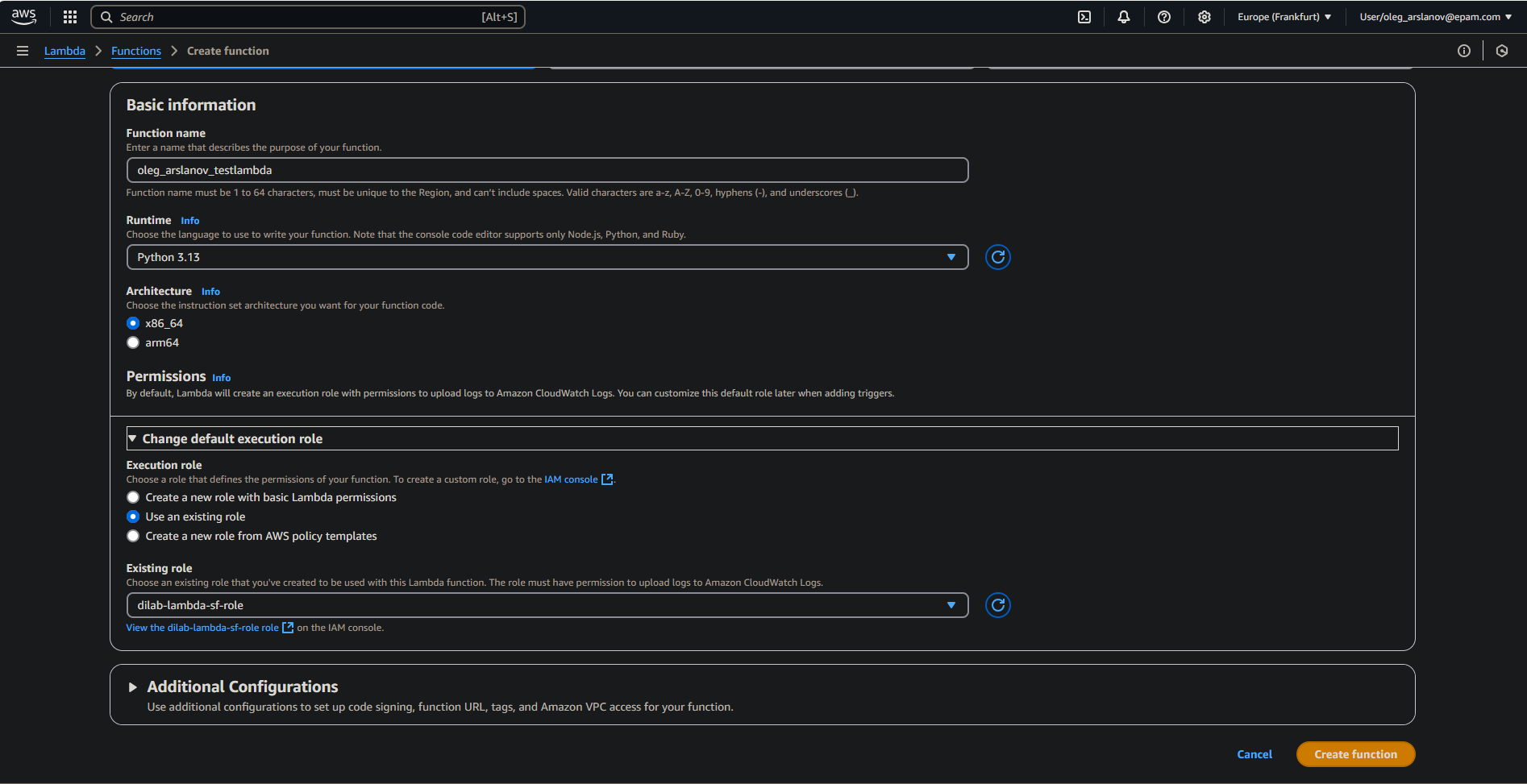
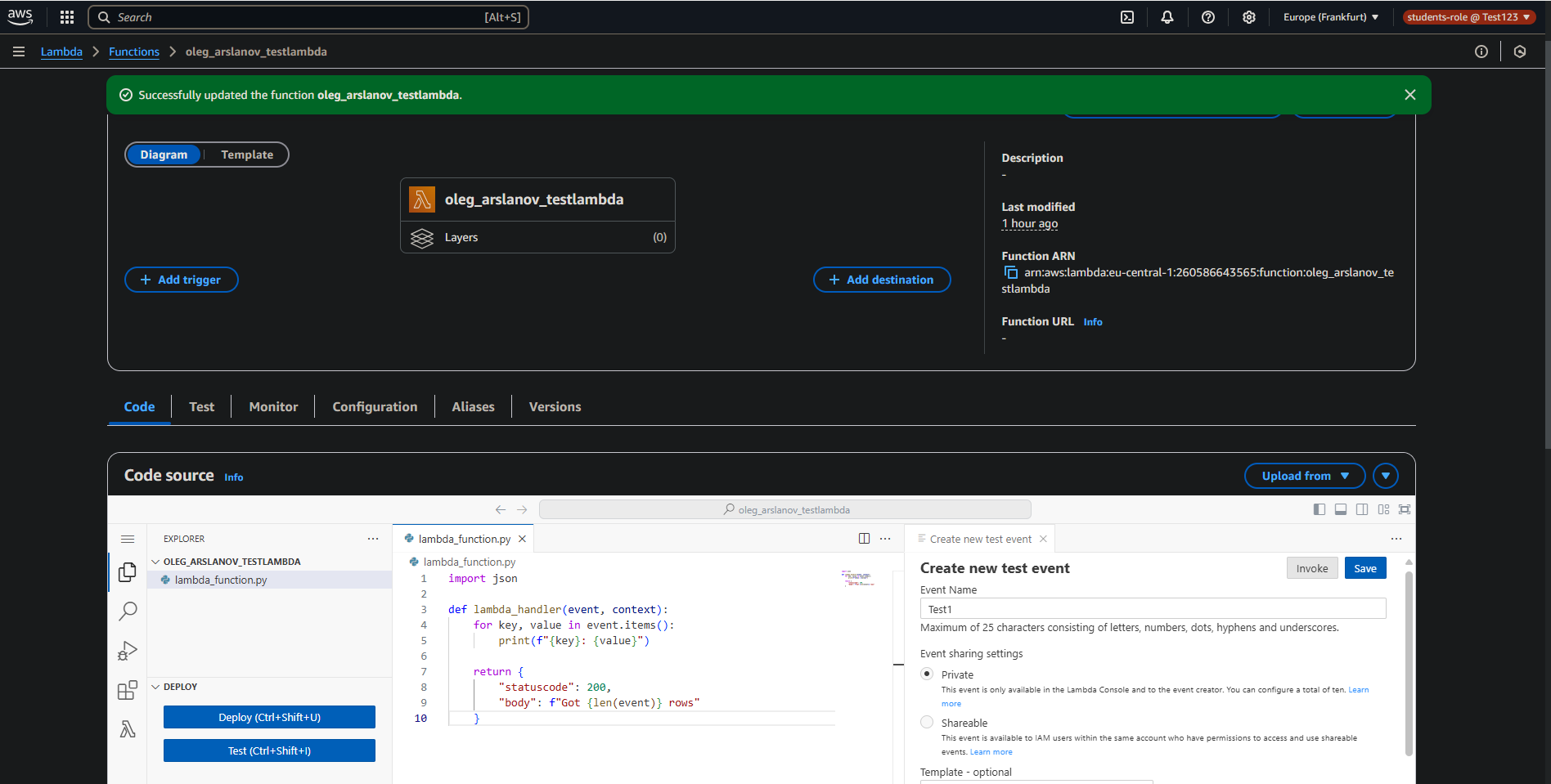
**Task 7**

1a) Create your first Lambda and show how you will parse test JSON using Response or Function logs.

