Dimensional Data Modeling

Dimensional Data Modeling Day 3 Lab
Data Modeling - Building an NBA Player Network Graph - Day 3 Lab

Transcript:

0:00
[Music] welcome to dimensional data modeling day
0:07
three lab in today's lab we are going to be doing a Hands-On exercise where we are going to be building a graph data
0:13
model to see which NBA players play with each other in games and which NBA players are a part of which teams at
0:19
which time so we will be building very data agnostic thing in postgress that will allow us to analyze these complex
0:27
relationships between players uh to be ready for this last make sure you have Docker installed make sure you have the
0:32
repo in the description below cloned and that you have Docker up and running and you can connect to postgress with a
0:38
visualizer something like data grip or DB visualizer or D Beaver one of those
0:44
SQL editor tools is going to be super important to get you to where you want to go here you could use PG admin too I
0:50
guess but I hope you enjoy the lab today and if you want to do more of these Hands-On exercises with hot Technologies
0:55
like Iceberg and trino definitely check out the data expert Academy in the description below and I hope to see you
1:01
there so in the presentation right we talked a lot about vertexes and edges so
1:07
that that's what we're going to do is we're going to do a create table here we're going to call this um call this
1:12
vertices right because that's the actually the correct name here so then we're going to have identifier that's a

1:18
text and then you have a type which is going to be a Vertex type oh I already
1:24
have vertex type in here okay well I need to drop vertex type then because I
1:30
want to make a new one can you drop it like
1:37
that okay fine got to do I'll do it this way so I can start over Okay awesome so
1:45
now that's dropped okay so what we need to do is like you'll see we're going to have a type here that's a Vertex type
1:50
that is going to be an enumeration obviously it's red right now
1:56
but that's fine but like the last column here is going to be properties and like the thing that's dumb about postgress is
2:02
it actually doesn't have um a map type uh so we're going to be using uh
2:09
properties as a Json type which is essentially the same thing uh it's a little bit more flexible than a map but
2:15
not that much more flexible uh anyways uh we have our um verticy table here and
2:21
uh the primary key verticy table very easy identifier and type right makes
2:26
sense uh so we need to do a create typ type here vertex uh vertex type as enum
2:34
and then uh our enum values here uh are going to be player
2:40
team um I kind of want to put game in here as well let's put game in here as well I
2:47
think game will work because we can have like an entity which is a game because that's in our uh we have a games as a as
2:54
a table here so I'm I think I'm good I think I'm good with that I need to I
3:00
need to refresh this this is actually off right because there should be okay
3:06
well that's fine um so we can go ahead and create this type here makes sense
3:11
and then uh because we're there might be Arena but AR I I when I I remember when

3:17
I was doing this in the rehearsal I realized Arena and team are one to one because a team only has one Arena that
3:23
they play in so it's like that's a dumb thing to add like you might as well just put team so in this case we have our
3:31
vertex type created now we can create our verticy table vertices already exist
3:37
okay well we need to I need to drop that table that's wrong I have all this like
3:42
dumb data in my freaking database right now that's not good okay so now we can go ahead and create this table great we
3:50
have our new vertices table um so let's get all of our ddls out of the way real
3:56
quick so what I want to do is I want to say I'm going to drop drop type Edge type as well and we need to do a Cascade
4:02
here we need to get rid of this guy and we can say drop table edges y'all
4:08
probably don't need to do this because you don't have these tables but um let's go ahead and say we're going to create a
4:15
type say Edge type as enum and in this case uh we're going to do a couple here
4:22
right um so we got to think about the the the relationships here right so I
4:28
think we're going to say um plays against right is probably one um maybe
4:35
shares team or something like that like because there's like because you have plays against is like player versus player but
4:43
you also have like if they're on the same team right where it's like you have like Michael Jordan and Scotti Pippen or something like that that's why I like
4:49
shar's team but like you could almost you could do that through the team way
4:56
as well but that doesn't cover the case where what if two players are on the
5:03
same team but they're on the same team in they're on different teams right now that is where things get like a little

