Apache Spark Fundamentals

Day 3 Lab

Testing Apache Spark Jobs in CI/CD

Transcript:

41:44
for p spark so we're going to be starting off here at the data engineer handbook go boot camp materials three
41:50
spark fundamentals so this is where we're going to be having both spark running stuff and as well as the testing
41:55
of P spark stuff so in here we just want to make sure we clone this repo and then
42:00
we go into that spot here so then we can say pip 3 install dasr requirements.txt
42:06
and that's going to uh install everything and that includes if you look in the repo that's going to in include
42:13
chisa Pi test P spark and P spark SQL which is all we need to run our tests
42:18
and then if we want to make sure everything's running we say python 3-m piy test and then this is going to fire
42:25
uh the tests where we have we have three test tests in our repo if you see in our tests here we have test monthly user
42:31
sight hits test player SCD and test team vertex and uh you'll see all of them
42:36
passed and that means that we are ready for the lab so let me uh open up the terminal real quick so that y'all can
42:42
see like how this works um so I'm in the this spot here you'll see
42:50
I'm in the same I'm in this like directory here so I'm pretty sure if I type P test does this
42:58
work okay yeah it does it does the right thing so um if y'all don't have this all
43:05
set up yet but uh is it going to run okay there we go
43:11
why did that take 11 seconds that should have not taken 11 seconds but um anyways
43:17

uh let's kind of go over like how this works and then we we can I think it'll be more clear so we have a couple
43:25
different folders here that are going to cover all sorts of things so you'll see in jobs here jobs is where you're going
43:31
to have like literal like jobs right like you see this main
43:37
here where you have like this is like building the spark session and then it's creating a data frame output data frame
43:44
and then it's actually like inserting that data frame into a table right so this is like an actual like spark job
43:53
and so and so is this one this is also a spark job and so one of the things that you can do
44:00
to make spark jobs uh uh more testable
44:06
is you create another function outside of where you build and write that just
44:13
does the transformation logic because if you do that then what you can do is in
44:19
your test code you can then test just this just the transformation logic cuz
44:25
do we really like we don't probably need to test like if the spark session is created or if the data was written out
44:33
like obviously there can be like weird edge cases on the failure there but the main thing that we want to test like 99
44:40
times out of 100 is just does the logic work right is the transformation logic
44:45
doing what we expect it to do and so that's literally all like we're trying to do here with these jobs is that's why
44:52
that's why there's this abstraction right where we have this like do player STD transformation
44:58
and so the idea here is so that's jobs and you're going to need a like for the
45:04
homework we're going to be creating more of these jobs because we wrote a lot of queries uh over the last uh couple weeks
45:11
and we're going to be moving some of them into spark so then we can test them so then let's look at the tests folder

45:18
so first off let's look at uh this file called conf test so what this does is
45:26
this gives us spark um anywhere that spark is referenced in
45:33
a file right so like for example you see here how we have a test monthly sight
45:41
hits and then we have spark here but like you'll see spark is like nowhere
45:47
like but like this still runs right because if we do pie test you'll see this actually runs right and it actually
45:54
is doing the test monthly user sight hits actually test it it runs this code here so how it
46:00
works is that like in comp test we give it spark right here right and then this
46:05
is just going to be the one spark that is created for this for these runs that we create like for both the test player
46:13
SCD and for the um test monthly user site hits so this is kind of the idea of
46:21
like how we're going to be testing our data pipelines and you'll see like one
46:27
of the things I want to show you though is okay so we have these tests and you'll you're you'll see they're working
46:33
let's go ahead and go into the job real quick and what I'm going to do is I'm
46:38
going to just change one thing here what I'm going to do is I'm just going to put
46:45
like a I'm just going to throw in a plus one here just because like I know it's going to break the test but I'll show
46:52
you how this works so I'm just fiddling with this quer say I'm some like naive Junior data engineer and I'm like I'm
46:58
just going to throw a plus one in here because I'm like I don't know I'm I'm you know I like to mess around right and
47:04
you'll see like it's G to like this is going to run and then this is going to test to see hey this is actually see
47:12
there we go it fail right so it's like test see it says test SCD generation
47:18

