<u>Data Quality Patterns</u>

Day 2 Lecture

Transcript:

0:00
[Music]
0:05
cardi B ruined this data quality pattern
0:08
there's a pattern called WAP pattern uh
0:10
stands for write audit publish not wet
0:14
and like it is how you guarantee good
0:17
quality data gets published into
0:19
production how it works is you write
0:21
data to a staging table that has the
0:23
same schema as production then you run
0:25
your quality checks or your Audits and
0:27
then if the audits pass then you publish
0:30
the data from staging into production
0:31
and if you just follow this one small
0:34
pattern that cardi B ruined you will
0:36
prevent 80 to 90% of the data quality
0:39
headaches that are generated from
0:41
publishing bad data such as breaking
0:44
trust with the analyst wasting a bunch
0:46
of time running stupid SQL queries and

0:49
many many other painful things that you
0:51
will avoid just by implementing this one
0:53
pattern so I hope you like the lecture
0:55
today uh if you want to learn more about
0:58
how to do this stuff in the cloud
0:59
definitely check out the data expert
1:00
Academy in the link description below I
1:02
can get you 20% off enjoy the course
1:05
what causes bad D this is not a
1:08
comprehensive list but this is a pretty
1:10
solid one so logging erors I'm just
1:13
going to talk through a couple things
1:14
that could happen with logging uh one is
1:17
a software engineer could change the
1:19
schema of the logs and then the schema
1:21
of the logs doesn't match the schema of
1:23
the table that you are ingesting them
1:25
into that usually causes NES to show up
1:28
or like has issues like that you could
1:29
have have different bugs with logging um
1:32
so here's an example uh say when
1:33
someone's like submitting a form on a

1:35
website you know when you like click a
1:37
button on a website most of the time
1:39
when you click a button the button
1:40
becomes disabled while the form is being
1:42
submitted but that's something that the
1:44
engineer has to code and if they don't
1:47
actually disable the button then uh you
1:49
can have logging and data errors because
1:52
people can double click and then they
1:54
get two records and it says that they
1:55
submitted the form twice and that can be
1:58
problematic as well well then uh so
2:02
logging has all sort I mean and I just
2:05
just just scratching the surface there
2:07
on like the ways that logging errors can
2:09
happen and that like we're not going to
2:11
go too deep there that's more covered in
2:13
the infrastructure track uh that is
2:16
we're going to cover that in more detail
2:18
uh tomorrow if you're in the combined
2:20
track um then snapshotting errors so so
2:25
logging errors is more for fact data

2:27
snapshotting errors is more for
2:28
dimensional data uh snapshotting errors
2:32
uh are actually kind of rare I actually
2:35
I think in my whole career I've probably
2:37
bumped into a snapshotting air like I
2:39
don't know less I can count on one hand
2:42
one so it's like less than once a year
2:44
do I run into this problem but generally
2:47
speaking like what can happen here is
2:49
when you snapshot the data it's you're
2:51
missing some Dimensions or you're
2:53
missing some users or you have too many
2:55
users because of the next one down which
2:58
is production data quality issues so in
3:01
that case is where you have uh actual
3:05
bad data in production like in your
3:07
production application so you have data
3:10
in production right that's just
3:12
Incorrect and you want to uh you pull it
3:14
in and then you have to filter it out or
3:16
deal with it some way and a lot of times
3:18
those are things that you have to work

3:20
with software Engineers to figure out um
3:22
schema Evolution can happen both for
3:24
snapshotting errors and for logging
3:26
errors cuz the logging schema can change
3:28
in production and then it doesn't match
3:30
your table schema same with the
3:32
production table can change and it
3:35
doesn't match the table in your data
3:37
Lake those are problems uh pipeline
3:41
mistakes right uh where you have an
3:44
error in your joint or your case when or
3:48
you all sorts of different pipeline
3:51
problems that can happen and then those
3:53
those get into production because you
3:55
don't follow the contract process that I
3:58
was talking about earlier here um and
4:01
then uh here's that item potent word
4:04
again like and it's going to keep coming
4:06
up uh so you have a non- item potent
4:08
Pipeline and you backfill and that that
4:11
causes different errs and then the last
4:13
one I think is actually the most common

