can you write a SQL query that shows the total amount of successful posts per user type in the last month for those of you who aren't already familiar with exponent exponent helps you get your dream Tech Career with our online courses expert coaching peer-to-peer mock interviewing platform and the interview question database check it out at try exponent.com [Music] so basically let's suppose that you work as a data analyst for a company that is a social media company and they're actually invested in growth hacking right now so one of the key features that they really care about is uh how frequently people post and also um how how how frequently successful those posts are because they really want to increase the usage of the post feature as a proxy of Engagement for their platform so this following series of SQL questions is going to cover the descriptive analysis to identify both the crimp Baseline for how frequently and people are posting and also how successful those posts are and also um new ways to increase the post engagement level okay so uh let's kick things off first by looking at the data tables that we have available to us so um I see we have this up on the screen already so the first table is the um there's the post table it's actually on the right side so each post is uniquely identified by post ID there's a date there's the user ID so that's the the ID of the user who made the post there's the interface so that can be like a Chrome web browser um mobile uh and so on and then is successful posts that uh will just indicate whether or not um the post was successful in uh being posted okay and then there's a user table where um we have a user ID um the user type uh and also the age of the user awesome okay yeah so given all that information um I'd love for us to first start with a baseline so can you write a SQL query that shows the total amount of successful posts per user type in the last month awesome no I'm super excited so this ecosystem is really around user and um kind of the overarching goal is to try to understand and potentially root cause like successful posting as a proxy for engagement right um if you don't mind I'd love to take a step back and just ask some clarifying questions on just the kind of the daily table and data structures and some of the assumptions made um I think you know diagram's super helpful the one thing that I'm seeing is like a user ID to kind of post ID um trying to understand the cardinality between that like is it between a post and a user so let's say for a particular post one and a user one they fail the first time and let's say they try to post again will that essentially be a new post ID or it would be the same post ID yeah so that would end up being another record in a post table which means that it would have a separate post ID perfect so it's actually like the post ID sequential um so like someone could like unsuccessfully do like three times and then finally the fourth time like it will post id4 for to be success yeah exactly yeah awesome um I think that's the main thing around just trying to make sure all those uh keys are are denuding out uh now going back to your original question which is like understanding the Baseline uh post successful rates um so I think the approach or what the outline probably looks like is like by user um basically in one column and then uh essentially another column would be like their success rate and then we can include any intermediary columns to kind of just showcase those numbers um so from the output perspective I'm going to do by user and essentially their post success rate um what I'm going to do first is I'm going to try to pseudocode it just to walk you through like the thinking process and then I'll go do coding and then we can talk about outputs from there okay yeah sounds great so I think the one thing that we're gonna first look at is essentially uh so I'm gonna do a select statement but basically we'll do a column um so by user um we're gonna do basically the sum of successful posts um and then some of uh essentially post attempts and then we'll essentially take um let's see the sum of so I'll just calls SP as successful post uh divide by some of essentially or we can do count but some of uh poster types uh all right sorry just to clarify when you say some of post attempts what is that referring to in the table oh that's essentially like assuming that is successful post is like a it looks like a binary uh one and zero so essentially count every time they tried to post um from a particular user so it's essentially instead of some I guess more semantically it would be accounts uh is there a correct way of looking at it and that should give us our post-successful rate okay so like number of unique post IDs exactly records yeah exactly it's a hence taking off right um and then essentially I think um to clarify I think you mentioned potentially there's a Time component I was wondering if you can just reiterate uh what kind of time uh timing reference you you wanted for this particular or we want the whole ecosystem um yeah so that's a good question so if you could get the total amount of successful um post per user type in the last month okay sure um I'll do like in essentially the last current month um so today the last card um and of course because we're doing us the aggregation we're going to do a group buy um and then we'll do assumption order by as well so uh essentially it will probably care about people who are failing the most so we'll order from the success rate uh descending so this should at least gives us the by user and then uh post success rate any concerns otherwise we'll just get to the coding part here uh no that sounds good yeah let's see the awesome all right all right let's do it uh so let's do our select statement from and then we're going to be pulling from the post table um let's call it SP just for simplicity's sake um then we're going to do blue user ID oops sticky and then first we're going to sum the p uh big successful posts and then we'll essentially do this as post success