might take you two more days to set up the presentation with somebody
14:22
anyway so like cuz businesses they don't move very quickly anyway so like a lot
14:28
of times if things are a week or two weeks delayed it's not going to be that big of a deal and uh and that like you
14:34
want to be careful about like how much pressure you're putting on yourself to deliver this uh these values because I
14:43
mean I don't know like I I I look back on a lot a lot of my career especially like a lot of the time near the end of
14:49
my time at Facebook and through most of my time at Netflix where like I did not learn this lesson and I just felt like I
14:56
could Sprint the marathon and that's like and that's exact what I did was I just sprinted the marathon the whole time and gave up so much and like what I
15:04
gave up was I don't know my social life I gave up uh like kind of my peace and I just was like running on anxiety and a
15:11
dream and uh like I don't know I look back on those times as like I don't I

15:16
don't I don't necessarily view my those times as necessarily I was happy that like I could have uh
15:23
definitely probably had the same impact as I did but was happier
15:29
because I was able to view the work through a lens that felt less critical
15:36
that felt like it was it was at the it it was at the accurate amount of uh
15:42
urgency not like emercy room urgency like we have to save someone from dying
15:48
but also not on the other end of like it doesn't matter at all because like if you put it all the way over there then
15:55
you're going to get fired because like you don't care enough about the job so you got to have like obviously there's
16:00
like there's something in the middle there that is the actually the healthiest amount that you should like
16:05
care or the the healthiest amount of urgency that you should uh use in your
16:11
own job especially this problem with urgency is something that's a lot more common among people with ADHD where they
16:17
you know a lot of people with ADHD are considered they're considered time blind where it's like people with ADHD they
16:23
can think in two times it's either now or not now and there's no like in between and so uh that's a like one of
16:31
the things that I've been kind trying to shift more in my own life and that was uh and that's why I feel like I was able
16:36
to have a lot more balance and a lot more uh be able to do a lot more other things when I worked at Airbnb that's
16:43
why I was able to like post on LinkedIn and become a content creator and build all this other cool stuff because uh I
16:51
had a better relationship with my work and I was able to think of my work as a marathon not as a sprinted Marathon so I
17:00

hope you learned something here a key thing here is we're not saving lives right we're saving companies money but
17:06
we're not saving lives okay so we're going to talk about ad hoc requests a little bit so a big
17:12
thing uh that is this is another common uh uh case of data engineer burnout
17:18
happens because of ad hoc requests uh I'm going to talk about like kind of a strategy that I've used to essentially
17:24
eliminate this problem or make it a lot more manageable uh so analytics need to
17:29
solve uh like they need to solve Ur solve urgent problems like but to them everything is urgent a lot of the time
17:36
and uh but one of the things that I like to say when I'm talking with my analytical Partners is like if
17:41
everything is urgent nothing is urgent because then everything is the same and
17:47
then like urgent means that you have like you you have some sort of Stack rank right and there's a priority que
17:53
right you can sort things that way uh generally speaking uh I like to think of it as every quarter you should spend
18:00
about 5 to 10% of your time on uh ad hoc requests which means that half a day
18:08
keeping in mind that's half a day a week half a day a week or across a quarter
18:14
it's like a week maybe two weeks of a quarter is how much time you should be spending on ad hoc requests the other 12
18:22
12 13 weeks or whatever should be spent on uh um more long-term in
18:28
infrastructure building projects so uh a lot of times like this is where when you're doing ad hoc requests like you
18:35
should have you should operate from an angle of uh uh like Roi where um if the
18:44
ad hoc request literally is like quick like you know there's that uh that Meme
18:49
that Meme ad hoc request which is like can you just pull this data and if it literally is that easy which like I
18:54

don't know one in 10 times it is then just do it and like do the 10 15 minute
19:00
thing and unblock them but a lot of the times it's not a lot of the times it's actually complex and it's actually a
19:05
project that they need and most of the time uh what you want to do at least like the thing that I've seen work the
19:12
best is that is something that needs to get put on the road map so that means that like when uh if they're asking for
19:18
something that's really complex then uh they should be asking it in January in
19:24
March in uh June or in August I I guess would be the times August you know
19:30
August September is those are going to be the times that are going to be the best and most effective times to ask so
19:36
that like they can get into the quarterly plan so that you can as a team
19:42
you can figure out the right ways to uh uh kind of process this work the right
19:47
ownership models the right data models right so that it doesn't feel like you have to just slap something together
19:52
real quick but you can actually think about it holistically and uh I've noticed that like if you have a strong
19:57
engine engineering leader as your leader that this is something that works really well but if you have a kind of a weak
20:04
engineering leader what ends up happening is analytics overwhelms the data engineers and uh I've seen both
20:10
happen I've seen both happen and like I I've definitely been in both situations so like a part of this really does
20:15
depend on who your manager is on like whether or not this pattern can work but generally speaking that's the way I
20:21
would think about it is like if it's a lwh hanging fruit ad hoc request just do it and that's some of them it's a it's
20:27