4:16
uh cause is when you are going to
4:20
release a data set you don't validate it
4:23
enough so you know in the um the speaker
4:26
series today when Alex was talking about
4:28
he had like he had JD and who was a data
4:32
engineer he worked with and there was a
4:33
lot of back and forth between him and JD
4:36
when they're building data sets that's
4:38
something that you should be working
4:39
with or you should be doing a lot with a
4:42
data analyst or a data scientist some
4:44
sort of like analytics counterpart
4:46
should definitely be involved in this
4:49
process if they're not like you're going
4:50
to have um a harder time but that's kind
4:54
of the idea behind um what causes bad
4:57
data um other things that I think are
5:00
important to talk about is if you are
5:02
ingesting data from a third- party API
5:05
say you are working with like Salesforce
5:08
or slack or Discord or I don't know
5:12
stripe or whoever third party right and

5:15
they uh they give you an API to ingest
5:17
data from uh they can change their
5:20
contract just the same right and they
5:22
can mess with you that way and that can
5:23
break stuff as well because that's a
5:26
different contract that you don't have
5:28
control over so you have to be
5:31
especially careful with data that you're
5:34
ingesting from third party apis because
5:37
that is like you can't just like go and
5:40
talk to those software engineers and be
5:41
like hey change it because like they're
5:44
like a company and you have to like go
5:46
through that process and they're
5:47
probably not going to listen to you
5:48
anyways even if you were able to get to
5:50
them so that's a thing to think about uh
5:54
like so bad data lots of lots of ways
5:56
that bad data can happen but what we're
5:58
trying to do is prevent it from showing
6:01
up in production that's the main main
6:04
purpose of this presentation and the

6:06
main thing that I'm trying to instill in
6:08
y'all as we go through
6:10
this okay so we're going to try
6:12
something different today um this is
6:14
going to be literally the first time
6:15
I've tried this in the um in the boot
6:18
camp V1 or V2 what we're going to do is
6:21
we're going to split into some breakout
6:24
rooms and then we're going to uh we're
6:27
going to do four things right and this
6:30
should only take maybe 10 15 minutes uh
6:33
where you're going to go in you're going
6:34
to introduce yourselves and then you're
6:36
going to say how you pronounce that
6:38
those characters there and then um and
6:41
then you're going to talk about uh a
6:43
time when bad data was handled correctly
6:46
if you have a time and then a time when
6:48
bad data was handled incorrectly and
6:51
what happened and like what were the
6:53
consequences of that like whatever you
6:55
can talk about like obviously don't

6:57
break any like don't get fired from this
6:59
breakout room or whatever but uh that's
7:01
kind of like what I want to do here so I
7:03
hopefully all can like get in and and
7:06
meet some more people from your boot
7:07
camp as well so we have validation best
7:10
practices so uh what that was one of the
7:13
common themes I saw from the breakout
7:14
sessions was uh
7:18
like not enough validation not enough
7:21
validation rules and like or like that
7:24
was the like having proper validation
7:26
rules was like a good way to handle bad
7:28
data and not having enough was a way
7:31
that bad data showed up more
7:33
so this is a great thing like especially
7:36
when you're like creating a new pipeline
7:38
at Airbnb what they would do is they say
7:40
okay back fill a small amount of data
7:41
maybe like a month of the data set and
7:44
then uh uh check all all your
7:48
assumptions about the data like the

7:49
nulles the duplicates
7:51
the the enumerations the time series the
7:55
row counts all of it those things should
7:58
all be in like you should have like a a
8:00
dock of some kind that has a bunch of
8:01
charts in it that says like here's what
8:04
this data looks like and we feel
8:07
confident in this one month so we can
8:09
probably backfill the rest of it and
8:11
like if it doesn't look good then you
8:13
probably want to fix whatever uh is
8:16
problematic before you go into the next
8:18
kind of steps of the puzzle so that's a
8:21
good thing to think about as you kind of
8:22
like go through this process of uh
8:27
like finding bad data like I found this
8:30
process to be more on the analytics side
8:32
like the data analyst or the analytics
8:35
engineer is going to be the one who's
8:36
writing this like because that's the
8:38
other thing you see on this slide I say
8:40
have someone else check all your