then we'll do counts p is acceptable opposed let's post attempt and then we'll do uh since we can't necessarily pull a derived column which subject is recreated line uh we'll call it uh that's right um and then we're going to do the where Applause where P post dates is in a date time format uh since we're doing just last current month we can hard code it we can come up with a more Dynamic one afterwards if we want to but just for a sake of time um we are going to do assume everyone uh last month okay fine and we'll also order by um some post success rates um descending order so just double checking um we have a user we there's post success post attempts let me summarize we've divided the only things be careful just in case this is the integers make into a float so the percentage um and then we do a word pause group and descent so I leave this assist or we'll just try it out and then if there's any issues we'll just uh troubles from there so looks like it ran um we have essentially our user nine with the you know best post rate grades and user four with uh such a really bad pulse rate I think one thing to take away at least from an inside perspective that just seems to be a pretty big spread um suggesting that there's some types of improvement right it's not a systematic issue that's affecting all users equally um and it'd be interesting to kind of double click into that try to observe well what's a potentially driving the issue here yeah okay so you exactly anticipated what one of my next questions is going to be actually so it seems like because there's so much variation in the post success rates I'm curious um if we can then try to isolate the users who um not only post very frequently but um they also have a successful post rate that's actually below the current monthly average um and we can call these like um either super users or whales um because they're basically the users who want to post a lot but are frequently failing at posting I see uh perfect so it's like trying to segment essentially the users that we want to really work on so like they're the type of people who want to post but essentially are not being able to right yeah exactly okay perfect I think that that makes a lot of sense um so let's uh same thing approach due to some pseudocoding here and then we'll do the coding from their perspective at first okay so I think when you look at this type of metric you're trying to compare essentially the post success rate that we just previously provided in the kind of previous section and then compared to kind of a global uh limit right um and I think this the easiest way to do that is probably doing kind of a kind of a with a statement Clause to kind of create the aggregated metrics and then compare them and then basically generate the table again and filtering out what we think are the the two things right one around post attempts and the second is around post successful rate okay yeah that sounds good to me awesome so could you clarify what's actually going in that aggregate yeah yeah so let's uh from an aggregate metrics let's let's talk about it first so I think we're gonna throw with basically create a table um so basically create some table um and then what we're essentially going to pull is the average uh post attempt by user uh we're gonna perform essentially the average uh post success rates uh by user um and then we'll kind of generate from the table that from before we'll create those metrics and then essentially we'll pull the original table that we had before and then essentially we're going to filter out uh with two things uh filter out with a where clause um on two things one is going to be where close attempts so a user's post attempts is greater than equal to the average post attempts and then two is where the users uh post success rates is less than equal to the average uh post s uh success rate I think that's the approach there any any questions or concerns with that um no that makes sense so number one basically looks for our like high frequency users and then number two is looking at the high frequency users who have a lot of failures right yep yeah right and just combine it you're busy just building on the the aggregate filter that we're doing here right yeah okay that makes sense to me okay um so let's just get to coding um Robert meets the road here uh so let's just do either metrics uh just to make it easier on us so we're gonna select um so I'm going to put this as a stand holder now um because I think what we're going to need to do is basically pull the user metrics first and then summarize it on the outside here um so I'm gonna have essentially uh from a smaller table here so I'm going to pull it essentially what we did before which is uh super simple so if you use your ID so I'd actually just copy so it's easier post successful attempts um from post p and then we'll go by one so this essentially get us the user um and now from here the only thing we just give it an alias and so then we can now pull T1 average uh post attempt which is from here as average posting and then we'll also will average with a comma average uh oh success ful success times 1.0 divided by the plus attempt so this is essentially one line to gonna do that as a average of success rates so kind of it's always good to kind of just check in between so what I'll actually do is I'll just run this right now to make sure that this is producing appropriately so we can see on the right hand side the average posting is about 250 which makes sense there's 20 people it sounds like it's about thousand uh just a sustainabase and then um the average Professor dot 50.4 so realistically when we pull the second table we should see everyone's posting that's you know above that amounts and then but above the posting amounts I've been below the success rate here okay yeah makes sense to me awesome so that's uncomment just back out so now we have the tables uh running now we're going to do essentially the the final post which is really bringing back that last uh half again um which we can just pull from