Exercise note tutorial. So you will. Like for tonight, the reading for tonight you will you will just have the reading again. No quiz. Don't tutorial. And then tomorrow after class, he wanted meet new exercises for. Yes, you're right. And then Thursday, we'll go ahead and do the the unit testing. I'm not even sure I'm sorry, DO testing. Which is a form of integration testing. So we're going one step beyond the unit testing. And I am not gonna be here on Friday. And so I have to fly to Boston. And so you will have pairs. Work, which I will. I should have announced all of that on Thursday, so you know, know who you're working with and and and give you your assignments and all that kind of stuff. On Thursday, you're going to have to find. Help elsewhere, perhaps because I am actually very available, I will have my laptop, so it's possible I will get out and do something, but I won't be very available for weekend. So but Tori and David and Colt are all. As round as they ever are, we so I mean certainly on Friday. And then I moved back on Monday given that I'm flying back Sunday night, I'm flying back on Monday, weather permitting. There's a very good chance that I'll be doing Monday's lesson from uh, you know. Airport hotel or something like that. But in which case, you know, we'll do our best to that's what we have to do, OK? Alright. So today. Sorry, I'm doing my. Cricket. OK, today we are talking about budget. I gotta be honest with you guys so. We could pretend and nobody would be fooled, but we could pretend that all topics in this course are of equal interest to us and that we are equally excited when we have, you know, one topic and another, and then we are all engaged instructors we'd like. You know this and it's not not quite true. I am perfectly happy designing databases, but database design day is probably my least favorite of the entire cohort. So much of it is just academic and really doesn't. It doesn't speak to me personally, alright. So just so you know, that's that's part of the reason why I like, I'm not going to run it long. I'm not going to do that much with it. It's not that I won't cover all the material I have to cover. I'm just not going to get excited by it. I know there's some instructors love this topic and well, you know, I know one instructor who who went to one o'clock yesterday. So much to talk about. Great, I'm. I'm excited. And if you had him, then you might be excited too. Or you might not. And whatever, OK. But now we are getting into the part that actually does excite me more, right? And database connectivity. Is that the idea today has has different. Names for data access and go is is that I think the official name, this cohort, they seemed to rename it every cohort. So I'm not I know that last court was called database connectivity. We're now at that point where we know how to work with the days. Right. And we, we can make our queries, we can, we can do our. We can insert new data. We need design a new database. We do all those kind of things we do. Then we have that database. But let's back up for just a second to the idea of what the whole point is. The point is that there is persistent storage of your information. And it that you can make changes day after day, different people can make changes. Different people can make queries. You have some place that stores all that data. And and this is now speaking as the programmer who works in Java kind of person. You don't have to worry about that garbage, right? That stuff is off there somewhere in a database, right? You don't have to maintain all of that data. Instead, what you have to do is interact with small portions of that data. So I'm going to use I think I. I haven't used a whiteboard in awhile. I used my whiteboard from in here. Imagine that we have the and and abused a similar example once before, I think. But imagine that we have the data access warehouse. Right. He. A. Oh. Probably saw in your reading DAO stands for Data Access object but. The. Idea behind this is that there may be different windows. That you can go to in your in your data access warehouse. I mean, you walk up to that window. Can you say I want a rubber duck? Or I want a surfboard. Then somewhere in the bowels of this thing, they come up with the duck rubber duck. When they come up with this airport, you may even say I want the rubber. I want the Unicorn rubber duck. And then they go out and they find the Unicorn rubber duck and they give it to you and then in Java your job is simply to say now I have a rubber duck. OK, so I want to say come on. Trying to say OK, so I'm gonna say I have a rubber duck. Enormously large, OK. And I'll say Unicorn. I may make some changes to the rubber, the unit Unicorn rubber duck. And I may walk back and say, can you put this back in your storage? Right. So then from that point of view, let's see if we can give ourselves a little bidirectional arrow, have our bidirectional arrow. OK. So now we have our rubber duck. It comes out of here. From the Java side of things, all I have to worry about is that I know how to deal with a rubber duck. It has whatever interest, you know, whatever values or things that it has, including this name, which is Unicorn, right? Or maybe that's his type. Maybe it has a real name which is Max. I don't know. But you know, you know how to deal with a rubber duck. You don't know how many rubber ducks are stored in here. If you go ahead and put in another, you create a brand new rubber duck or buy one or whatever you give it to the warehouse. They'll put it in there and they and they may store it. They may store it lined up on little shelves with little names, or they may have a very sophisticated electronic conveyor belt system that takes it over little drawers and turns it to you. You don't care. You care that you are able to interact with that warehouse and say I wanna rubber duck. So what is this? Up until now, and by that I mean basically your capstone 'cause. We haven't done that much up until now with with persistence, with semi persistent storage. All of the objects in your database when you loaded your inventory in your vending machine. all of that was in in some way, and then it will give it back that one object, which I will then deal with. Right. Similarly your bank account or whatever, let's, let's say that you were that that the idealistic people who wrote the the readme and then and said we were going to enter integrate this with our bank account. Let's say that they we really had done that. Maybe the whole bank account and everything else is also stored in a database, even a different database. You are then going to interact with it simply by saying what like you. You define actions that you can do against it, and then you go and say. You go do those actions. You don't have to worry about. How the actions are implemented internally? And in that sense, it's a it's all about abstraction. But you don't have to worry about it, because then you have to tell the database what to do. But you don't really have to. Manage how it does it. So. That is. One level of abstraction. Now we're going to do a new share. We're gonna go back to the screen here. Indeed dear, we have. Two levels in this other abstraction. My application wants to talk to. It wants to do certain kinds of actions. So it's gonna say I want to get. The rubber duck with a particular name. And in this image right here. That what we just say as DAO. Is an interface level of. What are the things I can do? And what are the actions I can take? What are the requests I can make? How can I update things, delete things? What can I do? And we implement that as an interface. And then our what we call our model, which is the Java side and the app. Only have to deal with the items that we deal with that we want to deal with. And these sort of general methods that are in the interface that are defined in our interface which just say what kind of thing do we want to do. And then we have a third piece, which is the specific. DAO. OK, it's a little confusing. They had the same name, but it is. This is just gonna be your general day and I'll show you an example of how this works in in code in a moment, and then you have your specific DL, which actually does the how am I going to go from get a rubber duck mean Unicorn? To figure out how it is actually stored and having a level where you go out and retrieve it the way you should, whether it is from a shelf or a box or or whatever else. However it is stored. This level handles these slightly dirty or details of the specific place it stored. That is the theory behind what I like about this. The reason why I get excited by this topic and whatever else is because this to me feels like the way I. The whole idea about. Programming that we're teaching in this course. Is the idea of taking complex problems and breaking them down into simpler ones? To simplify the, the app needs to be able to do certain methods, and there's this layer that does those methods. Then these boxes all the way before you get to the to the JDBC DL or file VA or service day or whatever. This interaction is is clean and simple. If we wanted to go and change the database completely, replace whether we are storing in POSTGRES database or whether we were storing in Mongo DB or whether we're restoring it, you know on end files like you did dot CSV files. We could do all that without touching the way the application works. And the way that the the DA, oh interface level works. This then becomes a separate piece. That allows me flexibility and as a programmer I like to. I like flexibility. I like breaking down problems and and I I like simplicity. So the goal is, yes, let's say that right now I have a database and that database is sitting on your computer 'cause we don't have. We don't have you guys interacting against the database somewhere else. Your Postgres database is on