Apache Spark Fundamentals

Day 3 Lecture

Testing Apache Spark Jobs in CI/CD

Transcript:

0:00
[Music] in this course we're going to be
0:06
covering all of the things you need to know to test your pipelines in development too many quality errors are
0:12
caught in production or even later in a dashboard and in this course we're going to be talking about how to unit test and
0:18
integration test your spark pipelines so that you catch the errors in development before you deploy to production and it
0:25
causes a lot fewer headaches for you to deal with later on so if you want to start building your data pipelines with
0:31
a software engineering mindset this 2-hour course is for you and if you're interested in learning how to deploy
0:36
these pipelines in the cloud you should check out the data expert.i academy uh there's a link in the description below
0:42
where you can get 20% off I'm excited to see you there a a quality bug uh like where uh uh where can you catch quality
0:50
bugs um usually there's going to be three three cases uh the first case is
0:56
you catch the quality bug in development so when you're writing the PIP line you catch the bug and then you fix the bug
1:03
before you deploy to production which is great because then it's like almost like
1:08
the bug never existed because the bug never surfaced to anyone and like you don't have to talk about the bug you
1:14
don't have to like do a postmortem about the bug you can just freaking like fix
1:19
the bug in development and be on your Merry way that is the best case scenario

1:25
it's beautiful I love catching bugs in development because that just means it's one less bug in production
1:30
um then you have two ways that bugs show up in production uh one is when you're
1:37
doing write audit publish and what happens is you you write to a staging
1:43
table and then you run your Audits and your audits fail and then you're you
1:49
don't end up publishing to production because your audits failed and then you have to go in trouble shoot why and um
1:56
this is something that uh you want to like minimize the amount of time that you're doing because this can be kind of
2:03
onerous on your data engineering teams especially if you have like a ton of production data quality checks that
2:09
false positive and then it's like I hate false positive data quality checks they
2:14
are common and uh so these types of checks are more expensive in that regard
2:21
than the ones that catch stuff in development because of the fact that you have to check them out in production and
2:27
they can fail and false positive and all these other things and then the worst case scenario is that you have a quality
2:35
air that by passes your quality checks and ends up showing up in production and
2:42
uh usually how you learn about this is oh ah too far there we go usually how
2:49
you learn about this is a data analyst hits you up and is like yo why does this
2:55
data look weird or like or there's a dashboard that has a spike or there's some sort of anomalous thing like that
3:02
that happens and sometimes these airs actually go unnoticed for weeks at a
3:07
time especially if they're more nuanced airs then uh they can go unnoticed for a substantial amount of time so definitely
3:14
know that uh you can uh have bugs that persist in production that you aren't as

3:20
aware of three ways to catch three ways to catch quality bugs in development in
3:25
production but not in production tables and in production in production tables the last one is the one that you want to
3:31
avoid cuz that's the one that damages the company the other two like the middle one is uh is better but it still
3:39
kind of like is a little bit stressful because you have to like troubleshoot something that's in production and it causes a data delay right data ends up
3:46
being backed up and like some people might complain to you like why is the data behind and it's like oh it's because it failed a quality check so and
3:53
then obviously catchi and development's great and we're going to be working in our in the lab today we're going to be working on that uh that first one on
4:00
like how to actually um create test cases for stuff in
4:07
development okay so how do you catch bugs in development that's probably you know I was just talking about that like
4:14
what is the right way to go here um so there obviously you can do two types of
4:19
tests right you have unit tests and integration tests for your pipelines and they both are very useful uh in this
4:27
case uh you could think of a unit test as if you have like udfs in your code or
4:33
you have like in spark say you have like a specific function that like does a
4:39
lookup or does some something like that you want to you want to write tests for each of those like individual functions
4:45
that stuff is very powerful especially and critically if those call another
4:52
Library like for example one of the things for me when I was working in pricing and availability at Airbnb is I
4:59
had to integrate with the pricing and availability libraries that Airbnb used
5:05
and so when I and so my pipelines would need to call those pricing libraries but
5:11

