

Documentation: (*there are comments within the code to supplement)

Setup:

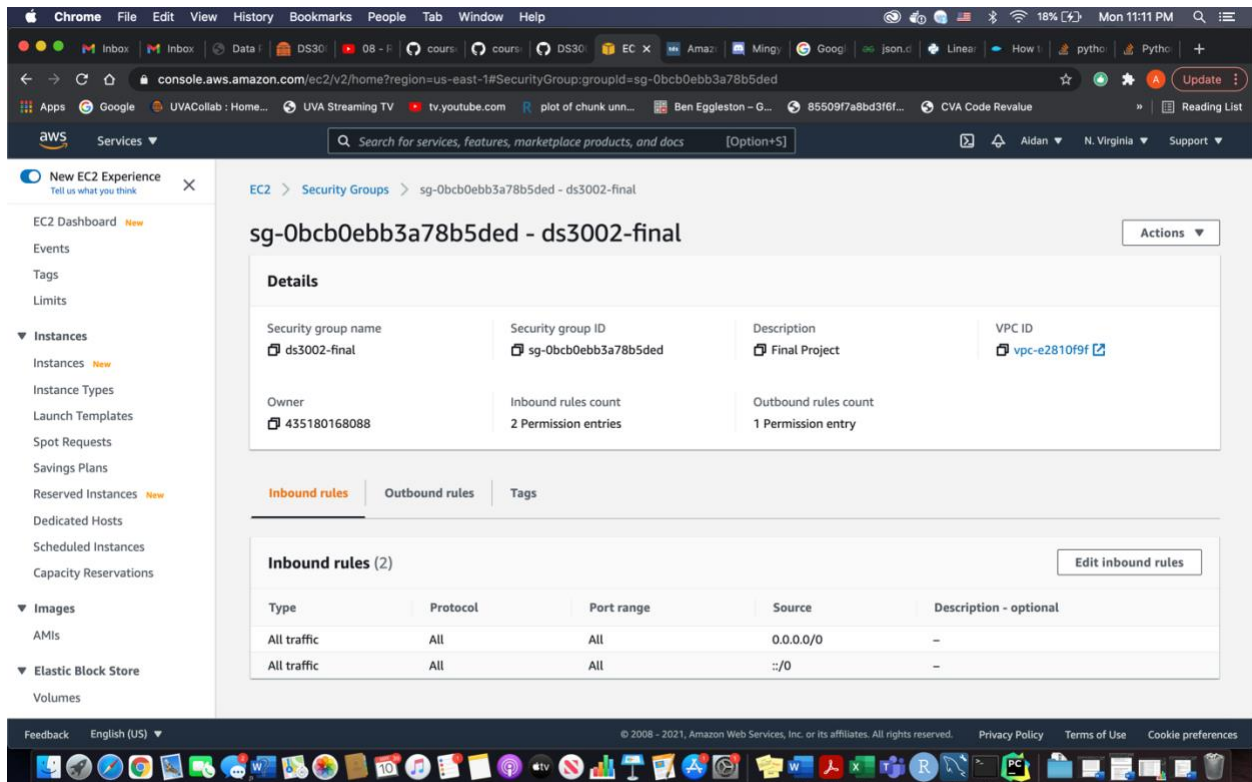
1. I created an instance in AWS RDS
 - I made sure to create a username/pw combination
 - Ensured the correct availability zone among other default settings

The screenshot displays the AWS Management Console interface for an Amazon RDS instance. The browser window shows the URL `console.aws.amazon.com/rds/home?region=us-east-1#database:id=ds3002final;is-cluster=false`. The left-hand navigation pane lists various RDS-related options, with 'Databases' selected. The main content area is titled 'Summary' and provides a high-level overview of the instance `ds3002final`. Key details include its role as an 'Instance', a CPU usage of 2.62%, a status of 'Available', and a class of `db.t2.micro`. It also specifies the engine as 'MySQL Community' and the region/availability zone as 'us-east-1d'. Below the summary, the 'Connectivity & security' tab is active, showing the endpoint `ds3002final.cmwmxkayq8t.us-east-1.rds.amazonaws.com` on port 3306. Networking details include the availability zone 'us-east-1d', VPC 'vpc-e2810f9f', subnet group 'default-vpc-e2810f9f', and subnet 'subnet-bb1f998a'. Security settings show the VPC security group `ds3002-final (sg-0bcb0ebb3a78b5ded)` is active, public accessibility is enabled, and the certificate authority is 'rds-ca-2019'.

