**Now DynamoDB is a public space service, much like S3, but it's private by default. To access a table inside DynamoDB, you either need to give an IAM user, role, or group in the same account access to that table using identity policies.** Or alternatively, **if you want identities outside of your account to access DynamoDB then you need to have created a role inside your account and allow an external identity or anonymous identities to assume that role. So unlike S3, you can't apply resource level permissions.** **So you can't control permissions directly on a table. Now, from a resiliency perspective and this is really important to understand for the exam, DynamoDB is resilient on a regional basis.** **So when you create a table, DynamoDB stores at least three replicas of any data in that table when you use the service. So when you write to DynamoDB, you actually get a HTTP status code of 200 when you've written data to DynamoDB and it's acknowledged that it has stored it persistently. So if you get a status code 200 that's more applicable if you use in the APIs directly, you don't see that when you're using the console UI but for the exam remember, a status code of 200 means that you've actually written data to DynamoDB in a persistent way. So you've got three replicas of your data and they're stored in independent availability zones and DynamoDB handles the replication between the storage devices. So DynamoDB can survive the failure of an availability zone without any additional configuration.**

Now, in the next lesson, I'll be talking about how DynamoDB handles consistency and performance. **But from a resiliency perspective for the exam, you need to be aware that you do have at least three copies of the data that you store DynamoDB.** Now from the console, as well as being able to put items into DynamoDB by creating an item and get items from DynamoDB by clicking on a specific item like I showed earlier. **You also have access to the scan and query operations. Now, remember earlier when I said that an item can be up to 400 kb in size and you always read the whole item, well keep that in mind as we talk through the architecture of scan and query.**

But before I move on, I just want to add additional data to this table. So I'm going to pause the video, and I'm going to resume it once I've added additional records for two more weather stations. Okay, so I've added that additional data, we've now got weather data for three weather stations 001, 2, and 3 and I've maintained the format of each set of records, so we've got two items that don't have any additional attributes and then we've got two items, one of which has a temperature in Celsius and one has a temperature in Celsius and an amount of rainfall. So now I want to talk about this scan and query operations.

So we're going to start with scan and **scan is the most flexible of all of the operations inside DynamoDB. It doesn't actually need to be passed anything. You can simply run a scan operation on a table without any additional parameters and if you do that, it's going to list or retrieve every item in that table.** So that's important to understand for the exam, if you don't specify anything for a scan operation, it can be used to list every single item in the table. Now you can also, with scan operation, add additional filters. So if you're looking for a specific value for a specific attribute, let's say I wanted to add a filter and I wanted to show any items where the rainfall so rain is equal to zero. Well, if I wanted to do that with the scan operation, I could add that as a parameter, click on "Search" or execute the operation from the command line or APIs and it would show only those items but this is a really important thing to understand about the scan operation, and this does come up in the exam, even though I'm only showing these three items because I've added this filter, the scan operation has had to read every item in the table. It's taken that data that it's read from the table, applied the filter, thrown away the items which don't match this filter, and then delivered the remaining ones to me on my screen. But I've still consumed the capacity for the entire table so that's why it's so important to understand that if you are using items, which are the full 400 kb in size then a scan operation can consume significant capacity. **So the scan operation is super flexible because it can filter on any attribute and any attribute value in an entire table and it isn't limited to looking at specific partition key or sort key values. It can use any attributes, but it does consume a lot of capacity units. It's not efficient for larger tables,** and I'll be explaining what capacity units are in the next lesson.

Before we move on, though, I want to talk about the query operation. **So query is a much more efficient operation than scan. Query operations allow you to perform lookups on the table without having to read every item as long as you only want to perform filters based on either the partition key or the sort key. So query can only ever retrieve data for one single partition key**. So let's say I wanted to retrieve all records for station ID 0001. Well, if I did a search, it would pull all of the items from the table which used that partition key and **that would only consume the data the total data size for all of those four individual items. But I could also add a filter based on the source key**. So let's say that I wanted to check for a specific date and time value. Let's say the first one 8:00 a.m. Well, I could say that I wanted the sort key to equal this specific value and when I search now it would only retrieve one item from the database. **Now, if you use a query operation and you filter only based on the partition key and the sort key, then this is the whole capacity units that it consumed. It only consumes the capacity units for the number of items highlighted. So if you pick a specific value for the sort key, you'll get one specific item. If you specify a range of values for the sort key, then you'll consume that specific range so only the capacity for that range of items that are returned.** So an example of this could be that I want to check only the items where the date and time strings begins with, and then let's say that I eliminate everything after 08 for the time value. Now, if I search I'm going to pull both records for 8 a.m. so 8 a.m. and 8:30. **Now, in this case because I'm using query and because I'm filtering based on partition key and sort key. I only consume the capacity for these two items, so two items rather four. So that's why query is really powerful because it allows you to efficiently retrieve data from a DynamoDB table but a limitation is that you could only query on one single value for the partition key**, so I couldn't, for example, use a query operation to show me all data points that occurred at 8 a.m. I can only do that for a single partition key value if I wanted to search across different partition keys I'd have to use a scan operation. **So with a scan operation, I could easily add a filter and then for the filter, I could say date and time. I could say the string begins with** so exactly the same as I did in the previous example. So the string begins with 2019-08-05 for 8 a.m. and if I do that it's going to show all records which occur between 8 a.m. and 8:59. Now the problem, of course, is that **because it's a scan operation, it's actually had to read the entire table and then it's filtered out the results**. But that illustrates how scan is a lot more flexible than query but query is, of course, more efficient. For many exam questions, **query should always be preferred.** But for the exam, remember, **query can only ever operate on a single partition key value, and you can only filter based on the sort key.** Okay, so this is the end of part two of this video. Go ahead, maybe take a small break or grab a coffee, mark this video as complete, and when you're ready, you can join me in part three.