they can edit right they can they can uh like I'm not in control of those libraries and uh the team that is in
5:17
control of those libraries can change them so one of the things that's very powerful if you do write unit and
5:24
integration tests is if you depend on somebody else's code then if they change
5:30
that code and break your test they have to fix your test before they can merge
5:36
because it will break your test and then they can't merge because cicd will fail and so that's a very powerful thing that
5:42
you get from unit tests is they actually allow you to safely depend on other
5:48
people's code and then the other people don't even need to know about your pipeline they just need to know that like hey like this like if I make this
5:55
change things are going to fail right you can catch bugs and Dev right and you
6:00
mostly catch them with tests that's going to be the big way that you catch things um there are other kind of
6:07
important ways uh that you can catch bugs in Dev uh I'll talk about just
6:13
quickly here one of them is going to be around having a linter so lint is a way
6:19
to enforce coding standards so you might not be able to catch a bug this way but
6:24
it makes your code more readable and if you make your code more readable you're more likely to spot bugs and it forces
6:31
everyone to code to the same standard and the same style so definitely look at
6:36
how to implement a linter into your coding environments and that will force everyone on your team to essentially
6:42
code the same way I do that for exactly.com and a bunch of other places so that like whether it's me writing the
6:49
code or someone else writing the code we're all going to do the same like thing right the same standards of like
6:54
spacing and uh variable names and all sorts of stuff like that so anyways

7:00
that's how you catch stuff in Dev let's go to the next slide okay well production's pretty uh
7:06
straightforward to catch stuff right you use write audit publish uh we're going to be covering write audit publish in
7:13
painful detail on Thursday so that y'all can like have a a lot clearer picture of
7:19
how all that stuff works there's another pattern called the signal table pattern
7:24
that we will also cover on Thursday that is similar uh I think it's less good but
7:30
you know Facebook actually prefers the sign signal table pattern over WR audit publish but they both work and uh so
7:38
definitely know that like essentially how this works is you write your data to like um a test table or a staging table
7:45
that has the same schema as your production table then you run your quality checks and if they pass then you
7:52
move the staging data into production and so that's kind of like how write audit publish works and so uh that's how
7:58
you that's how you catch stuff in production and then uh so what's the the
8:03
worst situation right uh the worst situation for data quality is the data
8:09
analyst will be uh they find bugs in your data and then they yell at you and then you feel sad because you're like I
8:15
thought my data was high quality and then it makes you know it ruins the trust of people looking at the data and
8:22
it ruins the mood of you and probably the data analyst as well because you're like why is this data bad and nobody
8:28
wins and so this is uh the situation that we're trying to avoid obviously you could have even more uh painful
8:35
situations where the data quality bug is not discovered and then bad decisions
8:43
start happening uh that's definitely something that I noticed uh when I was working at Facebook there was definitely

8:48
like a quality air that caused too many accounts to be labeled as fake for a while and that kind of like hid under
8:54
the surface for a bit until someone discovered it and so in those situations that's even worse right is once you have
9:01
people making people or algorithms are starting to make decisions based off of
9:06
bad data that is the absolute worst case scenario because then you're actually making a negative impact on the world
9:13
and that's something we're also trying to avoid this is like essentially the the last uh Bastion of uh data quality
9:22
is if the data analyst happens to find it like but you can't rely on manual
9:27
queries to uh enforce your data quality standards because that's like not scalable also
9:34
not reliable also just like uh that's just not what a data analyst is about a
9:39
data analyst is about finding Insight from data not finding bugs in data
9:45
obviously they can do both because that's like kind of like you have to be able to do both because of like the data cleaning requirements of the of the jobs
9:53
nowadays but that's kind of the idea uh behind like how this stuff can be really
9:58
go apocalyptic and so definitely um understanding what we can do as data Engineers to avoid the
10:05
apocalypse is great so yeah well now I'm kind of done talking about the apocalypse let's talk about things
10:11
something a little bit uh lighter mood um this slide right uh software
10:16
engineering actually has higher quality standards than data engineering and um that might sound bold to some of you uh
10:24
but it's 100% true I I found this to be the case uh especially as I transitioned
10:31
from uh when I was a data engineer at Facebook to a software engineer at Netflix just the amount of testing and
10:38