your laptop. What if that Postgres database were, you know, at Tech Elevators Office in Cleveland or in Dallas, or, you know, wherever? The changes that would need to be made would be entirely at this this the later part of this in this layer. I'm working with different layers gives us that ability to separate out and only think about business logic up here. And implementation back here. Hey. If I said you know what, I don't wanna maintain my database at all. I want somebody else to do it and there's a service out there that has a Web API and I can send requests to it and it will handle all the data and I don't care how it handles it. I wouldn't have to change my application at all. I would never change anything around here, I would just change the layer here which actually interacts with that service. So. So yeah. Let's talk about. I actually I should step back and just say as we as we go to this, what? This is a popular new and I say new because it's probably in the last 20 years as opposed to, you know, the last two years. So relatively new, the idea of software patterns have become more and more popular. A software pattern just says there are a whole bunch of different things in the world that have to get solved. And. They they all fall into certain buckets, certain kinds of things that have to get resolved, and we have come up with a set of interactions which tends to make that work well. So this abstraction layer, this idea of separating out the interface layer, which is the, you know what are the things you want to do from an implementation layer, which is how we actually do it. This is all this caos thing is called a software pattern. The DEO pattern is out there and you might use it in different languages. You might use it in, you know, in in all kinds of scenarios where you have persistent data somewhere, whether you're Amazon, storing all the information about all the products that are out there, or whether you're the library shrink. Will books or you have? Or a pet shop or or whatever else you know there is. There is persistent data out there. How do we get at it without having to muddy up our application by knowing too much about how it's stored? Right. And that DL pattern is then is is defined. As a general purpose software pattern, so when we refer to patterns, that's what we mean. There's not a package out there. That. There's not a package that Java defines, for instance out there that handles all of this. What we are using to help implement all this is something that spring, spring is this framework that works with Java and so this spring JDBC template presence that you're going to see in here that spring JDBC is is the implementation of this. Knowing how this works so it is a, you still don't just drop a pattern in a pattern is a thing. A pattern is a series of rules and abstractions that work that that you can learn how to use and then implemented by different people in different ways. But if you went to a company and they said we're doing, you know, adieo. With JavaScript and I know something you've never heard of. You would still understand. The idea is that you keep your application and what it what commands it is able to do, what queries is able to you keep that separate from the implementation layer of how it's going to actually interact. OK, so let's look at what this actually looks like. If you open up your lecture. Make sure I'm in the right place for that. So many things going on. OK, if you open up your lecture folder and you there's that actual, you know, palm dot XML get all excited. And you open it up. There's a whole bunch of this that is already implemented for you, and then we're going to implement some more to talk about how it works. But let's start. This should be familiar City Park state. Every. The idea behind a model. Came. See how we have being broken up into the model and the DARK. The model is how do I define an entity? A thing just one of them, not a collection of them, not groups, not whatever else. But how do I define an entity like we are familiar with using? In Java so that I can use. So if we go and look at our. You know, here is our United States erds. Right. And we're going to look at city, so let's let's focus in from in its city again, we're not talking now about the relationships we are talking about the object itself. So it has a city, Ivy, a city name, a state abbreviation population area. Those are the things we have decided we need to store about a city. In our. In our application we that will be part of our model. We look at our city, it's going to be very familiar as a city ID, a city name, state abbreviation, population in the area. We have all the same things. There is some matter. OK city idea can be a law. Strings for the things that are bare, carrying whatever else. Population is along areas of double, you know. However, we store these things. We are going to try to map them to something that works in job. You'll also notice that the names are. Java like names. And so they don't. The names don't have to be exactly the same. That that's an implementation. Then we'll get to about how we do that. But so we have an object here. And this object by itself is pretty much gonna be just a values that are in my one row of my one into T so in in SQL that's going to be one row. It may be stored in some other way and some other database, but it assuming we're using our post graphs. Better relational database. Whatever else, this is a map not to the city table. But to the individual row in a city in the city table. So that is our goal and underneath it, what we're really gonna have is by and large just getters and setters. I. And we're gonna we is common in these to over override the two string method. So we have a way of printing it out. We don't have to do this but this is a common thing to do in your model, right. So that's a really simple class. OK, I'm not putting a bunch of methods in here. I don't have fancy constructors. I don't have whatever else, I just have my getters and setters in the variables. OK, my model tends to be very simple. And then we've got three different files that go with city, three different Java. Fans OK, this is, by the way, gonna be in the package com dot echo dot model. Usually down here under the model. Under D AO, we have something called the city DL. And the JDBC sitting here. So we have three different things all related to the city. If I look at the city there, it's an interface. And it's a very simple interface that justifies what are the? It's almost like, you know it. We've been dealing with classes is with normally be the methods you would have in your in your in your class. It's a little like that. You've got get cities, you've got update cities, you've got whatever else, but it's just an interface. It doesn't implement anything. So you never put any implementation in here, you just say. I have some way of getting a city bat by ID. I I can get a list of cities you know by their state abbreviation. Notice that. What we don't have in our model. Is any of the relationship tables. We don't have the park state table. That's because there's no object that like we don't. In their Java we don't do any of the worrying about private, you know, foreign keys and primary keys and we don't worry about any of that kind of logic. We don't worry about relationships between the objects. Those are in your database. And you may use the stuff in your database. Well, this is the clear example where you use. It is. If I look at my I'm getting a list of cities get cities by state. That may use whatever implementation it does to go out and find that. If I had my heart DAO. Let me see her. Alright. And you have get parks by state that's going to need to use your park state thing. If you have your state deo, it probably has a get. Might have guessed state by capital which is going to have to look up and relate to them. You could have been here in your state or you could have, you know, get parts by state. Get parks that are in this state. At the interface level, we know that we need to do this stuff. We don't care how it's done. We just know we need this information, but we don't need to keep the relationships. We're keeping a single row at a time. In our model. And in our. You know, just playing DL, not the implementation. Go we have an interface which says what are the methods we can use against that data. And then finally, we have the thing that actually goes out and does the stuff. Right. And this is the one that is the layer that it it just like exact terms it's it's the way that you connect to a relational database. So let's try running this and see what this actually looks like when we try to do it. Hey I got my and I got my nice command line interface. Who is slightly fancier later on about this. There is a cities and parks CLI. That's gonna have my command line stuff. Maybe the milyar with that from the vending machine. You know, the way we the general purpose things, we're gonna save you or modify city information. So let's just try, we're going to say. Alright, we're going to view which state is it in, say, Ohio like? And so here are the cities in Ohio. Now I can say, alright, I wanna look at the. Ask and say there is date. Right now. In Java. We don't have any. We don't have all this information. Right. We have, if you look at what we're actually running at when we do this, I'm not going to go deeply into the CLI 'cause it just gets. Complicated, right at the moment. Alright, this is our run thing. Please enter an option. We have managed city info. We're going to manage city info. I'm just trying to feel weird actually, say update or no. See where we got? We got updated city. We have delete city. But each of these things, whatever it says here, I'm gonna see. plans. Display City, for instance. OK, we look for. Just play. Find. It's likely that's just that's afterwards. When