it's less than half though if it's a more complex one push back and um ask them to essentially wait till the end of
20:34
the quarter and uh and for them they might be like wow that's a long time and then uh like that is where it's great
20:41
though because that it's going to make them angry it's going to make them frustrated because you're like yeah I'll do it in two months right and then
20:47
they're gonna be like wow like you don't have like you really can't squeeze it in in the next like two months right and so
20:53
uh but the big thing there is that like you need to cause your analytical partner's pain cuz the only way that you
20:58
can change behavior is with pain only way cuz if you just do what they say every time and like you just burn
21:05
yourself out because like you just keep saying yes to every single request they're not going to change how they
21:11
interact with you they're going to keep doing the same thing and that's where like if you cause them a little bit of
21:16
pain cause something to be delayed right and then they have to like explain why it's delayed then they're going to like
21:22
try to figure out a better working model with you and that's where I love I love quarterly planning where then it's like
21:27
you work with them and then like you come up with a nice master plan and then you're going to execute that over the next 3 months and then it feels very
21:34
planned out and methodical and then you also give then you also have time to like buffer for things like on call and
21:42
things like data quality issues and other problems that can come up as opposed to having to buffer for more
21:49
work that's like I think a very important thing to think about like when you're doing these ad hoc requests it's
21:55
a probably one of the most important parts of managing your workload a data
22:00

engineer okay uh just uh kind of some more pieces of wisdom on this uh slide
22:05
here um most analytical questions are less urgent than they appear comes back to we're not saving lives um there are
22:13
some small exceptions here that I've actually noticed uh I just want to uh
22:18
give another anecdote here really quick uh so I like when I worked at Netflix uh I worked on this project called EDR
22:24
which is external data request which is similar to like when you know know if you go to like Facebook or LinkedIn or
22:31
like whatever website and you're like give me my data and you do like a gdpr request and then they have to give you all the data that uh we have on you uh
22:38
it's like that it's a lot like that um uh that's what Netflix has that they have EDR and but like the thing is is
22:44
like sometimes you get these EDR requests from like law enforcement because they're like trying to like solve a kidnapping or something like
22:50
that and then I don't know like I I guess that that's like an edge an edge case where it's like okay maybe this ad
22:56
hoc request like they're solving a kidnapping Maybe is a little bit more urgent than like I thought right that's like maybe that is like the one case
23:03
where it's like Yeah we actually are saving lives right now right uh that like at least in my in my experience
23:08
like like that wisdom of we're not saving lives that's the one time that I've been like okay maybe we actually
23:14
are saving lives but like generally speaking that's not the case and things are less urgent than they appear you
23:20
don't need to have that data ready as quickly as possible and especially if it's coming at the expense of like your
23:27
longer term impact work definitely consider that um and
23:33
this is what I was saying before like make sure to bring all your analytical Partners together at the beginning of the quarter and plan stuff out and don't

23:41
just let it be like week to week like what you're going to work on because that is not sustainable because like
23:47
they're going to keep putting more and more work on your plate every week and like by the end of the quarter you're going to have two quarters of work so uh
23:55
that's like the way to do it cuz then then it also causes them to think think about things more strategically as
24:00
opposed to tactically cuz if they think about it like uh if if they just give you work as it comes up then they're
24:07
just going to just keep being like oh here's new work here's more data I need here's more data I need right and then
24:12
like they're not even thinking about the order that you should do these things in they're just saying I need this I need
24:17
that I need this I need that I need this I need that and like it's not like I need this the most then this the second
24:24
most and this the third most they're not doing that prioritization effort that
24:29
they should be because data engineering isn't free and like I think that that's like a
24:36
thing that a lot of like analytical Partners get wrong is that they think that like you can just slap a pipeline
24:41
together real quick and it's essentially free but it's not that's why dat engineers make a lot of money but anyways that's kind of the idea behind
24:47
ad hoc requests uh like they are tricky and uh you definitely want to work with
24:52
your manager and Leadership to like essentially come up with uh systems where the teams can interact in a way
24:58
that's more appropriate this is tricky to do when you are like a mid-level uh
25:03
mid-level or Junior I but as you become senior or staff then you kind of wield more influence at that point okay
25:10
ownership models so uh I think one of the big things that can happen uh that
25:16

can also cause a lot of burnout for data Engineers is they can own too much stuff
25:22
I know this happened for me at Netflix cuz at n i mean not Netflix at Airbnb at
25:27
Airbnb this happened for for me where I was owning like end to end and like I freaking was like dude this is too much
25:32
why why do I own all this stuff it's too much so like um there's five things
25:39
right you have data sets data pipelines data documentation metrics experimentation those are like the five
25:45
things that uh generally speaking are have ownership we're going to go over
25:50
essentially the two most common ownership models and how like where there are problems and where things can
25:56
kind of have issues so okay so this is going to be your most common ownership model uh where you'll
26:04
see on the left you have logging in data exports that's going to be owned by a software engineer and then the data
26:10
engineer is going to own the data Pipeline and the master data and then the analyst is going to own the metrics and the dashboards and the
26:16
experimentation that's like I'd say this is the most common uh ownership model uh
26:22
the thing about it is about this model is that there is lines on this where uh
26:28
like for example data Engineers my own aggregate Master data and aggregate
26:34
Master data and metrics are like the same thing and that's where
26:40
these lines are not as clear sometimes and then same on the loging side right
26:45
where it's like a data engineer might work with a software engineer on a logging specification and that can be
26:51
another big another big way that like they can kind of impact and that line can also be kind of blurry so these like
26:58
especially around the edges of ownership like metrics a lot of the time can be owned by both logging can be owned by