before so P user I'm going to drop the time uh where Clause I'm assuming that's not something that we were concerned about at this time if that's correct but to check in with you do we still want the time clause in um no I think that's fine because we want like the averages so we don't want just the last one yeah no worries uh the only reason why I bring it up is because if we do want the time reference like the way that we create the aggro metrics like we're comparing apples to oranges so um depending on the phrasing question we would have to put that time Clause back into the aggregate metrics as well so just something to think about um if that's something that we want to do but yeah so we're doing that having clouds instead of aware Clause because we're doing it a filter based on an aggregated metric instead of like a actual inline metric so you can't really do a where here um so what we want to do now is we want to say the post attempts um is essentially greater than equal to and it will essentially do a select Clause here to get the AG uh metric from before and that metric before the average posting uh from aggregated metrics so that's the first part and then we want to do the and Clause right because we want to find the intersection of the two so we will do the post success rate now because that's what we want to compare to is less than equal to again create metrics so this would be average success rates from so I just want to check this still makes sense um oh let's have that select out there first um so you're gonna select and you're going to do sum and then you do Post success rate and do the group by and then do the workings that we talked about before um so we run this if we experience any issues we'll just fix it on the fly so it looks like we have out of the 20 users it looks like we have six users that are considerable uh whales and this is where there as you can see they all post more than 250 and their post success rate is less than that 50 point uh so 50.4 percent that was all before yeah okay that makes a lot of sense to me okay so that's one way of identifying um some of the users who are specifically having this problem more often um another way that we could segment our user bases Maybe by age because as we know um younger users tend to use social media differently from older users so um could you write some SQL that instead shows the difference in success rates but um of like posting drop off between young adults so those are people who are aged 0 to 18 and not non-young adults uh by each month Oh by each month okay so that's an interesting element so is the hypothesis here saying not only is age segment but maybe there's a seasonality to their essentially their posting rate is that the idea here perfect okay um so let's do some pseudocoding here so I think that the interesting element here is that we're gonna have to do segmentations uh um within a certain column so likely we'll use uh like a case in one statement um and then we're also introducing a new element of time right um where we need to segment by the month yeah can you quickly summarize at a high level what the case and when uh statement does oh uh a case in one is essentially like um so a case is essentially a wrapper around as a if and then statement so essentially you'll take a column let's say age for example right so let's say when the age is between 0 to 18 you say hey let's classify this as young adults if it's greater than 18 we'll classify as long as an adult right but this is the beauty of case or case and then when then else and kind of syntax is that you can do multiple conditions for this so let's say in the future we want to do like zero 18 uh like adults 18 to 20 like 35 and like older uh customers or users and like 35 and older whatever the case may be like this is kind of how you sequently can segment out your data without having to kind of break it up into multiple queries okay perfect that sounds perfect for our use case awesome so I think the interesting thing about this is if we want to compare the two of them um there's multiple ways around it I'm going to try to I'm going to index on the flexibility side of things here so I think the first high level before we even do any pseudocoding it's like one let's create the aggregating metrics uh uh based on why a and non-line a by month um and then this is this is kind of less efficient from a coding perspective but it's more flexible because you can utilize these tables but we'll split the tables um by y a versus non-ya into kind of two subtables to to to sorry the two subtables and then essentially then we'll combine them uh with a join uh to kind of compare the two okay and essentially find a difference right um and that if we can highlight the differences maybe it'll like be very poignant on what month what particular segment is doing better maybe there's a difference between the two okay cool um does that does that make sense or anything we want to adjust there um no that makes sense to me okay um so from the Aggregates uh submetrics um let's just list it out I think pretty quickly here because I think it's very similar to everything that we've done before right um so we're gonna essentially have uh by month from the date time perspective um and then we'll do the case and when uh case uh conditional statements to segment outs users the beauty of it is that we're utilizing the same metrics that we talked about before right so it's going to be post success post attempt um and then essentially post success rates so in this case since we don't really I've shown it many times we'll just kind of just jump to the plus success rate um just so that's easier to look at and then because we're doing by two Dimensions we're going to group by one or by one and two sort of the first and second column uh that we see here um then we'll create a y a table filtering an AG metrics uh with the Y A segments uh and then we'll create a non-wise a non-yong adult table uh doing the same filter filter on none non-ya and then we'll essentially join them based on