8:41
assumptions don't do it yourself I tried
8:44
to do that at Airbnb I remember like
8:45
when I was releasing like the pricing
8:47
pipeline because like the analytics
8:49
engineer was out sick and I was like
8:51
it's okay I got it I got this right I'm
8:54
a good engineer I'm a Staff engineer I
8:56
freaking got this and I totally didn't
8:58
have it like it's it's impossible like
9:00
if you build the pipeline it's totally
9:02
impossible to validate it dude it's like
9:04
I don't get why that's the case but it
9:06
totally is or you need a you need a
9:08
separate human to do it and because I I
9:10
I thought I I validated everything right
9:12
and then I was like oh wow I did not
9:14
because uh like you can't like see your
9:16
own data sometimes it's it's kind of
9:18
kind of interesting and intriguing so
9:20
definitely don't uh break that rule
9:21
definitely have someone else look at
9:24
it okay so we're going we're almost to

9:27
the memes uh so writing writing to
9:30
production is a contract this is the
9:32
second kind of big theme of uh today's
9:37
lecture and lab um and the contract has
9:42
a couple pieces to it so the first one
9:47
is the obvious one which is the schema
9:49
which is like the columns and data types
9:51
of the data great uh um but in the data
9:57
Lake that like that in the old world
10:00
right in like postgress and like
10:02
relational database world that contract
10:05
was like stronger right you can do
10:07
things like put a unique constraint and
10:10
a not null constraint on your data and
10:14
that's pretty cool uh in the data Lake
10:17
you can't though all those things kind
10:20
of go out the window in the data Lake
10:22
because the data lake is just like files
10:24
and you don't get the same level of
10:28
constraints like where you know in
10:30
relational World a lot of this stuff is
10:32
the same thing so you don't have to like

10:34
really worry about it but in the in the
10:36
lake you do have to kind of worry about
10:37
it a little bit more so you have schema
10:39
and then you have quality checks uh
10:41
quality checks being like row count
10:43
checks and duplicate checks and notnull
10:46
checks and all the different types of
10:48
checks that you want to put in your data
10:50
table uh those need to happen as well
10:53
and then the last one is how data shows
10:55
up in production and that last piece uh
10:58
is the one that is a little bit
11:00
different depending on the contract that
11:02
you choose and we're going to go over
11:04
there's essentially two big kind of
11:06
competing ways of doing things uh for
11:09
your like if you're writing out a data
11:12
contract so let's uh dig into it a
11:14
little
11:15
bit essentially we got two flavors of
11:19
the contract you have the WAP pattern
11:22
which

11:23
like preceded cardi B like cardi B like
11:26
ruined the name and like and obviously
11:29
to put cardi B in the slides cuz like
11:31
owed to her like she doesn't even real I
11:34
I don't think she even realizes how much
11:36
like of like a data engineering like
11:38
Rockstar she is like by like kind of
11:41
co-opting that and like taking taking
11:43
WAP away from us but it is what it is
11:46
and uh we're going to be uh talking a
11:48
lot more about WAP pattern and then also
11:50
there's another pattern called the
11:51
signal table pattern so when I was
11:54
working in big Tech
11:56
uh Netflix and airbeam B they use the
12:00
WAP pattern and Facebook used the signal
12:03
table pattern and they are uh they have
12:06
uh pros and cons and benefits and risks
12:08
and all that stuff they are a little bit
12:10
different and uh we're going to go a
12:12
little bit deeper into both of them so
12:14
yeah let's let's talk a little bit more

12:15
about WAP right WAP stands for write
12:19
audit publish that's what it stands for
12:22
so write means if you look at this
12:26
diagram you have a pipeline and it WR to
12:29
a staging
12:31
table the staging table should have the
12:33
same schema as the production
12:37
table
12:38
important then you run your quality
12:42
checks you say okay did they
12:46
pass if they passed move the data from
12:50
staging to
12:52
production and then once it's in
12:55
production that means that the down
12:57
whatever pipeline's Downstream are ready
12:59
to
13:00
fire great uh there's one more thing
13:03
here where uh your qual like quality
13:08
checks can be what's called blocking and
13:10
non-blocking so a blocking quality check
13:13
is like a very serious data quality
13:16
issue that you want to you have to

13:18
troubleshoot whereas a non-blocking one
13:21
might be like more instead of like it's
13:23
instead of calling it like a quality
13:24
issue it might be more classified as
13:26
like data weirdness and like essentially
13:30
how it works is like you see if you
13:31
follow the path here it says like okay
13:33
did the quality um did the quality
13:36
checks pass no okay then you fire the
13:38
alert because there's something weird
13:41
with the data and then we say okay is
13:44
the check blocking no and if it's not
13:46
blocking then we just we publish anyways
13:49
um but if it is blocking then we stop
13:51
the pipeline and we manually
13:53
troubleshoot and we have to uh figure
13:56
out like what's going on because it's
13:58
worth it to us to do that as opposed to
14:01
uh cuz even in those cases when uh you
14:04
have a blocking check that fails there
14:06
can be cases where that's just how it is
14:08
and that's just the data like I know for