requirements and all of these things that go into software engineering and building live systems is just it's
10:45
different it's like a whole it's like an order of magnitude more quality in tests than in the data engineering domain
10:53
and I think one of the things that's going to happen in the future and is already happening is a lot of these data
10:59
engineering jobs are going to need to adopt a lot of the quality standards that you find in software engineering
11:07
especially where people are building production systems so uh yeah let's uh let's go into like why that's the case
11:13
right because that seems kind of strange that uh software engineering would have more um uh quality standards so uh
11:20
software engineering has you know higher quality standards why um okay uh let's just go over the big ones so if uh
11:27
Facebook the website goes down does Facebook lose more revenue from that or
11:33
if a a data pipeline that powers notifications goes down so uh I mean
11:39
obviously I would think that if people are not accessing the website and they're not being served to ads that's going to cause a problem right and a lot
11:46
of times in data like it's okay if things break even for a day or two and
11:51
because you can just use um uh stale data or data that's a little bit older
11:57
and so you have to um like when the consequences in software engineering when things break is generally an order
12:04
or two an order of magnitude or two higher than the consequences of things when they break in data engineering and
12:11
especially like in terms of like response times like like if you are a software engineer working on the
12:17
Facebook website and then the Facebook website goes down um you need to respond to that at 3 in the morning right and
12:23

generally speaking like for me if a data pipeline breaks in big Tech at 3: in the morning I can wait I can wait until 9:00
12:31
a.m. or 10:00 a.m. I can wait till I'm back at work and I don't have to troubleshoot it in the middle of the
12:36
night and I think that's another big important difference between software engineering and data engineering is the
12:42
fact that you have to really um in software engineering you have to have
12:48
all your bases covered because there isn't as much tolerance for risk as
12:53
there is in data engineering and like it's a good thing I mean that's one of the things I like about data engineering as well because like
12:59
having to deal with where you freaking Wake Up 3 in the morning to troubleshoot on call like okay like like
13:06
that's a that's a that's an issue right so I I think that that's a big one so you can think about risks so like for
13:12
software engineering there's an immediate consequence for lack of quality there's an immediate consequence
13:18
of a bad push there's an immediate consequence of a quality issue and so that's a big thing um I think a second
13:24
area is uh software engineering is just more mature than data engineering
13:29
uh it's been around a lot longer data engineering has only been around like 10 15 years whereas software engineering
13:34
has been around I know like 50 years so big difference there and uh especially test driven development and behavior
13:41
driven development like those two types of development in data engineering are
13:47
very new and they uh for a lot of people they are not as aware about these other
13:53
types of development whereas like if you worked in agile at all or you worked as
14:00
a full stack engineer or anything like that then you are at least aware of test
14:05

driven development if not also behavior-driven development so those things are things that are going to be
14:11
more common in data engineering going forward especially for the data engineer
14:16
software engineer archetype because our data needs to have the same quality standards as our servers do and so and
14:25
just because the risk is lower if they break doesn't mean that we shouldn't hold ourselves to a higher standard and
14:31
because that also allows us to do um have better guarantees and increase uh
14:38
trust and open up new capabilities if we can hold ourselves to a higher standard
14:43
of quality and one last piece of why data engineering and software engineering have different backgrounds
14:50
or is or have different levels of quality is data Engineers come from a much more diverse background than
14:56
software Engineers like you have data Engineers who came from software engineering that's me I'm a Swede I talk
15:02
about that I make that joke all the time like I'm a swed right and then you also have people who came from data analytics
15:08
who are like data analysts data Engineers those people generally have less uh coding breadth and they have uh
15:15
less of the software engineering fundamentals obviously because they didn't come from software engineering but then you also have people who come
15:20
into Data engineering from like other areas right I mean I've seen people who were like e economists and freaking all
15:26
sorts of different other places as well whereas like software Engineers usually like are a lot more likely to just come
15:31
from like oh I have a computer science degree and boom done right so and because of that
15:37
like uh data Engineers are going to be more variable on their own software
15:42

