**QUESTION-1:**

Building a manual CI/CD Pipeline:

Code your Java application: Start by coding your Java application and create a Maven project.

Create a Git repository: Create a Git repository to store your code.

Define your manual pipeline stages: Define your manual CI/CD pipeline stages, such as Build, Test, Deploy.

Write a build script: Write a script to compile and package your Java application using Maven. The script should also copy the packaged JAR file to a designated location for deployment.

Write a test script: Write a script to test your Java application. For example, you can use the JUnit testing framework.

Write a deployment script: Write a script to deploy your Java application to a server or container. For example, you can use a shell script or Dockerfile.

Automate your pipeline: Once you have a working manual pipeline, you can automate it using a CI/CD tool like Jenkins.

Automating the CI/CD Pipeline with Jenkins:

Install and configure Jenkins: Install Jenkins on your server and configure it according to your needs. You'll need to install the necessary plugins for Java and Maven.

Create a Jenkins project: Create a new Jenkins project and configure it to pull the code from your Git repository.

Define build triggers: Define build triggers for your Jenkins project. For example, you can trigger a build every time a new commit is pushed to the Git repository.

Configure your Jenkins build: Configure your Jenkins build by specifying the Maven command to build your Java application and package it into a JAR file.

Configure test steps: Configureyour Jenkins build to run your test.sh script.

Configure deployment steps: Configure your Jenkins build to deploy your Java application using a shell script or Dockerfile.

Test our automated pipeline: Test our automated CI/CD pipeline by pushing a new commit to your Git repository and verifying that Jenkins automatically builds, tests, and deploys your Java application.

And that's it! You now have a manual CI/CD pipeline for your Java application and have automated it using Jenkins.

**QUESTION-2:**

**Create an EC2 intance for a Reactjs application and deploy the application and provide the public ip for it, also enable the cloud monitoring on this instances**.

**Step-1:** First we create instance with ubuntu image.

**Step-2:** Now connect the instance and give the commands.

**Commands:**

**To update ubuntu instance**: sudo apt-get update

**To upgrade ubuntu system**: sudo apt-get upgrade

**To install ngnix:** sudo apt-get install ngnix -y

**To check ngnix version**: ngnix -v

**To restart ngnix**: sudo systemctl restart ngnix

**To install nodejs ubuntu**: sudo apt-get install curl

curl -sL https://deb.nodesource.com/setup\_14.x | sudo -E bash –

sudo apt-get install -y nodejs

**To create the react project**: create-react-app

npx create-react-app

react-tutorial

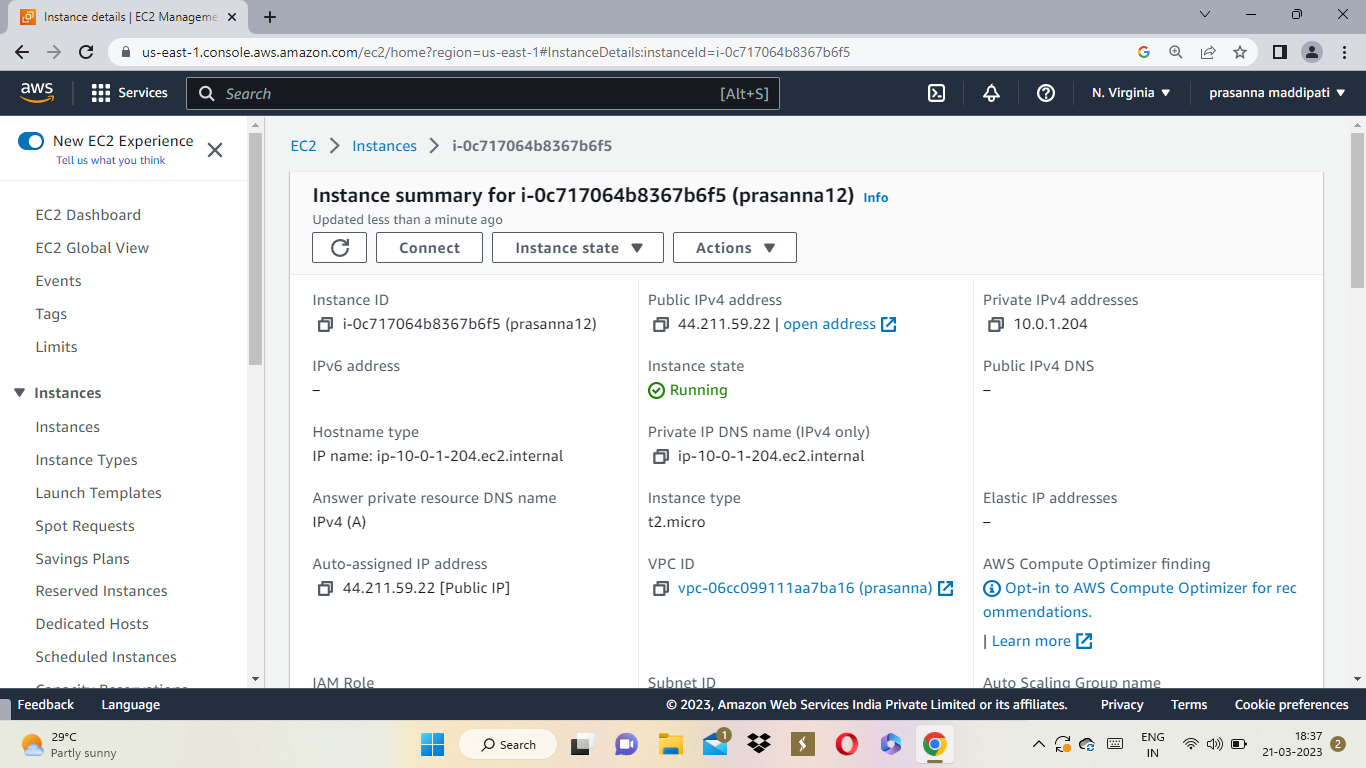
**To build the react project**: cd react-tutorial