5:09
bit dicey so I think that I'm going to keep shar's team as that's like a that'll be a player connected to a
5:16
player but they're on the same team and this is player connected to a player but they're on a different team and then
5:21
we're going to have um we'll say maybe uh plays on um plays we'll say plays in
5:29
and plays on so in this case a player plays in a game and a player plays on a team I
5:37
think that that's pretty pretty good I like I think I like that as our kind of
5:42
edge types that we can work with here um okay so let's go ahead and create uh
5:48
this Edge type here um great so now we have another uh
5:55
table here say create table this case this is going to be edges
6:00
and then uh remember we go subject identifier text and we have subject type
6:07
this is a vertex type then we have object identifier
6:12
text and then we have object type which is a vertex type and then we have Edge
6:18
type which is an edge type I need a comma Edge type and then we have
6:25
properties which is a Json all right so one of the things the great debate about
6:31
this is like what is the freaking um primary key of this edges table like and
6:37
the primary key of this table is actually um and it's really nasty it's
6:42
actually a subject identifier sub it's like essentially all the columns right except for properties subject type
6:50
object identifier object type Edge type that's the um this is the primary key
6:57
you have all of those together some some um graph places they actually have
7:02
another column here called like um Edge Edge ID and they put like a text here so that you have like a surrogate key I

7:10
think for our intents and purposes we don't want to do that because we have freaking 45 minutes to do this and I'm
7:15
this is more like a coding interview exercise than building out an actual graph so this probably makes sense so
7:22
far um so what we want to do here is I want to look at uh game details real
7:29
quick quick so we're going to I what I want to see is cuz game details has uh
7:34
the actual game information where it has like the player and the team ID all that kind of stuff but we also have um uh the
7:42
games right so I think games might be game is probably the first one that we
7:48
want to add because I think that one will probably be the easiest to add um
7:54
so let's go ahead and create game as a um
8:00
as a Vertex type so in this case uh I'm gonna so this is like the ddl one I'm
8:06
just going to make a new sketch pad here so what we're going to do is we're going to say um select star from games and
8:14
obviously this is already duped right so we don't need to aggregate at all we
8:20
just need to essentially uh move this over into our um into our other values
8:28
like do we have teams here what's in teams does teams have an ID okay good teams does have a team ID
8:36
here so we'll we we can you do use that to join um okay this this should work great then so in this case we have games
8:44
um and what we want in here is this is going to be our verticy right but it's
8:51
like this is where it's interesting is is a is this a Vertex or is this an edge
8:56
that's another thing that I think is an interesting uh thing but I I think of it as it is mostly a vertex in this case so
9:03

uh the identifier here we're going to say game ID as
9:08
identifier and then the type right game and we want to do colon colon vertex
9:13
type as type and then uh we have uh so
9:20
this is this cool thing called Json build object uh so this is essentially where we can get our
9:27
um our like values here right so in this case we
9:32
have I I like a couple of these I like points home and then there's points
9:38
home and points away points away right and then it's
9:44
like you want to see who is the winning team I I think the winner here is also a
9:50
good idea to see like winning team and losing team and I mean the rest of this
9:57
like we could put in there but like I I mean I don't necessarily like it so it looks like there's actually this column
10:03
at the end here home team wins right so we can say um winning team and this is
10:09
going to be uh case when home team wins equals one then home
10:16
team ID else visitor team ID end right so let's look at when this is
10:24
as properties so this is probably going to be a good enough like sample for for us
10:30
so um you'll see uh we now have the winning team and we have all that stuff
10:37
right um I I like I like this as our kind of edges because we can then um see
10:44
how these connect because we also have a teams participate in games as well right
10:49
but this is the main idea that I think like is this is probably good enough for us to create our first vertices right so
10:58
we can say insert into vertices and this will give us our uh okay good Done Right
11:06
games done easy so one of the things that I also want to help you guys understand is that like this is actually