engineering fundamentals and that is that's fine because data engineering doesn't need the same level there but
15:49
like my whole point of talking about this at all is that data Engineers that
15:55
do have these strong software engineering fundamentals are going to be able to uplevel the quality and maintain
16:02
really good standards for their data pipelines going forward into the future so that's kind of uh my perspective on
16:09
like why software engineering is higher quality it's mostly like from one angle is because it has to from a second angle
16:15
it's been around longer and from a third angle the people coming into the field in data engineering have more of a
16:22
diverse skill set that doesn't necessarily include a lot of these like
16:27
software engineering fundamentals so how does data engineering uh become
16:33
riskier uh this is a good one so when I worked uh at Airbnb uh I worked it with
16:39
this machine learning algorithm called Smart Pricing smart pricing for a lot of hosts on Airbnb they don't even pick
16:45
prices at all they let aim ml do it and uh I was responsible for the data that
16:50
fed into that algorithm and that algorithm was responsible for a very large chunk of airbnb's Revenue and so
16:57
um one of the things that we knew was okay if this if this model is trained on
17:03
data that's one or two days delayed how much do we give up like what percent of
17:08
Revenue do we lose for every day that the data is delayed and that's something that we went over and we like measured
17:15
that to understand like okay what are the risks and like how much like can we
17:20
uh be okay with when we are working in the data domain to do these things right
17:26
and so uh like that's a thing to think about right and that's like another
17:31

reason why these things are different is because in software engineering when there's like a delay or a something
17:37
broken a lot of times it's not like a 10% drop but it's like a 100% drop whereas in data engineering a lot of
17:43
times as your data gets more and more stale you see a rapid like kind of a
17:48
like a slow and then rapid degradation of the effectiveness of your models and
17:55
so that's a very important thing to think about when you are building building out your pipelines is what is
18:00
the consequence of something breaking and how do I mitigate that and so big
18:06
important thing there uh another one here uh data quality bugs that impact
18:11
experimentation so if you have data quality bugs that are um persisting and
18:17
they're kind of hiding out because like you aren't doing enough checks uh what can happen here is in the
18:24
experimentation layer when data scientists are running AB tests they might get the wrong idea because there's
18:31
a bug in one of the metrics and they might be like oh this this is this experiment was a success or the other
18:37
way around this experiment was like there was no impact when there actually was an impact and so that can be another
18:44
like thing where like bad data can impact the quality of decisions that are
18:50
made uh it's an important thing to think about and lastly here is like we we have
18:56
data driven cultures uh and hopefully we are as data Engineers we are spearheading that idea of we need to be
19:03
more data driven and uh or data inform data driven whatever words you want to use there and um as we rely on data more
19:11
to guide our decision-making processes the more risk there is in data quality
19:18
issues so as and so it's like as our impact as data Engineers rise we need to

19:25
hold ourselves to a higher standard because people rely on us more we have more responsibility we are carrying a
19:33
bigger weight of the company on our shoulders and since that's the case like we have to uplevel our game we can't
19:39
just be writing SQL queries that dump garbage into the lake and call it a day like we can't do that anymore we have to
19:46
take our skills to the next level we have to really go and build things that
19:51
actually have provable quality constraints and that is going to be so
19:57
important for data engineers in the future future and going forward so remember that like data engineering is
20:02
going to get riskier and especially as you climb the ladder uh as you become like senior and staff engineer you're
20:09
going to be responsible for pipelines that are riskier so that's like just like how I was talking about Smart
20:15
Pricing at Airbnb that's like the reason why I was responsible for that is because I was a staff engineer I was
20:20
like a pretty high level engineer at Airbnb and that's they when you're a high level engineer they give you the
20:25
riskiest stuff so definitely uh that's where having good fundamental
20:31
knowledge here around quality will also help you minimize uh the impacts of of
20:38
quality errors and issues as you progress throughout your career as
20:43
well let's talk about like why organizations miss the mark uh so many places I've worked at uh they don't do
20:51
quality right they don't do it right at all uh so for for example when I worked
20:58
at Facebook I didn't like in my first 18 months you know I worked at Facebook for two years so in my first 18 months at Facebook I didn't write a single data
21:05
quality check not a unit test not an integration test not a production data