First I created Lambda function with provided IAM role: *dilab-lambda-sf-role*



Next I created Lambda function with python code



Python code:

import json

def lambda\_handler(event, context):

for key, value in event.items():

print(f"{key}: {value}")

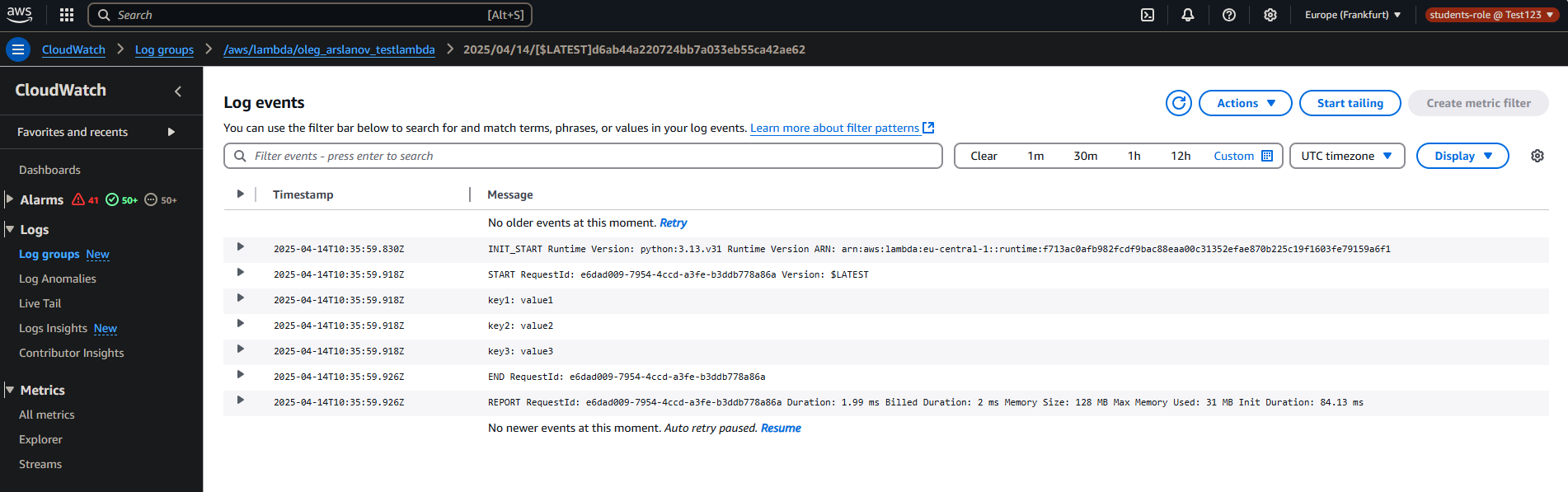
return {

"statuscode": 200,

"body": f"Got {len(event)} rows"