27:05
both and like there can be a mess messy stuff on kind of both sides of that I've definitely noticed those kind of
27:11
patterns in my career uh like one of the things that I happened for me when I was working at Airbnb was that that metrics
27:19
layer uh there wasn't enough like analytics people on my team so I just owned the metrics layer as well and that
27:27
was like a lot of work uh and because then you're owning like the big chunk of the chain you're owning like 60% of the
27:34
chain at that point instead of 40% of the chain and uh but here's the idea
27:39
right so you won't like I personally think that this is the most ideal setup
27:44
and then what you do is uh the one thing that you want to ask here is like okay
27:49
but what if someone has like a business question uh usually business questions are going to be related to metrics
27:55
metrics are going to be mostly where the business questions go and they should go to the analyst or the data scientist uh
28:02
like there are some edges there where like if like someone's trying to ask a business question that is like not
28:08
supplied by a metric but like it needs like you don't have the right aggregation for what this person's question is then maybe they need to go
28:15
up a layer and they need to go to the master data and then they might ask you a question about the master data on like how to query the master data and that
28:21
can that's a very common pattern that I've seen as well and those are like for people who are asking like a very very I remember there's a question about like
28:27
the Olympics like like the 2024 Olympics that I had to answer for people about like availability around the 2024
28:33
Olympics that I was like like like of course we don't have metrics for that dude that's like in a long
28:38
time but like anyways uh this is kind of the idea this is the most common ownership model I'm going to go over to

28:45
like the next slide where we're going to talk about another common ownership model um so this is another common
28:53
ownership model where uh where instead uh like just like what I was talking about in the last slide where uh I at
29:01
Airbnb I was owning pipeline Master data and metrics and then you have the data scientist and um analyst who work on
29:07
experimentations and dashboarding and you have software Engineers who work on logging this pattern is pretty common at
29:13
Netflix one of the reasons the Netflix has two names for analytics Engineers they call them analytics Engineers but
29:19
they also call them spanners and obviously it's pretty obvious why they call them spanners because it's like
29:25
they go they go all the way from logs to all the way dashboards sometimes they they even have analytics Engineers own
29:31
the dashboard too and it's like wow like that's a lot there's like like they just own like the front end the back end
29:38
everything and uh I generally find this pattern to be also can be kind of good
29:43
uh I want to go over why I think this pattern is good uh compared to the last one uh in some ways like so if you have
29:50
the data engineer and they own the metrics as well then uh it's a lot easier for them to uh kind of update the
29:57
Met metrics because they just they already know the master data very well so updating the metrics is pretty easy
30:03
for them but uh the trade-off being that like a lot of times the data engineer doesn't have enough time to actually
30:09
develop enough of the business context that they need to get good metrics
30:14
because a lot of times that requires a lot of conversations with product managers and customers and ux people and
30:21

designers and just like tons of other people right and so so this pattern not is not necessarily the best even though
30:27
from a technical perspect perspective you might be able to make an update metrics faster so um that's kind of the
30:33
idea behind uh the two kind of common patterns but keeping in mind that there can be tension especially around the
30:39
edges of these boxes okay where do problems ownership
30:46
problems arise right I was talking about this the there's at the boundaries so you can have data Engineers that own
30:51
logging and uh that can be a problem because then you have data Engineers that are going too far upstream and
30:57
they're like going into the server and like then they're like owning I remember that was a big thing I worked like at
31:03
Airbnb I was like I I went a little bit too deep into the server and I'm like why am I doing this like why why is this
31:08
on me when no one else wants to work on this problem and like that was also something that like I found to be kind
31:14
of one of my own problems where it's like I wasn't like respecting the boxes as well as I should have but then on the
31:20
other on the flip side like you can have like maybe the data engineer is the guy who's not doing his job and then the
31:25
data scientist is the guy who's like okay I'm I'm going to write pipelines now because like I need my data and I'm
31:31
going to do I'm going to own that square even though like I'm going to probably write a pipeline that's not that great
31:36
because I don't have all the skills that I need to write a good Pipeline and then you also have what I was talking about before we have data Engineers that own
31:42
metrics and then they get overwhelmed because they own the pipeline and the master data and the metrics and all the
31:48

business questions and like the maintenance for that one person is overwhelming I know for me that was how
31:54
it worked for unit economics at Airbnb where I owned the pipeline the master data the metrics the business questions
32:01
I owned all of that and like and that was data that was consumed by like 20 teams like or there was 20 Downstream
32:08
teams and like like 30ish upstream and so it was just like a mess such a mess
32:16
and so um and that's where uh you can have these like Islands you can kind of become an island that's why like I don't
32:23
like this analytics engineer pattern as much because it essentially says like okay
32:28
make it make tribal knowledge have one engineer own everything that's what that essentially this pattern does and like
32:35
in some ways I find it to be more unsustainable than uh the where the data engineer just owns the pipeline and the
32:41
master data because then they don't have to have all the business context and like and kind of uh and field all the
32:46
business questions that's the part that is better right about um the the the pattern where the the an the analyst and
32:53
scientists own more so uh these things can happen you can have a lot of issues here um if you
33:00
uh don't obey uh the boxes right because like then you're going to either overextend yourself or someone else is
33:06
going to trample on your work and so definitely be aware of these things and try to try to like I've found like for
33:14
my own self that like I like like or like I found myself that I'm just like
33:19
whatever it doesn't matter I'm just going to do it myself and I'm going to do things that way and as I've gotten
33:25
older I've recognized that like you really don't want to solve the problem that way because that again that just
33:31

goes back to uh you're going to create a team structure that is inherently unsustainable for yourself and that you
33:38
are essentially sewing the seeds of your own burnout by uh going too deep into
33:43
someone else's box because you're essentially owning their work and doing their job too and that is not what you
33:50
want to be doing like uh longterm as a data engineer because that's going to just burn you out real quick and so uh
33:57
keeping in mind that as like you have like different ownership problems that can arise as you are working your jobs
34:04
so what happens right what happens when the boundaries are crossed right uh there's a lot of things that going to
34:09
happen uh burnout people quit people leave people get angry people get
34:15
frustrated uh that's a big thing that can happen uh you can also have a bad team Dynamic where uh people blame the
34:21
data engineer or people blame the data scientist or people blame the software engineer or they like they or they or
34:27
they or they think that you're taking credit for their work or doing their job or like not giving them enough time to
34:33
shine and kind of trampling over their uh potential for promotion and like so
34:39
sometimes you might think that you're like you know relieving someone of their work but then they hate you for it
34:45
because like they're like that was my work and I was supposed to do that and that was going to be my promotion but now like you're trying to steal my
34:50
promotion from me right and so like you can get a bad team Dynamic from that so you don't want to do that either you
34:55
want to have good boxes right and that's like otherwise you get the blame game that happens and then this last one I
35:01
think is actually probably the most important and it like I noticed this especially with like uh like when I