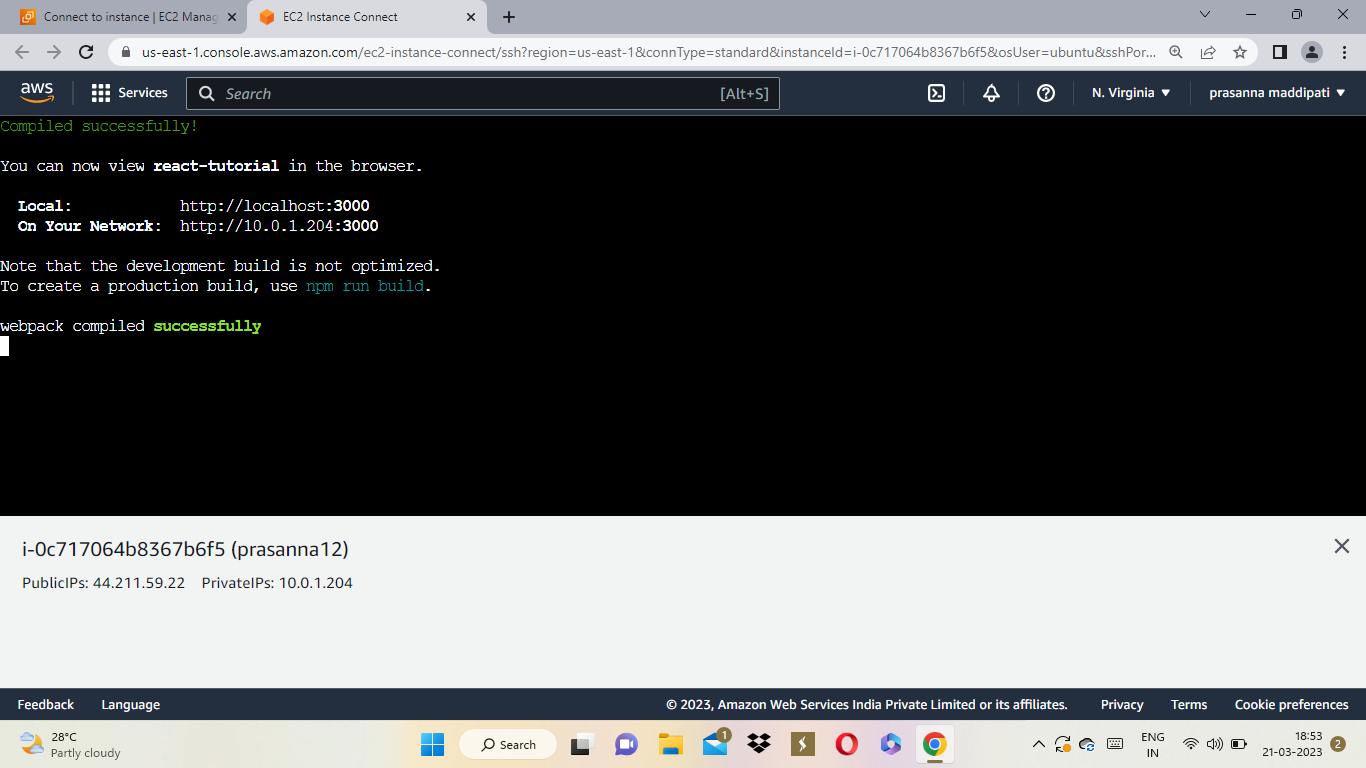
**To run the project**: npm start

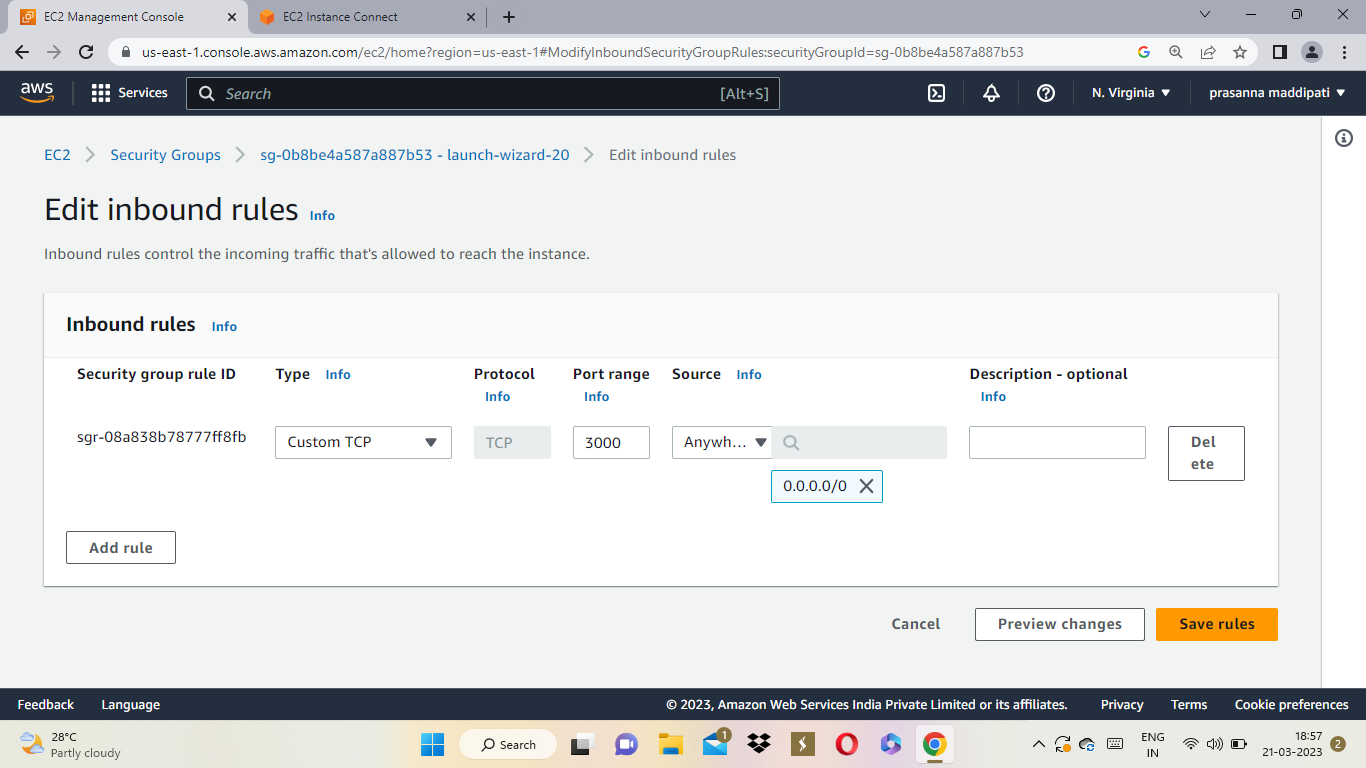
**Step-3:** Now go to instance security groups and edit inbound rules give the port range as 3000 and save changes.