11:12
not too crazy so awesome um I think for
11:17
uh player player is going to be a little bit trickier but we can uh essentially like I actually um so I know that like I
11:25
actually pulled some of this data from different data sets so we have player seasons as um one table but this is different I
11:33
actually pulled these this is like a mix of a couple so I actually want to get player from Game details because you see
11:38
how we have this player name here as a column so what I want to do is like we
11:44
can essentially use this uh we can aggregate this one up and that will give
11:49
us uh our player details because then we can get like their entire career stats
11:55
like their Min and Max Games all sorts of stuff like that so um uh let's go
12:02
ahead and uh one second let me let's do this so we can say uh select star from
12:07
game details and in this case we want to say uh we'll say player ID and this is
12:13
going to be as um uh identifier right and then we're going to
12:19
have a player vertex type as type and then uh we need a group buy
12:26
here actually so let's get rid of this first let's we need a this one's going to a little bit more complicated so what we want to do here is we want to say
12:33
count um count one as number of games right U maybe
12:41
[Music] sum uh points as total
12:48
points uh what other things maybe uh I like um
12:59
oh yeah because we also want um we can get the the teams the we can get that connection here as well probably so
13:06
maybe array a uh Team ID but in this case you want to put distinct here this
13:12

is as um teams played on or we would say teams we'll call it teams so this case
13:19
if we say a group by here say Group by player
13:25
ID okay so you'll see here how then you have okay this person played on three
13:31
teams and you have like all their points and all the connections right of everything here so this is a pretty good
13:40
uh like I like this as our kind of full query here so I'm going to say with
13:45
players a we probably want name as well though right you see how like we have
13:51
identifier but we also probably want name right so we're going to say max player name as player name this is
13:58
probably like a very strange thing to see here this Max but like you can put Min as well right it doesn't matter
14:03
because it's always the same value we just need one of them and that gives us because we want the name of the player
14:09
as well because this ID is kind of worthless um so what we can do here is we can say with players a as then this
14:17
is going to give us our aggregated view and then what we can say is we can just
14:23
make this pretty easy on us we can say select identifier and then we have play er um
14:30
vertex type and then we got to do that Json thing right Json build object in
14:36
this case we have player name player name then we have
14:43
uh number of games number of games let's put from
14:51
players EG here and then kind of just make this query a little bit
14:58
nicer so then we have uh total
15:03
points total points and then we have uh teams and then we have uh
15:09
teams and so this should work and it should put like that teams

15:16
into like a little array okay there we go so now we have
15:21
our see we have our player vertex type and then in here we have all sorts of
15:27
interesting like teams and stuff like that so this is pretty good I like this
15:32
as our um next thing so we can say insert into
15:39
vertices oh is it because oh you need a colon here or semicolon
15:44
here this will okay so that ran that ran great now
15:50
we have our so let's get the last bit here right because if you remember we had the three we had games players and
15:57
teams so teams should be easy teams is going to be easy like the last one so we
16:03
can say like select star from teams let's just see like if there's any other um like maybe metadata we might want to
16:09
pull in okay so maybe like the arena I like Arena and
16:17
City and then obviously this is the identifier but then we have to bring in like the nickname and stuff like that as
16:23
well so in this case we're going to have a team ID as identifier and then we have
16:31
Team vertex type as type and then uh Json build object and
16:38
then in this case we have uh maybe abbreviation
16:43
abbreviation a nickname nickname uh city
16:51
city [Music] Arena
16:57
Arena um I'll put year founded in there too but I'm putting an underscore in
17:02
there because I hate that that doesn't have an underscore there should be an underscore there um uh I don't care
17:09