failed and then it will even it'll even say why right so it says see it says uh
47:23
chisa data Frame comparer data frames not will air and then you'll see here
47:30
right how I have this I added the plus one here and this is like 2002 and it's
47:36
supposed to be 2001 and this one's 2004 and it's supposed to be 2003 so you can see that plus one like it shows every
47:43
single record that actually broke but like here's the question like what if that plus one was actually valid in that
47:50
case you would actually need to go through and edit the test so then you can enforce the New Logic that you came
47:56
up with with but the idea here is okay we have our test and it actually now if
48:03
we edit the SQL query if for some reason we change the logic then we know that the test will work so let's go over that
48:11
real quick and I think this will make more sense so I remember in the
48:17
presentation today there were questions around like hey like how do you generate
48:22
data for your tests and this is literally how we do it is we uh like see
48:28
how we have Source data here which is going to be these right and then what we
48:33
do is we create the data frame with spark spark create Source data frame and
48:39
then we do the transformation with the source data frame and Spark and then we
48:44
have our um expected data which is another data which is going to be another data frame that we want to
48:50
create and then that's when we then assert the equality of these but what we're going to do is we're going to get
48:55
another one of the SQL queries that we had from um a previous lecture and we're
49:01
going to create a new uh test so that y'all can see how to do this we're going
49:07
to create a new job and a new test so y'all can see how you can actually
49:13

create these tests yourself so I think in this case we're going to just probably do one of the um one of the
49:20
ones in here like uh what's a good one here um I like this
49:27
uh that one's too easy um what
49:34
about uh okay I think in this case let's go actually I think this this one is
49:40
probably good enough we we'll use we'll use this guy um so I'm going to move this query over into our um into our
49:50
jobs and I'm going to create uh we're going to create a job here we're going to create file this going to be called
49:55
team uh team vertex job. PI right then I'm just going to
50:01
create query we going to do like a triple quote here throw this guy in here um this query is not GNA quite work in
50:11
spark uh for a couple reasons uh one is
50:16
um vertex type this colon colon vertex type is not going to work uh you're
50:21
going to need to put type in uh ticks like this because type is more of
50:28
a spark is more strict about when you use keywords so we have that and then um
50:36
then in here this Json build object uh will be an interesting one I'm pretty sure we're going to need to change this
50:42
to struct and that will work but um we'll we'll we'll test some stuff out
50:47
here to see I'm pretty sure this is what we want so we have our um vertex job but
50:54
what we need to do is we're going to make a DE Main and Main is where we're actually going to run the code and in
51:00
this case I'm just going to copy a lot of this over real quick we're going to because a lot of this is just the same
51:06
so in this case oh yeah we need to import spark session grab this guy so we're going to
51:13

import spark session we have our query and you see this like do player STD transformation in this case we're going
51:19
to create def uh do team vertex uh
51:24
transformation and this is going to take in spark and uh it should take in spark
51:30
in like a data frame or something like that like one second let me see how like this working again oh yeah so it creates
51:35
spark and then a data frame yeah that makes sense so and then we're going to have our data frame
51:41
here and then we're going to say like return something right so uh in this
51:47
case uh what we want to do is create a view of this data frame here which and
51:53
then we're going to run the query right this spark query so in this case we need to create the view teams because that's
52:00
what we're going to be querying most of this on so what we can do here is we can say um so create or it's on the data
52:10
frame okay so this we say data frame. create or replace Temp View then we're
52:17
call this teams and then what we want to say is we're going to say spark. SQL and we're
52:23
going to put query we're say return and so this will then give us our query
52:30
and then the last thing we want to do here is pop that guy there and then that should give us what we are oh I forgot
52:37
you got to have the double enter otherwise pie charm yells at you so this
52:42
is close we might need to do something with struct here uh in a second but I
52:47
think for now we're pretty close to having what we need uh now what we need
52:54
to do is we need to create our test our test um object and then we can actually
53:02
the cool thing is is like when you create the test we'll learn if this struct syntax is right because I don't
53:08
know if it's right but we will find out so what we want to do is we're going to create a new uh file here we're going to
53:15

call this test team vertex uh job dop so
53:22
now we have our test team vertex drave so what we want to do is we're going to
53:28
uh kind of take a lot of this stuff here and uh move it into here but this one's
53:34
not what we want our import here is going to be from jobs uh Team vertex
53:39
jobs import do team vertex transformation and then we also want named tupal and then this will give us
53:46
some uh data frame comparison stuff that we want uh so these are our Imports and
53:52
then you'll see every function that we want to create create has to start with test otherwise um chespa and P or P test
54:01
won't pick it up and run it as a test so I'm just going to grab this guy real quick and we're going to say test we're
54:08
call this uh vertex uh generation so now what we need to do is
54:16
we need to create the input data and the
54:21
uh the we need to mock both of them but we got to come up with the schema first so in this case we're going to have um
54:29
probably two we have a team right we have a team schema and then we have like a team vertex schema so let's go I think
54:37
team vertex is obvious uh and obviously we need to put as properties here
54:42
because that's actually missing um so I think the team vertex schema is more
54:48
obvious so we're going to start with that and then we'll build the team schema in just a second so we're going
54:53
to say team vertex equals and then is like name Tuple team vertex and then
55:00
then in a list here in this second uh string you got to put the The Columns so
55:07
you have identifier type and uh
55:14
properties those that's the schema that we're looking for and then we're going to have a team here as as well this is