}

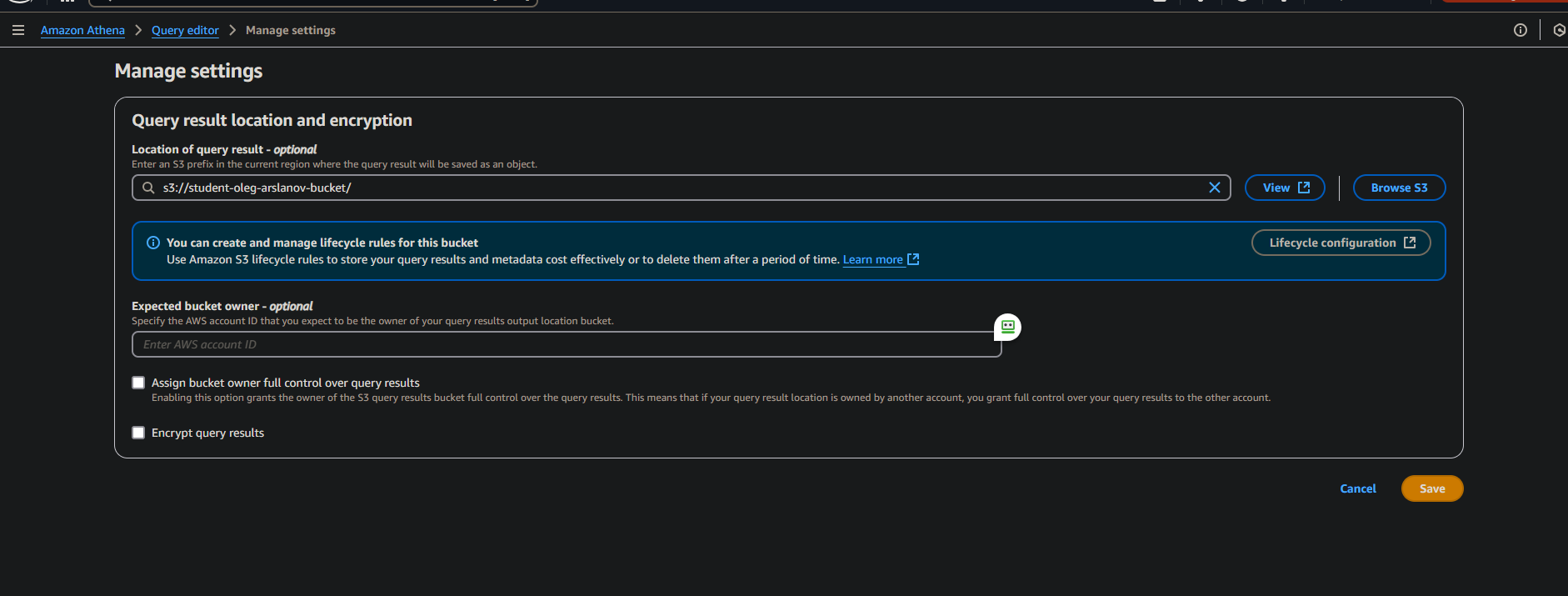
Next I test Lambda and succeeded

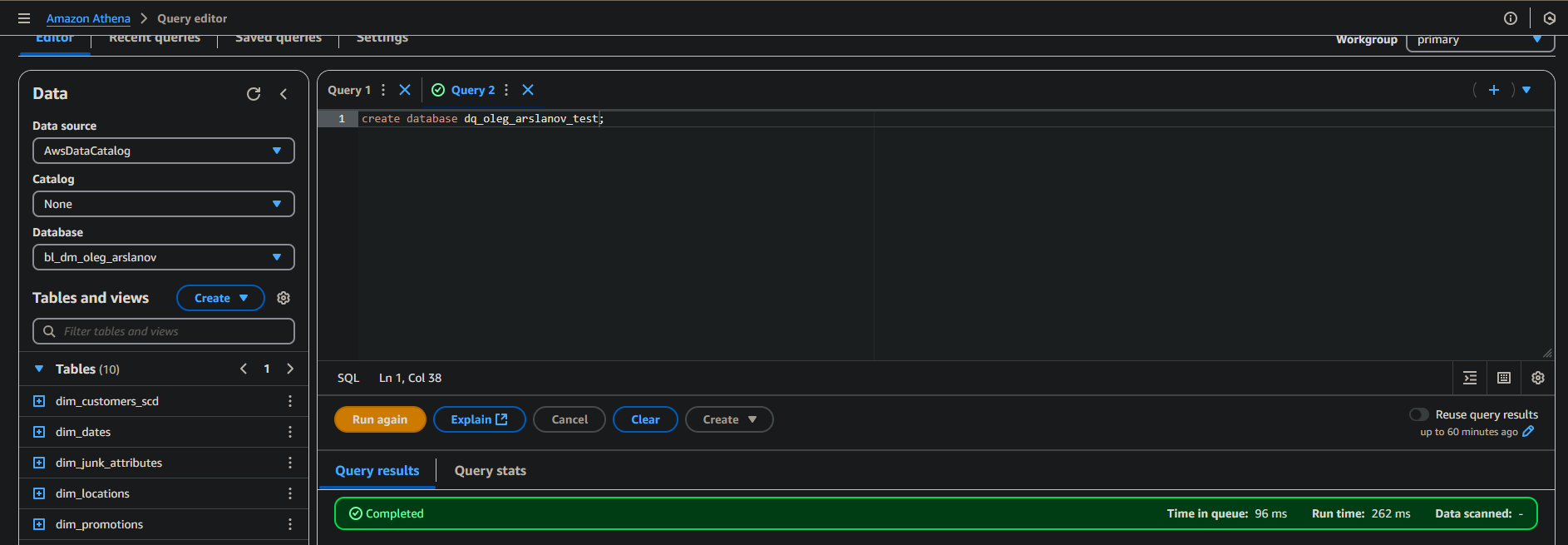


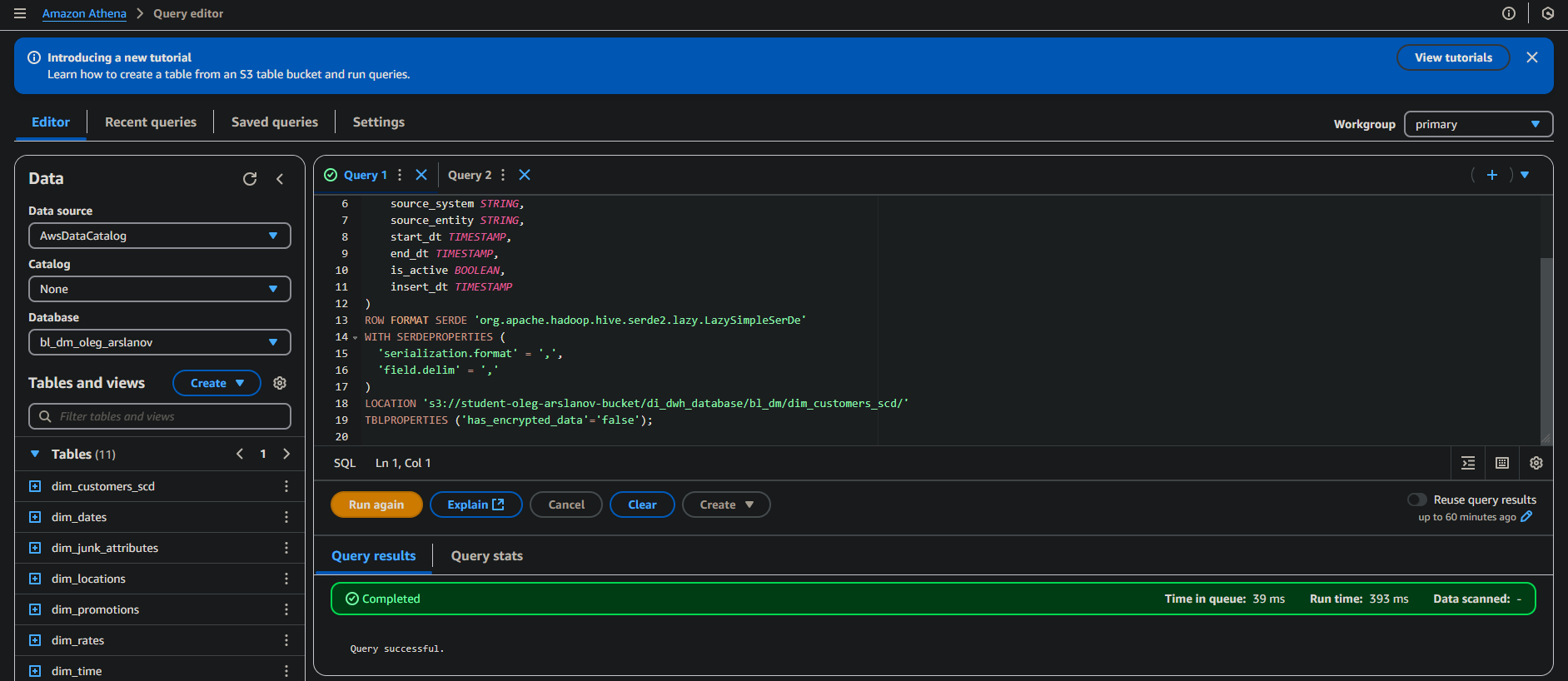
1b) Create a new one or change the existing Lambda function and connect it with the Athena service

1. Select rows from any table you choose and show them in the Function logs (don’t forget about LIMIT clause).

First of all I need to setting ATHENA for save results in S3



Create DB in Athena

Create test table 

SQl code:

CREATE EXTERNAL TABLE IF NOT EXISTS test\_table (

customer\_sur\_id BIGINT,

customer\_type STRING,

customer\_telephone STRING,

customer\_src\_id BIGINT,

source\_system STRING,

source\_entity STRING,

start\_dt TIMESTAMP,

end\_dt TIMESTAMP,

is\_active BOOLEAN,

insert\_dt TIMESTAMP

)

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'

WITH SERDEPROPERTIES (

'serialization.format' = ',',

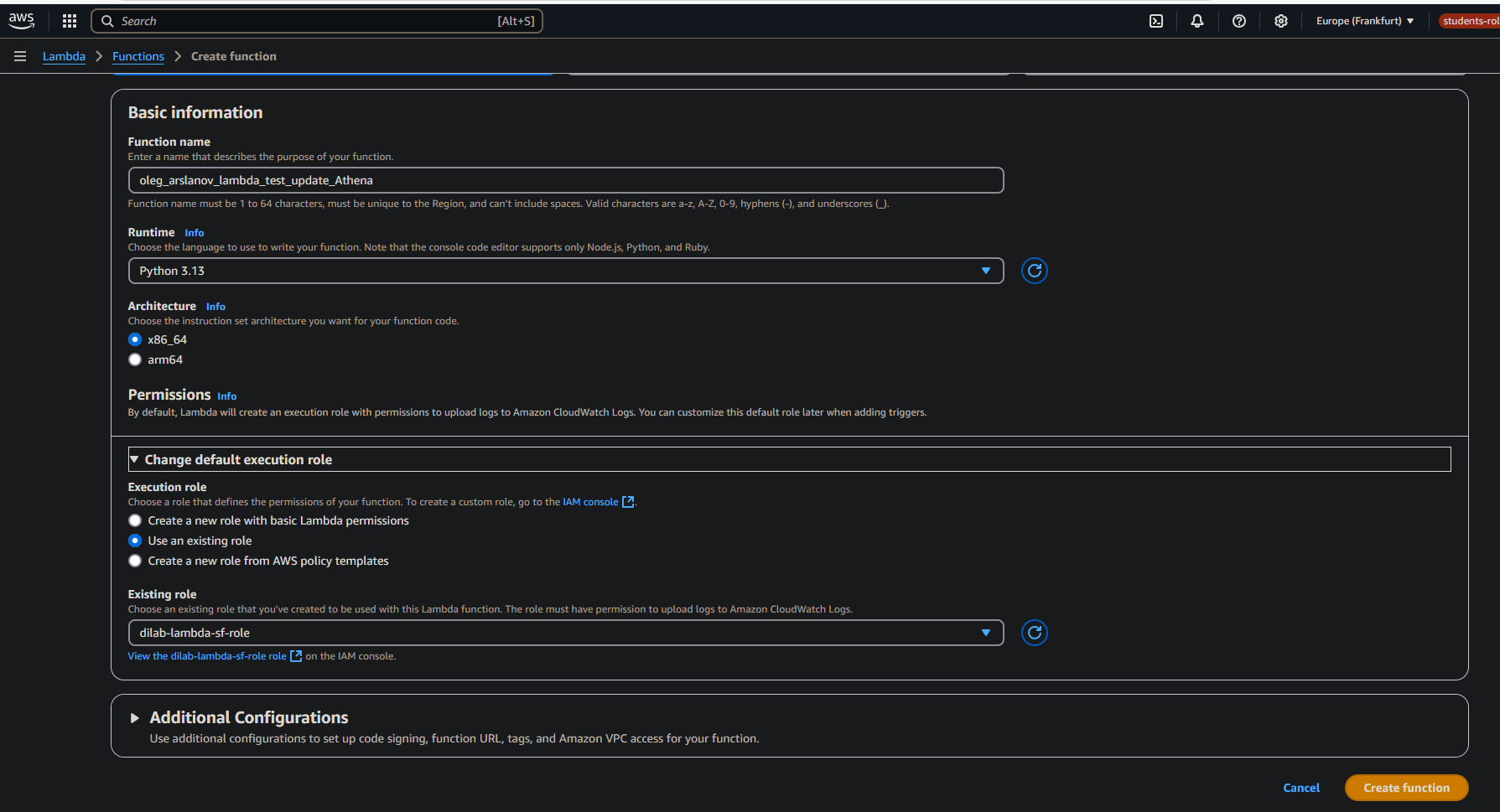
'field.delim' = ','

)

LOCATION 's3://student-oleg-arslanov-bucket/di\_dwh\_database/bl\_dm/dim\_customers\_scd/'

TBLPROPERTIES ('has\_encrypted\_data'='false');

Create LAMBDA function



Python code:

import boto3

import time

athena\_client = boto3.client('athena')

def lambda\_handler(event, context):

try:

print("Start query to Athena ...")

query = "SELECT \* FROM dq\_oleg\_arslanov\_test.test\_table LIMIT 5"

database = "dq\_oleg\_arslanov\_test"

output\_bucket = "s3://student-oleg-arslanov-bucket/athena\_output/"

response = athena\_client.start\_query\_execution(

QueryString=query,

QueryExecutionContext={"Database": database},

ResultConfiguration={"OutputLocation": output\_bucket}

)

query\_execution\_id = response['QueryExecutionId']

print(f"Query ID: {query\_execution\_id}")

while True:

result = athena\_client.get\_query\_execution(QueryExecutionId=query\_execution\_id)

status = result['QueryExecution']['Status']['State']

print(f"Query status: {status}")

if status in ['SUCCEEDED', 'FAILED', 'CANCELLED']:

break

time.sleep(1)

if status == 'SUCCEEDED':

result = athena\_client.get\_query\_results(QueryExecutionId=query\_execution\_id)

rows = result['ResultSet']['Rows']

for row in rows:

print([col.get('VarCharValue', '') for col in row['Data']])

return {

'statusCode': 200,

'body': f"Query completed with {len(rows)} rows"