about the rest of these columns I don't think they freaking matter so uh this is probably good enough for teams so you
17:16
can say um insert into um vertices and we just got to put another
17:22
semicolon here then this will give us our team
17:29
wait there's dupes yeah yeah I'll share this code just um I just like what the hell
17:35
there's like there's dupes in this why are there
17:41
dupes did this like this is like not there's like three of everything in
17:48
here okay fine we will just uh I I think I I when I loaded this data in postrest
17:55
it's like broken like you see how there's like triplets it's like like all of it's the same but there's like three of everything I mean that's fine all we
18:02
got to do right is I think if we just put like group by um oh wait we got to put that group
18:09
by first here but like okay so if we put oh this is going to be actually super
18:14
annoying but um City Arena oh wait no no no no I know what to do I know what to
18:20
do this is going to be where you just put you do a CTE right so you say like um with teams duped as you say like uh
18:30
select star from um teams and then you want to put a row number in here right
18:36
so we can say row number um over
18:41
Partition by team ID as R num right and
18:46
then what we can just do is we take team duped here and we say where teams oh no
18:54
we change this to teams D duped and we say where R num equals one and now this will work cuz apparently there's
19:01
duplicates in there I don't know what happened there like there there there shouldn't be but there is so now this
19:07
will work this will run so um we now are essentially we have the all of our data

19:14
for the most part now at least the very first bit of this we have all of our um
19:20
we have all of our vertices in our table now let's um let's go ahead and just like look at some of those real quick so
19:25
we can kind of I I want to I just want to illustrate some things to yall to see like how this works so we can say select type from vertices right and then we
19:33
let's just do a count real quick so we say like count one group by one right so
19:38
this will give us our okay so in this case we have um 30 teams 15 almost 1500
19:46
players and 9,300 games uh that's pretty cool um so what
19:52
we want to do right is if you remember we want to start looking at some of these edges right so um let's I think
20:00
the easiest one here to start with is going to be this um plays in I think
20:06
that's probably going to be the easiest Edge to start with uh because that's
20:12
just game details it's already at the right grain for that uh I think that
20:17
that's probably going to be the because some of these other ones we have to actually aggregate uh and play plays
20:23
against and shares team those ones are going to be those ones those ones are the nastiest of but we will uh we're
20:29
going to start here with uh this plays in so in that case right so we're going
20:35
to start adding to the edges table a little bit so we're going to say select star from game details and in this case
20:41
right we have um we have all the stuff right and then we need to get our
20:46
subject identifier subject type right so in this case we have a player ID as
20:52
subject identifier and then subject type here is player
20:59
castes to a Vertex type as subject type and then we have object identifier so in

21:05
this case object identifier is game ID and then we have object type which is
21:13
a game right and then we have our Edge
21:18
type right so Edge type in this case is going to be plays in is Edge type and this is as Edge
21:26
type and then we have our properties right so then we have uh this one's kind of cool so we say Json build object and
21:33
then our properties in this case are what about this so this is a player in a
21:39
game right so in this case we have their start position
21:45
right which is uh like Center forward
21:50
whatever um I also like the points right there's going to be points probably right there's going to
21:56
be yeah points is at the end here um anything else that I really care
22:04
about uh I like the team ID I think like because you can put te but team ID is really more of like an edge right
22:10
because you can like but because that's going to be in this game but no but you actually have to have in this game this
22:17
player because there's all like all three of those together so we're going to have um Team ID team ID right and
22:23
then probably um Team abbreviation
22:30
so this is going to be um as properties right so this is pretty close I think
22:36
that this is pretty much what we're looking for in terms of our like getting everything into the right schema so uh
22:43
what we can do here is we can say um insert into
22:49
edges and oh there's a dupe
22:58
okay why is there a dupe there um that's interesting there's a there's
23:03