Summary			
DB identifier ds3002final	CPU 2.62%	Status Available	Class db.t2.micro
Role Instance	Current activity 1 Connections	Engine MySQL Community	Region & AZ us-east-1d

Connectivity & security		
Endpoint & port Endpoint ds3002final.cmwmxkayq8t.us-east-1.rds.amazonaws.com Port 3306	Networking Availability zone us-east-1d VPC vpc-e2810f9f Subnet group default-vpc-e2810f9f Subnets subnet-bb1f998a	Security VPC security groups ds3002-final (sg-0bcb0ebb3a78b5ded) (active) Public accessibility Yes Certificate authority rds-ca-2019 Certificate authority date

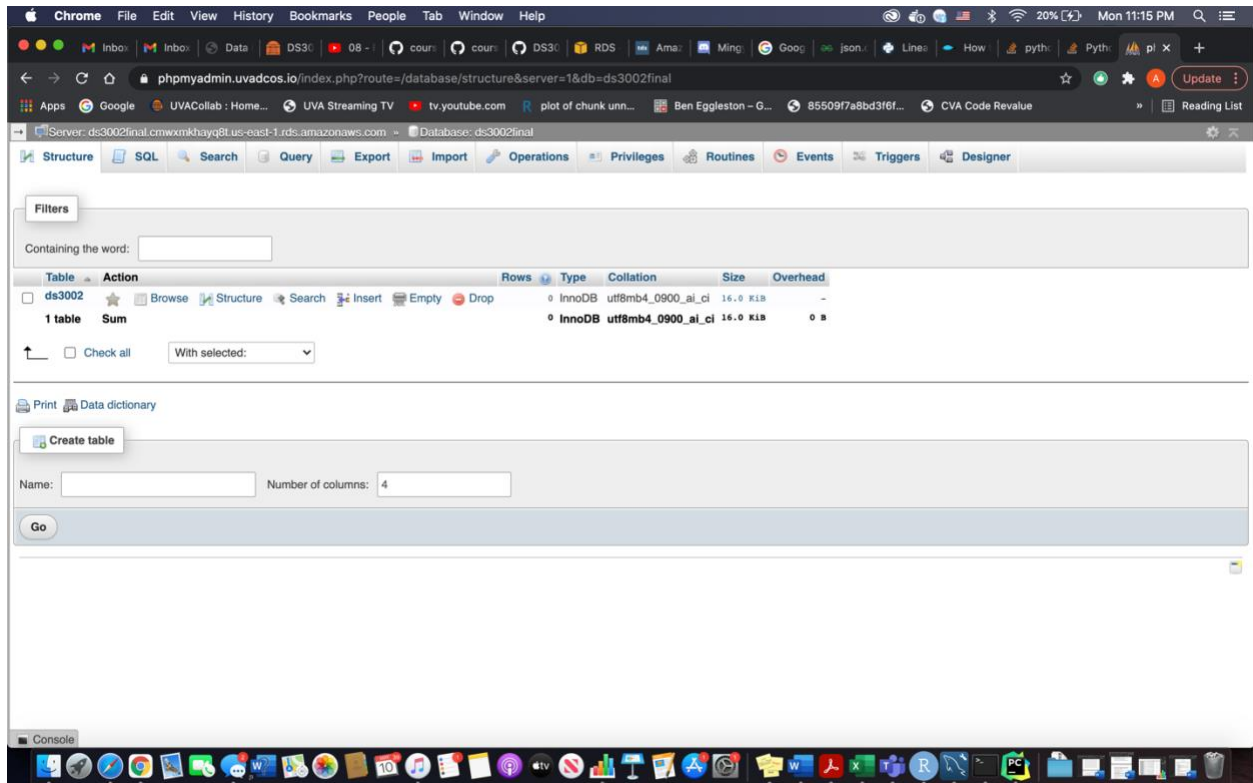
2. I created a VPC security group for my RDS instance
 - I added an inbound rule to allow all traffic from any source
 - Then, I changed the VPC security to my newly created EC2 security group
 - I also checked the outbound rules, which also allowed traffic anywhere and from any source



(see code for the following)

3. I accessed my database using my Python script using the information I made when I created my RDS instance.
4. I created a function to iterate 60 times, once every 60 seconds
5. I read in the API and stored the response
6. I tried to read it to the SQL database using both JSON and text, but both were unsuccessful

7. Then I created my SQL database, and connected it to my AWS RDS instance
 - Within that database, I created a table with columns for factor, pi, and time



(see code)

8. After the function had finished running, the data needed to be queried back to the script for analysis
9. Analysis was conducted