35:06
worked at Airbnb was when I was owning all that stuff end to end and it was just me then it was like one of the
35:13
things that would happen it was like I would go on vacation and then everything would just
35:18
melt it was like if I took a week off then like the profitability stuff was
35:24
just done it was just like and then I would come back and there would be like 12 slack messages and I'm like like why
35:31
why haven't we freaking solved this problem yet like why won't someone else pick up this context right and so that's
35:37
another thing that you can H do right and this is an important part of a lot of like pipeline ownership stuff is that
35:44
you there should be a secondary person on most of the stuff that you're working on there should be a backup person so
35:49
that like if you do leave or go on vacation that like the business can still like get their questions answered
35:55
maybe not to the same level of completeness because like there's no way the secondary person will have the same
36:00
level of context as you do as the primary person but at least they can maybe you know answer simple to you know
36:08
intermediate questions about the data sets so big thing to remember here is like
36:15
solve these problems uh organizationally not technically so if you are getting
36:23
blocked like talk to your manager about it talk to uh the analytical the
36:28
analytics Partners manager about it right and like try to figure out a way to solve this problem where the person
36:35
that's supposed to do their job does their job and you do your job and you're going to be a lot happier like and it's
36:42
better to do that than to be the hero it's way better to do that than to be the hero i' I've noticed that throughout
36:47

my whole career that that's definitely a way that I've like burned myself many times so good thing to think about

36:53

crossing boundaries so like don't do it that's the main thing I'm trying to say here okay so we we're getting close here

37:01

to um uh I think we have like one or two slides left um and like I want to talk a little bit about this this is something

37:07

uh Stephanie brought up uh like I don't know like a week or two ago I even did a poll and I was surprised the poll link

37:13

didn't let me down link didn't let me down so bad on that poll so when I asked like uh what are better centralized or

37:19

embedded uh data teams and like uh it was like it was like it was like 6040

37:24

60% said embedded and 40% said said centralized and I'm all about the

37:29

centralized data teams I think that it's better that way uh because on call is easier right uh and you can knowledge

37:36

share and like you can grow each other it's great obviously the tradeoff is

37:41

that like those uh you're the teams that you support they have to come to you and

37:46

then you have to be able to prioritize it versus other teams asks as well so like sometimes those teams might not get

37:53

their data that they need because they get uh other teams priorities take higher than them and so then they often

38:01

times might want to go and do something on their own and unblock themselves stuff like that that definitely happens

38:06

so that's one of the things uh you know in the first slide I wanted to talk a little bit about with on call uh in big

38:13

Tech because I think that's something that I like we didn't cover quite as much quite yet so on call generally

38:19

speaking uh in big Tech uh if you are a mid-level engineer or higher you're

38:24

going to be put on an on call rotation us about at the six Monon Mark after you've started a job and you're going to
38:32
and usually depending on how big the team is like it might be once a once a month it might be once a quarter right
38:39
or like I mean for me at one point it was twice a month and and even at one point it was I was on call for for three
38:46
months straight that was insane so like on call can be freaking uh really messy
38:52
depending on uh what happens so what on call is is like pipelines break break
38:58
pipelines break and some pipelines have uh are very sensitive where they need to land like in a couple hours because
39:04
they're Upstream of a lot of other pipelines so you have like a lot of pipelines that depend on this pipeline so we need to unblock it and so that can
39:13
happen and a lot of times like generally speaking data engineering on calls are not as intense as software engineering
39:18
on calls I've been both I've been on both like I was a software engineer on call at Netflix and uh but data engineer
39:24
on call generally speaking you can do it throughout the day like and like most of
39:30
the on calls I've been on on data engineering is just like business hours like nine it's just like 9 to 99 9:00
39:36
a.m. to 9:00 p.m. business hours and they might expect you to look on the weekends but not at like 3:00 in the
39:42
morning but they might expect you to look at some point on the weekends like to see like if uh anything has failed uh
39:49
but it depends it honestly depends on the data engineering team that you're on and like the sensitivity of the data
39:54
that you have where like if you are working in some spaces like I don't know like say you're working on like the
40:00
personalization data team at Netflix where that data is absolutely critical for the company and it's very sensitive