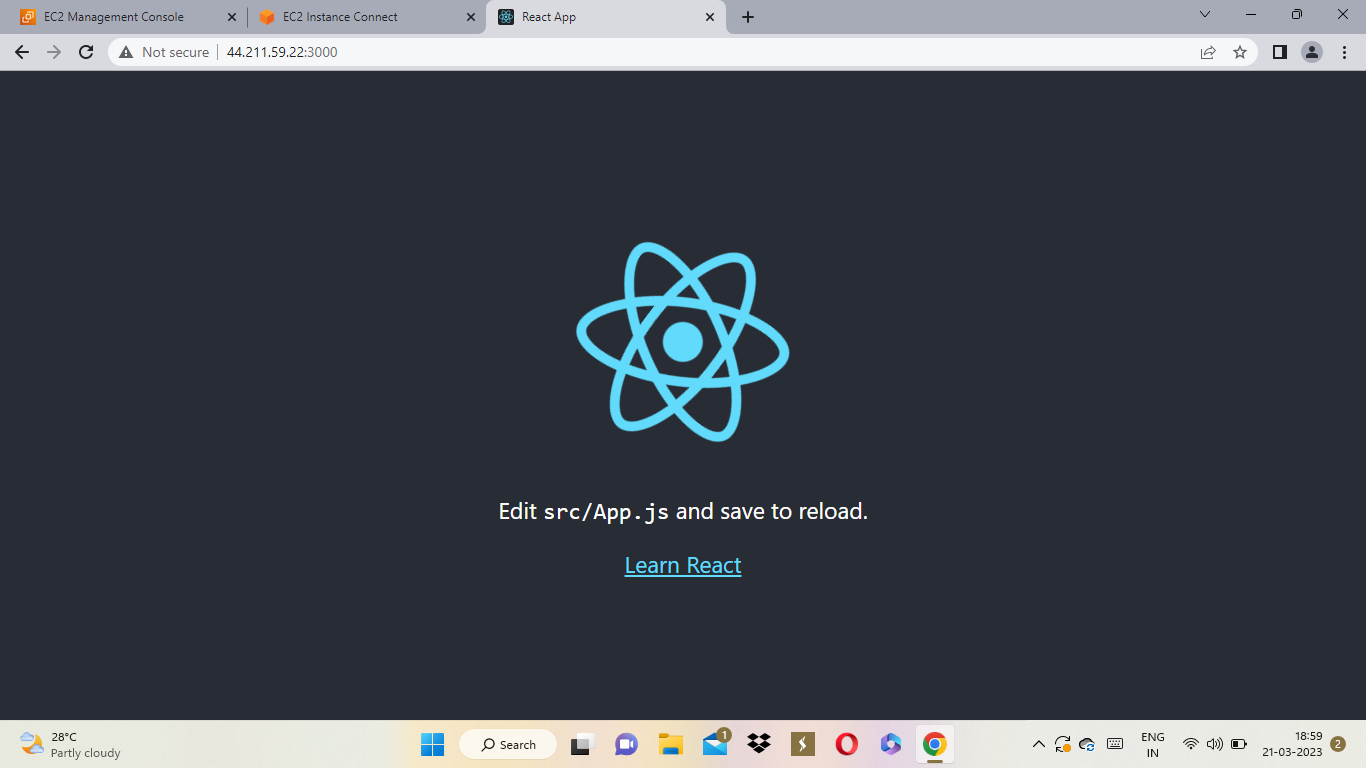
**Step-4:** copy the public ip and paste it in the new tab we get required output.