months so buy a month the same one um we're gonna compare Y versus non-ya um and then also essentially calculate compute the difference so I believe this should be handled everything um if there's no further questions I'll just I'll just jump right to the quoting um okay uh sorry just trying to clarify like the comments you have written here so uh is this meant to be okay so there's not two separate joins right this is just like uh just depending on yeah you're gonna join them based on month because you want to compare like months to month but then you compare it like to two groups right uh Y versus non-way right so that's just the corresponds to what you have on line 85 right they're not two separate things oh yeah yeah so sorry line uh 83 85 was like the high level idea and then 87 through 97 is more like the pseudocoding um just at least so you can track along because I might be writing coding and I don't um just want to make sure that you're kind of walking step with me that's all okay so then when you're saying compute the difference at the end um so that's in uh absolute difference in percentages correct uh so it probably won't be absolute um because a positive negatives do mean something um so likely I will I think what we care about is probably young adults just because we're into social media like everyone cares about young adult so we'll compare like Y A minus uh non-ya so if it's a negative it means that Y is performing worse if it's positive means why it's important better Okay and like what about the magnitudes does it make sense to get like a percent difference instead of like a value difference uh it's fair so I think it really depends on what you care about right so I think the so right now I'm just looking at the nominal difference right so like both percentages they so that's a success rate so that normalizes like the posting that's happening I'm assuming that's what you meant while the percent difference is is fine but the it could be uh misleading right with if that a base because you can have a percentage of five percent but like what if that's like not very high to begin with right if the base is really low so that's why I prefer the nominal difference versus the percentage difference okay sure so yeah then I guess with the nominal difference then if you want to uh reference the basis but we also have to have that output in the table right yep yep exactly okay uh sure let me just make sure so I have the output of success rate in the table perfect no I appreciate that um if not it's known for a dude that's then we can get started coding yeah I'd love to see the sequel for this awesome so let's do um let's call it uh I guess because the segments we'll do accent or add metric segments so we'll do ads and then we're going to send shape for what we did before so this is going to be interesting because we've done this before but it's functionality and basically pull the month um cable first so P um believe D is called post dates so I don't mix that up close dates and then we'll essentially call that as post month now we'll do the case statement that we talked about before so we'll we'll close on to one just parentheses just in case so case one um so in this case we forgive me uh so one thing I didn't I made the Assumption uh if it was kind of implicitly implied is that in the previous pseudo uh coding the thing that we need to do in the ad summary we do need to do a join uh to the user table uh to get the age we'll get it from there so yeah um before before we get there so let's just do the join right now um okay so let's join from the I believe it's called the posts user table so post user table um you can get their age that's correct yep yep okay so we're gonna connect on key user ID uh so on P user ID equals uh you user ID so great so the reason why that matters is because in here we really want to pull the H which I believe is just called age so age um is less than equal to 18 hopefully there's no one that's negative 18. but that's the equal 18 uh 18. then we'll call this uh y a for young adults and then else I will call it non-ya or um and then essentially end and this essentially split the two into two brackets or segments we'll call it as age brackets um and then this is where we'll do the similar metrics that we did before which then it's really easy I'm just gonna copy and paste from before foreign counts um as post success rates great um then we'll pull from pool and then as we talked about before because we have two Dimensions here we're gonna Group by one and two um from there we're gonna do the next section which is then pulling essentially a y a group which this is a pretty like um you can do a cleaner or faster way but this allows us to be more flexible in the future but we're gonna pull from the uh metrics segments uh where and then age brackets equals 5A and then we'll pull the non-ya section here and then we'll a H matches the top and then we're going to essentially pull the last uh assignment now we're going to combine the two to compare them right so I'm going to do select and then we're going to do from we'll just pull y a as the basis so full y a as six table one and then let's join the non-ya table uh it's table 2 on uh T1 plus one so we are using an inner join here so the only thing that I'm making uh the only caveats to this output is that I'm only going to compare where y a has a value if y a doesn't have like a month for example but non-non young adults did then that essentially would potentially be eliminated from this particular view um because it's it's kind of trivial to not look at uh if you don't have a comparison between the one other um and then we are going to try to do an order by just to kind of make sure it's like a sequential order month but let's pull in the post month that makes sense um then we're going to pull in the T1 success rates so I think it's called post success post success rates uh and then we'll call it just so it's easy to look at my success success rates uh T2 plus success rate as non-ya why success rates um and then because