40:06
like latency sensitive then in that case like yeah you're going to be it's going to be treated like a a software software
40:13
engineering on call where you will get a page so I don't know if y'all have ever heard of pager Duty it's an app that uh
40:19
can like essentially send you notifications and make your phone freak out if there's like a um a bug that you
40:25
need to troubleshoot and it can be it might be bug that you need to troubleshoot at 3: in the morning and um you know it's interesting that that's
40:31
actually how it was uh at Airbnb when I started they like they wanted me to like
40:36
uh troubleshoot stuff at 3:00 in the morning and um I eventually was like no
40:43
I'm like not doing that like and because like what I showed what I showed was I uh intentionally didn't do it for a
40:51
couple days and I just let the pipeline fail for a couple days and then it was essentially like a mini experiment where
40:56
I was like see the machine learning Downstream is so I looked at it it was like for every one day that it's delayed
41:03
the machine learning is half a percent as effective like so it's like it loses half a percent of Effectiveness and I'm
41:09
like dude that's like not that not that bad like not that bad like it obviously
41:14
would probably drop off as like it got later and later like if it was like five days it's probably not like half a percent half a percent half percent it
41:20
was like probably half a percent 5% 10% Like or something like that is probably going to be like a kind of more of an
41:25
exponential drop but I showed that right and I was like see because that's one of the things that you can do that can
41:31
really help the on call like especially if you have latency sensitive data then
41:36
uh one of the things that is important to talk about to maybe make on call easier on you and your data engineering

41:43
teams is okay what is the consequence if we don't troubleshoot at 3 in the
41:49
morning what is going to happen like what is going to be the problems that arise and like and then if you can see
41:55
what the consequences if if the dat it is a day late or 2 days late and just understand what the impact to the
42:01
business is and like for some things like it might be like a million dollars like for Netflix and the
42:08
engagement that they would lose if the data was that late it would probably be like a million dollars and then it's
42:13
like yeah okay well if it's a million dollars I'm going to wake up at three in the morning right but if it's like ,000
42:20
like like I don't know man like freaking that's not worth it to me man my mental health is worth more than that like
42:26
freaking like cuz it's like if it's $1,000 and I have to wake up at 3:00 a.m. like 10 times I'm going to quit and
42:33
then the turnover is worth more than that me leaving the company because of
42:38
bad mental health is going to cost more than $10,000 because they're going to have to replace me give that new person
42:43
more stock give that new person a sign on bonus and there's and probably pay a recruiter right and there's like all
42:50
like there's like this other cost that they potentially have so that's where it's important when you're working in on
42:56
call to understand like okay like okay if everyone drops the ball and everyone is just asleep and no one actually
43:01
responds to a an outage or a breakage like what is what is the actual consequence so that you can have
43:08
conversations with your Downstream Partners so that then they have a clear expectations of like what you're going
43:15
to do when things break so that's on call that's on call and like I hate it

43:21
but like it's also an important part of the job for reliability um the other model here is
43:27
embedded teams right and how embedded teams work is like you essentially have one de per team uh like I don't like
43:35
this model as much because of the fact that uh this data engineer is dedicated in one business area but like a lot of
43:42
times they're not going to have as much contact and information sharing with like other data engineers and uh how
43:48
does and how does on call work on call can be a mess in these situations where like you essentially have like um like a
43:55
a heterogeneous uh group of people on call where you have like software Engineers data scientists and data engineers and they're all on call like
44:02
for data pipelines and it can work it can generally work I've I I had that experience and I've like gotten that to
44:07
work as well but generally speaking it's better when like everyone on call is the same job title because then you have an
44:14
expectation of what the skill set is of of all these people but uh yeah there
44:19
these are two different ways that um you know companies can kind of set up their
44:24
data engineering teams I've mostly worked on Central teams but like earlier in my career I was definitely working in
44:30
embedded teams so um that's kind of the idea centralized versus embedded um
44:37
so uh I like it centralized more because it just feels like you're like on a team
44:42
it makes it feel like you're like building something cool with like a bunch of people that you like especially if it's a centralized team of people
44:48
that you like then it's freaking awesome so um oh yeah we still got a couple more
44:53
here so um so we're going to talk about four areas here that are probably the most
44:59
common uh things that happen during on call when things break you have out of-
45:05

memory exceptions uh you have missing data or a schema change uh you have back
45:11
fills and then you have business questions remember I was talking about how like business if you own the metrics
45:16
you kind of own the business questions as well so these are going to be the four that you're definitely going to run across when you're a data engineer let's
45:23
kind of go over each one individually here okay freaking skew so skew happens when you
45:31
you're doing a group buy or a join where like there's one key that has a lot more
45:36
data so you can imagine like if you were looking at like say notifications and you did like a very very very viral post
45:43
that say it says say you got like a post that did a got 100 million likes then
45:49
there's a chance that you got a 100 million notifications sent to you as well for every like that you got and then that uh whatever the executor gets
45:58
that data in spark is probably going to choke and is probably going to die and is is not going to handle all of that
46:04
extra data because it's all skewed um so there's essentially three ways to
46:10
fix this the best way is just upgrade to spark 3 if you're on spark obviously I'm
46:15
going to be talking about this in the from the perspective of spark but the best option is to upgrade to spark 3 and
46:22
uh enable adaptive execution spark. SQL adaptive do execution if you if you can enable that slam dunk done solved it
46:31
will never skew again so that's one of the things I like about spark 3 is that like this problem of skew and out of
46:37
memory for the most part is going away and uh is going to save a lot of data engineer sanity uh your second best
46:44
option is to bump up the executive memory add some more executive memory and like because it's easy it's like
46:50
something you can try in like half a second and then be like okay give it more memory and see if that fixes the problem and then like hopefully the next