}

else:

print("Query not completed")

return {

'statusCode': 500,

'body': f"Query failed with status {status}"

}

except Exception as e:

print("Execution error Lambda:")

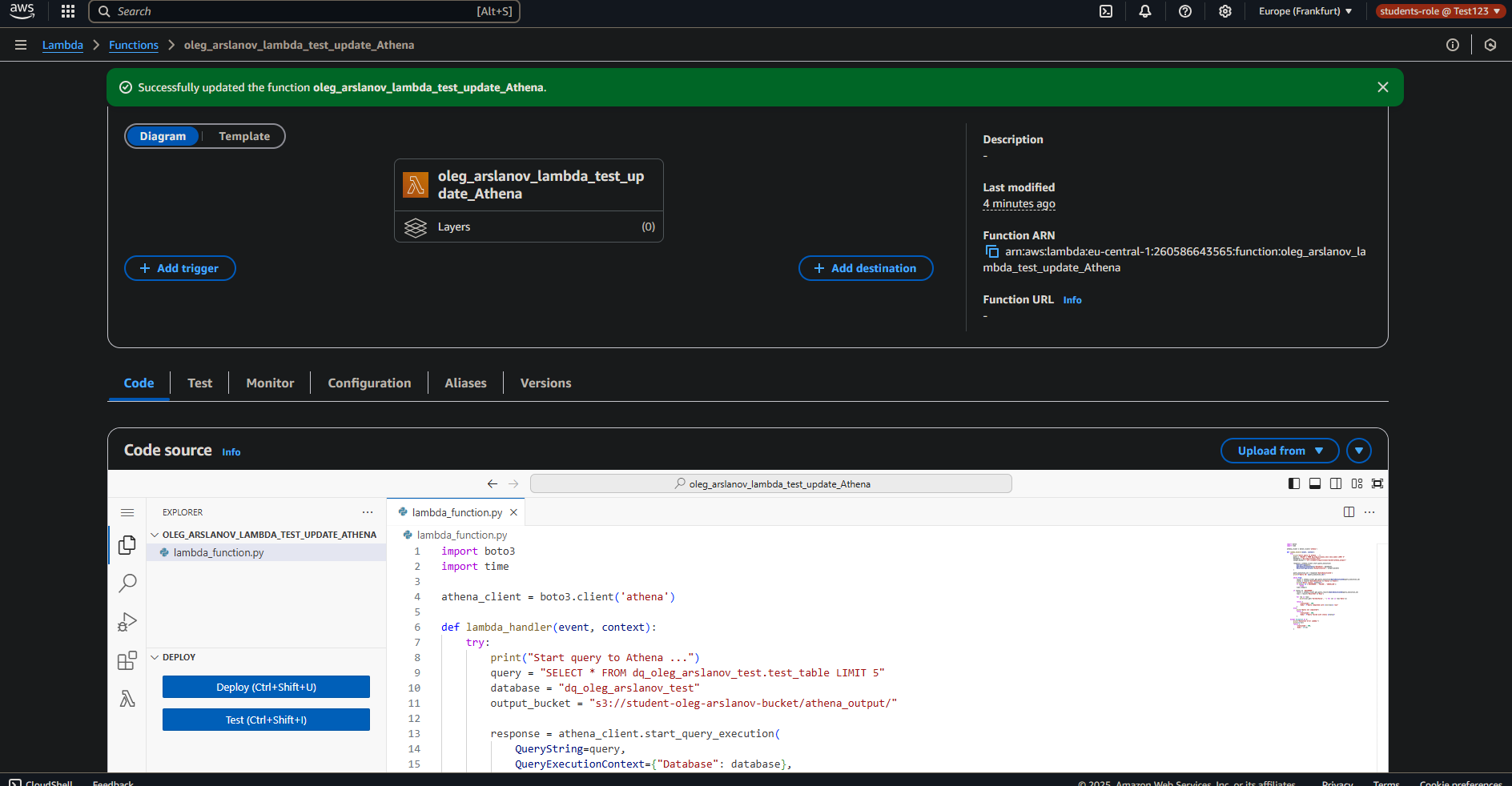
print(str(e))

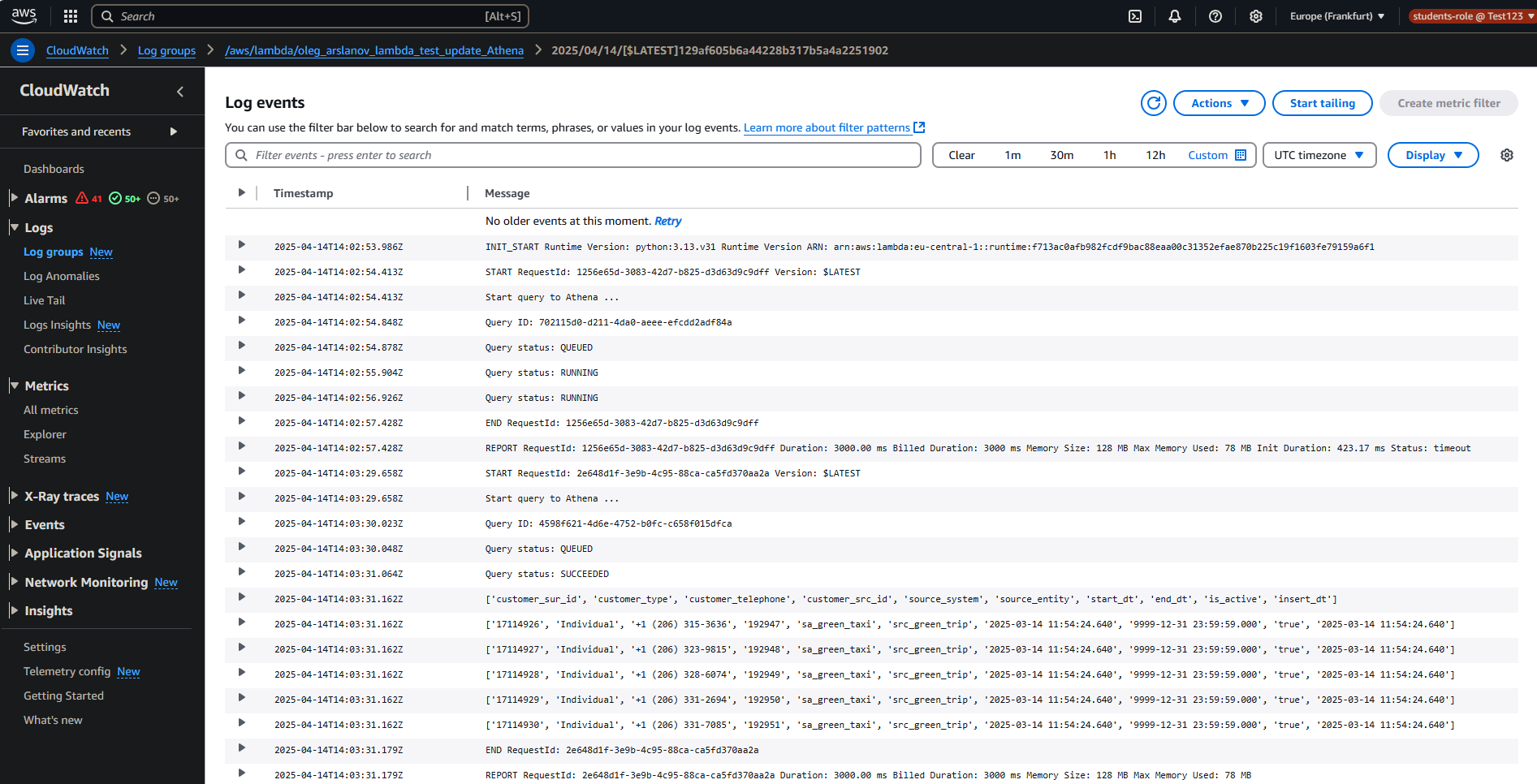
return {

'statusCode': 500,

'body': str(e)