some there's like a data quality issue or something with uh the game details apparently because there's like a
23:09
there's a there's a player who has two records in here so what I want to do is
23:15
I want to look at that record real quick so we're going to say select star from game details where player ID
23:26
equals this and game ID
23:32
equals this like why is there two records here that's
23:38
weird okay there's not two records there's freaking six what is there is there six records
23:46
of everything though one second I got to see this now I say player ID what's game
23:52
ID count one do I like mess up here as well is there freaking like a lot more data in here than there's supposed to be
24:03
like there's always okay so it looks like there's always three essentially okay there's
24:11
like I so when I was like importing some of this data in the postgress like it like duplicated again so uh we have the
24:17
same problem here as we did in the um in that last example right where we need to uh essentially uh put in that R num
24:25
right so we're just going to say DD here as say uh select star from game details
24:32
and then in this case we can just throw in that row number and then uh the Partition by here
24:37
is a little different right we're going to have player ID game ID as R num and
24:44
then we can just essentially use this instead of uh that I'm going to say where row
24:51
num equals 1 the cool thing about this though is that like even if there is no dupes this query still runs so this
24:58
gives us are uh kind of edges and we need to actually uh essentially I'm going to work with we're going to need
25:03

to work with this dded essentially the whole time B right this is doesn't do like multiple degrees but this is what
25:10
this ends up doing where you get your um properties plays in and you this is a way to and then you can aggregate these
25:17
things right so imagine like what we can say here is you see how we in here we get points right we get points so in
25:25
this case what we can say is we can say e do properties and then we can do points I don't know
25:32
if y'all have seen this before but this is uh so then we can say uh average or we can say Max say Max here and then
25:40
what we want to do is we're going to say um v. identifier or I think it's even easier v. properties again and we can
25:46
say player name and then Max of that and I'm just
25:51
going to put Group by one here so this is going to give us uh there we go so
25:58
now this is every player and um we can put an order by in here too we say order
26:04
by two descending so this should Kobe Bryant should be number one here right no oh nolles it's cuz oh
26:14
these guys it's because these guys just never never played why is it always
26:20
nine that seems incorrect I must have like when I made the edge this is not
26:27
right whatever whatever's in points here is not right because nine like how is it all nine oh oh oh oh I know why I know
26:37
why okay so here's here's this is a great this is a great example of how
26:42
working with graph data is a problem right because what I'm trying to see here is like what's the most points that each player played in a game but the
26:48
problem here is it's actually treating this as a string right because it's a
26:54
Json so if we change this we say cast as integer this will not
27:00

work there we go that was the problem and then oh null is still first
27:05
for some reason but uh then you have uh Devin Booker I guess Kobe Bryant is not is is his his 80 point game isn't in the
27:13
data set apparently but okay but he's up here he's got he's got a 60o game in here so uh this is how uh I would
27:22
imagine that like you see how like we can get new kind of relationships here
27:27
that this one isn't that interesting because we could just aggregated game details
27:33
right and that is fine and that is like that's why I want to like show you other
27:39
uh kind of connections here besides just plays in so this is just our first kind of way of going about it so now what
27:45
we're going to do is we're going to do one that's a lot more complicated so what we're going to do here is we're
27:51
going to create uh um some more data here let's let's go down here and just
27:58
going to here I'll I'll paste this to y'all too if you want to see this query like it's like a it's more of like an analytical one that's like not as like
28:05
critical to uh like to the rest of it uh yeah go ahead
28:22
seanm uh yeah I mean uh oh the the oh this data model right and everything and
28:28
this like yeah I mean there's going to be other options there like there's this thing called post graph right I mean the
28:35
that visualization of the graph and everything like uh is kind of outside the scope of this uh presentation I'm
28:41
I'm this is more of like very narrowly focused on the model itself right uh but
28:47
like yeah that that'd be a cool thing to that'd be a fancy obviously that's like a an in a later version if that's like
28:53
an easy thing to slap on right then to give people the wow factor for this that's definitely something I would want