14:11
sure one of the times that that happened
14:12
for me when I was at Facebook was we had
14:14
these growth pipelines and uh we had a
14:17
call a blocking quality check fail
14:19
because of the fact that like um
14:22
Ethiopia um had uh like we we saw almost
14:27
everyone in Ethiopia leave Facebook and
14:31
uh from our pipeline perspective it's
14:33
like oh that's a quality air but then if
14:35
you look if you look at the world events
14:37
you can see oh no the world leaders of
14:39
Ethiopia turned off the internet and
14:42
it's like okay well then it's not a
14:44
Quality Air it's actually like the
14:46
actual data right so then you have to uh
14:49
and then so in that case we we it was a
14:52
blocking check we troubleshooted it and
14:54
we're like well that sucks for Ethiopia
14:57
but this is not actually quality air and
14:59
then we exchange the partitions and we
15:01
go on with our lives this is right audit
15:03
publish um it's like my favorite I like

15:08
this the most I I think that Facebook
15:10
does it wrong and I think that Netflix
15:11
and Airbnb do it right uh and this is
15:15
the pattern I like um we're going to
15:17
talk a little bit more about the signal
15:18
table pattern and uh we're going to
15:21
actually kind of alternate between the
15:22
two slides because I think it will
15:23
really clearly illustrate the
15:25
differences so so here is the signal
15:29
table pattern um in the signal table
15:32
pattern how it's different is there is
15:35
no staging table you see the pipeline
15:39
writes directly to
15:41
production and then we run our quality
15:43
checks on the production table and we
15:46
get the same blocking thing but um if
15:49
the quality checks pass what we do is we
15:51
have a thing called a signal table and
15:53
the signal table tells the downstream
15:56
pipelines that that the data is ready to
15:58
go

15:59
so what it means is the downstream needs
16:03
to wait on the signal table not on the
16:07
production table and if you do that like
16:10
this pattern is I mean obviously if you
16:13
like if we jump between these two you
16:16
can see
16:17
that the the signal table pattern is a
16:22
fairly it's a simpler pattern right but
16:25
it comes with a lot of kind of tradeoffs
16:27
and risks one of the things that I do
16:30
not like about the signal table pattern
16:33
is that it does not cater to ad hoc
16:37
queries very well because say you have
16:40
your pipeline it runs it writes it
16:43
writes directly to production and then
16:45
it fails the quality check um and then a
16:48
data scientist goes and queries
16:49
production because the data scientist is
16:51
not going to look at the signal table
16:52
they're not going to do that like that's
16:54
freaking Madness like the data engineer
16:57
might look at the signal table but the

16:58
the data scientist and the data analyst
17:00
is going to query the production table
17:02
and like there's a good chance that like
17:04
if the quality check fails they're going
17:05
to query production that has bad data in
17:07
it or it's going to have data that's
17:09
incorrect in it so that is like my
17:11
biggest beef with the signal table
17:13
pattern even though I also recognize
17:16
that it is a simpler pattern and it
17:18
actually uses less compute and less IO
17:21
so in some ways it is more efficient in
17:23
that way so let's talk a little bit more
17:24
about like the pros and cons of each
17:26
sides of this contract so
17:30
so write audit publish uh one of my
17:34
absolute favorite part about write audit
17:36
publish is there's no chance that data
17:39
in production gets there without passing
17:41
your quality checks it has to pass the
17:44
audit step to show up in production
17:47
which is beautiful that's like and then

17:49
so that means that like if you have an
17:50
ad hoc query user who is querying your
17:53
production data they get the same
17:56
guarantees as the signal table people
17:58
people do in the signal table pattern
18:00
which is great um and like obviously you
18:04
don't even have to worry about signal
18:05
tables you can just work with production
18:07
tables and they can be they can be your
18:09
signals for your Downstream pipelines
18:11
which is a lot more
18:13
intuitive uh right audit publish has a
18:16
minus the minus being you have that
18:18
partition exchange where you have to
18:20
move the data from staging to production
18:23
you actually have to like depending on
18:25
you sometimes you can do an ex like if
18:27
you're an S3 you can do a partition
18:29
exchange and it just moves the file
18:32
right or it moves the folder it does
18:33
like a folder rename so you don't
18:36
actually move anything you just like