}

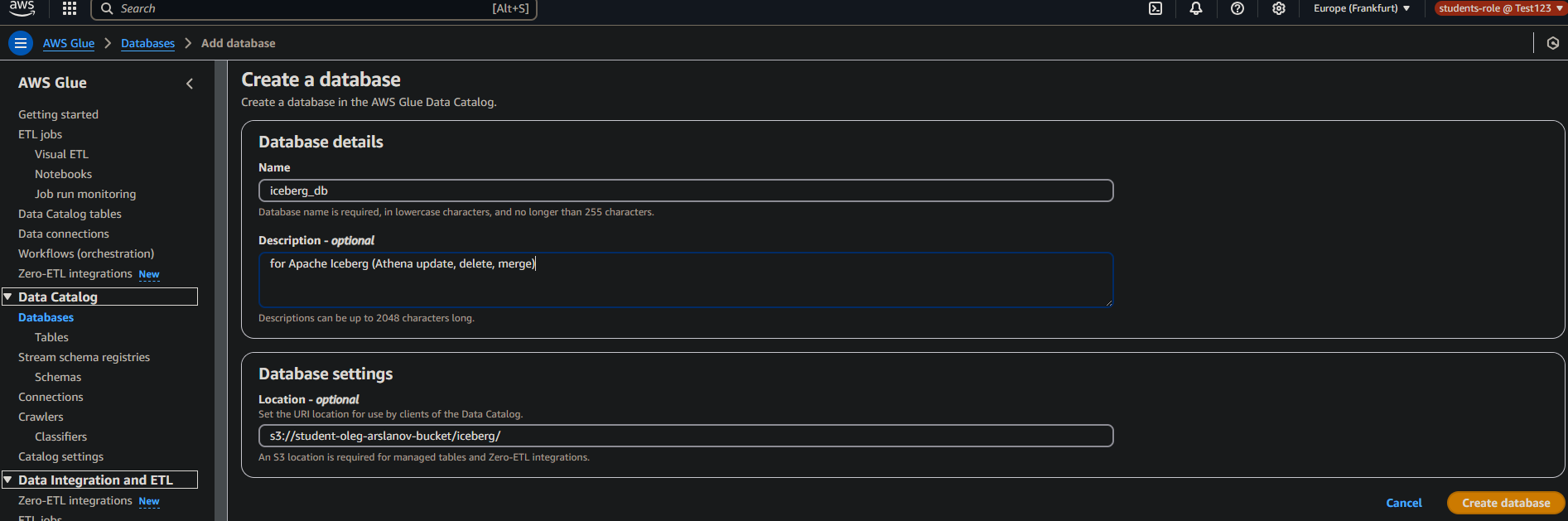


Successfully added to log rows from test\_table

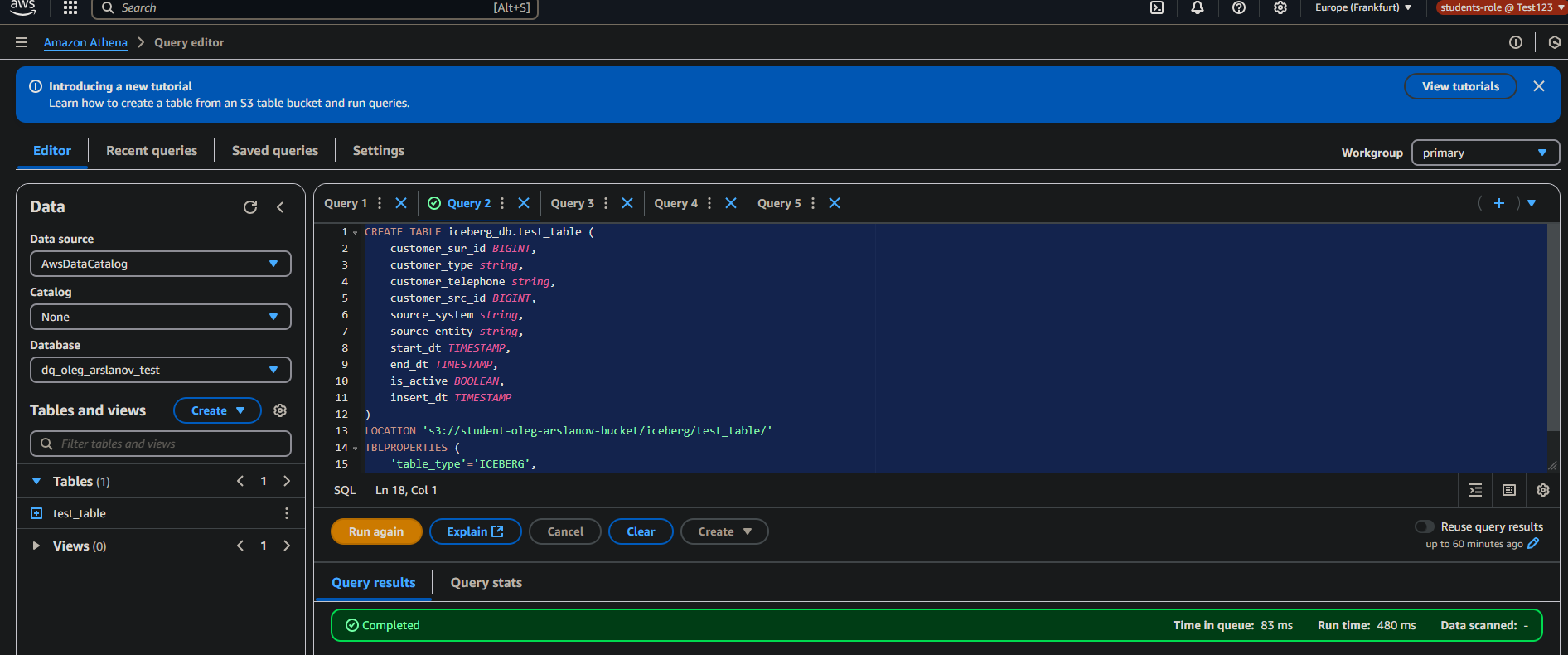
ii. Update any value from any table you choose.

For update rows in Athena table we need to setting Apache Iceberg (we will can update, delete, merge)

With AWS Glue created DB (inside Athena can not create Iceberg table)



Then back to Athena and created Iceberg table



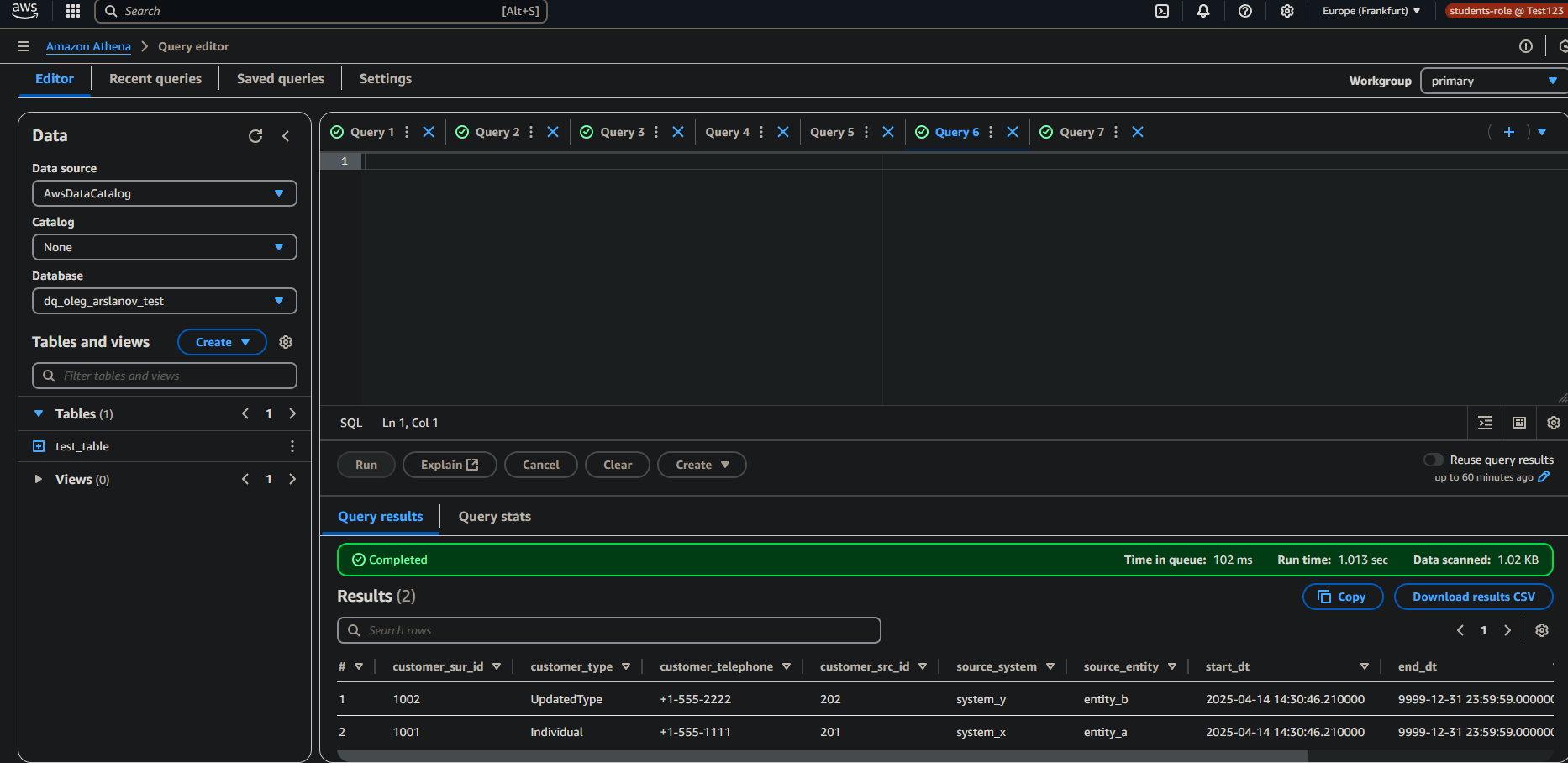
SQL code for add some rows:

