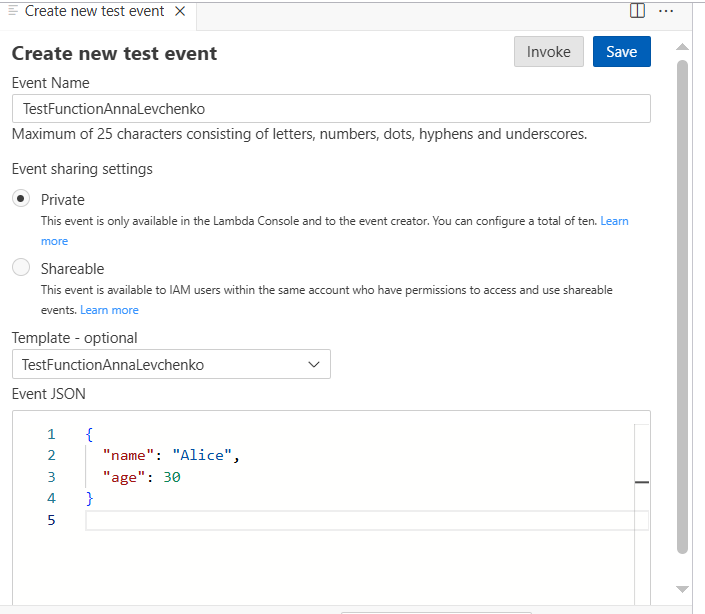
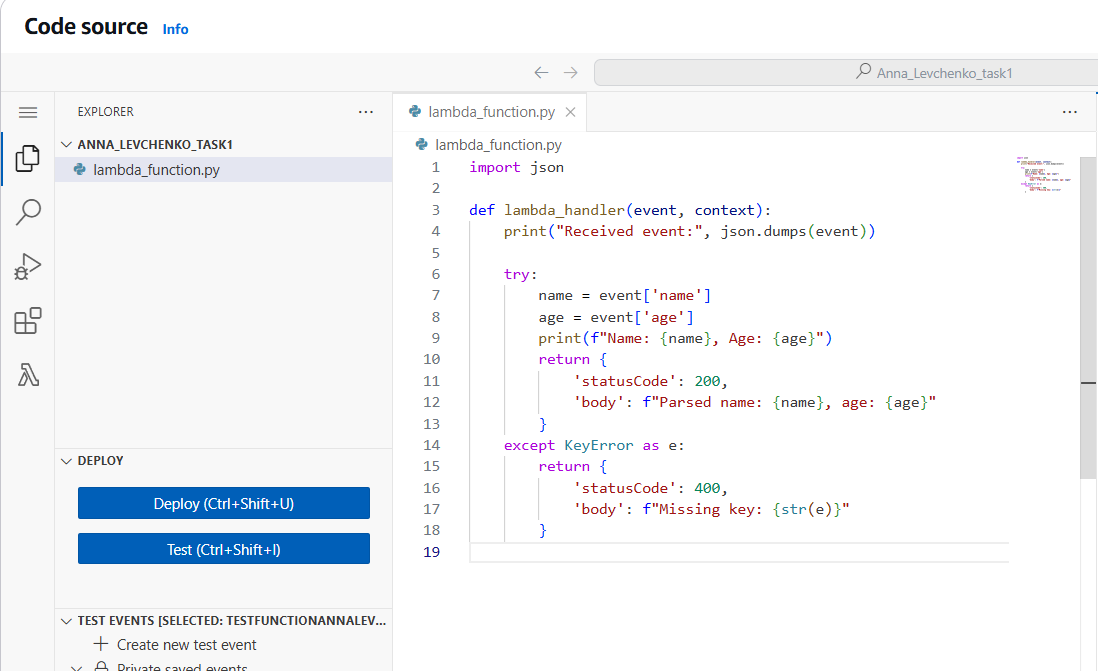
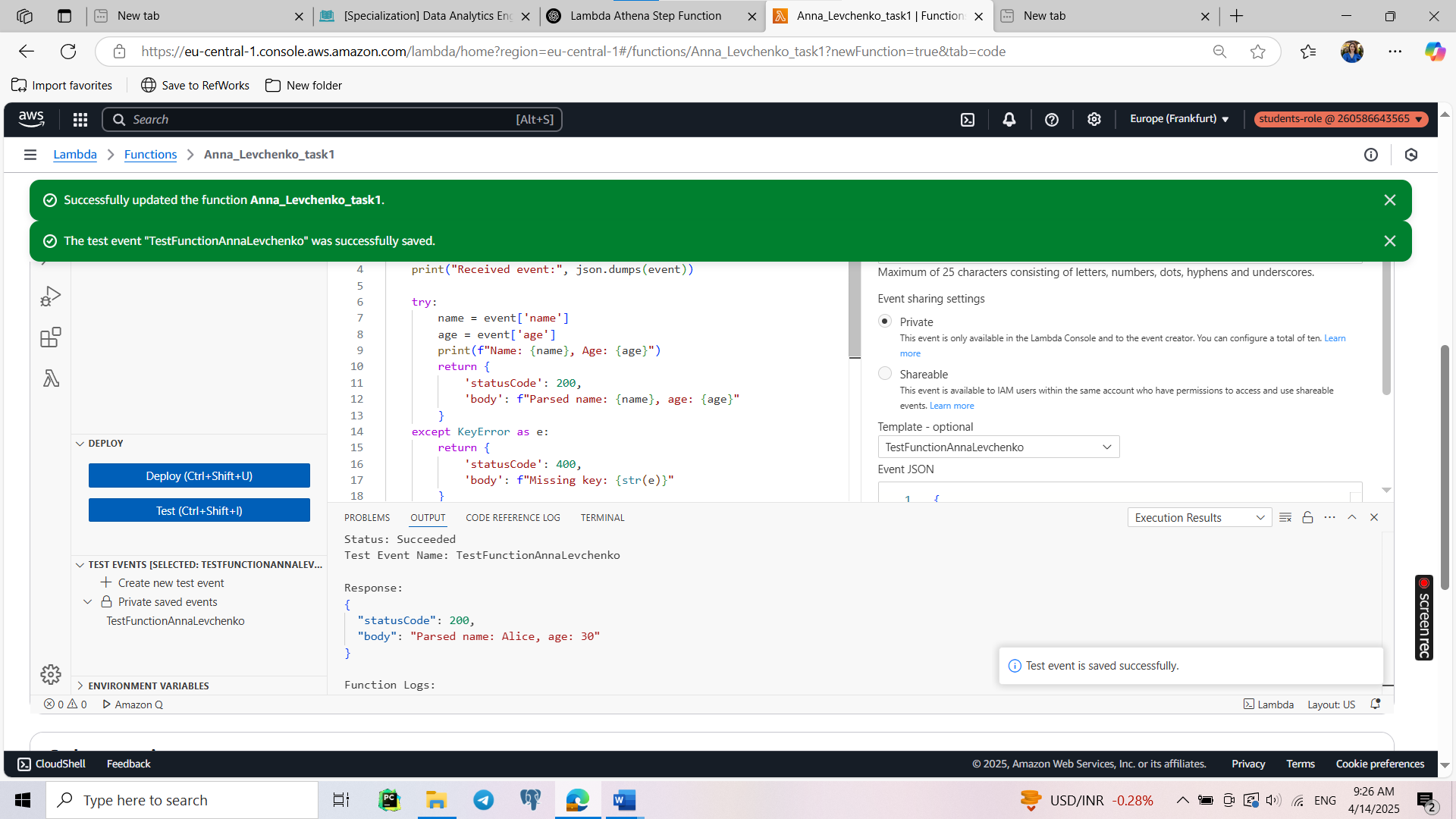
**Task 1a:** [**https://eu-central-1.console.aws.amazon.com/lambda/home?region=eu-central-1#/functions/Anna\_Levchenko\_task1?tab=code**](https://eu-central-1.console.aws.amazon.com/lambda/home?region=eu-central-1#/functions/Anna_Levchenko_task1?tab=code)