we interact and we want to do something. We will use the DL. I always try to avoid getting into the depths of the CLI because I don't get confused. OK, there's an example. OK, so we have. If we're going to update a city. We get whatever information we need from the user. And then we call this general purpose city VA O to update the city. Now City VA oh is the thing we defined in here. Because it is an interface. What we're really calling is this right here. The JBL and DEO to do the update part. But we just interact with a level that's at the interface part. So let's say I wanted to update the city. OK, we're going to go. Here, let's update the city. OK, what's our new name? And we don't want to change it. It's our new population. Then how has 150,000? And then. We don't wanna change the area. OK, so we've updated. If we were to view the city again. Hi Al, this is how it's doing it and we say Dayton again. Interesting enough, notice that it came in a different order. Now I don't know why Aiden was. Yeah. Dayton was here before. Probably comes back and turn in the order that it was last modified or something like that. Doesn't really matter to us. We don't care. But I'm saying that that's a shorten implementation choice. Get an alphabetic. OK, so I'm gonna look at dating. Look at Dayton Hello and say something. I don't know what it's doing. Hello. I didn't like the truth, OK? We're not going to worry about what it's theoretically doing, but what we will do is let's go out and take a look at PG admin permit. what? Let's look at the United States database. Do a. Query tool. Alright, so let's do a select star from. City. Where? Stage abbreviation. Peoples. OK. So we have 10 and you'll notice that Dayton's population is now 150,000. Both PG admin. And my Java program running over here with the command line interface are interacting with the same persistent database back there. They're both clients. And Pro Scratch is the server that is handling the data. Now for me it's just two different applications running on my laptop, but if this it's a database row, anywhere else, if we put in one database for all of you and you all started doing stuff like this, you'd all start seeing each other's changes happen. So my. I am interacting with the database here just the way. Now if I let's see if I can stop it and restart it. I don't know why it's. Running into problems, but. Try running again and just see if we can. OK. So we're gonna have you. CEO H. Sex is dating. Let's see if you managed it. OK, so it pulled it in the population at this time. If we go back to PG admin. And we want to say we want to update this value. Talk me through how I update this this population. Any volunteers? How about you could do? I'm sorry, it would be update. Uh, let's see. We're doing the city database, so we update city. Set up and then you could do we. I mean, we know the city ID. It might be this easiest way to do that or do it just do set city name or no actually sorry. Or change the population population. OK, so said city.com. Hey, by the way, you're right. We know, and that's fine. Yeah. So set city dot population equals 29 million. You know they've they've had a population boom where city, under score name equals Dayton and state abbreviation equals Ohio. The way to do it, or to city ID equals 86. OK, so this is what we're going to do. We're going to run this. Now I want you to notice a couple things about you because we're going to focus on how this shows up in Java. It's going to come up and it's going to say how many roads were updated. When I go back and run my query. It's not going to have Dayton and Dayton now has 9 million people right in Cleveland is very jealous. Right. If I now go back into my running Java program. And I say. Let's see, man just hang again. OK, we list 6. Right now the population is 9 million, so the interaction between the two both of them are interacting with this database that sits out there that stores all the information. This is very powerful. It means that you can have different things running and interacting with your data. That needed like you could have one application out there running that was just monitoring cities and any of them that had explosion in population growth, you would start your new ad campaign there. And it can just check once in a while. OK, so all of that is possible at same time as you are managing it here or through PG admin or whatever. It's all a persistent data store. That is the goal. OK. Have you done all this? Let's see if we can exit. Now let's. Look at. OK, here we did. This is how we updated something here. One quick thing which I did not was in your reading, but we didn't really mention it because it's not very relevant until you ripped here is that when you were doing insert. You can return a value. So in your insert. If you say returning and then you give the name of the the column that is going to return. Then that becomes the the the result of your insert. Now in Java, that's going to be really important. Whereas it wasn't very much in SQL 'cause, we don't really care what it is. We can go back and get it in Java. And again, we're used in SQL. This is the way we tend to do things. Alright. Just the way Elliot said you're going to go ahead and do things like this 'cause. You don't know the city. When you were in Java, we store the city. And the city ID. I'm sorry, so I can use it to update things. I don't ever have to do queries to fight, figure out what updates. Remember, I'm only dealing with one. Item at a time one row. So let's take a look at what it looks like. You know our we have our city D AO has cat city. And we have the city ID. So let's take a look at our JDBC. Right. It is going to. It's a class that implements city there. Just an interface, it implemented. That means we have to have a. You know a method in here. For every method in here, we have to have an implement key. There are a couple of things in here that you will. You don't have to understand in any super way, but they are the pieces that let Spring JDBC work. You're gonna have this. Private JDBC template that we're going to work with. And then it's just going to call a data source thing, which goes ahead and kind of sets up our our scenario right. Again, if you can get all excited about these, you want to explore how all this works. That's great. Mostly what you're going to do is go looking at a working one and do that same. Alright, just to be clear, I I wanted there's nothing wrong with understanding more and I will tell you I will talk a little bit more about the data source and how it works, but for now I wanna just say those things happen and they allow you to do this other stuff. So there is a get city method. And whatever, you know, we have a slight query because we are now in our implementation. We now know we are dealing with a a relationship database takes equal. We can create our select state. To run against that database and then get results back from. Right. It is just defined as a string. I'm putting together in various lines just for the pluses at the end. It's just one long string. However you wanna do it. It's one long string. That's the thing we're going to run. The one tricky thing here that well. You'll hate the semicolons because they'll make your life, you know, because they're so confusing the at first there's going to be a; Inside the string. There's going to be a; After the string because there's this Java. The one inside is related to SQL. The one outside is related to Java, right? You'll get used to it, but just seeing it? Is very important. The way question marks work is when we do this magic query for rosette. That is, running the query against the database. And when we do this sequel right now, we tend to call it sequel or something like that is the strain. It's just that you put it right in line here, but it's much, much, much better to go ahead and make a separate strength. It's easier to understand what you're doing, but it's going to be here. And for every? That it finds in here. There is going to be one argument passed in. So this has a single? So there's going to be a single value that gets passed in. And then we check to see if there are any results done next just. Basically, it's like when you're doing a scanner or something like that and you say that doesn't have an X line. Right results dot next just kind of populates this results object with. The next thing, it returns false if there is no next thing, so it kind of does both things at the same time. And then calls this back row to see which is a method in here that we implement. I mean, show one more thing about this and then we go back and like try to make sense to this if if I've lost you. So if I'm going to look for map rhoticity, it's usually. Macro to see is the thing that says how do we take that data from our table? And from our results set to get passed back, how do we map that thing into a Java object? And it's it's it's very straightforward. It's just every single one of the different things you're going to you're going to call the setter, you're going to create a brand new city and empty city. Remember that you're going to call the Center for each one of the items. Using. This special logic and then all you really have to learn about these. If you have to learn what is the way you translate from some data type that you've got in your database to the Java equivalent. So the city ID is like get law. And that is row sentence. The thing that got past it. That's the sequel, Row said. Well, sequel Row said is the current row. In a result set. When you do next, it will get. It will just re populate its its internal stuff so that I can one buy one, get all the values. That's a little confusing because you don't get a new object, you just keep using the same object and it keeps changing what it is showing. It's moving its position along through