again this is a derived metrics I can't just pull a derived column here we're just going to do t1a minus uh T2 which is um and then we'll say this as the diff so just kind of thing through your T1 your T2 you're on and then we'll order by uh T1 post month ascending so that we get months basically chronologically and we can look at this so we'll run this again if we hit any bugs we'll resolve on the way but uh thanks for the notebooks but just kind of give some quick insights from looking at this we'll see like one two and three I'm assuming that corresponds to January February and March um what's interesting is that there is some variance um young adults tend to form out of the three months at least so within this data set performs better than non young adults um and then they all need discrepancy is really within the month of February um uh off the top of the head the only thing I can think of is maybe Valentine's Day but we can kind of explore further that really has a driving Factor here but it is it is interesting to see that the segmentation here yeah yeah okay I think that makes sense as a hypothesis and like just out of curiosity then if you were to try and explore what's behind that discrepancy do you have any suggestions for what types of analyzes you would do yeah I think so from a young adults um what would be really interesting is like even in so let's separate the two things out right so so you don't confuse like a multi-factorial problem potentially so like in the month of January and March you really want to see like why are they consistently like all young adults are posting better is it a particular time of day that they're posting that let's say young adults are posting throughout the day versus not mean adults maybe they're posting like after work and maybe it's overloading on the system that's causing them to fail like that's like a Time components that could happen so that's just from an age segmentation from uh potential seasonality component I would be really interested to see like any anomalies around like holidays so like is it Valentine's Day really driving the February uh indication is it's you know nominals maybe you know they're more established relationships or wounds opposed while young adults maybe maybe they're still in that kind of early phase uh that's why they're not posting that much but there's a lot of uh other segments and cuts that we can go after this too yeah okay that makes a lot of sense okay so so far we've been exploring um this like post-success rate as a function of the user segment um whether that's been before by like whether or not there yeah a a whale or a by their age group um but perhaps we could also hypothesize that this is actually a systemic issue it might be something to do with the tech like maybe a specific user flow in uh the interface is actually causing this issue so if it is a um systemic issue one way we could look at that is by looking like at like whether or not it happens consistently so if a user tries to make a post it fails and then they immediately retry again um does that second retry fail as well um could you write me a query then that shows each user's um success rate of posting when uh their previous post has failed perfect I I think from a Simplicity perspective uh just off top of my head we can do essentially the next post that they do um I guess the one caveat that we we're not considering is like how long between the post but maybe that's something we can say for another time um but more than happy to do that here so I think the interesting thing about this is unlike the previous uh kind of sections that we're talking about this is like uh you have to almost create a sequence within SQL to kind of know what is the next post for each user um and this is like a perfect application for like the partition or basically a windowing function here yeah um so I think the kind of a high level before we do pseudocoding like I think the first thing you want to do is that essentially create a post sequence um comment this out so it doesn't run uh post sequence uh for each user so basically like if you look at their total history you'll take their very first post as one and then the sequently go down um the reason why this is important is that you start indexing each of their posts so that when a post does fail let's say post four fails then I know to pull post five and so on right by using essentially indexing here um okay I think what we'll do next is we'll do the post pairing which I kind of talked about uh post pairing uh where your uh where we're filtering out the previous post as the fail as a fail and then basically uh pulling the next index and then just and then it kind of brings us back home and we keep hitting this on the head but essentially accounted for in the post success rate of of the next post uh or basically the yeah the next posts here so I think at a high level that's going to help us solve our issue so I'll just jump right into the kind of the SQL coding here or the pseudocoding um so I think what we'll do first is we'll basically we'll create a subtable uh and the we're going to basically Dimension it by user and post um ID and then we'll actually create a new column called uh row number or basically post sequence basically this is gonna create the window for each user uh user perspective and label each post with an index and then we'll also pull in what's most important probably is the fail or not fail so we'll pull the is uh is put a successful post or not okay um then we will create uh I I guess I the easiest way to talk about is a post pairing table um where we will get easier to look at um we'll pull from pull from the index table that we just created before but we'll do a where Clause where the previous post uh has failed um okay okay and then we'll also we'll combine the two so like so for example the output here would be essentially be user post ID and then the next post and then you're only filtering on essentially the post that has failed and again I'm breaking this out steps so it's easier