Assignment 2

Weight: 40% of your final mark

Due: Week 6 Monday (5 June 2023) at 23:59 PM QLD time

Specifications

In Part A, your task is bda function that is invoked after a “Book.json” object is created in an S3 buck he Lambda function reads the content in the “Book.json” file and inserts them table. In Part B, your task also includes preparing an employee salaries da me pre-processing techniques. Your submission will be evaluated based on:

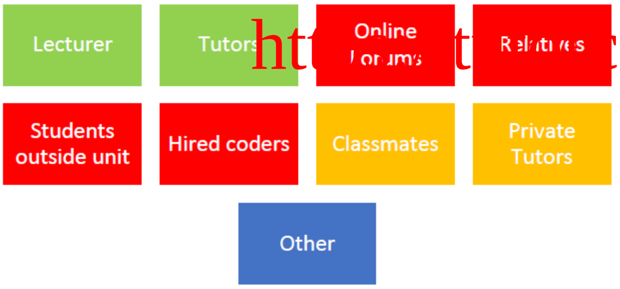
- whether i followed;
- correct functionality;
- correct implementation; and
- comments inside the program.

Getting Help

This assignment, which is to be completed individually, is your chance to gain an understanding of the fundamental concepts of Lambda and Lambda events on which later learning will be based. It is important that you master these concepts yourself.

Since you are mastering fundamental skills, you are permitted to work from the examples in the MySCU site or textbook, but you must acknowledge assistance from other textbooks or classmates. In particular, you must not use online material or help from others, as this would prevent you from mastering these concepts.

This diagram will help you understand where you can get help:



Encouraged **Attribution Required** **Not acceptable** **Ask tutor**

Be aware that if you do get help from one of the red sources, you will be reported for academic misconduct, which may have serious penalties. Please visit the following link for the guidelines: https://policies.scu.edu.au/download.php?associated=1&id=326&version=3

(Go to the next page)

ATTENTION: Assignment 2 must be completed in the UA-provided AWS account. Personal AWS accounts will not be accepted, and there will be a significant mark deduction. If your AWS account is not accessible for any reason and the marker cannot check your app, you will also lose significant marks. No excuse will be considered. You must use Java and relevant AWS SDK to implement your app.

Details: The details of the assignment tasks are given below.

Part A: (30 marks):

- Create an **S3** [illegible] uusername-a2-bucket”.
- Implement a [illegible] ed “yourscuusername-a2-function”. Add a “create object” **trigge**[illegible] scuusername-a2-bucket” to this Lambda function.
- Use the “LabR[illegible] **role** of the Lambda function. You do not need to update “LabRole” and do not need to add any inline policy.
- The Lambda function is invoked when a new object named “Book.json” is created/uploaded in “yourscuusername-a2-bucket”. The “Book.json” includes a JSON array with more than one JSON objects. An example of the JSON array is given below. A sample Book.json is attached with this assignment. You can add new elements to this JSON file.

  [{"Title": "Never Get Angry Again", "Author":"Felix Daniel" , "Year": 2019, "Price": 25},
  [{"Title": "The Power of Letting Go", "Author":"James Clear" , "Year": 2016, "Price": 28},
  [{"Title": "You Become What You Think About", "Author":"Vic Johnson" , "Year": 2018, "Price": 24},
  [{"Title": "Think and Grow Rich", "Author":"Napoleon Hill" , "Year": 2017, "Price": 35},

- Create a Cloud9 app named “yourscuusernameA2App”. Implement a handler named “handleRequest()” with appropriate parameters to receive S3 event notifications.
- The handler **retrieves** the bucket name and object key associated with the S3 event and conducts the following tasks **only for the “Book.json”** object. For any other object created/uploaded the handler will not do anything following.
- The handler **reads** the JSON array in the “Book.json”, **parses** the JSON objects, and **inserts** the values of the fields to a DynamoDB table named “Book”. **The books in the JSON file will be inserted if only the book doesn’t exist in the table**. Make sure you perform this checking before inserting the books. The schema of the table is given below, where **Id** is the primary key. You can add a test item to the table as shown below.

  Book (**ID:** *Number*, Title: *String*, Author: *String*, Year: *Number*, Price: *Number*)

- The handler must retrieve the **latest value** of “Id” from the DynamoDB table before doing any insertion.
- You must use AWS SDK v1.x, appropriate exception handlings, loops, and **separate methods for insertion and reading the latest Id**.
- Add detailed **comments** to your Java source code. JSON parsing must be done by “org.json” package.

Testing the Lambda function: You can test the Lambda function by just simply uploading the **Book.json** into the specified bucket.

Solution Helps: Check the following hints for implementing Part A.

- You need to use the “org.json” package to retrieve the JSON array and parse each JSON object. Please check this link: https://stleary.github.io/JSON-java/index.html. You can use the latest maven dependency from here: https://mvnrepository.com/artifact/org.json/json.
- A *JSONArray* is a collection of *JSONObject*s and a *JSONObject* is a collection of values of each data filed. You need to understand this concept and parse the fields.

Part B: (10 marks):

- You need to create, publish, and import a recipe named “yourscuusername-recipe” to complete the following tasks on the “employeesalaries2020.csv” (attached with this assignment) in the Glue DataBrew project. Create a project named “yourscuuser[illegible]A2Project”.
  (The employe[illegible] public dataset and was taken from https://catalo[illegible]nployee-salaries-2020. Some of the rows are removed to limit the usag[illegible]
- Replace all mi[illegible]umns which have missing values. Use the most frequent value and nur[illegible]npute the missing value.
- Identify the d[illegible] them with “yes or no” values.
- Flag outliers for any one column as “yes or no” values.
- Compute the total salary (“Base Salary” + “2020 Overtime Pay” + “2020 Longevity Pay”) for each employee, and add a new column at the last position (after “Grade”).
- Compute the total salary (“Base Salary” + “2020 Overtime Pay” + “2020 Longevity Pay”) for each department (“Department”) and add a new column to the dataset.
- Use “LabRole” while creating the project. Upload the “resolution.csv” to “yourscuusername-a2-bucket”.
- Import the recipe as a JSON file.

Solution Helps: No solution hint is required for implementing Part B.

Submission Checklist: The marker will access your AWS Academy workspace (provided by the UA), check your app, and directly mark your app from there. You have to zip the app project folder and submit it in Blackboard as well. Blackboard submission list:

- A zipped app folder for the Handler app.
- The recipe JSON file.