the Rose. But. So some of these are fairly obvious and straightforward. You have to get long, which is long. You have to get string which is a string get double which is a double. OK get Boolean is going to be a Boolean get you know it's it's it's pretty straightforward. OK, it gets a little trickier with dates, but. Mostly this is fairly straightforward, and again, it's not so much that you're going to memorize it. It's you're going to go look at one that actually works and and take it. Or you can sit here and say well. What is rosette? You know, rosette dot. Remember back in Intellij, you can go and say, oh, what are all the things that it does? And let it do its thing. Notice by the way, that each one of them has an INT I and a string as as two possible ways of calling. That's because the result that comes in has a name, but you can also just say I want the zero with one and it's an offset like thing. I want the the first column, the second column, the third column, the 4th column. So you can do it by column without even knowing what the names are. Or you can do it by the name that is stored with it. Notice these are the database names. So the reason why you might do something that was not by that is if you didn't know the exact database names but you did know the position. For most of what you're going to do in this class, you're going to use the names. It is best to keep the two in mind, but that's just so it's well and the reason why that's important is if you're going to do sequentially, then you'd like you have to know the sequence. If you just know that that there will be a city name in there somewhere, they may use the string. Let it worry about which indent. You know which roads. So. From your point of view, and we'll again, we'll show this in a minute and we do this something with the park. The most important thing you always do is you want this math row to whatever the object is. Everything else is then going to use it. So when Iowa and get the city. Alright, I do mind query. I get back where I resultset I remember. We don't know whether it's coming back with one or two with zero or one or 100 results. Just in general, we don't know. In this particular case, we happen to know that's a primary key. And we happen to know that this is only likely to return one result, but but the the query for Rosette doesn't know what. What the query is, it just knows that it's going to run a query and it might have results. So let's look at something a little more complicated like the get cities by state. Alright. Now again. You know, if you can even you can even kind of cheat here and say, OK, what are we going to get if I do Ohio? You go in and you say. You have to fiddle with it a little bit. Right. But. Right. I'm gonna say I don't want that. I don't want that. This feels like overkill when you're doing it, but I'm doing it and and trust me. You will get to places where you want to just say I just. I need to see this actually run. OK, so the city ID we're going to do is let's is dating, so it's 86. I'm trying to run this and. When we run this query. Notice it is not star. It can actually tell it all the different things because we want to be sure we know exactly what order is coming in. Right. Isn't gonna do a thing. 'cause, I said state abbreviation, didn't I? My mistake. Yeah. Someday I'll hit tension, OK? So this is going to come back with a result set. Which has six values. 66 different rose. And then when we go into our code here. And we call get cities by state. I'm creating a brand new list. I call my Gettys by state and this is where I'm going to say. While there is a next result, well, there's another row. Add to my list the results of my retina. Rose City Macrotis city, remember. Not only does all this, but it created a new city and then returned it. So what I'm doing is creating a new object for each of the cities for each row I'm creating a perhaps even. What is this Java thing that he acts as if we want to know, you know, and and and, you know, intelligence. I don't recognize this all of this me. It's kind of feel like that a little bit. And. You're going to take a couple of iterations. In terms of this trying to get to. not just what is going on with the arrow but but like just getting yournew city object. And then in here I am adding that new object to my list of cities. And then I returned my cities. For the get cities by state. And then out here. OK. Yet. Cities by state. OK, I'm gonna. So. I'm gonna get cities by state, right. And then I have a list which I can do like cities outside. Or I can cycle through and do for I = 0. I less than cities outside I plus plus in print out by format my cities dot. Yet by by offset my new your Java here. This is where I now heavy list. I'm going to loop through it. And I'm going to be able to do this gets the city of the specific city object at that offset. And then just calls the gutter to get the city. So the way that I ran this thing that we saw where it said, you know, tell me the state abbreviation. And then I will tell you, the cities in that state. Is was right in here we asked what the state is. We pass in a prompt. Let me do all this. Let's try running it one more time. Just because I want to show you. So what happens? I now do my I want to view sitting information one more time, so I do OK what state is the city in? And I say \*\*. What is it doing in this case? It's gonna say invalid state abbreviation. Now think about just before we go look at code. Where is that happen? In which one of these things? We are now getting cities. And we have our we know that we are calling the method get cities by state. Right. What is giving us this invalid state of revision? OK so. We go to our job, Kitty, whatever, whatever. Now what? Before we? Before we dive into this, there's a little complicated scene where this is I just wanna let's let's let's take a break and we'll come back at 10:10 and we'll go through it. So as everyone is gonna like shade, alright, so a lot going on here. she 1010 Not just what is going on with the AO, but but like just getting your your head back around, I want you to do is try not to get too hung up on. Whether you understand the details. And try to understand the concept first. If you just if you like, don't you don't have to worry about how you're exactly going to do this. I'm trying to make sure that you understand what the concept is. So. And I will circle back on it. But it's not even a time when it's great to ask if you have questions because the questions are so like, what is life at this point? Then why am I in this career than they are, you know, helpful for so. So bear with me for a little longer. We will circle back. This is going to get more clear, but it's not all going to come at once. So when I ask God. Invalid state abbreviation. Now I ask you. How do I know that a state abbreviation is invalid? The answer is not really. I'm not really asking for where do you see this in the code 'cause first of all, you're not looking at the. They're like you could be browsing through the code and looking for it, but the point is not that. I now have a database. The way that I know a state is invalid is because I query the database and say is it a valid state. So literally the way this works in our code, going to see my code, let me make sure I'm recording again. Recording in progress. The state is invalid. Is I've gotten? I don't. I don't care. A lot of the the code part of it is I call prompt for state OK which is the thing that asks me for input about the state. Right. And I say, let's let's look for it and proper state, it's going to just ask me a question, get my input, you know user input is my scanner that is all set up. But what's important here is that I go say. Now you could actually put a question mark, the list State's don't worry about that. So we're supposed to enter a city abbreviation. So if I enter a istudy brevi etian like \*\*. I'm going to use my video. To go query my database and say can you get me the state? With that abbreviation. So. When we do that, we're going to go, we can get the state because we've defined that with the state abbreviation. In the state, it's going to get the state right here. And here's the important part. Now worry about the exact limitation. We define a a state as an object state is just our Java object. We go out and run a query on it. And we try to create that state or not create, we create the object from that row. If we don't, if we, if our result returns at zero, like if we go over here and we say. Yes, we try to run this query. It doesn't give an error if it's completely valid from what it does is it says there it returns a zero ropes. So over in our Java we don't look for an error. We don't look for, you know, whatever we look for, we ran it, we got our results and we if if results dot next. Is false. That means there were no results at all. So our state is still just null. So the place they called prompt receipt now. Is going to say. I tried to set my state to the result of this this get state and it it returned a null and if the object is null then I can. I can do my display error display errors can put the stars around or whatever. It doesn't really matter. The point is I query my database. My database doesn't have any result and then I know that's an error. I go back to my. This is the goal here was all in my prompt fruit for Selective city. What's T City state is that city and I asked for what state it was. I used exit says that's invalid. So I try again and I put in a valid state. So something longer. And it's gonna give me. It's now. It's not an invalid state, so we can run my second query, which is my get cities by state. So I query the database once to find out whether that's a valid city state. And notice I could have a valid state that has no cities at all, and that's a different kind of error. It's still