to kind of manipulate again you can kind of consolidate this as necessary but this helps if you ever need to do anything else with it typically you want to just break them out so that it's easier to work with in the future um and then essentially the last table is essentially doing the summary summary Matrix um where you're going to join essentially um you're going to join essentially the original uh post uh kind of the original post information and the reason why here you're getting the post information is you want to essentially tag uh the second the tag the second show post uh if it failed or not which you couldn't really pull from before you could um and then we'll just summarize from there summarize the metrics so the only thing about this join is just to kind of uh list out it's going to be post ID uh post ID on the next sequential host ID it'll make more sense once you kind of go through the coding here but um just want to at least give a preview on what we're looking at I'm just not right there just checking with you this is my thought process anything you want to adjust or any questions you have here I think that makes sense to me and what do you mean my summarized demand oh it's it's always uh the same metric that we talk about um so it's really going after uh post success or secondary post success rate okay um so essentially we should have essentially user and then next uh uh next post success rate as the next column over okay perfect awesome so why don't we get uh added here so again we'll to create that width and then I'll sequence as so we'll pick the subtable um so let's do our select from host as P um where this is we're going to pull the user ID we're going to pull in the post that we talked about before and then the new thing that we're doing here is the subject of row number function um and then what's interesting is that we we want to do it over a window where we are partitioning by so basically a subgroup by user ID um and then we're gonna order by essentially because we care about the sequence and the sequence is really driven by the postings here so this is going to be the posts sequence ID and then the very last thing we're going to go on and of course is the star is successful post so that will should be the first uh sequence or kind of create the sequence now we'll do the post pairing table pairings as still select we're going to pull in essentially the holding what we're going to pull the post sequence table uh let's call this PS just for simplicity's sake and we're gonna pull in PS dot user five fingers here uh post user ID we're gonna pull in essentially DPS posts uh post sequence ID okay we'll call this as the fail posts ID for example and then we're also gonna pull in the posts uh sequence ID oops ID and then we'll do plus one so this is going to get the next uh posts so as next post ID and of course we can only call fail plus ID if we are filtering the PS I think successful post equals zero so this is essentially going to pull in only the failed post and also the next IDs post yes so the reason so this was a little user and then like say four five and then six seven for whatever case example so this is really important because now we can combine the two of them so let's Force let's First Rate the let's pull from closed pairing because that's the one that makes no sense let's pause PP and then we'll join at with posts um as as P2 just so it's no confusing so we're just pulling from the original table now where pp's uh next post ID equals 32.e um and the reason why I want to do that is now we're going to pull from user ID and then we're essentially going to pull the metric that we've always pulled pulling before but there's a caveat here um so we're gonna pull p2s a successful post and then p2s is successful both to kind of get the As and then we'll call it so it's easy to read next posts XD rates and then because we have one dimension here we are going to do a group by one and then just for simplicity's sake also do the next post success rate and I guess what makes the most sense probably a sending order to say like it was the most problematic folks okay so And to clarify again Group by one is grouping by the user ID yeah correct so the the index of like one two or three especially the the order the columns so I'm just say in this case I'm saying Group by one I'm trying to buy user ID um I could specifically write out user ID but this is typically just allows more sorry flexibility um yeah okay that makes sense yeah well so again uh you know but we're run it and so there is just a error and I'm just want to make sure everybody's a row number over partition by user ID just want to make sure oh just a syntax here to pull a partition by um awesome so we ran to an issue um this all looks correct you know fix the buy or but I think it's because it's uh simple uh simple issue here is just every time I'm because I'm running on multiple lines of Coach add that semicolon to kind of cut off the previous code and then now uh simplest thing gets you right um and perfect I think we now have essentially by user and the next uh post successful rate um so I think just some quick thoughts here just looking at the discrepancy when we previously looked at the user ID at success rates it's a lot higher right it was like 60 to 40 span range now we're seeing it down to like the range of 35 to 51. so I think your original hypothesis hey maybe there is a technical issue like maybe people are like like trying to hammer in a post that it's not being successful and that's what's potentially causing a pretty low rate um and this sequential analysis kind of points at that right direction um of course we need to do a little bit more dating to make sure that's truly the case right yeah so essentially you've showed here that the probability of the uh next post attempt failing is um uh higher for if the initial post failed right exactly yep yeah yeah okay perfect so um I think this is a great place for us to pause I think you've answered a lot of really cool questions and formed a lot of