55:20
going to be a named tup be a team and then inside here what do we need let's
55:26
go back to the vertex job so this one's a little bit more gnarly so we have Team
55:33
ID uh so team ID abbreviation nickname City Arena year founded those are going
55:39
to be the ones that we have so we have uh Team ID oh there's no comma though
55:45
it's it's so weird abbreviation abbreviation nickname City
55:51
Arena nickname City Arena and year founded and it's like it's like
56:00
stupid okay so this is great so we now have our
56:07
two values here that is kind of like what we're expecting um so what we want
56:13
to do in this case is we now want to create
56:18
our um we we want to create oh this is vertex generation forgot the r there but
56:25
what we want to do is we want to create fake input data so we're going to say um input data is equal to array in this
56:32
case we want to create let's just create one team real quick we'll say the ID is one we'll say the abbreviation is
56:39
GSW and uh the nickname is uh gold gold or Warriors right Warriors city is uh
56:49
San Francisco arena is Chase Center and year founded on I will put
56:56
like 1,900 doesn't really matter that much so that is our input data or data right
57:05
and then let's create our expected output I think it will this should be
57:11
pretty straightforward here so we say team vertex and then we have our identifier in this case is just one
57:18
because we're just taking the team ID and then uh team is our uh type and then
57:26
properties right so a lot of times I like to actually put the
57:31
um the actual like equal signs on this right because oh this is identifier my
57:38

bad identifier type and then properties and
57:43
properties in this case is going to be like a map right it's going to be a big old map here where we have
57:49
abbreviation and this will be GSW then we have nickname
57:56
this is Warriors and then we have City San
58:02
Francisco we have arena is a Chase Center and then we have
58:08
year founded this is 1900 so this is what this is the
58:15
transformation we are expecting with given this input we expect this output and one of the things that I want to
58:22
also show you is you see how in this query we actually dup we have like
58:27
another row so what I want to do is I want to test that as well so what I'm going to do is I'm going to create a
58:33
second object here what I'm going to do is I'm just going to say this is like bad Warriors I'm just going to have but it's going to have the same ID so that
58:40
like we can know for sure that like we only have one because we we honestly genuinely expect a DD to happen here so
58:48
we have our input data so then what we want to do is I want to say spark or so I want to say um input uh data frame
58:54
it's equals spark. create data frame and then we say input data so now this is
58:59
going to be our input data frame to our you we have like do team vertex transformation in this case we're going
59:05
to say spark and input data frame and this is going to be our actual um actual
59:14
output and then we have our expected output and then I think there's just like the last thing there's only like
59:20
one more line of code we need to add here and that's going to be the uh
59:26
assert at the bottom here there's G like some sort of assert line here yeah this
59:32

guy oh we need to create one more data frame we got to we got to uh because you got to compare data frames to data
59:38
frames not data frames to array of name tupal so in this case we need to do one more we're going to say expected um DF
59:47
is equal to spark. create data frame expected output and then we're going to
59:54
call this actual DF just to keep everything keep our code nice and clean
1:00:02
so most likely this is not going to work on the first try if it does like I am like a freaking unit testing wizard but
1:00:08
we have enough of the code written now so that we can actually try running the test so if we go back here and we run Pi
1:00:16
test see now we have three items that are going to be running and then one of the things that kind of sucks about
1:00:22
spark unit tests is that they are kind of slow because okay see it barf
1:00:30
great let's see why Okay window function row number requires window to be ordered
1:00:37
please okay see this is a great example of where Spark and
1:00:43
um uh spark and postgress have different syntax in terms of row number so what
1:00:50
that means is we need to go back into the job and we got to put an order by here so we're going to say order by and
1:00:56
we can just throw in maybe also team ID CU it doesn't matter that much but like
1:01:01
uh cuz we're just trying to get the dupes out but you'll see like spark
1:01:06
doesn't like that right so now that should fix that problem but one of the things I'm hoping y'all see is that like
1:01:12
I'm actually running spark code right now pretty quickly I'm not deploying it I'm not submitting it but there we go
1:01:20
now we have like a we have some sort of crazy thing going on here that is uh
1:01:27
what is going on here cuz we have like whoa dude okay schemas not equal so
1:01:35