a valid state. I would get through to this. I would call my get cities by state. If it return nothing. It was then give me a different error which is no cities in selected state. But the point here is that you don't. Stepping back from the Cove. You in your Java program aren't supposed to know pieces of information like this. You're supposed to ask the database for information like this. You don't keep track of whether they're that's an invalid. You don't do validation on the state names. You don't do any of that cop. You have stuff. You just pass it in. So back to our sorry JDBC. So this is our. This is the thing we're implementing. By now we know because we asked, we know the state abbreviation is valid. When invalidated it then it would just return. Again, same sort of thing here. If we create a list, we get all the results we put them into our into our list and then we return the list. What happens if there are no results? We're not returning null now. We actually have a list, it's just an empty list until we've added things to it. When it was California, it had 75 cities in that list. If I if it's, if I click, I go back here and I say. Yeah, yeah. 75 West Covina. Proving. So let's go, let's. Let's try what is Guam? You may remember what? Guam, maybe Gu. No idea. Guac is why we do our. So just I guess right Gu. OK, so we're going to say, Gee you. Right. What the heck is that? Right. Well, there's, there's the only city in Guam. And it goes through and it does its thing with all 1000 people in it, right, smaller than, you know, the smallest cities around us in Cleveland. OK. This is then there, so you have a whole series of things that are just that, feel like a bunch of different steps. But if you focus on what the the goal is, which is I know how to do a SQL query that gets this information. Is this just how I call that SQL query? The only way to get this to understand the only way to. Get it working is to start building it to do it yourself. For those of you which is, I think many of you who did not do the tutorial. It's fine, I understand, but you might want to go back through it and actually try to do it as you get a little closer to this because. It's it's just you need to go step by step and say I don't have to get old. I I need to understand the concept and I need to be able to implement it at the lower level and the stuff in between where I understand why I'm implementing what I'm doing. That will come overtime. Any questions based on like what is the most confusing part of this to you so that I'm sure I'm, I'm I'm focusing on the. The right parts. It's valid to say all I'm. Can you break it down in steps? Step forward on this. I'm saying, can you break it down in steps like step one, step two, step three, step four in probably that sequence will. I know you have, but. We'll go through that. This is a tough one because because it's so easy to get lost as I step through the the pieces don't, but let's let's take. Let's try that with the parks, OK? The parks in this case are not. OK, our perk deo. Now remember, actually let me step back the model. A single park, a single park has an Ivy, a park name, date, established area and camping. Skype. I want to go out and I want to get a single perk. My name. Right. I'll go and look and see. Is there a in my park? VA. Oh, what are the things I can do with a park? I can get a park if I know the ID. I can list out all the parks by state, create a park, or update a park, delete a park at a park 2A state. Or remove a part from a state. Those are all the things I've defined that I need to be able to do. Right. Forget Java for a minute. How would you get a park? From a database, have the park ID. That's where I go over it and practice. I just say, OK, how would I get a park? Helmet. Comment that stuff out. Let's let's do the get a perk I am getting. Get a perk I'm gonna select. Star from park. The park idea is 16. I want to get. All the stuff from it where? Park. ID equals 16. Right. This is what I am model. This is what I am doing in Java. Or what I want to do? OK, so I get my one value. Because we want to be super explicit. I'm going to do this as get park ID. Park name. Date established. Area. And has camp. I'm just do it like that. It's going to give the same result, but I'm going to be explicit about it. OK. This is what I want to be able to do. If I go to my job. And I say I wanna get it park. Then I go and say, OK, this is my interface. Just tell me what I have to be able to do. How do I actually do that? So now we go to our park, VA. Oh, and we have get a parking. You'll notice that it does nothing but return an empty park. Actually, we haven't implemented yet part yet. We haven't implemented anything on here yet. For the parks. We did implement. B. The city start the city staff is already done for you. But the park stuff, let's go through and say, what do we need to do? Well, if we're gonna get a park. And we know the query. Let's start by saying, OK, then let's let's do that thing right. Now, are we going to start from scratch like this? No, we're not. We don't know enough to start from scratch. So when they do two things, first, first thing we're gonna do is we're going to put our query in here because, whoops, wrong query, we're going to put our query from here. OK. And we're gonna do, let's see if I can get it to copy over to Intellij. Now we're going to put our query in here. And then 'cause all we need it for is just as reference. We're going to comment it out. That is the query that we're going to try to do. And now, because we don't know what we're doing. We're going to go to the city and say how do we do this same thing for a city? Or renew the state which I think is already done. OK, there's the state. Let's do it for a state because we don't have to deal with the the city abbreviation thing. You know which city is in? This is the code we're gonna do. I'm telling you, I'm gonna do it like this. I'm gonna say, OK, let's just do this again. In something else. I'm not gonna be trying to be fancy here. I'm gonna. I'm gonna say I don't know. So I'm gonna do this exact thing and see if it works. Except that I'm going to say, OK, I don't want. I'm trying to return a perk. Remember what I said in Java? I'm gonna return a park. So the first thing I do is I create that park. Or in this case I'm just setting it to no. And the last thing I do is I return the park. That makes sense to everyone. It was pretty good on that. Now I know I need to have. This SQL statement. But what I want is this SQL statement. So I am going to go ahead and just. But. Say. String SQL. Equals votes. My SQL statement in I get rid of my comments. I'm gonna put it all in as one. I'm not gonna get fancy here. I'm gonna just. Just the way they did right here. It's just too much for me. Do that. Do that now look at what we did in here. We said we're state abbreviation. Equals? I need to do that thing I'm going to say, OK, I don't know what the. Get rid of that for a moment. And we put a question mark right here. And then put a; After it. So at the end it's going to be way area where I wear park ID or whatever just the way they said where state abbreviation equals whatever. I am just copying and trying to make do that same thing. Right. So now I go ahead and I call my query for roset. With my SQL statement, which is now the one that I know worked when I was over in PG admin. But instead of passing the state abbreviation, I'm going to pass in the park ID 'cause. That's the thing I know I have. And notice that in this case I don't even care what type it is, where there's ones a string I I just pass in the thing that is going to match that? I let. Jabba do his fancy stuff and figure out how to what to map this to, or how many different possible, you know, overloaded statements row in order to do this I'm just going to say I see give my SQL and for every? That I have in here I'm gonna put in a value. So now I have my select statement. I have my query for rosette, so I'm gonna call the database and you got a series of roads. Everyone good on that so far. If you're not, feel free to ask. I'm just. And I'm not saying, could you do it? I'm saying, could you copy and paste it do it right? This statement right here, all that says is there is at least one result. That happens. I only care about the one result, but it there is at least one result. If there are no results, I'm just gonna. I'm gonna drop through passed my if test. When they return, park, which is set to null. And we're going to be done. I couldn't get the part because I got no results from my day. That's the only kind of validation I'm doing. I don't know, it just wasn't one. And finally I get 20 right? How do I actually create the park from my row? I need to change this to park. Macro to park. Try. We're almost done. With this one, we spelled. Project is project is Cobb, Gect is Cobb, Gect is here. This is what I'm trying to fill in. I know that. I am. Going to create an object here I need to actually write math road. I'll use it over and over again in here, but that's I need to write that thing that actually maps that row of those row columns. Back into my my Java object. We already the reason it's not showing up with a red squiggly is 'cause. We went ahead and created it for you. It just has nothing in it. I don't really want to do this. OK, so now? Alright. Great. We're in good shape. Let's do this and I'm going to go back to the same logic. Let's go to our state 'cause. That's the one we have you from. How do we do this? We did it like this. It's easier to type it brand new, but if it's if you want to be sure, just do it like this. Just just copy and say we're just going to go into our park. Find one that works. And. And start over. Or returning a park. So the first thing we do. In our park. Find new park. Last thing we do, we return that perk. All right, now it's just