**Task 2**

1. 1st Select Test: <https://eu-central-1.console.aws.amazon.com/lambda/home?region=eu-central-1#/functions/Anna_Levchenko_Test_Table_1?subtab=permissions&tab=configure>

import json

import redshift\_connector

def lambda\_handler(event, context):

    try:

        conn = redshift\_connector.connect(

            host='data-bi-lab-redshift-cluster-3.cettexdsxw3v.eu-central-1.redshift.amazonaws.com',

            database='dev',

            user='dilab\_student51',

            password='aU24\_2025\_51',

            port=5439

        )

        cursor = conn.cursor()

        cursor.execute("SELECT location\_state\_name FROM user\_dilab\_student51.location WHERE location\_city\_name ='Kyiv' ORDER BY location\_surr\_id ASC;")

        row = cursor.fetchone()  # Get the first row from the result

        cursor.close()

        conn.close()

        # If we found a row, check if the location\_state\_name is not empty or None

        if row and row[0]:  # row[0] is the location\_state\_name

            location\_state\_name = row[0]

            print(f"Row found: {location\_state\_name}")

            return {

                "status": "pass" if location\_state\_name else "fail",

                "location\_state\_name": location\_state\_name

            }

        else:

            print("No rows found.")

            return {

                "status": "fail",

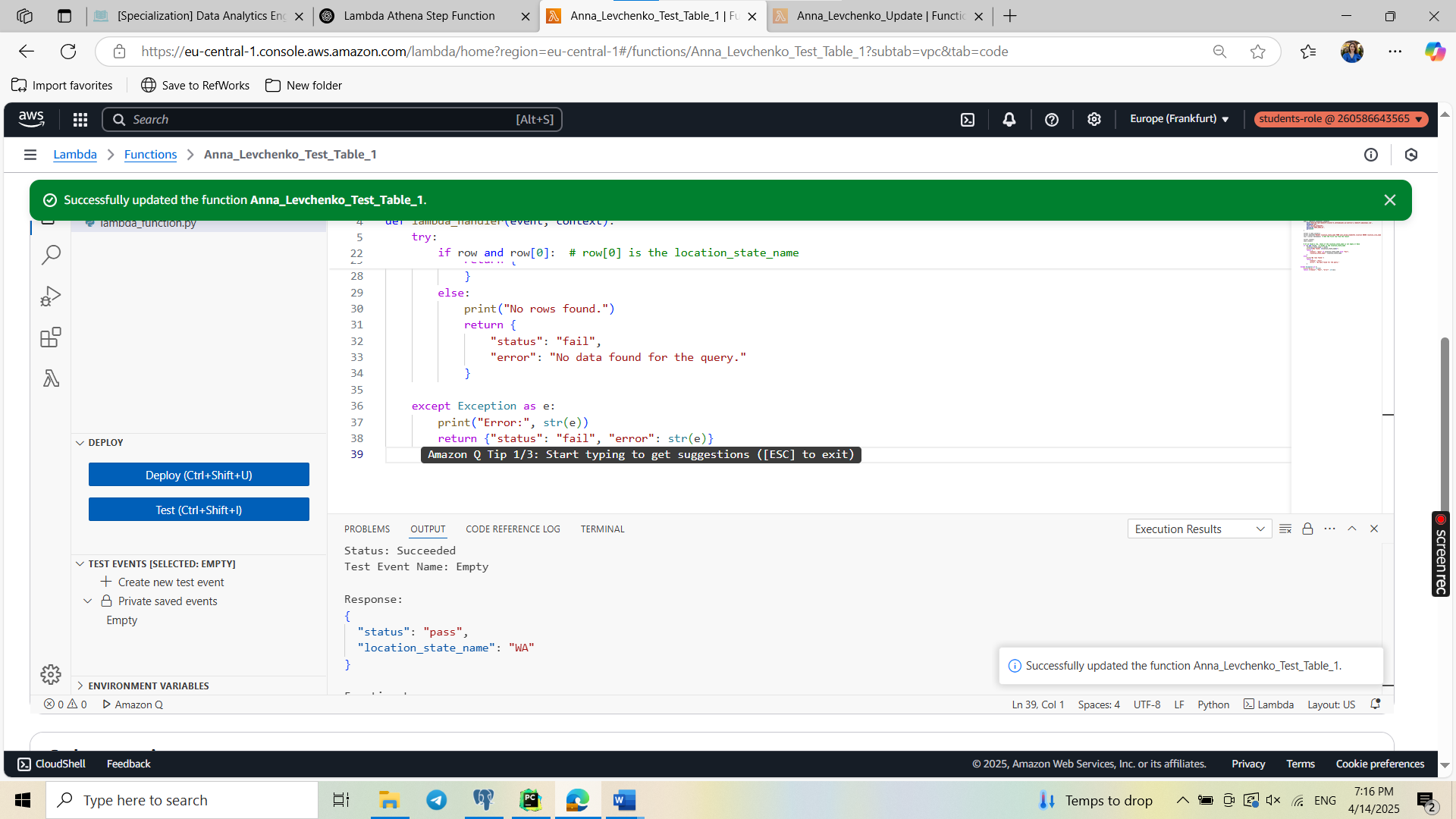
                "error": "No data found for the query."