INSERT INTO iceberg\_db.test\_table VALUES

(1001, 'Individual', '+1-555-1111', 201, 'system\_x', 'entity\_a', CURRENT\_TIMESTAMP, TIMESTAMP '9999-12-31 23:59:59', true, CURRENT\_TIMESTAMP),

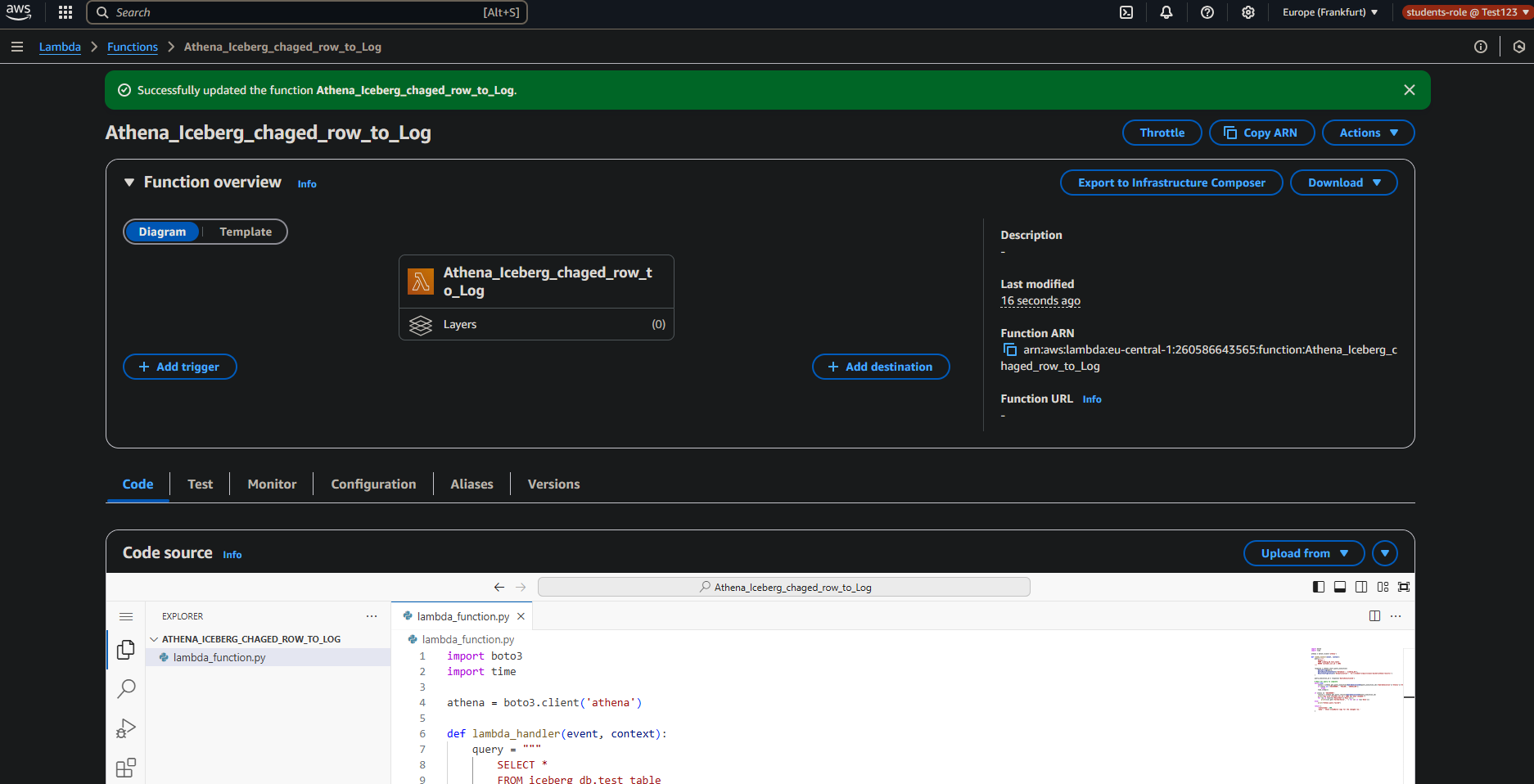
(1002, 'Business', '+1-555-2222', 202, 'system\_y', 'entity\_b', CURRENT\_TIMESTAMP, TIMESTAMP '9999-12-31 23:59:59', true, CURRENT\_TIMESTAMP);

Then we change one row



iii. Select row for which the value has been changed and show it in the Function log.

Created Lambda function



Python code:

import boto3

import time

ATHENA\_DATABASE = 'iceberg\_db'

ATHENA\_TABLE = 'test\_table'

ATHENA\_OUTPUT = 's3://student-oleg-arslanov-bucket/iceberg\_output/'

CUSTOMER\_SUR\_ID = 1002

athena = boto3.client('athena')

def lambda\_handler(event, context):

print("Start Athena query ...")

query = f"""

SELECT \* FROM {ATHENA\_DATABASE}.{ATHENA\_TABLE}

WHERE customer\_sur\_id = {CUSTOMER\_SUR\_ID}

"""

response = athena.start\_query\_execution(

QueryString=query,

QueryExecutionContext={'Database': ATHENA\_DATABASE},

ResultConfiguration={'OutputLocation': ATHENA\_OUTPUT}

)

query\_execution\_id = response['QueryExecutionId']

print(f"Query ID: {query\_execution\_id}")

# Check status

while True:

result = athena.get\_query\_execution(QueryExecutionId=query\_execution\_id)

status = result['QueryExecution']['Status']['State']

print(f"Status: {status}")

if status in ['SUCCEEDED', 'FAILED', 'CANCELLED']:

break

time.sleep(2)

if status != 'SUCCEEDED':

print("Query end with error")

return

# Get result

results = athena.get\_query\_results(QueryExecutionId=query\_execution\_id)

rows = results['ResultSet']['Rows']

if len(rows) > 1:

header = [col['VarCharValue'] for col in rows[0]['Data']]

values = [col['VarCharValue'] for col in rows[1]['Data']]