18:37
rename the folder and like the data
18:39
doesn't even really move so that can be
18:42
P that could be powerful but like most
18:44
of the time there's going to be at least
18:45
a couple minutes delay when you're doing
18:48
uh WR audit publish versus signal table
18:51
because of the fact that you have that
18:52
exchange step that does take a little
18:55
bit of time obviously uh signal table
18:58
essentially the pros and cons are
18:59
flipped signal table uh you have uh is
19:03
faster um it's going to hit SLA more
19:06
likely and the data is going to land
19:07
sooner going be more readily available
19:09
and then I already talked about the cons
19:11
like the ad H data scientists could end
19:13
up querying bad data and it's not
19:15
intuitive and like data Engineers might
19:18
not end up waiting on the signal table
19:20
they might wait on production and then
19:21
they're going to pull in bad data the
19:24
idea here guys is it's a contract and

19:28
like you can see how these two have
19:30
their different contracts and like how
19:33
the handshake is done and the
19:34
implications of that handshake are not
19:37
the same and so um that's a thing to
19:40
think about like as you're kind of going
19:42
through your uh processes of building
19:44
out your pipelines and stuff like that
19:46
so how does the contract look and like
19:49
what are the benefits and risks uh
19:51
personally I like write audit publish
19:54
signal table not as good but it has some
19:57
benefits if you has some marginal
19:59
benefits that are obviously I listed out
20:02
here so let's talk about um what happens
20:07
if these contracts are violated so most
20:10
of the time if you have a non-blocking
20:11
check uh it's fine but like if if you
20:14
don't obey the contracts like say you
20:16
write a pipeline that doesn't do write
20:18
audit publish
20:20
then you get bad data propagation and

20:24
bad data propagation is literally one of
20:26
the worst things in the entire world
20:28
like it's literally um man one of the
20:30
one of the very big issues here so let's
20:34
talk about what I mean by that like uh
20:36
bad data propagation like pipelines
20:39
write output data and then a lot of
20:41
times that data is then read in by other
20:44
pipelines that then output data and you
20:47
have like a um kind of a a tree like
20:49
that where you have like it keeps
20:51
writing out writing out writing out and
20:53
it's great uh it's one of the things I
20:55
love about Big Data it's a crazy tree
20:58
thing right so if you have a pipeline
21:00
that has say say you write a pipeline
21:01
and then the only Downstream is like a
21:04
dashboard then the consequences of bad
21:07
data aren't very large because there's
21:10
just the dashboard and an analyst might
21:12
look at it and be like this data is
21:13
funky and then that's it um as you work

21:17
with more and more critical data sets
21:19
things get more complicated like for
21:22
example like when I was working in
21:23
pricing and availability at Airbnb uh
21:26
those tables had over a thousand downam
21:29
pipelines so if I published bad data
21:33
then that bad data would then get picked
21:36
up by a thousand pipelines and then that
21:38
bad data would then be like for my bad
21:41
data would now be in a,1
21:44
spots so uh that's a problem right and
21:49
like essentially as you create data sets
21:51
that are more and more critical to the
21:53
business and more and more heavily used
21:56
the more you need to do this the more
21:59
you need to publish good data and follow
22:02
these contracts so that like you can
22:05
have all the quality checks in place so
22:07
that when you publish data you can
22:09
publish data with confidence and you
22:11
don't have to uh because backfilling
22:14
that gets very crazy because then it's

22:16
like oh we have to just like if if you
22:18
have a b if you have a very important
22:20
table that publishes bad data and then
22:22
it gets proliferated you essentially
22:24
have to backfill the entire warehouse
22:26
for a day and so it's like oh now
22:29
we just got to double up on our
22:30
warehouse compute for a day and it's
22:32
like that's expensive that it's really
22:34
expensive especially from an IO
22:35
perspective it's very expensive um so I
22:39
want to just talk about an example here
22:41
um so there's a table called Dim All
22:43
users at Facebook which you know the the
22:45
table I was those pricing and
22:46
availability tables I was talking about
22:48
that had like a thousand downstreams um
22:50
dem all users is another level a whole
22:52
other level it was like like 10 or
22:54
20,000 downstreams where um what ended
22:57
up happening was there was a migration
23:00
to a new uh compute engine and it