21:10
quality check nothing like didn't write anything so uh um it's like and uh
21:18
sometimes I think some people might find that surprising because they're like wow like what like why are like isn't
21:24
Facebook like a very technical really awesome company like why are you not like actually caring about quality and
21:30
one of the reasons for that is at Facebook they had this whole tenant of like move fast and break things it's a
21:37
very important part of the culture at Facebook was like we need to iterate quickly and that put a lot of pressure
21:42
on the data Engineers to actually build and uh build pipelines as quickly as possible which made the quality check
21:49
part was kind of a nice to have kind of like one of those things that's like whatever we don't really need it and uh
21:56
data analytics is more about answering question questions as quickly as possible and that's how you like win
22:02
that's how you get your impact is you answered questions as quickly as possible but it doesn't have like it it
22:09
data analytics and software engineering here do not have the same culture because software Engineers are all about
22:15
building something that lasts and data analytics is all about answering questions quickly and like they both
22:24
like like you can actually have both that's one of the things that uh I think we're starting to realize in data
22:30
engineering is that you can build a data high quality data sets that last and
22:35
answer questions quickly you can have both like you don't have to like pick one right even though they seem like
22:41
they're competing but they're not competing as much as you would think especially on a long enough time scale
22:47

like if you look at it on like a one month time scale obviously there's going to be competition but if you look at it
22:52
on on a two-year time scale they're going to be they should be together because if you want to go quickly right
22:58
cut out cut as many corners as you want but if you want to go far do everything
23:04
right do automated Excellence have all your tests right and I mean that was the
23:09
other thing at Facebook that I thought was crazy was that like um when uh essentially how quality airs were caught
23:15
was the analyst would be like this chart looks funky what's going on here and then we'd spend five days digging into
23:21
like what the hell is going on and then we would just I would update the query and back fill and then that would be it
23:26
and then I wouldn't like add any quality check or anything like that that was just bada bing bada boom done and I
23:33
don't think that and honestly these things like like these behaviors that I'm talking about are definitely
23:38
contributors to why I left Facebook because I was like like why are we just like running around with our like like
23:44
chickens with our head cut off like and not like just jumping from Quality Air to Quality Air like is this really the
23:50
best way that we can possibly do this and I don't know I I I think that it's pretty clear that the answer to that
23:56
question is no uh but that's what I'm saying is that like even Facebook a company that has enough Engineers who
24:02
has some of the best engineers in the world missed the Mark here on data quality so this is a hard problem to get
24:08
right and uh and so definitely understand that that's the case just like I was talking about in
24:14
the last slide there's a trade-off right between business velocity and sustainability and quality like
24:21

sustainability and quality are pretty linked and so because the business wants to answer questions as quickly as
24:26
possible they want to be like yo like I need to know I need this answer yesterday and then they're like can you
24:33
quickly pull this data for me and then the data engineer gets angry because they're like dude this is not like an
24:38
easy quick thing to do and so like how do you decide like do you answer questions quickly or do you avoid making
24:45
Tech debt or do you kind of do something in the middle and so I found uh when
24:51
I've been working in big Tech when I have like strong managers and I have
24:56
managers who know how to protect like their uh reports that you usually index
25:04
more heavily on not creating Tech debt and you go a little bit slower on
25:10
answering business questions and then when I've been working with more weak leaders or leaders who don't understand
25:16
how to push back then essentially analytics just eats the data Engineers lunch and you get burnt out and like and
25:24
I've been there that's was a big part of what happened uh near the end of my time time at Facebook and why I quit uh so
25:32
definitely they both sides of this can happen and you'll find kind of contention on either side uh that's
25:39
where you want to definitely learn how to push back right and don't just solve every ad hoc request and jump from like
25:46
little pipeline to little pipeline like that's where you need to have a more robust data model that's uh so important
25:52
and I hope that y'all learn some stuff from weeks one and two to help y'all build more robust data models
25:58
but that's kind of the idea uh and my perspective has changed dramatically and
26:04
now I I believe that like yeah you should very very very rarely cut
26:10