28:59
to add so um yeah for sure that like uh but like yeah like I I don't know of any
29:05
like easy ways to do that like just like out the box there probably is there probably is I just didn't do any
29:10
research on it so um what we want to do next here is we want to create
29:17
uh uh well I'm going to call one called here is filtered this is where like I
29:22
wish qualify existed in freaking postgress but it doesn't like is where
29:28
um where you have to like have like two CTE here right essentially so we're going to say d duped where R num equals
29:35
one or Rum equals one so now this is just like we just want to do this so that we have our DD data set so what I
29:44
want to do now is I want to create an edge that is where we have plays
29:50
against um between two players right and there's going to be some interesting
29:56
trade-offs here because of the fact that that like we're going to this is actually going to create two edges uh
30:02
where you have one on either side because we're going to be doing what's called a self join here right so if we
30:07
say select star from filtered F right let's just query this real
30:15
quick so you'll see okay we have our games all duped looking good but what we
30:21
want to do here is we're going to do a self joint so we're going to say select star from filtered F joint join filtered
30:28
or we're gonna call this F1 and F2 um so in this case we're gonna join
30:35
these uh and then our on here is going to say f1. game ID equals f2. game ID
30:41
right and um f1. uh player name does not equal f2.
30:50
player name so this will give us uh so let's look at this real quick so if we

30:56
say um F1 player name f2. player name and we want to say f1. team
31:04
abbreviation f2. team abbreviation just going to put that in here for now and uh
31:10
you'll see how this kind of this query kind of works okay so you'll see we have
31:17
our um we're going to have both here right where this is going to be a game where
31:23
um so this is Kyle Lowry and Steph Curry right but this is like a Toronto Raptors
31:28
versus Golden State Warriors game right but they these are people who are playing against each other but then
31:33
here's Kyle Lowry on the same team as this person here right so what we can do
31:39
is we can actually generate both types of edges in one query here around um uh
31:47
the what try to say here around like both the plays the teams with like the
31:53
team one and also the the other side the the one around um uh playing against
32:01
right so you have on the same team and playing against here so in this case we have uh all these things together so
32:07
what I'd say here is I say case when f1. abbreviation equals f2.
32:13
abbreviation then in that case we have uh teams
32:20
with right and this is an edge type and we have an else here and the else here
32:26
is plays against and this is an edge type and this is an
32:32
end right so now this is going to give us uh we need to put the player IDs in here as well though because we need
32:38
those identifiers right we got put player ID and then we can say um f2.
32:43
player ID we need all of that so then we have all of this right and then this is
32:51
uh oh is it not teams with what did I what did I call it shares team I called
32:57

it shares team that's the problem so see this is why you use enums though you see how like if we just used a string here
33:03
then it would have just been bogus right and we wouldn't have the that consistency this is just a great example
33:08
of how enums catch data quality issues right teams with is it oh it's oh wait I
33:14
messed up that that is this one's plays against and this is shares
33:20
team that's the problem okay so now you'll see how we have uh we got our player IDs and then we have play plays
33:28
against and shares team so now the thing is is like this is going to create an
33:33
edge per game and we don't want that we actually want uh an aggregation of all
33:40
of this right that's the main thing that we want here is we want uh an aggregation of both sides of this so
33:48
we're going to have uh let's go ahead and create that aggregation so in this
33:53
case what things do we want we probably want uh we we probably want points and
33:59
games right so we can say count one as numb games and we proba have some um f1.
34:05
points as I'm going to call this like left points because I like you have the player on the left and the player on the
34:12
right side of the join right and uh we need both right f2. points as um right
34:19
points I I I don't know the better name to call that but so now this this query
34:25
we need to group by here right we're going Group by it's going to be like one two three four five don't do this in
34:33
production if you put this nasty stuff in production like I I I'll I'll no I'm
34:38
I I I you know it's not do as I say but do as I do right okay um now this query
34:47
should we run all of this we should
34:52