the stuff in between. That we have to do. And we know by looking here. What the things are? He knows that there are getters for all of them. Hey there setters going the setters at this point. So we go in here and we say fact, I might even do this because I just wanted to make sure I cut. I did everything. I might go back into here and say, let's just say these are the things I'm going to have to do. OK. I will comment it. I am big on. Putting stuff over here and leaving comments for myself and figuring out what I have to do, and then I'm just going to do the things one at a time. So I am going to say, OK, our first thing here is a park ID. K. And I'm going to set. Park ID. Remember, we know there are park getters and setters. Is Park idea string? And remember what Park ID is? It's a long it's a long. So we're going to do along. OK. And now we're going to say in my SQL database it is called park under score ID, so I'm going to call it that. So that will get my park ID. Dan, I'm gonna do the next one, say park. Now I'm doing is she's just getting rid of him here, perk. Got it. Set. Park name. And it's going to be called Perk name. This one is slightly tricky, so I'm just going to say date. OK. Or what is it called date established, right? We'll get to it in just a second. Area Area is a double. I'm gonna say park dot. Set area. Right now, it's doubled. We're gonna say row set dot get double. And we're going to give it the name, which is area. And finally, we have the Boolean, so we say park. Start set has camping. Rosette. Dot. Get boolean. Right. And then here we're going to have has camping. So all we are doing in each of these is just copying over from my row that I in my result. I'm just copying it into the various. Well I'm I'm setting using it setters but I'm copying in the values to make a fully complete park. That has all the values. That's all I'm doing. The data established. Dates are just slightly trickier. If the. Make sure I am looking at the right. Yeah. There are. This is the sort of unsafe, slightly unsafe way to do it. And then there is a Safeway slightly safer way to do it, but. We're going to say. First of all, it's just what you might expect. It's going to set, you know, set data stablished. Ascent data Stablished is a type. Let me see if it showed it here data siders is a type local date. That's what we need to get. But local date you can't just there isn't a gap local date. Add So what you have to do is. This is our chaining thing. We're going to say row set just like we're used to. Whoops. Sorry. Wow. You doing? That get. We are gonna get a date. So, so far, so good. We're gonna give it data stablished. Alright bye. This is not enough. It does not return it that. Let's see if it get the. It returns a Java dot SQL dot date. That is not directly equivalent to a local 8. But there are methods that you can call so you can change this. Remember, we do this in Java where we say two local date. Right. This works and actually is probably all you should use for now, but just be aware that if you had a null date. If the data stablish is allowed to be null. You could possibly get a null value here for the rosetted getdate, in which case it would crash with a null pointer exception. When you got to the local date. Candidate you candidate. You can do that if you want to be secure. Secure is you just say date. 8. My date. Import the class you wanna do it from Java dot sequence. This is one case where you don't want to do it for Java dot util, which is normally what you do. I can do this. Say my date equals the rosette dot getdate. Try. And then you can say. If my date. Not equal. Jamal. And then just put the date. Then you could say you're my date. This is the safer way to do it. Be aware that if you do that then you you don't have it a date value, a local date value, and you're delete just left with a default. So you might want to think about what you want to do with the default. But this at least will not crash for the for the help line are exception. OK. Now, sorry, go ahead. You said you said default date. Is that something that's set up at the database level in this if we are doing? So I'm not exactly sure where Matt Map Road to Park is called. It's called in maybe three or of multiple places. I think in this could be called multiple places. OK so I when you say the default value is set for my date. Java the default value for the date has nothing to do with the database unless I unless I have some way of getting it from the database. I'm right here going to get a null value so all that I'm doing is not setting the. The data is stab Liszt, which means that it's a purely job at issue about what happens with the local date. That is. That isn't given a value. OK, OK. I mean good question because it's not, yeah, there's nothing in your database which says right now is there a default? I mean in theory you could have something which went out and queried your database for a default date if there was one or something like that. But you don't have an upset, not right place but. But the reality is this is just saying I will never set. I will never call the setter at all. So I'm left with whatever Java thinks the local date is when you it's done by default. OK, OK. I have now filled in each of these values. And so I will tend to delete this because that is only to help me get through. I now have a map road to park. Does everyone understand what this the value of this map row to? You know park or state or city or whatever. Do you understand what this is doing? All of you. You're sure you really know what this is doing? It's OK to say no. I mean, I give what's going on, where we have done the query from the database and now we're taking that information and parsing it is there we can you show us the step before this where it would be like how are we getting? Like how exactly? I'm not quite sure I get how? How is Java interacting with the database? UM 'cause. This is how we deal with the data, but how? What is the UM, the SQL ROWSET and all that stuff? Or how does that work? That is, that is that stuff I I don't need to concern myself with right now or you don't need to concern yourself with. But but this is where there's a whole. I think hold on. Let me just make sure this was where in your. Hold on a second. There was a nice thing somewhere which which showed the layers of. OK, it's this whole idea, OK? Your Java application and if you compare this to this, they happen to have done it vertically as opposed to horizontally. This is your Java application and your data going back and forth in your DL in here, what they're saying is that the level that you're interacting with. It's gonna be this JDBC template. It has all the IT has. The JC the database driver has a whole bunch more crap underneath. They are making that unnecessary for you to learn. And it would be hard to understand anyway. 'cause there's certain amount of it is public and you know it's just it's got a whole bunch of garbage in there that goes out and says let's actually do our stuff. Now I will say this is this is a good opportunity to stop and say. It does mean that. We have to like, think about it. We used start up PG admin. Can you connect to the database? What do you mean to give it? First time you open up PG admin, what does it ask for? Password, password, password. So how is it that our Java thing is not getting going in and asking for that password? Right. Well, the answer is because in in a violation of any sort of security whatsoever. The work looks. Let's let's look for the password that we know it is ghost. Fresh, right? So right down here in our main, very, very first thing that we do. And again, you don't need to worry about a lot of these details. Most of the time you'll get through them one time and you'll copy and paste from whatever else. But our main method is right here. It creates this thing called a basic data source. That is provided by spring, right? And it is a data source and that data source we have to set the URL and that's how we get to the right database. We have to tell it what our username is, which is by default Postgres in here and on your machines it's going to be a password Postgres 1. When I say this is insecure, it's like don't do this in your code 'cause you've just put both your username and password right there in your code. These things are really getting a poll from environment variables and cookies and like, don't worry about it, they're gonna get pulled from somewhere. We're just trying to do a demonstration here. So you don't care. The only thing I will say is you could put them as part of your URL and that's the thing you absolutely never. Right. You could, but it just totally insecure because it means that that the message you're sending out to the databases in clear text has the password, which you don't want to do. If you said it like this, at least it's never sending it out in clear text. It does its thing and the data is first logic, but we have to have these or we would never connect to. First of all, we wouldn't know which database. And that's the the URL tells you what databases. Where is it localhost just means my local machine and this is particular port. And then it gives it the name of the database. But if I wanted if you wanted to go and copy this and work on your own, the Pet shop database. It would all work, you would just need to go and change this to pet shop or whatever it was called or meetups that you know or whatever. If you want to do it, you would just copy this sort of thing and say you know I don't, you know I don't need to know exactly why this work, how this works, but this is the stuff that works that connects. Because you have done all this and because in the beginning of your. Heart peeo or JDBC Park, PA oh, you put in these two lines. The magic happens that you don't need to know about and then when you actually do your query for rossetter, whatever it goes off into. possibly hundreds of lines of code that you have to look at