46:56
day the data is skewed and like or isn't as skewed and it doesn't keep being more skewed because if it is then bumping the
47:02
memory is not going to help um then your third best option right is to include
47:08
what's called a a join salt uh I'll send you guys a link uh in and I'm going to
47:13
add it to the slide later but like I'll show you kind of a a link of like how to do this technique where you use random
47:19
numbers to make it so that like if you have a skewed row that you split it up first and then aggregate so that you
47:25
don't have to um uh like have all that data be shipped to one executor and it can kind of be split
47:32
up based on a random number and that can uh be another great way to solve your kind of skew problems so skew like I
47:39
always think skew is one of the worst words in data engineering um but that's kind of like I'd say those are going to
47:45
be your best ways to solve this problem uh and so definitely check that out um
47:50
if you ever are running into like weird memory problems but you can also get out of memory unrelated to you but that's a
47:58
lot less likely to happen those in those cases that's going to usually be a symptom that someone like wrote the job
48:05
wrong where they like were calling like do collect in the um driver and then
48:11
they try to collect all the data in the driver and then like and then the the driver only has 16 gigs right at the
48:18
most and so there's like some like weird gachas and Spark we're going to cover a lot of these gotas next week but like
48:25
generally speaking skew is going to be the main reason why you get an out of memory exception especially if the job
48:30
already did run before and this is like just a random out of memory failure that
48:36
you're seeing in production later on so yeah skew uh very annoying um okay

48:42
so yeah missing data and uh schema change um the biggest thing here is you
48:47
want to have pre-checks on this data especially like if it's data that you don't trust if it's like data from a
48:53
third party API or data that could change any time definitely have quality checks before your pipeline runs so that
49:00
that you can fail the quality checks and then your pipeline doesn't even ever have to start um this is where you want
49:07
to solve this mostly uh in a collaborative manner where you want to like find that Upstream owner uh and
49:14
then ask them to fix the problem if it's someone at your company if it's a thirdparty API then it gets kind of
49:21
Messier where it's like you might be able to talk with them most of the time you're not going to be able to talk with them and if you're not able to talk with
49:28
them you should change your pipeline code and hopefully you can you the new data that you get is good enough but
49:33
sometimes they might just turn off the API I know that's what happened for me in my startup when I was working in uh working on my startup in 2020 where I
49:40
was uh getting all these Call of Duty stats and then one day they just turned off the API and then I'm like oh I guess
49:47
my startup is dead because now I don't have API access and they're not going to give it to me again and so that was
49:53
depression uh but like uh there's a lot of different things that can happen with third party apis uh generally speaking
50:00
if it if if you know the owner get them to unblock it and then also try to find a long-term fix especially if this is
50:06
something that is like a commonly occurring problem a lot of times these are one-offs but like that's where you
50:12
want to be aware of like if it is a one-off or not and because on call changes hands every week whether or not

50:19
you can recognize it's a oneoff or not can be tricky because you are only on call once a month or like less even
50:25
maybe even less often than that and that's where run books and proper on call handoff can be very useful in
50:32
identifying uh kind of patterns in failures so that you can solve the root
50:39
cause of the failures and not just like unblock and keep putting Band-Aids on like a broken leg or keep putting
50:45
Band-Aids on like a situation that's not going to get better and so uh we're going to talk more about that uh on on
50:50
Thursday as well uh when we're talking more about run books but that's kind of the idea for schema change
50:57
uh those are going to be the main things that you're going to want to look at is and then once they fix the problem great
51:03
then uh you can move on with your life and unblock your pipeline and get it going or you might have to update your
51:09
code if the if the schem of change is something that you can't like undo so
51:14
that's going to be probably the next big thing that can happen all right backfills uh backfills
51:20
are freaking messy they're probably like uh just like what was said in the chat earlier backfields is one of the most
51:26
painful parts parts of data engineering so um generally speaking how backfills
51:31
work is you want to uh do a parallel backfill into a table that maybe you
51:37
have a table that's called table and then uh you want to name it table underscore backfill and you want to dump
51:44
all the data into there first and then uh validate it right and make sure that
51:51
things look good and if they look good then go ahead and do the swap where you essentially rename the production table
51:57

to production undor old and then you rename the table uncore backfill to the production table and then then you're
52:04
done like and that's for small migrations for things that are like uh where you're not changing column names
52:10
or you're not breaking things or you're not making a very big change to the metric definition uh that can be one way
52:16
to do it uh sometimes like if you are working with pipelines that have a lot of downstreams you can't do it that way
52:23
because you have to migrate hundreds and hundreds of people and and in that case you don't do this what you do instead is
52:29
you back fill into a table that's like table V2 and then you build a parallel
52:35
pipeline so then you have the production pipeline running and you have this parallel pipeline running and so they're both running and you have V2 and
52:41
production and then you encourage everyone to move over to V2 and then uh
52:48
after all of the references to the production table are U migrated to the
52:53
V2 table then you can drop the production table and then rename table
52:59
V2 to production and that's going to be the more painful way to go because then
53:04
like there's like a lot of PRS where people are like they update the table to like the V2 name and then you update it
53:10
back to the production name but you want to do this because if you don't then like your table names get really weird
53:15
because then the next time you want to do a migration then what you do V3 and V4 and V5 and V6 it's like you don't
53:21
really want to do that with data you just want like production right you don't want like the v on it like you
53:26
like you the the versioning should only be there like when you're in the middle of a migration not permanently and I
53:33