Corners uh to answer a question faster unless it's very obvious that it's a
26:15
one-off and that you're never going to answer you're not going to need this uh question to be answered again and again
26:21
if it's that case go and do it as quickly and sloppily as you want but like if it's anything else which is 99%
26:27
of the data that going to create you want to do it the highest quality way that you can so that it sticks around
26:34
right because that's one of the things to remember is as data Engineers like we're building roads and highways we're
26:41
not build like we're not answering questions this is something that I think people get wrong is we're building
26:47
information roads and highways and do you want to build like a shitty dirt road or do you want to build like a a
26:52
huge Highway that like that can answer that can answer questions in a way more information rich and high velocity way
27:01
or do you want to build a little dirt road that people complain about the bumps and the edges on obviously you can
27:06
build a dirt road a lot faster you can build a dirt road way faster right and you just throw a couple rocks and some
27:12
gravel and you're done right whereas like building the the information Super Highway takes a little bit longer but it
27:18
reaps more benefits over the long term and I mean even even after like six
27:23
months to a year if you really have your team focus on the stuff you you'll you'll start to see the benefits of
27:30
doing it this way as opposed to just like fighting the dirt road what I'm trying to hammer in here
27:37
is data engineering is engineering it's not analytics it's not like it and this
27:44
was this was another big thing at Facebook that bothered me a lot was that uh at Facebook they have uh the
27:51

individual contributor track and then they have the engineering track and data Engineers were not on the engineering
27:57
track were on the individual contributor track which I was like that is so ridiculous and so stupid and so dumb and
28:04
so I would that's another reason why I left because I didn't I was like I'm I'm I'm an engineer I'm not an individual
28:11
contributor that's like such a weird way to like word things so remember that that like you're you want to be building
28:19
things like we're building powerful infrastructure for answering questions and that is not the same as being a
28:27
sequel monkey for the data scientist or being a sequel monkey for the data analyst it's not about that like cuz
28:34
that's not how you build powerful robust systems so keeping in mind that data
28:40
engineering is engineering and so and I mean y'all are on the infrastructure track or the
28:46
combined track so I hope y'all probably agree with me on that some of the people on the analytics track might not but uh
28:52
so that's kind of the idea uh for this and since it's engineering we're trying to build cool stuff
28:59
since it's engineering uh usually things that are engineered get better and better over time right because if you
29:05
think about um like you think about like the first Tesla right the Roadster and it was like super expensive and it still
29:12
went pretty fast but it was super expensive and then it kind of iterated and it got cheaper and faster and then
29:18
like self-driving and like they're keep like they keep engineering more stuff into the Tesla right and it's like wow
29:24
like that is freaking nuts so um like that's one of the things that I don't
29:30
like about the narrative that like data Engineers are just trying to answer questions because that like narrative
29:36

really doesn't really speak to the fact that we're Engineers too and here are four different ways that we are going to
29:45
be creating more value over time as data Engineers right so you have latency
29:51
latency we're going to be moving pipelines from batch to streaming or batch to micro batch so that like they
29:58
data is available sooner and it's processed sooner and is ready and sooner
30:03
because for some things like you know like fraud detection and bad behavior detection or dynamic pricing you really
30:10
need as upto-date data as you possibly can get but one of the things that's a trade-off here is when you have a
30:16
streaming pipeline uh one of the things that's different between streaming and batch is streaming pipelines run all the
30:23
time they run 247 whereas batch pipelines usually run Maybe like 1 hour
30:28
a day or 30 minutes a day so in that way your quality standard is very different
30:34
because if something's running 24/7 the number of things that can break is a lot higher and so just think about it like
30:41
if you ran for one hour as a human you probably wouldn't get injured but if you
30:46
ran for 24 hours as a human you probably would get injured just because of the nature of running and like how things
30:53
work right and like just the longer it's similar also like how like men have
30:58
higher insurance rates than women and the main reason for that is because they drive more they're on the road more and
31:04
the more you're on the road the more likely you are to crash so that's going to happen in data engineering as well
31:10
we're going to have more streaming pipelines and microbatch pipelines which both run a lot more often than like
31:16
daily pipelines so that's going to be one way that we're going to kind of move more into that software engineering