g Let's create a real park objectivist each said no, and then we return it. So way back to our logic, but we wanted to get a park. We now have implemented gather park. So in our. People have to rerun this thing. Probably still running it which? So let's rerun our CLI. See if it already has logic commit for. OK, so we want to view or modify park information. So we're going to say 3. Which state is it in? Right. We're gonna say Ohio. We got problem here because there are no parks in selected state. Are there really no parks in Ohio? Now. What does this mean for us? You know how. Yeah, I mean like. It is giving that error because when we go into JVC parking info. We are the programmers and we haven't created the method. So it just returns a new empty arraylist and returns it. So it's gonna say that for every state that it's gonna say this. So we now need to do this same job that we just did. It's gonna be a little easier 'cause. We already have our macro to to perk. Let's go and look at our, let's say in our cities, we try to find an example that is similar when you're trying to say what do I do, go and find a similar example. I happen to know that I that get cities by state is kind of similar. To get perks bicet. Right. So I just happen to know that, you know, look around, figure out which one it makes the most sense and say, alright. Hang now. Gonna go. I'm gonna say, where's my PERC information. I need to do the thing here. So I need to change all the stuff I need to say. In the park. Perks. New Arraylist and you just say OK, what are the things we need to have in a park? Go two ways. Here we go over here and we say OK. Should we already figured out we need these things? Right. So let us. All about the. Call it cheating. Call it whatever you want. But I have all about the. You know. Get it done 'cause you have it. We need to go from the park. And we need to say where the state abbreviation. We still have the same state abbreviation for students. Does everyone understand where this? Thing is gonna go. If anybody is not sure again right now, go ahead do the. Do the queries in here and make sure you know what you're going to get. I should be getting. The same results here as over there. OK, state abbreviation does not exist. So what is it called? Oh, we we don't, because what do we have here? Can we get to the state abbreviation now? What do you park? State, State Park. My life? It sucks, OK? So. Somebody walked me through. What do I have to do? How would I join? Something here? What am I joining you too? Join Park state. On. Hurt State dot airpark dot. Mark State dot Parker. OK now. And then where was my where statement going to be? Per state. Yeah, it is fine to just try this thing. Here and see if I get enough. Maybe not. OK, select Park, ID, perk, name, whatever from perk. OK Park ID is ambiguous now, so I have to change this to say. First, not perfect. Again, it is one heck of a lot easier to do this in SQL and get it right. And then copy it over and change it OK. Only one part that's just straight. OK, let's say let's try something different. Just 'cause. Parking kind of, you know, state. What's the state that you think of as having parks, not California? Let's try that. Alright, good. It's got four of them. That's great. So this is the thing we're gonna try to do. So what do I do? I do exactly the same, copy the thing over. So let me put it over here. Right. Now I'm gonna. Slightly differently, here I'm going to go back to here. I'm going to do. I'm gonna say. I often do my. I hadn't, I say. And then at the end of each one I put the. OK, so I went with that. Actually this is I should put a space here because. Sometimes the next row doesn't. You don't want to run together, OK? And then I'm gonna say plus wear whatever. Whatever. And I need to put my. So this is my SQL statement. What do I need to change in here so that it works in this? UTI needs to be a question mark. Right now, here's where I want you to understand. It is not in quotes, even though the event end result will be in quotes. There is a very important reason why use the question mark here rather than simply concatenating in the state abbreviation. Like I could at this point, say alright, let us do we want. Plus state abbreviation. Thanks. Plus slash Hey. Or I could say wait a minute. I don't wanna do that because it's sequel. So the thing really should be like that. OK. You understand what I'm doing every other way, but what? I'm. Too many of them are in go. Ventually get down, OK. The first initial single quote that I have is the sequel part. And then I have an End Quote for the end of this string so that I can add another string and then I have the new string and the first quote is gonna be a single quote. First of all this is complicated. Whereas putting the question mark in was not constant. But there's a far more important reason. Far more important reason is. We don't have any control about what a user is putting. So what if the state abbreviation were? OK, what if state abbreviation? Equals. Hey or something like that? Somebody malicious this is called SQL injection. Somebody malicious who is putting in that state abbreviation. Could put in something that ended the strand. I'm not sure I got it alright, but I'm seeing end this string. Put a; Put in dangerous stuff. And because SQL just goes ahead and does thing after thing with the rights of whoever is calling it. You could completely screw up your database, or they could take it and they could get information from it. The return that they needed that they wouldn't have access to. Alright. You need something that will take care of this and. There are ways to sanitize your inputs and again I think somebody posted in the thing of the the FK CV one about, you know little Bobby drop tables that's that's the the the classic example of it. But the point is that you can do malicious things in here or you can do things that retrieve information that not supposed to. And the only way to prevent this is that you can escape characters in a string. To prevent it, like every quote, instead of just being a quote which might then be part of the query, you can escape it and the escaping. And this is just as simple as putting two quotes, but you don't want to be doing that job. You don't want to be responsible for that job. So you're gonna let somebody else do that job, which in programming is, you know, the most important thing in programming is delegating. You know, any of the craft work to somebody else. So in this case, you're delegating the crowd work to someone else, and the way you're doing that is you are. We will talk a little more about C1 Jackson, things like that tomorrow is that if I put the question mark here. Then we can let our JDBC template. Do all the hard, the heavy lifting when it does this, it will automatically sanitize the inputs to make sure that nobody can do SQL injection. So you are protected by the fact that it is done that. I can now get rid of this OK, because that was my. Getting rid of the right thing. Yeah. OK. So we have selected percote whatever, whatever we're perk. State abbreviation equals whatever, so I'm giving it this, giving it the full thing. We know this is going to work now because we ran it over on the other side. We're going to pass in this state abbreviation. And then while the results, now we have to add to parks instead of. Perks. I'm going to map grow the parks. Because remember, we're getting each park at a time and then adding it to our thing. We already implemented Map Pro to Park, right. Mistyping something somewhere. You got capital A and you list park parks. You equals new arraylist but where is it? It's not. I'm 41. I knew I had it somewhere. OK, there we go. Right. So it doesn't know what to do with that. So it's complaints. She's good. OK, does this thing. And finally we return our parks. Hey. So if everything has gone. According to plan. Now we can stop this and rerun it. OK, let's run our CLI. Now we are going to. What state is it in? We're going to say Utah because we know that that's the one we got. How many results do we have in Utah? We had five. It turns out I was not quite seeing the bottom there. We have five of them. So let's go back in here and see if we get five as well and here. And sure enough, we get our five parks in here. So we did that thing. We went from enacted showing any perks to it showing these perks. Now if we pick a park. OK, we did that thing too 'cause. We created the get perfect. So we got our. IP or established date or whatever it is. It says this park offers camping this whole thing, by the way. Which was built for us, so we they needed to, but it was done down in the. Oh, actually it might be done. It might be in here. There might be a display part, but if we could have done it in here and a two string. OK. So that then is our X. What else can we do? We can update , Or we could press enter. What are our choices? We could add a new park or add a new city. We already did add a new city. Let's look at that new park. Actually, let's take a 10 minute break. and then we will go through this before we hardly i want you to just follow it as send from the work is being done. The CLI is already pulling in most of the information you need. So when I get to. How do you get? You do get hurt yet? What do I do? I do something in here. They lose everything. Possibly. Hold on. Let me just make sure that I'm not looking at the wrong project 'cause. Yeah, it's quite possible. Wow, just a second. I'm gonna try it in here. I'm even working in here. That's OK. Stop right? Module 2. Sorry I got data security data test data access. In general, ignore the man behind the curtain. OK, back to where we were. Alright so. I had to get parked by state. I have my my map