think that's a mistake that a lot of data teams end up doing is that like they don't actually do that final step where they drop production and rename
53:39
stuff because it's like kind of a pain in the ass but it's better to do that because then like people won't get
53:45
confused because they're like if you see V2 in a table name you're like is this the latest version though like well how
53:51
do I know there's not a V3 right and that's why like having tables that are just called like production tables or whatever then those tables are going to
53:57
like obviously be the right table to query but that V2 can give people some doubts about whether or not it's the
54:03
right one obviously I just talked a lot about this backfill process let's go a little bit deeper into this um this is
54:10
probably a better example especially for that uh simpler um version right so you
54:16
start with your backfill and you backfill into backfill table and then uh you have this production table as well
54:23
and then you have a decision here where you're like oh did the Back Field finish okay if the backfield finished and everything looks good then you want to
54:30
rename the production table to table old and then you want to rename the backfield table to the production table
54:36
so then this backfield table becomes the production table and the production table becomes table old right great then
54:43
what you want to do is uh message your Downstream pipeline owners and tell them that like there might be a a change in
54:49
the metric or a change in some of the definitions or some it might break things because columns might be added or
54:54
removed or renamed uh and that can happen and and that's usually fine if things break cuz then you're just like
55:00
that kind of motivates people to migrate quicker uh but they might get mad at you too because it's extra work you're putting on their plate and then uh the
55:07

last thing is is like once uh all the downstreams are migrated then go ahead and drop the old table and then just
55:13
work with this new production table in backfill so like this is like the most
55:18
common pattern I've seen with backfill that I would recommend y'all like kind of learn um obviously there is that
55:24
other one with that V2 but that's only like the only time you want to do this like V2 one is like when you have a
55:29
table that has a ridiculous number of downstreams like if it has like more than 30 that's probably when you want to
55:35
like look at like that other way of doing it where you have a parallel pipeline that runs in production with the other pipeline but uh generally
55:42
speaking you don't need to do that because you can just migrate the downstreams like while you're working in this process so um that's essentially
55:50
how backfills work business questions uh this is another common thing that can happen where uh
55:56
if you own data sets like people are going to ask questions about how to query the data sets how to like get um
56:02
information out of the data sets and like uh you might have a channel where people post questions and then a lot of
56:07
times you're going to want to set a clear expectation of like okay we'll get back to you in two hours or we'll get back to you into in a day or like
56:14
whatever because then uh the the it won't frustrate people if they just post something and then you can't immediately
56:19
get back to them the key thing about this SLA is to give you space so that you don't feel like you have to immediately respond when you're on call
56:26
so that you could still have a life and still like do other things and maybe even still be productive when you're on
56:32
call because there's a little bit of a gap between when someone asks a question and when you have to respond uh so
56:37

setting an SLA there is great um also like you're going to get the same question over and over and over again
56:43
and put it in a put it in a dock just and then like because then you don't have to keep answering the question you
56:49
can just point people Point people to the dock and then you can get more efficient that way and then uh another
56:55
important question here is to is to ask like is the business question on call
57:01
the same as the pipeline on call is it the same or is it different um and that
57:06
comes back to who is owning stuff A lot of times like you know at Airbnb when things are working the right way then uh
57:13
they're different because this business questions on call is actually like the data scientists and analysts they do
57:19
this on call and the data engineer will just do the pipeline on call but then you know for me in some cases I was
57:24
doing both uh uh but you want to make sure that you have this in here and like
57:30
try your best to just not do this essentially like I know in my perspective like I don't get a lot of
57:35
joy of doing this on call and I like the data scientist to do this on call but I know that's just my perspective for kind
57:41
of business questions awesome everyone um thank you uh for coming to this one uh so next time we're going to be
57:47
talking more about run books and how we can uh essentially cover a lot of these
57:54
problems that reoccur in a way that makes on call more
57:59
streamlined so uh I just want uh everyone's back I just want y'all to like Drop in the chat like what uh you
58:04
thought was like the best part of data engineering like what was the thing that y'all like really uh what gets you to
58:11
come in and actually you know show up every day like we got like problem solving anyone else have any other uh

58:17
any other thoughts on like different uh pieces of the mhm I like that I like that like I
58:24
like the long-term value bar I like that like the creating data assets which can be harnessed for Value I like the idea
58:31
of that but like with like one more uh like uh caveat which is like I like to
58:37
create data assets that are going to provide value for many years like it's like a I almost think of it as like I
58:42
like to create information highways right where it's like being able to like really uh provide a lot of like truth
58:50
like almost like truth highways I don't know that's that's like when I think about like the part of data engineering
58:55
that really uh Sparks a soul and Sparks like fire in my soul
59:02
right dude I love these like unlocking other teams enablement right I'm
59:07
energized by enabling other people in teams that's so good streamlining other processes helping people that's so good
59:15
I mean that's awesome seems like y'all really uh like the like the being able
59:22
to build something and see the data flowing I like that the Builder kind of like the Builder mentality I like that
59:27
Builder mentality that's like the uh one of my favorite parts of data engineering as well is like when you like see all of
59:33
it like come together and you're like it feels like you like you made Frankenstein work right because you're
59:38
like oh I wrote all these queries and then they all came together and now we have a beautiful data set right and
59:45
that's great yeah okay yep uh I totally agree creal about that like uh that uh there's like
59:52
that kind of the long-term Outlook yeah yomi I agree that not having to obsess over pixels like I'm totally with you
59:57
there like I uh I do full stack development and every time I get into CSS I'm just