31:22
fundamentals is we're going to have these 247 running pipelines that are just really doing doing uh a lot of
31:29
really great great stuff uh obviously we have Quality quality that's what this whole week's about uh we want to be able
31:37
to test and prove that our data is high quality and answers questions and you
31:42
might want to use a framework like grade expectations or Amazon DQ we're going to be using one called chisa in um the the
31:49
lab today which is used for a unit testing and integration testing so
31:55
there's going to be another up level there you're not just be able to write a SQL query throw it
32:00
onto the data Lake and call it a day like that's not like no please don't do that like you're giving data Engineers a
32:07
bad name if you're doing that um then you also have completeness uh which is one that is more solved
32:14
through communication and understanding the domain so like you might become more of a domain expert uh as you kind of get
32:21
better and more well vered in data I learned that like after I worked in pricing and availability for 2 years at
32:26
Airbnb I was like dude like I could start my own hotel company dude like holy I know so much about this now like but like and
32:34
you'll slowly build up more of that completeness and you can be a do domain expert but you also want to work with
32:39
your stakeholders and other people who probably are even closer to the business in that way so that you can get a bigger
32:46
picture through communication and you can answer things more
32:51
completely so anyways the last one is around like ease of access and usability
32:58
so um most people or data Engineers they think oh uh our product is we give
33:06

people a data table and then they can run SQL on it or maybe we give people a dashboard and then they can uh play with
33:13
filters on it if we're like doing we're going above and beyond um there's more there's way more things that you can do
33:19
right like for example when I worked at Netflix I worked on a data product that uh surfaced all the data via API call so
33:27
then you can just hit a URL and then you can get whatever data you want and then you can have automated Integrations that
33:33
way where like people can read and write to whatever data store that you have and that's going to be something that is
33:40
going to be more common uh you have like a data product highly recommend that
33:45
checking out that stuff and then also proper data modeling because that's how you increase usability the idea here is
33:53
we're going to be doing more as data engineers and data engineer software engineers and that's going to require us
33:58
to have better standards and we're going to need to hold ourselves to a higher quality bar and so that's uh the main
34:06
point of this uh slide here so how do we adopt this new mindset of
34:12
like how do we create code and systems that follow these software engineering
34:19
mindsets how do we do data engineering with a software engineering mindset so
34:24
one of the big ones here is I like to talk about this first line is code is mostly to be read by humans and ran by
34:32
machines like on occasion like the fact that like uh like so you want to write
34:38
code that's meant for humans to read not for machines to run that's a very very
34:43
powerful thing that you can do I mean I've seen this problem in data engineering as well where people are like oh like if I do three subqueries
34:51
it's it's 5% faster than if I do two uh Common Table expressions and it's like
34:57

yeah it's 5% faster but like how much pain and suffering are you causing your
35:02
fellow Engineers to like troubleshoot stuff so because the more readable your
35:08
code is the more EAS the easier it is to actually troubleshoot your code by you
35:14
or by another engineer so remember that code is written for humans not for
35:20
machines that's a very very powerful tenant of software engineering fundamentals that if you can really take
35:26
that to home it will make you a better engineer and then we have two types of failures right so uh the failure type
35:34
that is your enemy is a silent failure if you have a failure that's happening or a bug that's happening that doesn't
35:40
show itself that's the type of bug that you don't like that is going to be the bugs that uh make you sad those are the
35:47
bugs that cause a lot of problems and so uh you want to avoid those things um for
35:53
example uh if you're in Java you can use uh if you come across cross an exception
35:59
a lot of times the the better thing to do is just use throw like throw the exception like and actually crash the
36:05
program because that's going to be better than like just having the the that silent failure because then at
36:12
least if it crashes it's going to alert you and then you you can troubleshoot like what's going on as opposed to it
36:17
just like silently like causing issues and then succeeding or whatever then
36:23
that is not like good you don't want to do that so you if you're in Java you can
36:29
use the throws keyword in Python they have another keyword that you could use it's called raise so you can raise and
36:35
ex like you can raise an air or raise an exception in python as well it's the raise keyword and that's another way to
36:42
kind of Crash Your programs in ways that make it so that your silent failures are