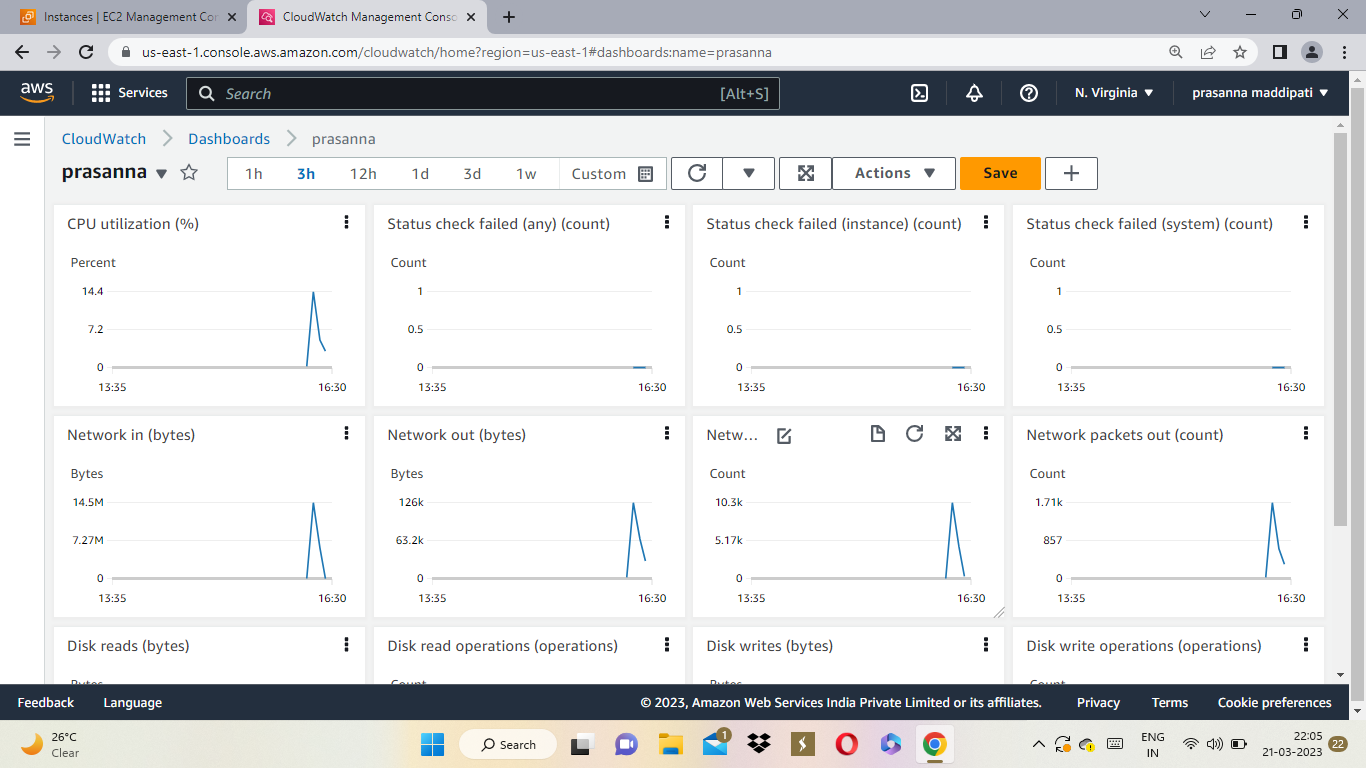
**Step-5:** create dashboard in cloudwatch and go to Ec2 select instance go to monitoring click on manage detailed monitoring and enable it add dashboard which we previously created and observe the resultant graphs.