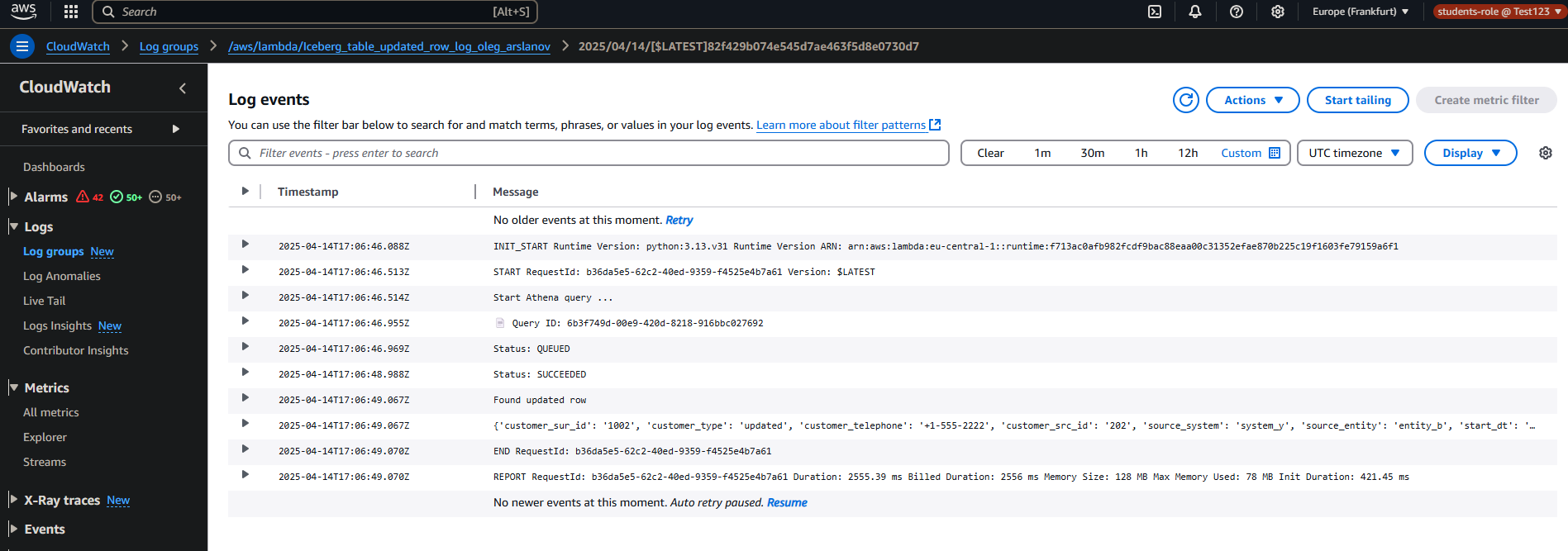
updated\_row = dict(zip(header, values))

print("Found updated row")

print(updated\_row)

else:

print("Row with customer\_sur\_id not found")

And Log looking good with changed value

c. (Optional) Create a file with SQL queries from the previous step and place it on AWS S3. Parse this file using Lambda and use the parsed row as an SQL query instead of a hardcoded one.

Sorry have no time all time on task 2 😐

2. AWS Step Function service

Note: To connect to Redshift, please assign to Lambda a Layer, VPC, subnets, and security group. You can find respective names in the Lambda function epam-aleh-shylin-test-lambda.

a. Create a Step Function that will have the following steps:

i. Perform the first test for the table from Redshift using AWS Lambda.

ii. Change data in the Redshift table using AWS Lambda.

iii. Perform a second test for the same table from Redshift using AWS Lambda

iv. If the first and third steps are passed then Step Function completes successfully, otherwise – Step Function fails.

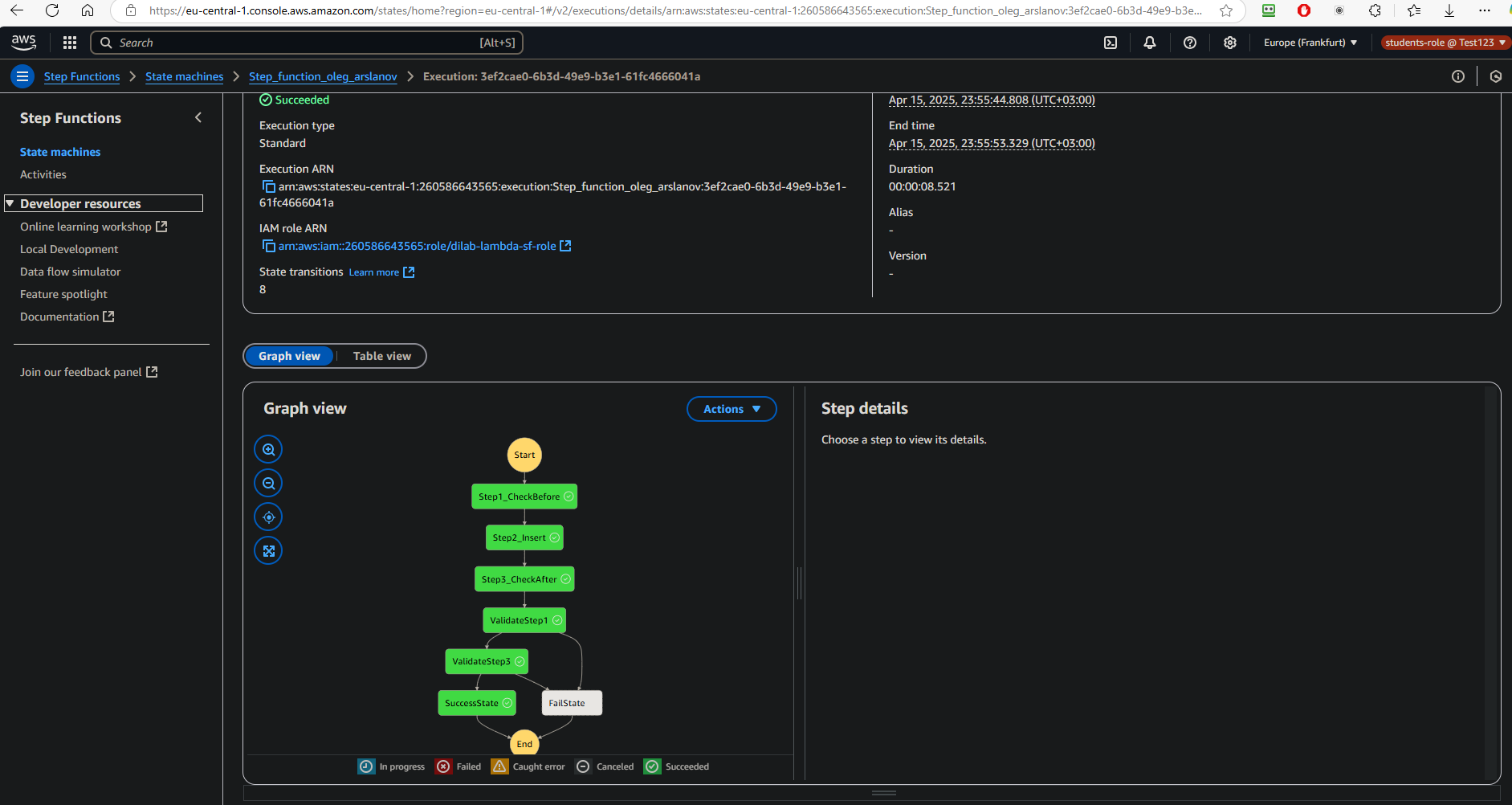
Link to step function and lambdas:

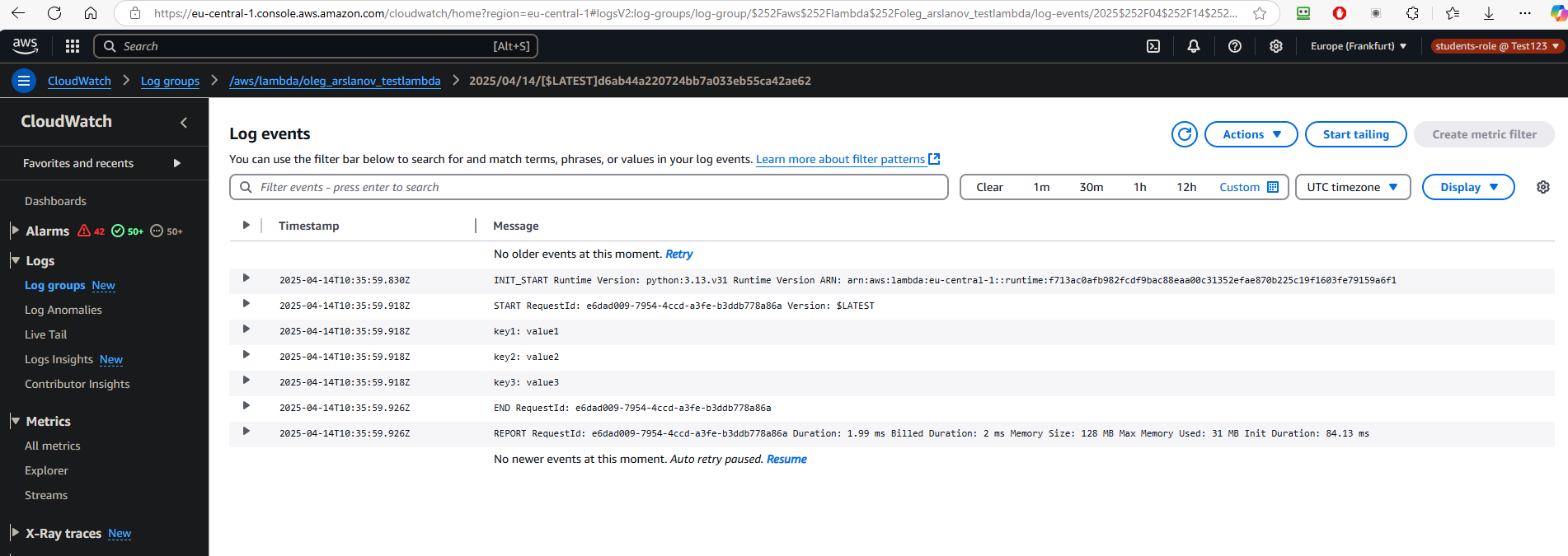
[Step Functions | eu-central-1](https://eu-central-1.console.aws.amazon.com/states/home?region=eu-central-1#/v2/statemachines/edit/arn%3Aaws%3Astates%3Aeu-central-1%3A260586643565%3AstateMachine%3AStep_function_oleg_arslanov?referrerPage=StateMachineDetails)