really cool hypotheses about what's going on here with the uh post success rate so I'd love to hear from you first um what do you think went well about this interview and uh what what do you think you would change going forward yeah I think uh it's a really interesting questions I really appreciate you walking me through this like uh it kind of brings me back to little days no matter but that being said I think what really went well was like being able to document the thought process like for example even though we hit some couple snacks in terms of like just syntax issues um I don't I think it was always it came across that hey I knew what we're trying to do and it was just simple um you know semicolons or like simple like syntax that was dropped off and I think it's really important as candidates go through this is that it's often more important to show your thought process than like trying to write the best code right um because I've seen the opposite before where people write the best code but they can't explain it that's really where like you you can't really understand like how how good they are um in terms of like opportunities um I think this is like a coding preference um you can be like the most efficient sequel out there when you look at kind of you know the way that we coded today it's not the most efficient right and but it's done within reason right and I think the important thing is to understand the trade-offs and to be able to explain that like for me uh It's always important to kind of build those sequential tables because I'm often doing analysis based on a previous analysis that I've built so instead of having to like you know strip apart a black box of code like I can say oh all I need is a subtable right and build off of there um so I think as long as you're explaining those preferences like those um those can kind of help you not have to write the most efficient code like even those last uh sequential coding I could have pulled this all the second last table into that second table right so I didn't have to do as many um that's something you think about just from a trade-off of like timing and also like memory usage you're using as you're running this code okay okay perfect yeah I agree with a lot with that point and it's efficient uh efficiency not just from the standpoint of performance but also in terms of whether or not the code is being reused too right yeah yeah okay so that makes sense so um some other things I had to say about this is that I really like that you uh asked clarifying questions throughout because uh this case study in general there's many different ways to interpret it you could have used different quantities to measure uh the things that we were talking about or to answer the questions so that was really important um and I also like that as you mentioned the thought process is really important right because you're trying to communicate uh what the data says to somebody else produce presumably your product team and so I really like that you had the pseudocode and you wrote out what the queries were gonna be step by step um yeah and so I also liked that uh you ran the intermediate queries instead of like just coding everything all at once and then uh running it at the end because that's more clear for whoever is reading your code to understand and also it just like helps you debug uh bugs earlier if you do yeah um yeah lastly I thought you thought you showed like great depth of knowledge of just various different SQL constructs like we went through some joints we went through filters we did group buys aggregations and window functions and I think all of that was really informative and educational so uh some things that might also be helpful is um so with a lot of case interviews like this uh oftentimes there will be quantities that can be somewhat ambiguous and how you measure them so for example one of our questions focused on the drop-off rate um of uh post success between like uh young adults and non-young adults so drop off rate could be it could be a percentage it could be like an absolute um value and I think it's just important whichever one you choose to implement for you to just explain and justify your choice of that um yeah and one last question for you is uh you sort of touched on this I know we kind of moved on past it how would you be able to avoid like hard coding the dates in the first question where I asked you to do only from the last month yeah um so in SQL there is like um so it really depends like you still need a reference but like a date ad or a current or interval ad so sorry a combination of a date ad with the interval function allows you to do um essentially um very Dynamic dates so let's say for example like I wanted to do the first day and last date like uh let's go let's say First Dates uh the easiest way to do it is like select uh so date add by the way I didn't I didn't know if you want me to show you but uh I I figured I figured it's uh better for folks who's watching so um you can essentially pull the current dates um and then you do interval um minus the day of current date so really what you're trying to get so let's say today is the uh beginning 20s I'm really bad uh 24th right so you're born in the day 24 um out and then essentially you want to do is a plus uh you're gonna do plus one and then you're gonna say I want to essentially minus 24 plus one day uh and that should get you your first day of the month right okay perfect yeah and that's all I had um thank you so much for joining us today Andrew I think we really learned a lot from you both about the data side and the business side of these types of case interviews and thank you everybody for watching good luck on your upcoming interviews bye everyone good luck bye thank you thanks so much for watching don't forget to hit the like And subscribe buttons below to let us know that this video is valuable for you and of course check out hundreds more videos just like this at try exponent.com thanks for watching and good luck on your upcoming interview foreign [Music]