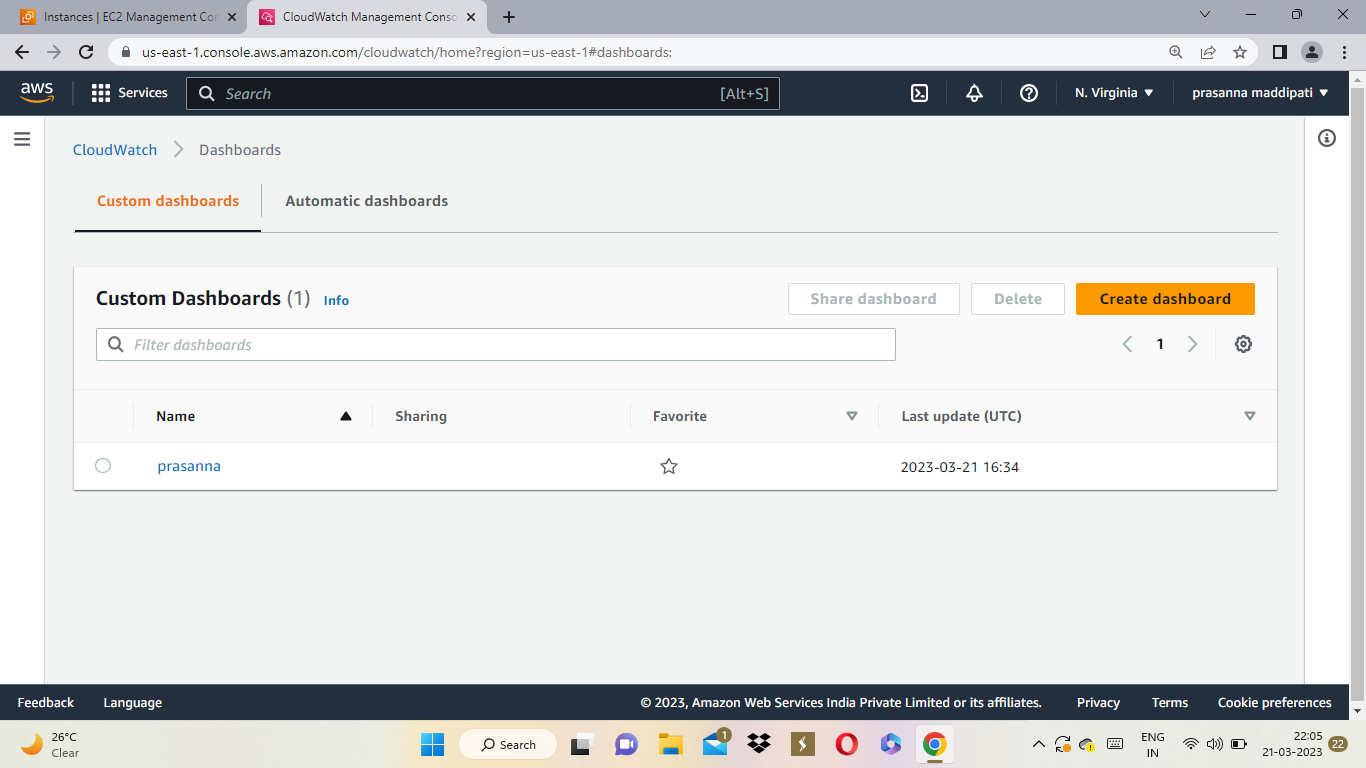












**QUESTION-3:**

**Create an rds connection with ec2 intance and use it to create an sql database and a sample table.**

**Step-1**: create database in rds service.

**Step-2**: create an instance with ubuntu image.

**Step-3**: connect database to the ubuntu instance.

**Step-4**: connect the instance and give the following commands.

sudo su