for me. I got my get park. Now I'm going to go to my create perk. Alright, so my create park is going to look a lot like what? Where would I go? Do I go to the state where? Where am I going to do? Create part. What should I look at? Either state or state. Liberty create city. We just did. Yeah, close to create city, I mean. I guess it's it's. It's it's similar enough. OK, so let's go to create city. I do. I'm really just. Copy and paste because I'm all into that kind of thing. Right. You may not find it easy to create whatever I'm I'm good with it either way. You want to do it, but this is for me the easiest way to do it, park. How do I know which things to do? Well, I'm gonna say this time I'm. I know pretty much that it's going to be the park name, date established area and has camping. I'm gonna steal them from there. Notice that again, I do not set the value. For the ID commonly used that I'm going to return the park ID. Make sure this is very critical. It happened to match up here, but doesn't hours. Make sure your question marks here match up to your. To the number of columns you're when it's best fide. And then I'm going to say Park dot get. Oops, don't worry, I went but. Park dot Park name. Dot get. Data stablished. Stab liszt. Area. Back. Chad has kept. One second. Alright so. I I'm gonna. I'm gonna do my thing here and then. I will come up with a new ID just like before. And. But to be clear, what I'm doing here is I am creating an insert into, creating my my insert query. I'm returning the new thing and all I'm doing is calling this query. I'm calling my regular query on a database, it's just like SQL, just like you've been dealing with, right? I get back to my new return and then the last thing I do is get parked. And that will then do. The. It will allow me to create a perk. Updating apart very, very similar. OK, I'm not leaving to go through it all, but we'll take a look at what it would take updated city. It's gonna look really familiar. It's going to be awfully similar to what you just did. You're gonna update city. You're gonna set the various values you're gonna put the question marks now here where they are. And you're going to say where the city ID is this. OK, This is why it's very important to have the city ID, because right now I actually need the city ID's. The reason I've been floating around with the city I have somebody. Let's say I went out and got that city. The reason why we need to know that city ID so badly is that now we're going to actually update that object. I don't want to have to go to a query to find out what that city ID is. 'cause I'm. I'm. Using it so I'm using the current city that I whether I created it or got it or however I got it, I'm gonna use that and I'm not gonna update that city. And it's just going to be an update city and set the values. In theory you could have and. This is an implementation choice, right? Right now this is going to update all the values in the city. If you think that it is important for your application. To have another. This thing here. Which updates not the entire city in all the stuff. But just. Like let's say, adds 100 people to their occulation. If that's important to your thing. Alright. You could what were the steps you would take? You have an object. In your model. You wanna add 100 to it? You know what the city is, right? All you would do in that case is you would say I go to my city DAO. When I say I should have one that is not just update City, but I should have a void. Add. Add. 100. 100. Two city. K. Some vague reason that's the reason why you want to do things. So you're going to have this add 100 and in this case you could just say I want a long cvid. Or you could pass in the whole city. I don't really care. Doesn't make any difference. We're doing something very fairly standard here. You would make this as an interface. Now anybody who wants to implement this is going to have to implement this logic right here. And. Just try and make sure you understand how these things work and so then. I'm gonna put in here and I'm going to say, oh, it's an override. I'm going to say. Add. 100. Two city which I may or may not have, and I'm gonna say along with city ID. OK. How is this gonna change? So since all you're doing is updating the population, uhm, you would just have it where it instead of being where it sets every individual part of it, you would just do update City, set up population equals. Uhm, continue. You'd have to do it like population equals population plus 100. Also, just so you know, online 6500 dispelled long, so that's gonna run to an issue. No doubt. What is field wrong 100 and add 100 to city there's it either missing there. So. I am going to change that. Now there's only one? Not only one? Is that I could do the city if our passing the whole city and but in this case, I'm going to say I don't really need all of that. So I am going to update it with just the. City ID. OK. OK. So you could have something like that. Now I say 100, it could be that you want to have an update that updates just an incorrect or just a counter and so that it adds one to it every time and that's how you would say every time I visit like a visit count. So every time a new user comes in, you can say that user has visited this page. You could increment it by sending off this query that said. You know updated by 1. The nice thing about something like this is let's say there are different people who might be calling it. Well, if it doesn't really matter which order it happens to hit, it's just going to add one 100 or one, you know, one to the thing I don't have to query it. And then return it. And the reason why that's important is that you've got to understand things like latency here, like if I go out and query the database. And then I go and updated it page. What is gonna stop that from somebody else has already go has a say that I wanna add one to the population. Now I go and I get my object and I work for a little bit with it and I want to update that population by one or I want to update that visit count by 1. Maybe three other people have already been in there, have already done that. So. I have to be careful when I update things that I end do not overwrite somebody elses changes that are already there. And there are ways. Again, we're going to transactions and stuff like that, but right now that's not included. So that's how I would add a new thing. I would, I would add the generic thing that I'm going to add, I would add to the interface and then my face, I would add these specific override for it. If I look at my. Here. How do I add my update? Should take moments here. It should be fairly simple. I'm gonna just do my thing. Go to my park. I'm gonna say add my perk. Update park. Per name set state abbreviation I have. Things I have I have today to stab Liszt. Boost. I had to be careful here, by the way. Not did I use the the names in my query foster where was I? I have to be very careful here that I use the names for SQL in my SQL part and that I use my names for Java in my Java part. Park ID. OK. So no. Probably have the wrong today I have data stablished but then I don't have a area it has. Say. Area. And. Hasskamp. I'm back. Yeah. That's the basic logic of what you're going to do. Delete. It's going to be straightforward. I'm not gonna show you because we're getting to the end of our time here. But if we had it delete. How simple is that going to be, right? It's gonna do that for the park. The one I want you to think about is how do you do an add park to state? Now I want you to think about it when I post the exercise, I will go ahead and have included that so you can see it. Remove the park from the state. How do you do it? Think about not how am I going to implement this in Java? Think about how I'm going to implement this in SQL. And then just look because you've got. The tools here to update to change to whatever else. So think about your SQL command. That would get rid of one of, you know, QWERTY or whatever it was. Whether it would add QWERTY to a different state. Hopefully not state this terribly far from Guam or Maine or whatever was that I was. OK. I'm losing track of it all, but. Think about how to add park and remove a park. And in about 1/2 an hour when I actually post this, I will will include the implementations in there so you can see. Again, reminder, there are exercises for tonight that will be due on Friday. Wait, what day is it today? Is it Tuesday? What day is it? Right? Really. Don't have this day we do Thursday. Is there do Thursday OK? Just my brain is completely Friday. That's cool too. I mean, Thursday is good. OK. So anyway, there will be homework. That is, there will be exercises there. That is Thursday. Tomorrow or there's reading for tomorrow. No quiz, no tutorial, and there will be no exercises tomorrow. So you get a little bit of a break. And then on you will have your exercises on Thursday that will be due Monday. And you will have a parish project which again I will tell you about at some point before Friday because of of the the current day. And if you want to get started on it by I may give it to you early, I have no idea whatever I'll do. I'll. I'll do something so that it's out of my hair before I have to travel, OK. Alright, this is all a lot to take in, but it's also. it just feels like more than it is if you actually go through a bucket at working in this and then go like keep two of them open and say here is the way you know it was done in my tutorial and in my you know. For the tutorial final, look at the exercise final. When I put it in there and then also look at the. And then look at yours and say, how do I how do I translate this? How do I do this? And remembering you always just getting back to a sequel and it's always just getting back to a thing that you already know how to do in C. OK, alright, have at it. Why? Thanks Ben.