            }

    except Exception as e:

        print("Error:", str(e))

        return {"status": "fail", "error": str(e)}



1. Update Test: <https://eu-central-1.console.aws.amazon.com/lambda/home?region=eu-central-1#/functions/Anna_Levchenko_Update?subtab=general&tab=code>

import json

import redshift\_connector

import random

import datetime

import time

def lambda\_handler(event, context):

    try:

        start\_time = time.time()

        conn = redshift\_connector.connect(

            host='data-bi-lab-redshift-cluster-3.cettexdsxw3v.eu-central-1.redshift.amazonaws.com',

            database='dev',

            user='dilab\_student51',

            password='aU24\_2025\_51',

            port=5439

        )

        connection\_time = time.time() - start\_time

        print(f"Connection established in {connection\_time:.2f} seconds.")

        start\_time = time.time()

        cursor = conn.cursor()

        query = """

            UPDATE user\_dilab\_student51.location

            SET location\_state\_name = 'UA'

            WHERE location\_city\_name ='Kyiv';

        """

        cursor.execute(query)

        conn.commit()

        query\_time = time.time() - start\_time

        print(f"Query executed in {query\_time:.2f} seconds.")

        return {

            'update': {

                'status': 'success'

            }

        }

    except Exception as e:

        print(f"Error: {str(e)}")

        return {

            'update': {

                'status': 'error',

                'error': str(e)

            }

        }

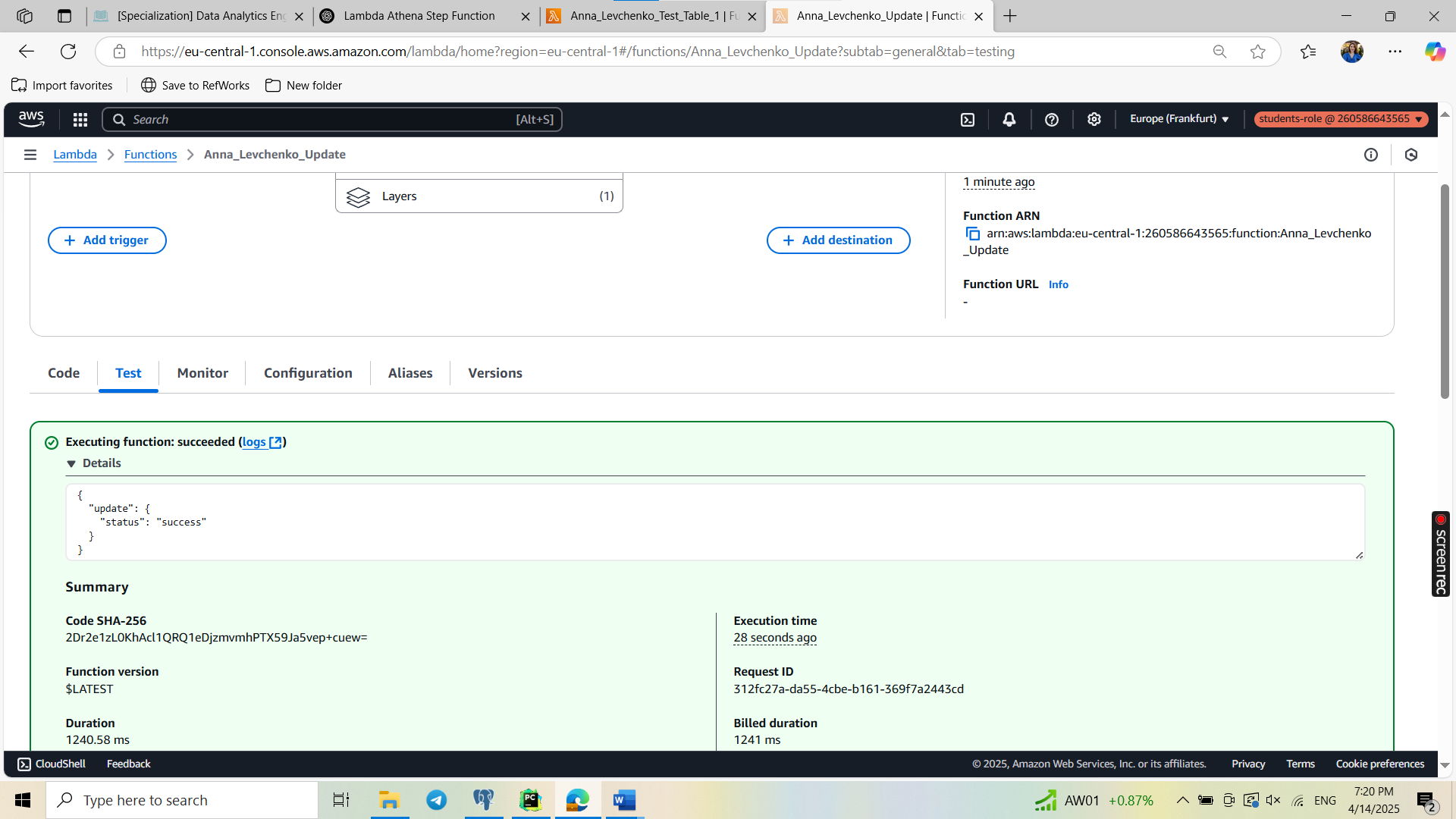
    finally:

        if cursor:

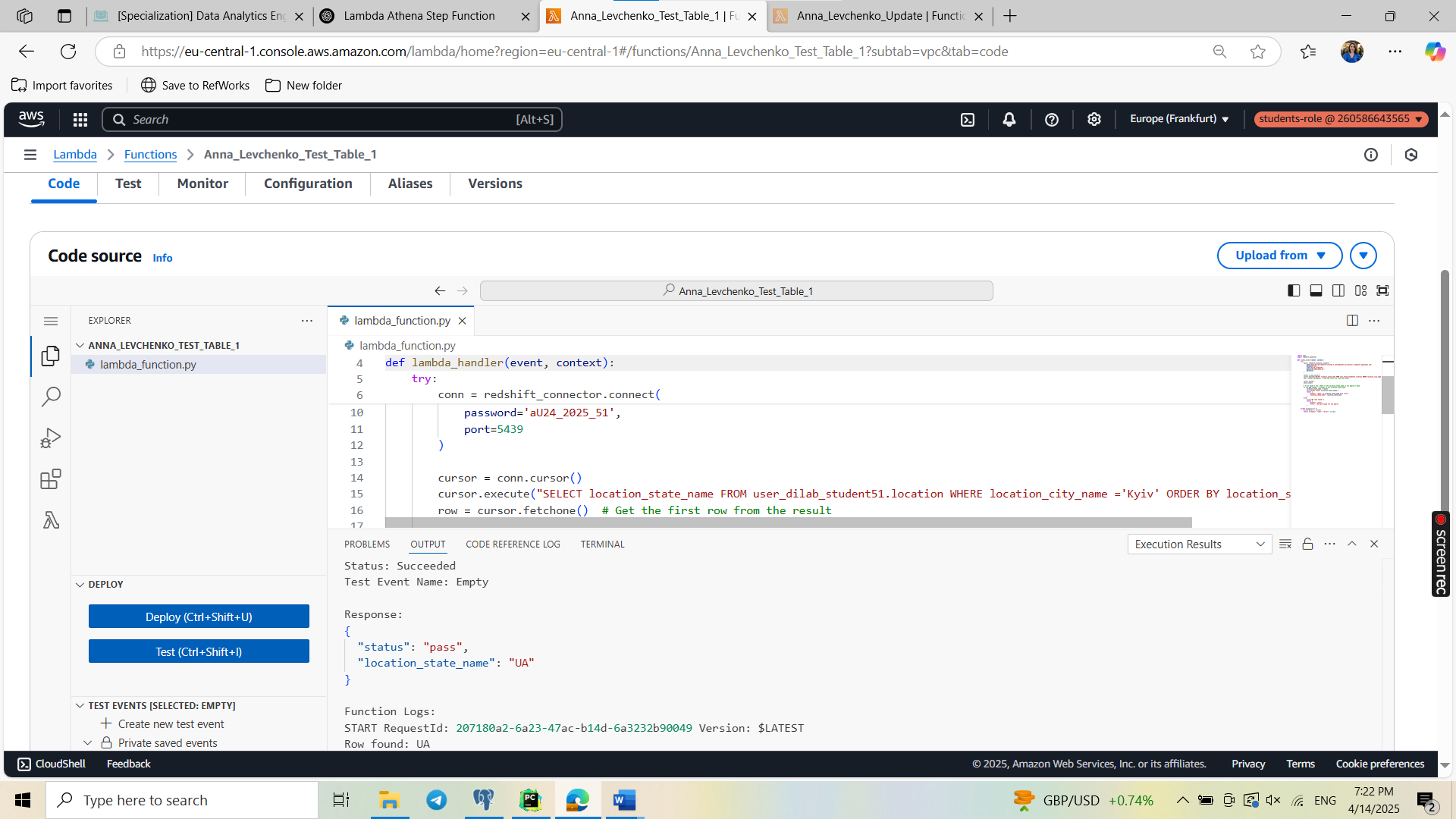
            cursor.close()

        if conn:

            conn.close()