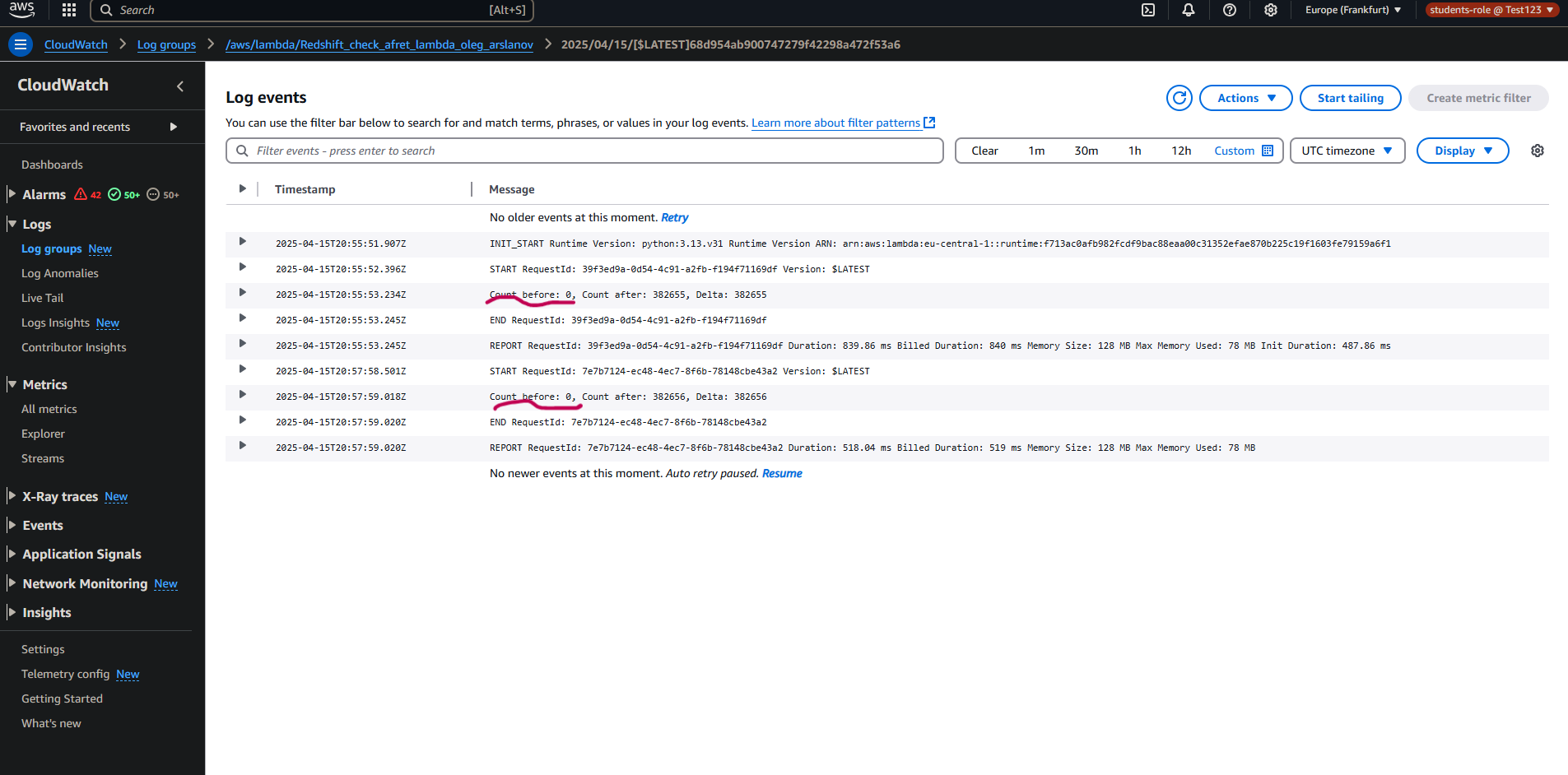
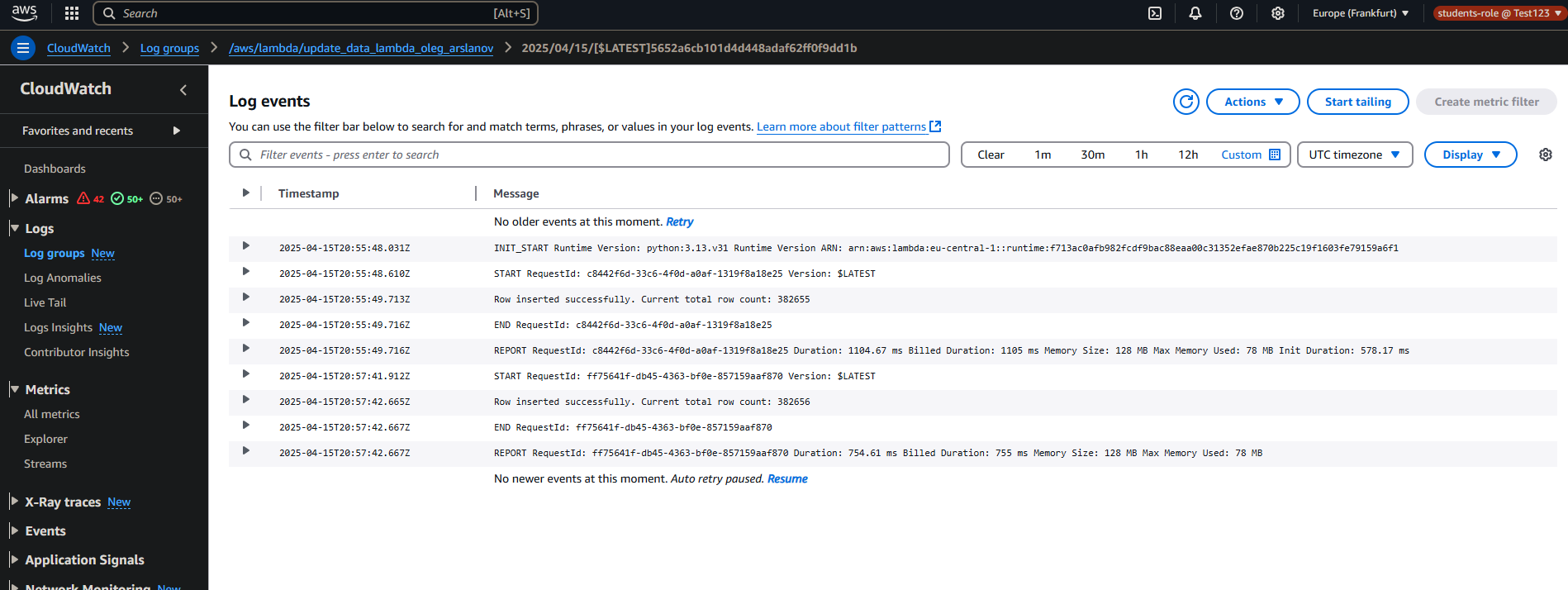
[Redshift\_check\_before\_lambda\_oleg\_arslanov | Functions | Lambda](https://eu-central-1.console.aws.amazon.com/lambda/home?region=eu-central-1#/functions/Redshift_check_before_lambda_oleg_arslanov?tab=testing)

[update\_data\_lambda\_oleg\_arslanov | Functions | Lambda](https://eu-central-1.console.aws.amazon.com/lambda/home?region=eu-central-1#/functions/update_data_lambda_oleg_arslanov?subtab=vpc&tab=testing)

[Redshift\_check\_afret\_lambda\_oleg\_arslanov | Functions | Lambda](https://eu-central-1.console.aws.amazon.com/lambda/home?region=eu-central-1#/functions/Redshift_check_afret_lambda_oleg_arslanov?subtab=vpc&tab=code)







In this case I don’t know why ‘count before’ = 0. Maybe You can explain where I am wrong :/