**Programmatically Utilizing S3 Select**

**Introduction**

In this lab, we'll work through completing a filtering function for a simple web application that displays user data. To accomplish this, we will edit a Lambda function and write the necessary code to utilize S3 Select.

The Development team at your company is working on a proof of concept for a directory of all the employees for the company. They have a very basic web application working, but want to add a filtering feature to the application. They've run into problems attempting to get S3 Select working to implement this, and have requested your assistance. They've provided a Dev environment with an S3 hosts web frontend, API Gateway, Lambda function, and another S3 bucket which contains the data for the application. You need to update the Lambda function to be able to filter the data and return the results.

**Note:** For a full walkthrough of the Lambda function components and how they work, be sure to watch the lab videos.

**Additional Resources**

* [S3 Select Object Content](https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html#S3.Client.select_object_content)
* [S3 Select Documentation](https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-glacier-select-sql-reference-select.html)
* [Lab Github Repo](https://github.com/linuxacademy/Content-AWS-Certified-Data-Analytics---Speciality/tree/master/Lab_Assets/programmatically_utilizing_s3_select/lambda)

**Solution**

Log into the AWS Management Console using the credentials provided for the lab. In another browser tab, open the **random-users** website provided for the lab.

Each website URL is unique and contains the account number for the account you log into.

**Investigate the Lab Environment**

1. In the AWS Management Console, navigate to S3 using the *Services* menu or the unified search bar. You should have two buckets provided:
   * The random-users-<ACCOUNT\_NUMBER> bucket provides the frontend user interface.
   * The random-users-data-<ACCOUNT\_NUMBER> bucket provides user data.
2. Open the buckets and review the objects you'll be working with. Feel free to make any modifications you'd like to the data.
3. In the random-users-<ACCOUNT\_NUMBER> bucket, select an object and use the *Actions* dropdown to select **Download**.
4. Open the file and review the user data.
5. Copy the random-users-data-<ACCOUNT\_NUMBER> bucket name.

S3 Select can only select and run the SQL on the contents of one object at a time. The advantage of using S3 Select is that you use a lot less memory and S3 does all the computational work for you to filter the data.

**Update the Lambda Function**

1. Navigate to Lambda using the *Services* menu or the unified search bar. You should have two functions provided.
2. Select the **Users-primary** function.
3. In the *Code Source* section, select **function.py** and review the code. You can see that you're importing boto3 and JSON. You'll need to modify the filter\_data function.
4. Paste your copied random-users-data-<ACCOUNT\_NUMBER> bucket name on the s3\_bucket = line.
5. Click **Deploy** to deploy the change to the S3 bucket.
6. Navigate to the *random-users* website and refresh the page. Your user data should now load. The Lambda function iterates through each JSON file and compiles the data, then returns the data to the site.
7. Navigate back to S3 and select the random-users-<ACCOUNT\_NUMBER> bucket.
8. Select the **randomusers.js** file and use the *Actions* dropdown to select **Download**.
9. Open the **randomusers.js** file and view the variables. You can see there are filter variables that correspond to the filters on the *random-users* website. The *self.filterUsers* variable is the piece of code that sends the request for the filters using map values.
10. Navigate back to the Lambda function.
11. Copy the *function\_solved.py* code provided in [GitHub](https://github.com/linuxacademy/Content-AWS-Certified-Data-Analytics---Speciality/tree/master/Lab_Assets/programmatically_utilizing_s3_select/lambda) and paste it into the Code Source text editor. This code must be able to handle the provided filter variables as well as empty filters.
12. Copy the random-users-data-<ACCOUNT\_NUMBER> bucket name again and paste it on both s3\_bucket = lines in the code.
13. Click **Deploy**. The code iterates through and filters the data using the s3 select API call.

If you do have filters, the s3 variable collects all the objects in the bucket and iterates through each object to make an S3 Select call against each one. The keys for the objects are then filtered out, and a filter string is generated to act as the WHERE clause in the SQL statement used to filter your data.

**Test Web Application Filtering**

Open the web application with the URL provided and test each filter, as well as combinations of filters, to ensure they function as expected.

**Note:** The filter is case sensitive.

1. Navigate back to the **random-users** website and refresh the page. All 1500 employees should load.
2. Test the filters:
   * In the *Last Name* filter box, enter a last name (e.g.Sanchez), then click **Filter**. You should now only see employees with the last name you entered.
   * In the *First Name* filter box, enter a first name from your current employee list (e.g. Marco), then click **Filter**. You should now only see employees with the first and last name you entered (e.g. Marco Sanchez).
   * Clear the text from the filters and click **Filter**. You should see all 1500 employees again.
   * In the *Age* filter, enter an age (e.g 60), then click **Filter**. Unlike the other filters, this variable returns integers. You should now only see employees of the age you entered.
   * In the *Country* filter box, enter a country (e.g.Spain), then click **Filter**. You should now only see employees who live in the country you entered.
   * In the *State* filter box, enter a state (e.g. Asturias). You should now only see employees who live in the state/country you entered.

**Conclusion**

Congratulations — you've completed this hands-on lab!