Question 3

We would like you to setup an environment on AWS with the following components:

Part 1 - An EC2 instance

When creating an EC2 instance,

Resource: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html

Instance type: t2.micro - since it was free tier

AMI: Linux
Created a key-pair and saved it as a .pem file.

Q. A script should start on boot that will read the ID of the instance or assign an ID if it is the first boot, the ID should be a 6-character hexadecimal.
Q. Register with a Lambda function to confirm ID and IP address, with a flag denoting if this is a new registration or an update of an old ID.

Part 2 - The Registration Function

Q. A Lambda function that acts as an API that takes an ID, IP Address and a flag (to signify a new registration or update)

What is a Lambda function?
Resource: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html
(What is AWS Lambda? AWS Documentation)

A Lambda handler function is the entry point for AWS Lambda to start executing the code.
There should be an event that triggers the Lambda function.

In this question, there are 2 events that trigger the Lambda functions,

- The registration function: The event trigger is the HTTP POST request made by the EC2 instance's startup script. The event includes the instance ID, IP address, and a flag indicating whether it is a new registration or an update.
- The key updating function

Creating a Lambda function using Python.
Resource: https://docs.aws.amazon.com/lambda/latest/dg/lambda-python.html
(Building Lambda functions with Python AWS Documentation) Contains the basic code using boto3 and the definition of a Lambda handler function.

```
import json
```

```
import boto3

def lambda_handler(event, context):
```

Q. The script should then, on loop, write to a time series database, of your choosing, the timestamp and a 'secret key' hash.

What do we need to run SQL statements in a Lambda function?
Resource:
https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/rds-data.html
(RDSDataService Documentation) Contains the definitions of the methods client() and execute_statement().
Says that you need to import boto3. Has a sample code to create a service client as well.

```
rds = boto3.client('rds-data')
```

Additional resources about the RDS service,
https://boto3.amazonaws.com/v1/documentation/api/latest/reference/core/boto3.html
Has the definition of the boto3 source code,
https://boto3.amazonaws.com/v1/documentation/api/latest/_modules/boto3.html#client
-client(service_name, region_name=None, api_version=None, use_ssl=True, verify=None, endpoint_url=None, aws_access_key_id=None, aws_secret_access_key=None, aws_session_token=None, config=None)
-service_name (string) – The name of a service, e.g. 's3' or 'ec2'.

To take the ID, IP Address and a flag, the following code is used,

```
    instance_id = event['id']
    ip_address = event['ip']
    flag = event['flag']
```

Q. Function should check the ID exists

- ***If it does and it is an update call, update the IP address of the ID in the RDS, then return a true flag.***
- ***If it doesn't and it is a new registration call, add it to the RDS, then return a true flag***
- ***Otherwise, return a false flag***

First check the flag

Store a query to be used in the execute_statement function().

```
if flag == 'new':
        query = f"INSERT INTO Instances (InstanceID, IPAddress) VALUES
('{instance_id}', '{ip_address}')"
```

```
    elif flag == 'update':
        query = f"UPDATE Instances SET IPAddress='{ip_address}' WHERE
InstanceID='{instance_id}'"
    else:
        return {'status': 'error', 'message': 'Invalid flag'}
```

Then, Check if the ID exists. Only if the ID exists, execute the queries.

Resource for execute_statement() function:
https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/rds-data/client/execute_statement.html

The execute_statement() function needs the database name, the resourceArn and the secretArn. These were defined at the top of the code.

```
database = 'TimeSeriesDB'
    resourceArn =
'arn:aws:timestream:us-east-2:010928184835:database/TimeSeriesDB'
    secretArn =
'arn:aws:secretsmanager:us-east-2:010928184835:secret:RDSSecret-p9BRKu'
```

The Arns were taken from the database information

ARN (Amazon Resource Name) are unique identifiers used to specify particular AWS resources across all of AWS.

arn: Indicates that this is an Amazon Resource Name.
partition: The AWS partition (e.g., aws for AWS regions, aws-cn for China, aws-us-gov for AWS GovCloud).
service: The AWS service (e.g., s3, ec2).
region: The AWS region (e.g., us-east-1, eu-west-1).
account-id: The AWS account ID.
resource-type: The type of resource (e.g., instance, bucket).
resource-id: The unique identifier for the resource.

```
# Check if the ID exists in the database
    try:
        rds.execute_statement(
            secretArn=secretArn,
            database=database,
            resourceArn=resourceArn,
            sql=query
```

```
        )
        return {'status': 'success'}
    except Exception as e:
        return {'status': 'error', 'message': str(e)}
```

Part 3: The key updating function

Q. A separate Lambda function that will read the time series data from the EC2 instance, confirm the secret key with the one stored in the RDS

```
    query = "SELECT InstanceID, SecretKey FROM Instances"
    instances = rds.execute_statement(
        secretArn=secretArn,
        database=database,
        resourceArn=resourceArn,
        sql=query
    )['records']
```

A query is written to get the instanceID and SecretKey columns of the Instances database in the RDS.

The execute_statement() function is used.
Resource:
https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/rds-data/client/execute_statement.html

The response of the execute_statement() function contains a dictionary with ‘records’ and ‘columnMetadata’ parameters. Here we are accessing only the records.

Q. It will then generate a new secret key and access an API to update the secret key stored on the EC2 instance.

The secret key is a hexadecimal string.

So to generate a new hexadecimal string, use sha256 from the hashlib module in python. Hashlib.sha256 uses the old secret key to create a new one. Making sure that the new key is derived from the old one but is very different from the old one as well.

```
new_secret_key = hashlib.sha256(stored_secret_key.encode()).hexdigest()
```

Send the POST request to the correct url of the instance and with a payload dictionary.

This requires an API to be created which was not done.