we have uh because struct one identifier type properties oh because this is
1:01:42
saying um oh this is
1:01:47
like because it's not a map right it's actually another tupal that's the
1:01:54
problem here because it's it's a struct right so spark is treating this like a
1:02:00
struct not like a map so there's kind of two ways to go here right so right now
1:02:06
we're comparing um a struct to a map because you'll see like in my job right
1:02:12
I'm assuming that it's going to be uh you see in properties here I'm saying it's a map uh so there's two ways to go
1:02:19
here either we change this to um like my
1:02:24
my my preference here is actually map is probably the right call here but then we just need everything to be the same data
1:02:30
type so I'm going to change this to 1900 just have them all be string but then
1:02:36
the problem here is we got to do it on the query side as well so in here this
1:02:43
is no longer struct this is now map and then the last thing we got to do here is
1:02:48
we got to cast and then we cast this as a string I think string works here we'll
1:02:55
see the idea here is we're trying to iterate and we're doing test driven development right now and I'm hoping
1:03:02
y'all see the value here in that like we can actually both test our code and make
1:03:08
sure that it's like running a lot of times okay this is look this is looking a lot closer though this is looking a
1:03:13
lot a lot closer so what do we got here we have uh okay so these are the two
1:03:20
types right so we have identifier long type struct properties type because then
1:03:27
this is saying map string string true and then this is saying false what what is that last one I think
1:03:34
that's like nullable or something like that is it because so it's saying that the types are not equal here because of

1:03:42
some funky reason like hold up struct Field properties
1:03:49
because stru field identifier stru field string stru Field properties map type so
1:03:55
it seems like it's this last like false nullable sort of property here that is
1:04:00
making this uh happen so if we go back into here this should be uh this should be right now actually
1:04:09
hold up let's see here because we have nullable here so that's what I'm pretty
1:04:14
sure this last map type is I want to I want to look at this map type real quick and just bust out uh map type
1:04:24
ppar CU I I want to say that it has like there's like nullable it's like some
1:04:30
sort of weird nullable problem that is happening here um because now they they are like the
1:04:37
same I want to say because you have uh there I wonder if chiso might have another comparison operator as well that
1:04:44
we might want to use instead and that's what's going on but so interesting so
1:04:49
this is a great example of where like when you're creating a lot of these tests like how you can kind of run into
1:04:55
some roadblocks of like oh I need to be able to recreate these two and you see how they're so close right it's like all
1:05:02
the way this is so it type here is AI type seems to also have the same uh
1:05:09
thing here where in S1 it's false and then in S2 it's true um because why is
1:05:16
this like not going like is my internet like freaking
1:05:23
broken like that's so weird because I'm talking to y'all but the internet's just like freaking so slow so anyways what
1:05:30
I'm trying to say here is we have probably another um there's going to be
1:05:36
another data frame equality thing okay there we go see that's what we need you
1:05:42
see ignore nullable there I'm so happy see that sometimes you just don't Google stuff
1:05:48

right and you see this ignore nullable we want to change this to truth nor n we want to say this is truth
1:05:56
so now if we run this we should get a
1:06:10
pass okay a lot closer though a lot a lot closer because
1:06:16
now we have uh now because now you're seeing uh it's not yelling at the um you
1:06:22
see now it's saying the data frames are the right they are the same types but now we have we're having a different
1:06:27
issue where it's saying that the rows are actually not the
1:06:33
same year founded oh I have a UND see because I actually like what is oh it's
1:06:40
because in the job here uh because yeah oh interesting okay well I'm going to
1:06:45
put I'm G to put the underscore in there then dude because like why have it be bad when okay yeah it's like it's a key
1:06:52
problem so this is a great example though of like how this really gets you
1:06:57
into the nitty-gritty of uh what you would expect right so now uh this code
1:07:04
is running how we would expect it to run and it's giving us the values that we're
1:07:10
expecting obviously we're not covering all of the cases of things cuz uh
1:07:16
obviously there's uh but like this is covering the duplicate problem right we are no we do not you see how we are
1:07:23
filtering out bad Warriors we don't expect that duplicate to be there and then we have our new record here that is
1:07:31
showing up most of the time here we probably want to include more uh data
1:07:39
right like maybe a couple more rows one of the things about this transformation is that it is uh kind of simple and it's
1:07:47
not going to uh require quite as much like and I'm hoping one of the things
1:07:52
that y'all try uh in the homework is there's going to be some of these
1:07:57

queries especially if you're going to do some of the uh like slowly changing Dimension ones like there should be some

1:08:04

of these oh the slowly changing Dimension one's fine but like if you're doing like the cumulative one where like you actually have to do a join between

1:08:11

two data sets then you have to have two input data frames instead of one input data frame and that can make these

1:08:18

problems trickier because then it can help you handle cases of like oh what if this side's null or what if this side's

1:08:24

null what if there's not match right and all the like the left join right join logic um awesome well so this is

1:08:31

essentially what I had for the lab today congrats on making it to the end of the unit testing and integration testing lab

1:08:37

make sure to like comment and subscribe and share this with your friends who are interested in learning about data engineering I'm really excited for you

1:08:43

to check out the rest of these lectures make sure to check out the ones on spark and Flink and data modeling there's a

1:08:49

lot of really powerful other lectures in this data engineering boot camp series I'm excited to see you there