23:03
actually ended up breaking this contract
23:05
and kind of ignoring the contract
23:07
completely and then uh bad data was
23:09
published and then it was picked up by
23:12
uh 20,000 Downstream pipelines and then
23:15
uh we had to uh then then it's like as
23:18
data introduced we have to figure out
23:19
how to fix that right that's like a
23:22
that's a big nasty problem and like it
23:24
becomes like it it completely derails
23:27
your other development right and it's
23:29
like wow I have to now figure out this
23:31
big giant data quality mess instead of
23:34
like actually doing my job of like
23:36
building new pipelines and adding new
23:38
features we have a dem all users right
23:40
and so what we have to do there is we
23:42
essentially have to find all the
23:43
pipelines so we have to fix the bad data
23:45
and then find all the pipelines that
23:46
pick that bad data up and then uh run
23:49
them again right because keeping in mind

23:51
in those cases like you can actually
23:53
have not just uh secondary um pipelines
23:56
but you can have tertiary Pipelines
23:58
where your bad data is picked up by
24:00
another pipeline which is then picked up
24:02
by another pipeline right and then you
24:04
have to think about oh what about the
24:06
tertiary impact of uh bad data so the
24:10
main thing I'm trying to nail home here
24:12
with y'all is like if you are working
24:13
with data sets that are heavily used
24:15
throughout a big company you have to do
24:17
this you have to do this like because if
24:20
you don't and you have and and and and
24:23
this happens twice where you publish bad
24:25
data twice you're gone you're fired done
24:27
get out of here like it's too expensive
24:29
like in fixing that problem for Facebook
24:31
probably cost them almost a million
24:32
dollars right so it's like like not
24:36
worth it not worth it right follow these
24:38
contracts these contracts are very

24:39
important and we're going to be going
24:41
over how to do one of these contracts in
24:42
the lab today so um uh if you have a bad
24:46
metric definition or you filter down too
24:50
far things get noisy they get very noisy
24:53
and like so you can't do data quality on
24:57
uh
24:59
uh on like metric definitions that are
25:01
too narrow because you the problem is is
25:05
you you get to a point where normality
25:08
doesn't exist anymore so you know my
25:10
comical one here is knowing the
25:12
engagement per qualified minute of
25:14
Ethiopian children who use
25:16
seven-year-old iPhone devices on
25:17
Christmas
25:19
day is going to be a noisy metric right
25:22
because like that's there's that's a lot
25:24
right that's a lot of cuts and like
25:26
what's what does qualified mean like
25:27
qualified minute okay like that has its
25:30
own definition and its own set of

25:31
metrics right and then you have like
25:34
okay like seven-year-old device that
25:35
seems pretty narrow and so like when
25:38
you're coming up with metrics like they
25:40
should be pretty Broad and you might you
25:42
can get away with like one or two like
25:45
dimensional Cuts you might be able to do
25:47
like female iPhone users or like
25:51
American iPhone users or female
25:54
Americans or you might be able to get
25:56
age in there as well but like generally
25:58
speaking like you want to have like your
26:01
metrics and like the your where you're
26:03
throwing in your quality checks on your
26:05
metrics you can't have them be so narrow
26:10
because like there's like there's going
26:13
to be like what because in this case
26:15
well maybe there's only seven Ethiopian
26:17
children who do this right and then uh
26:20
like then the next time around there's
26:22
five and then it's like oh it goes from
26:24
seven to five and then people are like

26:26
wow a 30% drop holy crap right and like
26:30
um like and like your thresholds are
26:32
going to be all funky and so uh that's a
26:34
thing to think about when you're like
26:36
thinking about metric definitions is
26:38
like you got to be careful right a lot
26:41
of this is common sense like and like
26:43
obviously I'm trying to illustrate this
26:45
in a very comical way where like this is
26:48
not a metric definition that y'all would
26:49
probably come up with but you might have
26:51
come up with one that is like that
26:53
sounds smarter right so but that's the
26:56
idea is like it's the same thing like um
26:58
if you know if you're trying to look to
27:00
get married and uh like it's like oh I
27:03
need a man who's like over six foot tall
27:05
he may he makes over six six figures
27:08
right he lives in San Francisco he plays
27:10
guitar and like you know once you have
27:13
like five or six or seven criteria like
27:16
you're down to like five people right

27:18
and like um because that's just how like
27:21
Dimensions work right they become more
27:22
and more unique as you throw in more of
27:25
them and so like you want to be careful
27:27
when uh coming up with these metrics
27:29
especially with dimensional cuts
27:32
[Music]