1:00:04
like I hate this I hate this so much right I I I every time CSS murders me
1:00:11
every single time I I love that I love that like um this is awesome all right y'all uh I I totally totally agree I
1:00:18
think that those are some very good uh you know solid reasons like you know what you know what you know what surprises me like something that really
1:00:24
surprises me from all of this is that like nobody not even one person said uh the money right no one said it no one
1:00:31
said it and it's like I don't know it's dat Jers get paid pretty well I I feel like data engineers get paid pretty well
1:00:36
but uh I think that's freaking awesome that y'all like have more of that uh it's more uh you know intrinsic right
1:00:43
you're like I like to build I like to enable I like to help and it's not like I like to get paid right I like to make
1:00:49
fat stacks of dollars or whatever right so um that's awesome um I'm overpaid by
1:00:56
a lot like a lot I care
1:01:02
you oh my God that's so good that's too good yeah I hope I hope by the end of this at the end of this boot camp all of
1:01:07
you are overpaid right that's the whole goal of this right so um okay awesome
1:01:13
everyone like thanks for uh you know coming up with a bunch of good reasons like I think that like you really do
1:01:19
want to focus on those reasons a lot cuz uh you know data engineering can also test you and it can make you uh feel
1:01:26
like you want to run away sometimes and you you want to like not do this anymore so um one of the things I am curious
1:01:32
about is uh now I wanted y'all to drop in chat like what's the hardest part
1:01:39
like what what about data engineering like do you uh find very challenging or
1:01:47
difficult yep on call yeah validating data especially as a perfectionist cuz

1:01:55
you can't get it perfect I remember that like Julio I'm going to tell a story about that in a second
1:02:01
about validating data as a perfectionist that I like uh that I figured out when is like I I think I told the story like
1:02:08
in like week two or something but I want to tell it again because it's a good story work life balance with being on call talking to stakeholders and
1:02:15
bringing everyone on the same page you know multiple stakeholder bosses on call nights troubling why job failed in the
1:02:21
middle of the night backfilling lack of
1:02:27
documentation yeah learn company internal knowledge that is not documented yeah I I I I hate that I hate
1:02:34
how like there's a freaking um like all that tribal knowledge right you have all
1:02:39
those people who like have all of this knowledge and it's just in their head and they haven't like written it down anywhere like uh and that's uh important
1:02:48
uh to like get yeah back filling we're actually going to I literally am talking about back filling today like you know
1:02:54
you know that's this this is great like I I nailed it I nail like I nailed this presentation just based on what y'all
1:03:00
are saying so like I think that there's like kind of like three sort of
1:03:06
um uh three sort of pains that really are that are coming up here kind of over
1:03:12
and over again right part of it is like communication like requests like that's one piece of it right which is like it's
1:03:19
like I like I I like to consider that almost like business on call um that's a
1:03:24
way that I like to think of like ad hoc requests especially like when people are like oh can you like go and build out
1:03:30
this new pipeline for me can you just whip up this data real quick and I'm like is it an emergency like oh it is
1:03:36

okay then uh then then that's then this is now an on call it's a business on call even though it's like development
1:03:43
right it's not necessarily the same as like unblocking or fixing a pipeline but
1:03:48
that's going to be one bucket I think another bucket here is going to be around uh on call like the actual on
1:03:55
call of just keeping your pipelines running that's going to be another one that I think is important and then uh
1:04:01
the third one I guess is around like documentation and knowledge transfer and stuff like that I think those are going
1:04:07
to be your three buckets that really are the the big ones right data modeling
1:04:13
with thousands of tables from entire business without their internal system knowledge yeah dude like if you have a
1:04:20
company that has a th000 tables dude that's freaking that's a lot of tables like I mean like the odds that you need a th000 tables like even if you're like
1:04:27
I don't know like I feel like you know at Airbnb I feel there was like like 300 and that was like all they needed
1:04:33
obviously there was like there's like 300 like trusted tables in the middle but then you have like uh the leafes
1:04:40
obviously there's like thousands of leavs but you don't like I mean most people I'm just talking about the master data that people should build everything
1:04:46
off of right but obviously like uh a lot of companies don't have their data stuff together quite as much as Airbnb but
1:04:55
okay yeah I mean I agree with all the stuff tons of tables a billion tools yeah a billion tools that one's like
1:05:01
interesting like I know cuz you know what's interesting about that problem with a billion tools like like I don't
1:05:06
know maybe this is coming from me from a a place of privilege uh working in big tech for most of my career but like I
1:05:13
don't know like I feel like for me I've been using like spark airflow

1:05:19
S3 and like those three I've been using those three tools like for like six
1:05:25
years and like nothing new like that's about it uh yeah Niti you had a question
1:05:31
they're just like they all about like five Tran and stuff like that like like
1:05:36
no code low code stuff like like the engineer inside of you just dies cuz you're like you're like I want to write
1:05:42
stuff better but like this tool doesn't let me right all right y'all uh I want
1:05:48
to do one last followup here just to see what other things are like what was y'all's favorite data type I need to
1:05:53
know I need to know like I know that's a very like strange question but like oh
1:05:59
dude you got dude Joseph over here Joseph with the date time as your favorite one dude not not date time with
1:06:06
time zone no dude I don't know I don't know about that date time column dude I like Tuple Tuple tuple's a good one um
1:06:14
Boolean Bo Boolean I like I like Boolean too because it's like very like uh it's like a it's as simple as you can go
1:06:20
you're like I like the one and the zero right I like the I like the binary data types right yeah like varar so great
1:06:28
like struct struct string yeah string varar that's so good that's so good I
1:06:34
like all of those like I think um for me personally I like uh I like map actually
1:06:39
the most like I like the like map string string CU that's just like so useful you
1:06:45
can essentially model whatever oh season stats oh my gosh like you're like not not just any struct but this very
1:06:52
specific struct oh man that's so good that's so good all right y'all all right y'all
1:06:59
great I mean I think that uh I learned a lot from y'all here and uh and I hope
1:07:04
yall learned a lot from each other and how like how this kind of data engineering World works like I think

1:07:09
it's important to remember just that like there are things that are painful
1:07:15
that you want to like look into congrats on getting to the end of the day one lecture if you're taking this class for
1:07:20
credit make sure to switch to the other tab so that you get credit