have okay there we go so now we have okay we have Anderson this guy versus
34:58
this guy they've played plays against oh did that not that didn't run like
35:04
because it didn't give us the rest of those columns can I like stomp this okay let me why didn't that not run oh oh
35:11
it's because it's still running there we go okay it was it's because this is actually a lot of data right okay so
35:16
there we go so now we see uh we have the number of times okay so you'll see that
35:23
um Tim Duncan and Tony Parker have played Four 42 games together right and
35:29
then you can see how much the left points got and the right points got so um there's going to be a lot of
35:37
combinations here there's going to be and one of the things that's a problem here is going to be the fact that uh
35:44
there's going to be two edges here you have the Tim Duncan Tony Parker Edge but you also have the Tony Parker Tim Duncan
35:51
Edge and you really don't need both right that's just like a duplicate Edge right it's just like it can be on either
35:57
side it all depends on like how you want to do your graph modeling so sometimes you want both edges so that like the query
36:03
patterns are a little bit easier cuz then you can always query on subject and that works fine but uh on the other side
36:11
like if you just twox your edges cuz this because this is a great example of
36:16
where uh this is a two-sided uh connection right and who is the subject
36:22
and who is the object here doesn't matter right because they are playing
36:27
with each other right it's like Tim plays with Tony and Tony plays with Tim right they like are like it's it's it's
36:33
just a double-sided Edge so in that case what you can do uh to like make sure

36:40
that like you get like just one set of edges which is what we're going to do is we're going to say where f1. player name
36:47
is greater than f2. player name which is a string comparison but this makes it so
36:53
that like uh like whichever one is sorted that way like this this makes it
36:58
so we don't have the double edges right we like we will only have a single amount of edges here so this query is
37:06
getting pretty close now so let's uh let's go ahead and put this in like a
37:11
one more CTE and then I'm going to paste this to y'all this is going to be our
37:16
uh um this is our kind of aggregated CTE so
37:22
in here we can say like select star from aggregated and this is going to give us our
37:29
uh oh yeah this is I forget we can't just like I need to like like actually
37:36
uh this this query takes a like just it takes a little bit of time right so um in this case we want to now build out
37:43
our identifiers and stuff like that so in this case uh I'm going to call this
37:48
uh as left player ID and this is as left player name and this is as uh right
37:56
player ID and this is as right player name uh just so we can have left and
38:02
right we have we keep everything in there and this is as Edge type here so I think we're pretty much here
38:11
we got like okay hold up I don't want to use left and right here because that's freaking inconsistent we want to call
38:16
this just subject we're just going to be uh canonical here subject and object so
38:24
yeah this okay so we have our subject I play ID object player ID okay we are getting real close here we we'll we are
38:31

okay so we have a subject player ID and we're going to call this as subject identifier and then we have um subject
38:40
oh no then we have subject type right in this case we have player so we have uh vertex type this is as a subject type
38:47
here and then we have object player ID as object identifier we're getting real
38:53
close here then we have player again as vertex type as a Vertex type and then we
39:00
have our Edge type which is just Edge type as Edge type right and then oh this
39:07
is my bad object type here and then we have the last one which is our properties where we got to do the Json
39:13
build object and then in this case we have um numb
39:22
games num games then we have a subject point
39:29
subject points object points object points I think that gives
39:39
us uh yeah that's everything so now this whole thing we can just say like insert
39:46
into edges right and this should
39:51
work this query takes like 10 seconds though because it's like we're finally
39:56
actually having a query that's like big data or at least a little bit of Big
40:03
Data interesting so there is it's saying that there's a duplicate key in here
40:09
where you have player player object cuz cuz we Group
40:17
by that because it should be where they're different and then we have our player ID player name
40:29
oh oh oh oh oh I bet okay I know what's going on here the problem here is that
40:36
these player IDs or that these names are actually not uh so someone could have
40:42
the same ID but a different name and that's that's causing issues so in this case this actually needs to be
40:48