36:47
are louder so that you don't have to uh like worry about that too much so loud
36:53
failures are great they're annoying because they like you have to actually go and troubleshoot them but if you have
36:58
a loud failure you know like at least it's telling you it's broken and that's going to be great so uh that is where if
37:08
you have a a failure that's loud that's coming from Bad data so for example you
37:13
have a record that failed to parse or you have a record that is not working the way you think or it has duplicates
37:20
or whatever fail the pipeline throw an exception and maybe even throw a special
37:25
exception like a throws like d duplicate record exception like you make your own fancy exceptions right do it that way
37:31
like and that that can be a really powerful thing to do and then obviously you also have uh loud failures can can
37:37
show up in testing and uh that's important because cicd will catch them
37:43
and then but and you'll have these loud failures that are caught in cicd before you deploy to production you know how to
37:49
do data engineering uh with a software engineering mindset right so you have dry code and yagy principles I think
37:55
these are very good uh dry code is don't repeat yourself uh which means you need
38:00
to encapsulate and uh reuse logic and then yag me stands for you aren't going
38:06
to need it which means you build things out as they're needed not try to build out the master awesome pipeline like in
38:13
the first go uh SQL and dry compete this one of like why I kind of love and hate
38:19
SQL because SQL is kind of hard to encapsulate and uh and it's kind of hard
38:26
to test and dry code is better to test because it's encapsulated into like one
38:31

function and so that's great obviously design docs are your friend uh in the analytics track yesterday we talked a
38:37
lot about design specs and how those work and how to get reviews and all that
38:42
kind of stuff on your design specs which can be very good at catching a lot of these airs and obviously care about
38:49
efficiency like data structures and algorithms Big O notation like how many iterations is going to happen when we do
38:55
a join and the shuffling and all those kind of things like is it going to be a cross joint cartisian
39:01
product like who knows right so definitely care about efficiency as well like those things are super important
39:06
not just like testability I think we're we're we're almost at time for the presentation so
39:12
should data Engineers learn software engineering best practices short answer yes uh I mean some might not benefit as
39:19
much if they are in a more analytics focused role if they're in a role that is very focused on like experimentation
39:26
metrics and Analytics like then that these best practices are less valuable but like for I one of my perspectives is
39:34
if you learn these best practices software engineering best practices then you don't have to be a data engineer
39:40
like you can go and be a software engineer if you want to as well like or a full stack engineer it opens the door
39:46
to so many other opportunities and makes you more uh of a leveraged candidate so
39:52
like like cuz one of the things I always think about like if you want to be really rich uh one of the things that
39:58
you should have is have a lot of options and so that was one of the things for me I was like I wanted to be a full stack engineer like more of like a an mle like
40:05
machine learning engineer data engineer I wanted to have like essentially all of those doors open for me and one of the things I did was like okay learn these
40:11

kind of Hardcore best practices and then you can have all those options and then that helps you make a lot more money and
40:17
so obviously like it's hard to learn everything but learning the fundamentals these like strong fundamentals can be
40:23
very important to making you better right and let's talk about l M for a second because I think large language
40:29
models are freaking people out uh one of the things about large language models
40:34
is they are going to be better at the analytics and SQL layer of the pipeline than they are going to be at other
40:41
pieces of the pipeline like the testing and the optimization layers of the pipeline that stuff is going to take
40:47
llms a little bit longer to figure out because it's like more nuanced and like it really does depend on like the data
40:53
shape so uh in some ways learning this stuff is how you can really kind of fend
40:59
off the lm's ability to automate your job and if you don't want to do this uh
41:05
go to analytics engineering then like don't freaking be a data engineer like like uh my perspective is like if you
41:10
don't want to embrace software engineering fundamentals like don't don't don't do data engineering like
41:16
that's not like what you want to do uh it's like the that's kind of where the
41:21
field is going and so in analytics engineering you don't really need as much of these fundamentals congrats on
41:26
getting to the end of the unit testing and integration testing lecture if you're taking this class for credit make sure to switch over to the other tab so
41:33
that you can get your credit for the lab in the lab we're going to be generating a lot of really exciting tests using a
41:38
library called chisa get ready welcome to the data engineer testing setup lab