sudo apt update

sudo apt-get install mysql-client

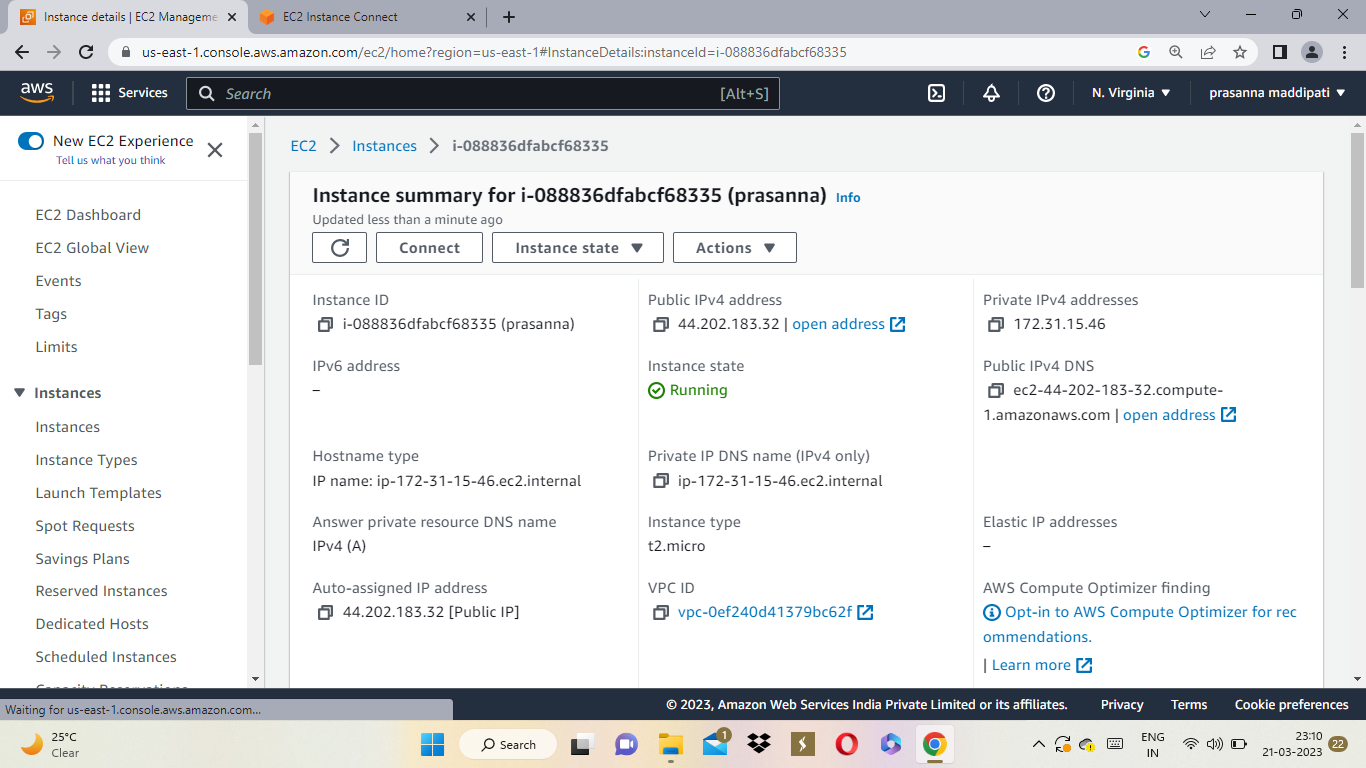
mqsql -h endpoint -u username -p

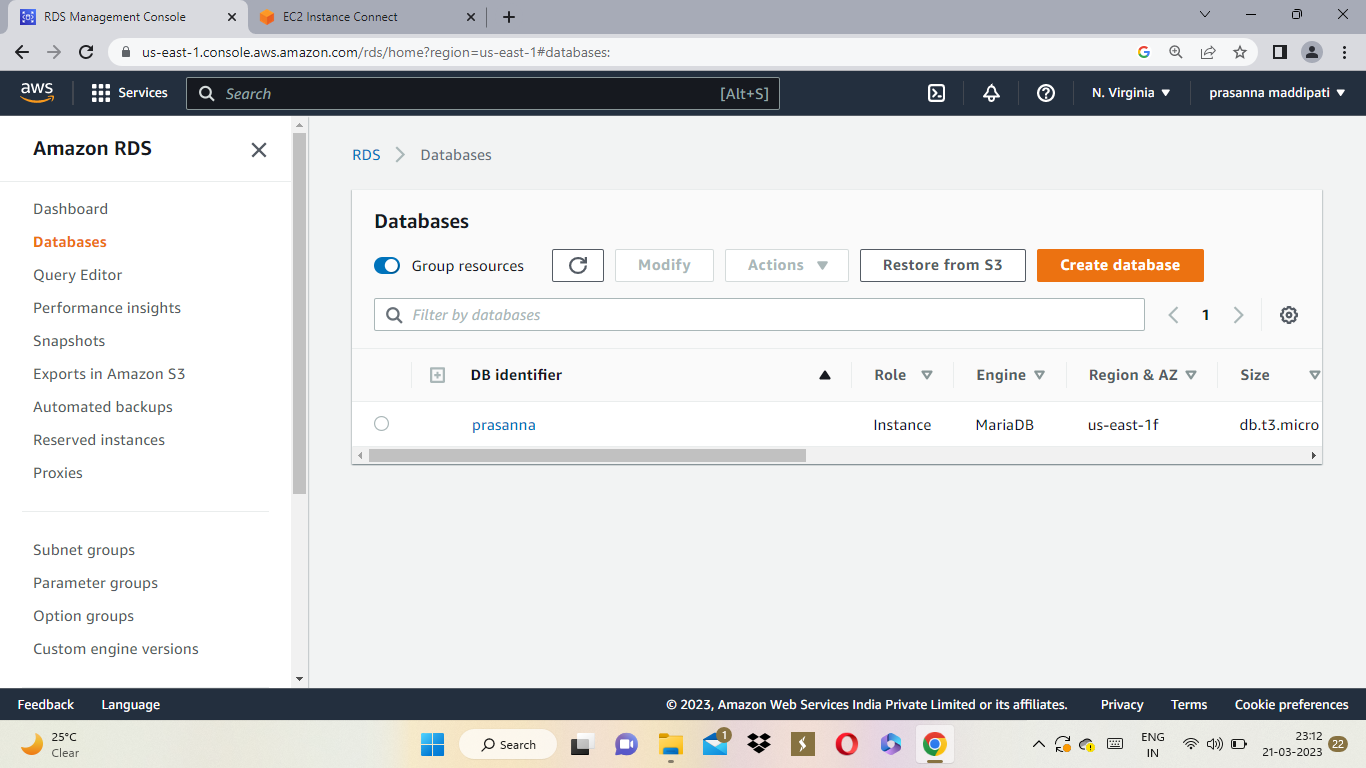
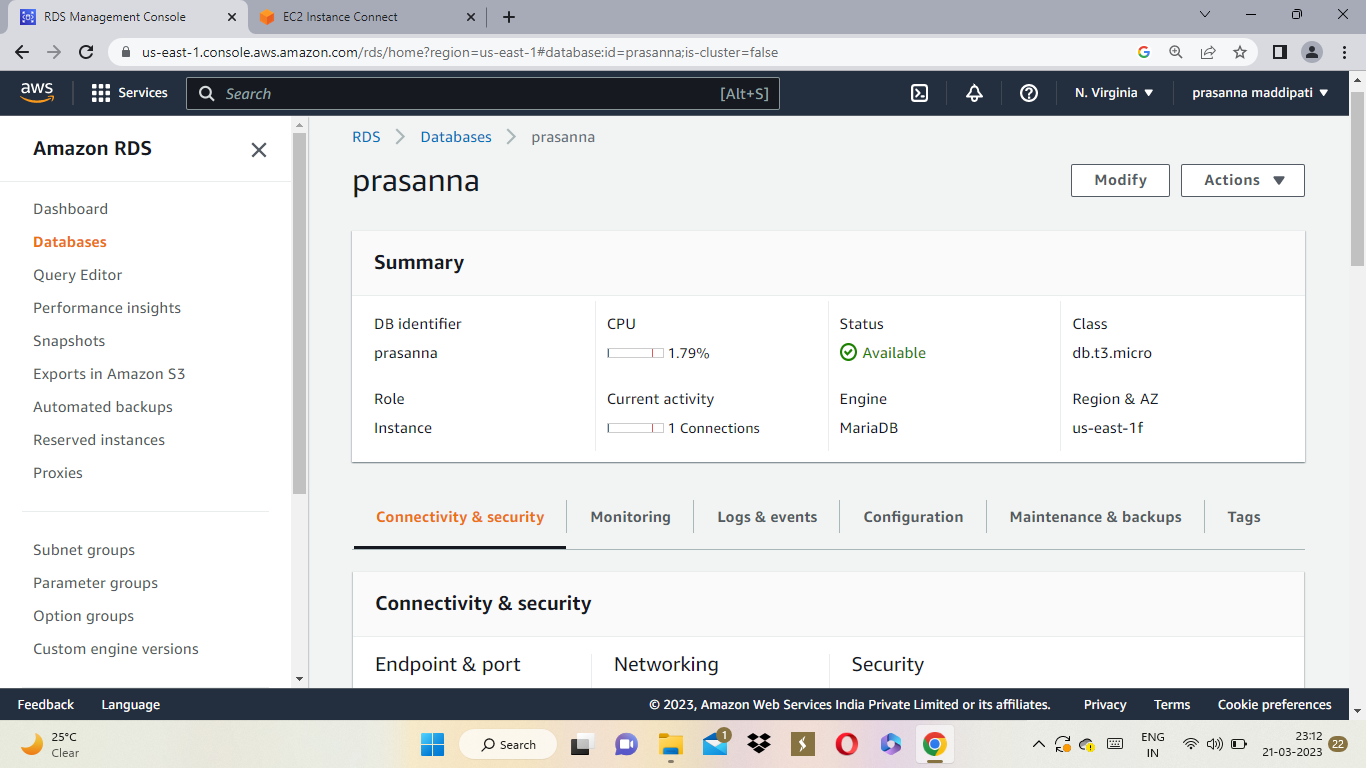
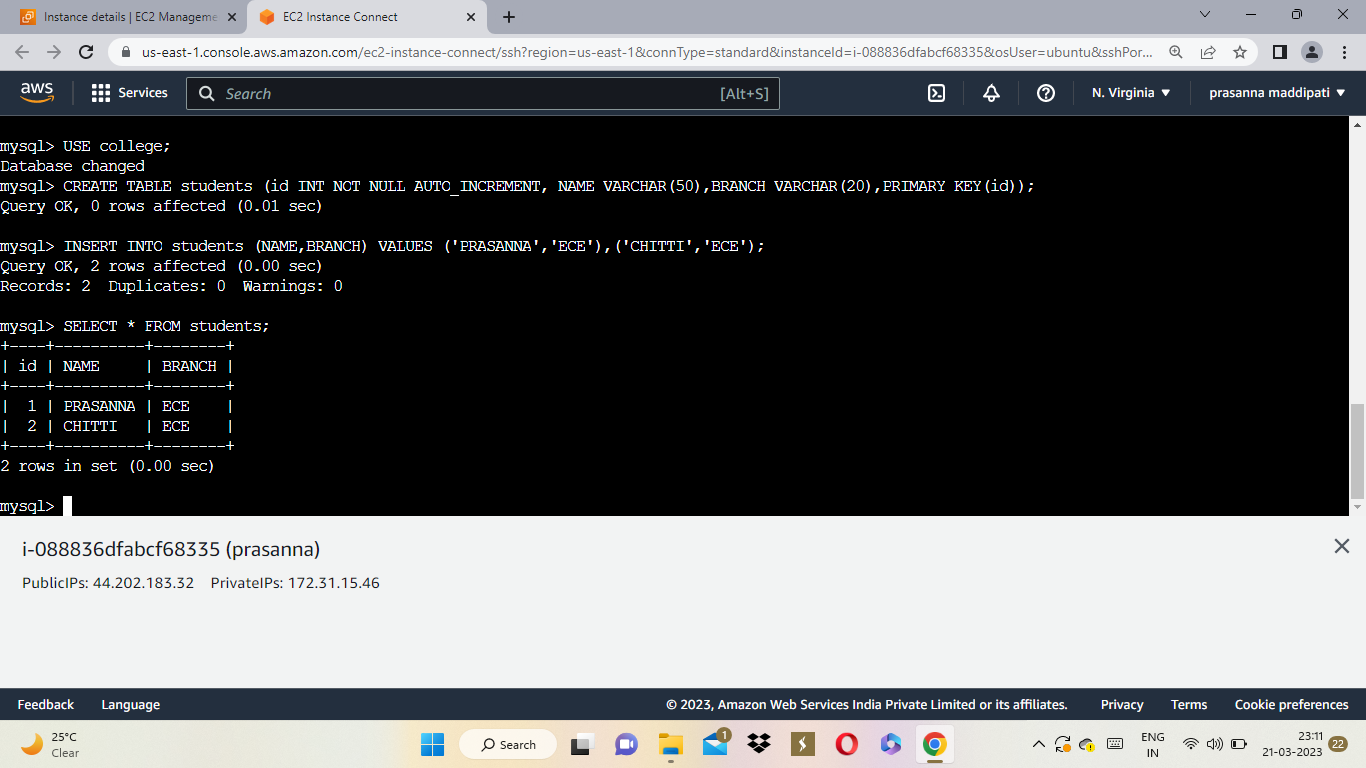
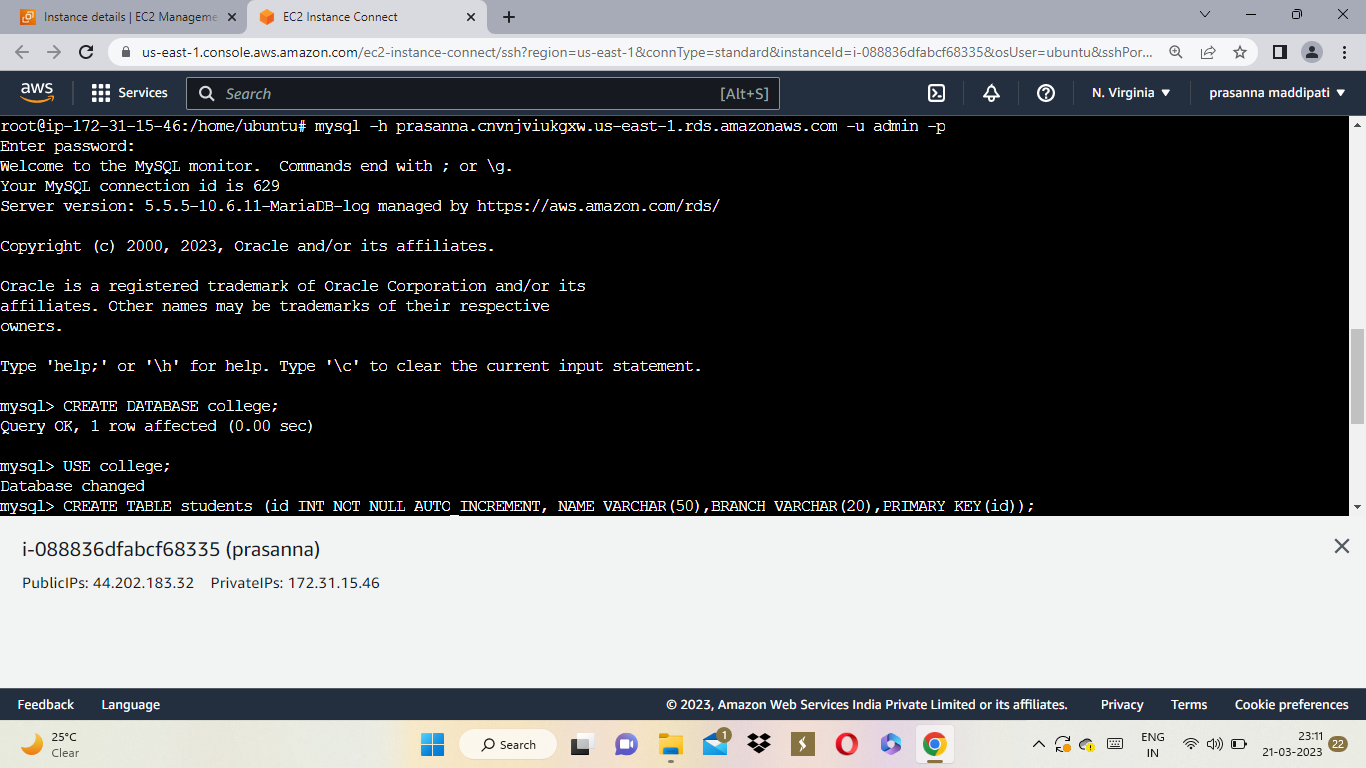
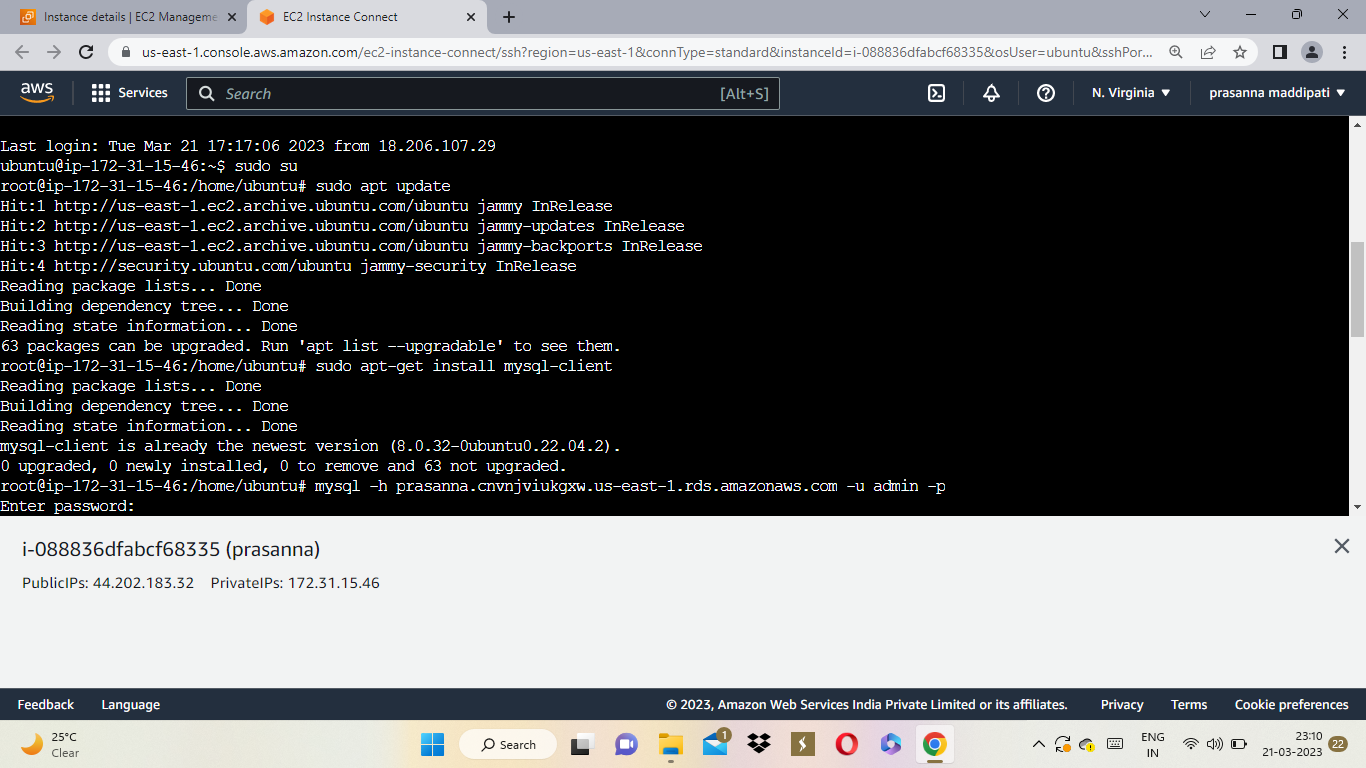
CREARE DATABASE database name;

USE databasename

CREATE TABLE tablename(id INT NOT NULL AUTO\_INCREMENT, NAME VARCHAR(50), BRANCH VARCHAR(20), PRIMARY KEY(id));

**Step-4:** By using these commands we get our required output.





QUESTION-4:

**Create an S3 bucket and deploy the reactjs static application on it and provide with the public url for use.**

**STEP-1:** create s3 bucket.

**STEP-2:** Give a bucket name and select acl's enabled option in object ownership.

**STEP-3:** give public access to the bucket.

**STEP-4:** click on create bucket.

**STEP-5:** Now upload files and folders related to reactjs application.

**STEP-6:** Then go to properties and make static webhosting enabled.

**STEP-7:** Now we have to give public access to the object, select the object go to actions and click on make public.

**STEP-8:** copy url and paste it on the new tab we get required output.