aggregated right so we just pick one of their names because they might like maybe he change maybe like Dennis Rodman
40:53
changed his name to something else at some point and then you got to take these out of the group by so that we
40:59
only group on the identifiers and the edge type that should now
41:06
work so this query is going to be very powerful oh look at that it worked oh yeah oh yeah okay so now um now I just
41:14
want to like kind of like we only have like a couple more minutes left in this lab but I'm going to show youall like what this actually did right so now like
41:21
if we look at this like we have like um this is like the plays in Edge right but
41:27
what we want to do is uh let's like put
41:33
our vertices V join edges e right on v.
41:40
identifier equals e. subject identifier and V
41:47
do type equals e do subject type so um this uh and then what we can do here is
41:54
we want to say where um e do object type equals
42:01
player this query should give us there we go so now we have our um our
42:08
properties and then see this gives us our number of games right so one of the
42:14
things that we can do with this though right is this is going to be pretty cool I think this is going to blow y'all's
42:20
mind a little bit so let's go ahead and get v. properties uh player name you'll
42:26
see in here we have v. properties and then in this case we can say player name which is what we probably want that's
42:32
the main thing but then we also want v. properties and then in this case we have um there's like number of games and
42:39
total points right so in this case we're going to put number of games and uh I'm

42:44
going to cast this cast as um this is going to be as real
42:50
just going to cast this as real and then we're going to cast the total points as real as well so or or we can
42:57
yeah we can cast oh wait this is yeah we need another one like that then a division here right what why
43:06
is it mad that unresolved symbol V doesn't it isn't that the cast cast oh
43:14
no it's because there we go that's why okay um so now like I just want to show
43:19
what this is doing real quick because this is just going to give us oh yeah we get the division by zero right case when
43:26
uh equals zero then one else got to
43:33
just add a little block so that we get the actual um this gives us like points
43:40
per game right so you see this is actually going to give us our uh points per game uh number this is their career
43:46
points per game but the thing is is we also have um in our e do properties
43:53
right we have subject points and uh this is going to give us um and
44:00
then we also have numb games right so we have eproperties and we have numb games and this will give us let kind of just show
44:07
you what what this does right so now we and oh yeah we have one we have to get the last one in here we have eot
44:13
properties in this case we have a um object name right or what what did I end
44:19
up calling it oh I didn't even it's oh it's object identifier that's what it is just it's easier e do object identifier
44:29
and then okay so this will give us like an idea right so what happens here is this
44:37
is going to be uh when this player on average when this player plays against
44:44

this player this is their career average right but this is actually the um the
44:51
total number of points that they have right uh divided by games right so you can get the get that division and you
44:57
can see like oh this player actually is he's playing better than average or worse than average right you can kind of
45:03
see like oh like who when he plays with these people or against these people
45:08
like what is the better kind of approach right and you can get all sorts of comparisons of like Global averages
45:14
versus like playing with this person's average and you can kind of like build out like all sorts of different things
45:20
like oh how does Michael Jordan play with Scotti Pippen or like all sorts of different kind of connections with all
45:25
these different Edge types right right so you can do all sorts of really power you see this but you see how this query
45:31
though in comparison to the last query you see this query is instant right because we aggregated all like the big
45:37
data and so like it's a lot better it's a lot lot better like um now that like it's actually aggregated up right and so
45:45
uh that's another thing that you know y'all can check out but that's that's essentially where I wanted to you know
45:52
end the lab today is at that point just to kind of show y'all how all of these kind of subjects and objects and
45:58
vertices kind of come together and you can really do really powerful analyses
46:04
quickly where you don't have to crunch that like Rowl data anymore you can actually Crunch at this higher level so
46:11
[Music]

English (auto-generated)

All

From Data with Zach

Computer programming

Related

For you

Recently uploaded

Watched

Learning