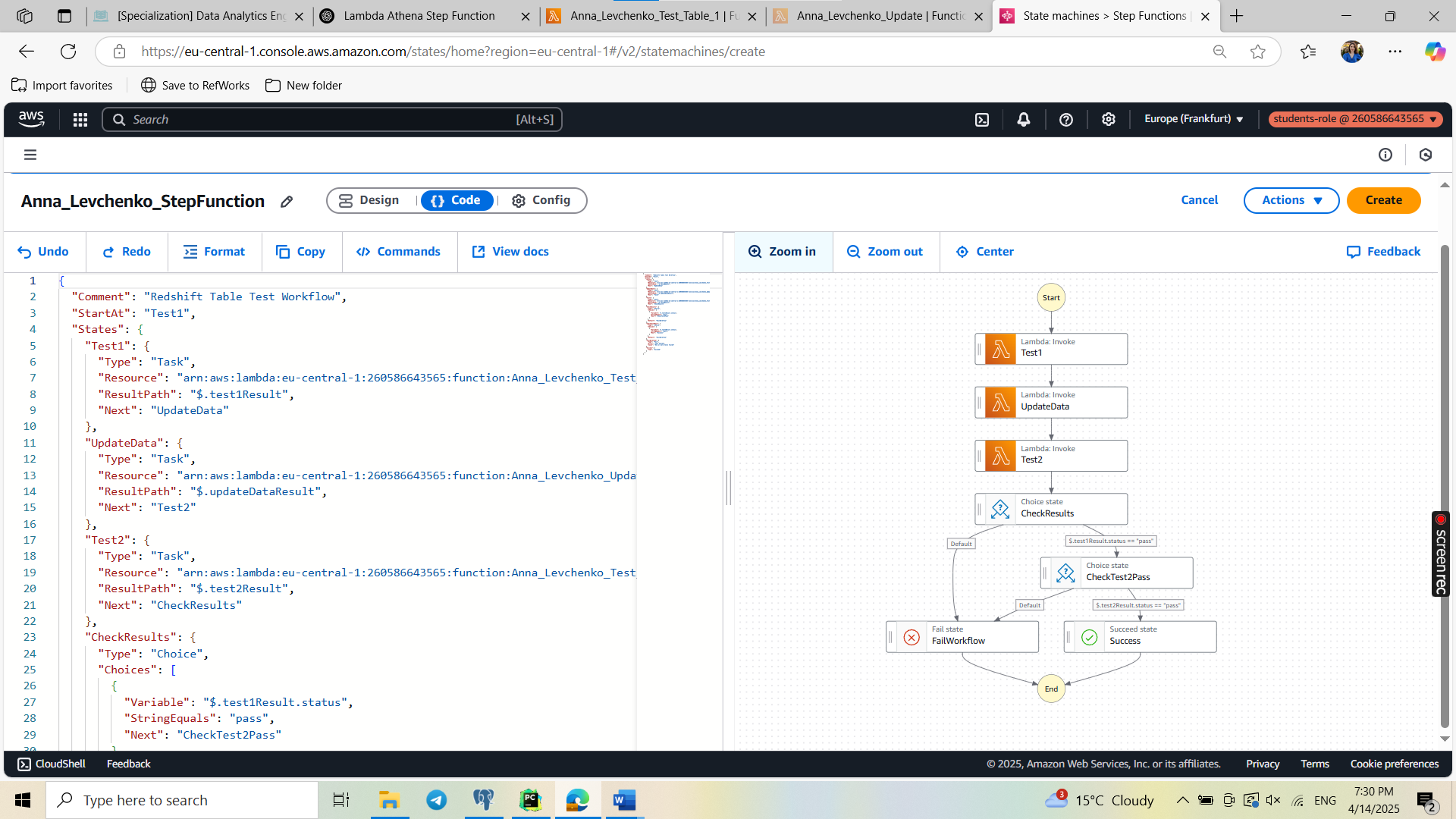


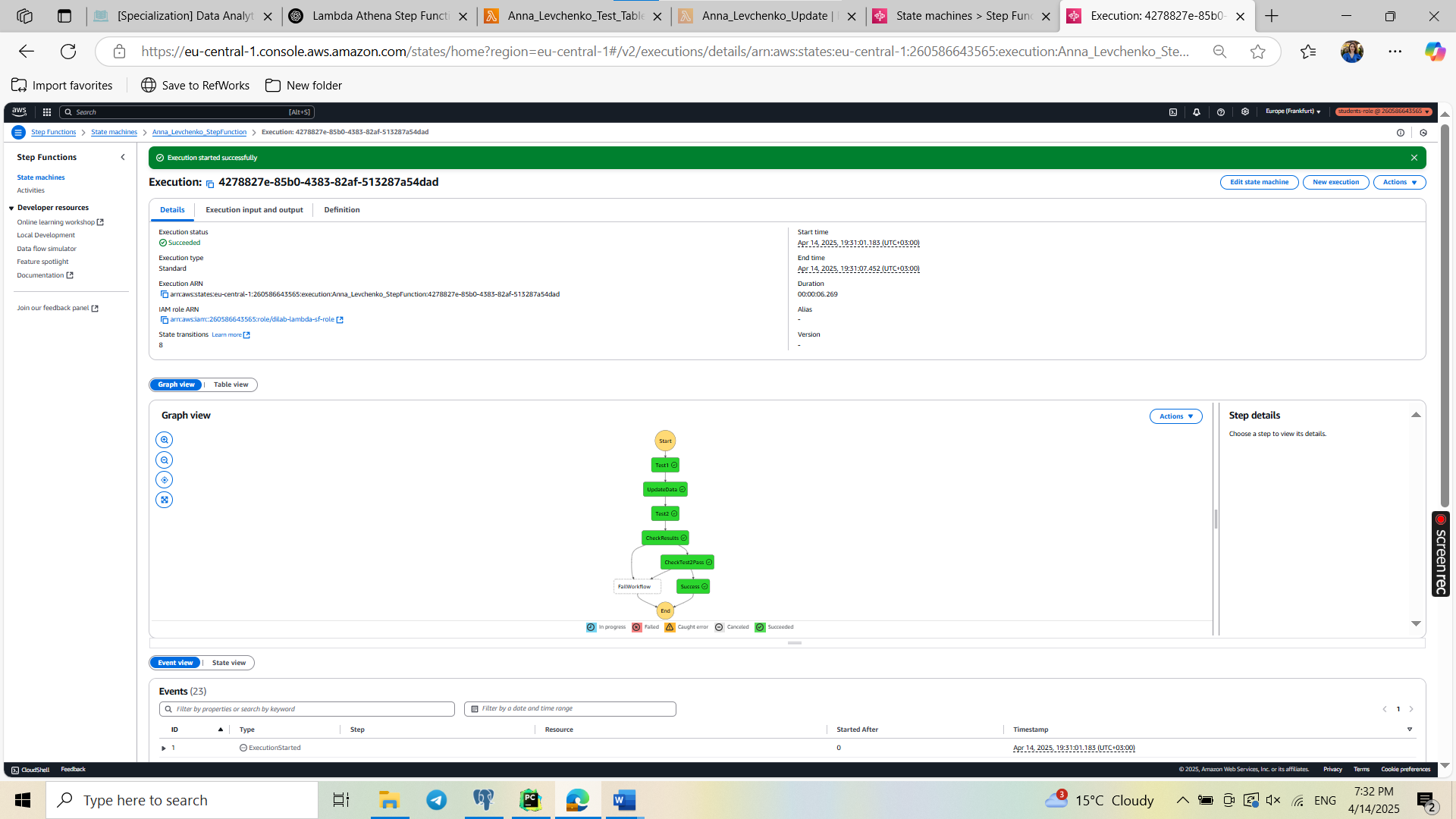
1. 2nd Select Test: success, from WA changes to UA: <https://eu-central-1.console.aws.amazon.com/lambda/home?region=eu-central-1#/functions/Anna_Levchenko_Test_Table_1?subtab=permissions&tab=configure>

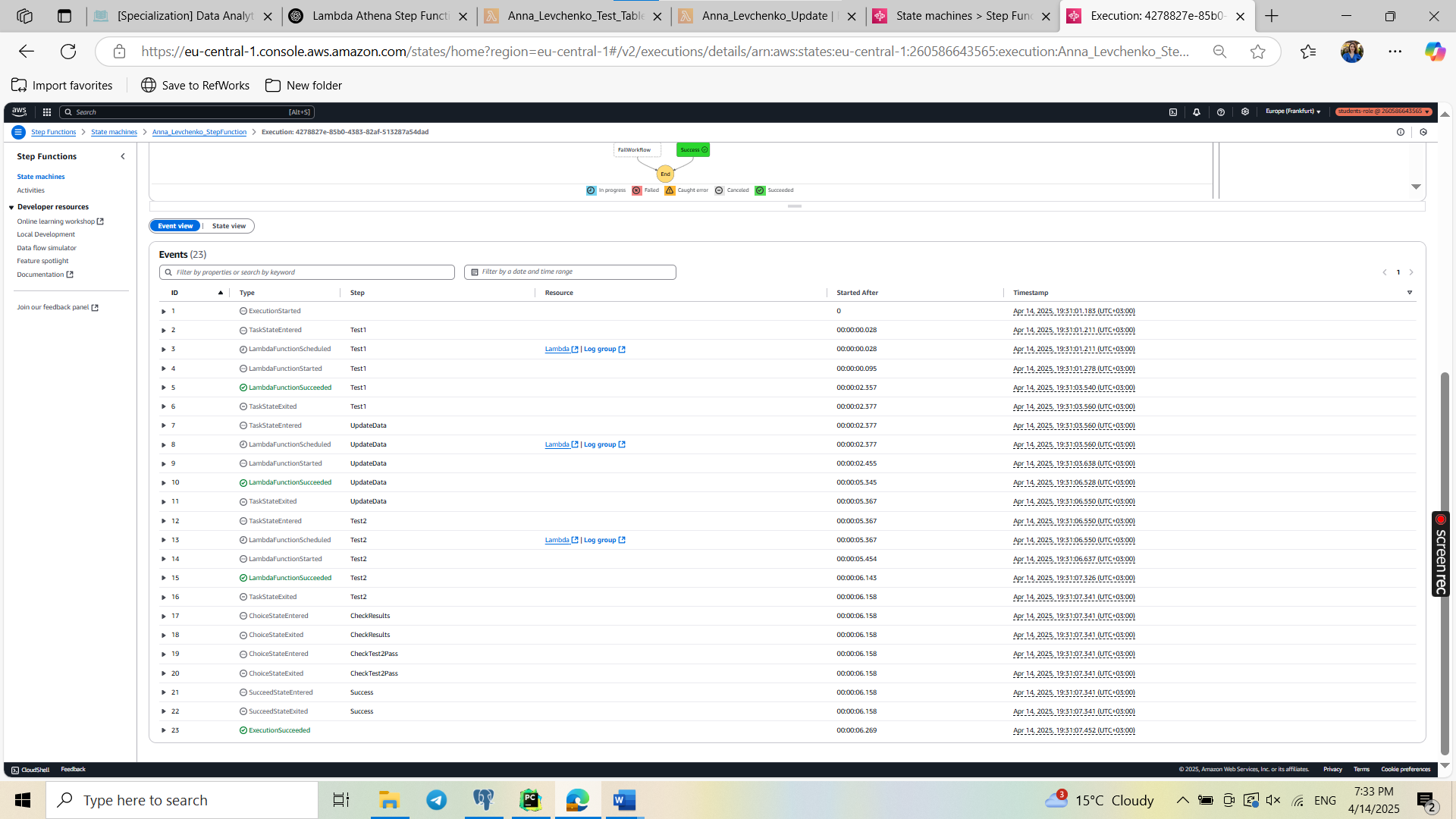


1. Then I set up State Machine with my functions, execution is succeed.

<https://eu-central-1.console.aws.amazon.com/states/home?region=eu-central-1#/statemachines/view/arn%3Aaws%3Astates%3Aeu-central-1%3A260586643565%3AstateMachine%3AAnna_Levchenko_StepFunction?type=standard>







**Lambda Functions:** allow to run code without provisioning or managing servers.

* Set up VPC, subnets, and security groups.
* Use layers for dependencies.
* Handle database connections and queries.
* Parse responses and handle errors.

**Step Functions:** provide a way to orchestrate multiple Lambda functions into a workflow.

* Define the workflow using Amazon States Language.
* Define states for each step in workflow. A state can invoke Lambda functions (with Task), make decisions (Choice), or handle failures (Fail).
* Pass data between states and handle success/failure logic.